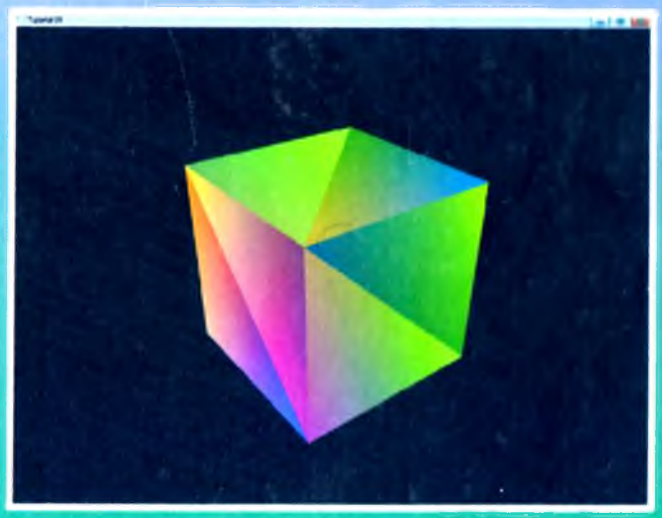
SH.A. NAZIROV, F.M. NURALIYEV,
B.Z. TO'RAYEV

KOMPYUTER GRAFIKASI VA DIZAYN



TOSHKENT

O‘ZBEKISTON RESPUBLIKASI
OLIY VA O‘RTA MAHSUS TA’LIM VAZIRLIGI

SH.A.NAZIROV, F.M.NURALIYEV,
B.Z.TO‘RAYEV

KOMPYUTER GRAFIKASI VA DIZAYN

O‘zbekiston Respublikasi Oliy va o‘rta mahsus ta’lim vazirligi tomonidan 5330400 – «Kompyuter grafikasi va dizayn» bakalavr ta’lim yo‘nalishi talabalari uchun o‘quv qo‘llanma sifatida tavsiya etilgan

TOSHKENT–2015

UO‘K: 004.92 (075)
KBK 32.973
N-18

N-18 **Sh.A.Nazirov, F.M.Nuraliyev, B.Z.To‘rayev. Kompyuter grafikasi va dizayn. O‘quv qo‘llanma. –T.: «Fan va texnologiya», 2015, 256 bet.**

ISBN 978–9943–990–80–7

Mazkur o‘quv qo‘llanma 5320600 – «Audio-video texnologiyalar», 5330400 – «Kompyuter grafikasi va dizayn», 5350200 – «Televizion texnologiyalar», 5330200 – «Informatika va axborot texnologiyalari», 5111000 – «Kasb ta’limi (5330200-Informatika va axborot texnologiyalari (tarmoqlar bo‘yicha))» kabi bakalavr ta’lim yo‘nalishlarida o‘qitiladigan «Kompyuter grafikasi va dizayn» o‘quv fani mazmuni asosida tuzilgan.

Qo‘llanmaning maqsadi grafik dizayn elementlari hamda kompyuter grafikasi asoslari: geometrik almashtirishlar, geometrik proeksiyalash, rastr grafikasi, rang, yorug‘lik va hokazolar haqida tushuncha berish. Bir so‘z bilan aytganda realistik tasvirlarni yaratish va ularning harakatini bajarish. Shuningdek, geometrik almashtirishlar, geometrik proeksiyalash, rastr grafikasi, rang va yorug‘lik bilan ishlashni **OpenGL** grafik kutubxonasi yordamida amalga oshirish, bunda **OpenGL** grafik kutubxonasi buyruqlarini asosiy sintaksisi va ularni ishlatish yordamida real tasvirlarni yaratish keltirib o‘tilgan.

Mazkur o‘quv qo‘llanma oliy o‘quv yurtlari talabalari uchun mo‘ljallangan bo‘lib, undan kasb-hunar kollejlari o‘quvchilari va barcha qiziquvchular foydalanishlari mumkin.

UO‘K: 004.92 (075)
KBK 32.973

Taqrizchilar:

N.Mirzayev – texnika fanlari nomzodi, dotsent;
A.Xaydarov – fizika-matematika fanlari nomzodi, dotsent;
A.Muhamadiyev – fizika-matematika fanlari nomzodi, dotsent.

ISBN 978–9943–990–80–7

KIRISH

XXI asr haqiqatdan ham axborot texnoloyiyalar asri ekanligini tan olish joiz, chunki biz yashab turgan ushbu davrda axborotlar oqimi shu darajada jadal rivojlanib bormoqda-ki, bu jarayonni ko'rmaslik mumkin emas va har birimiz usbu jarayonning ma'lim ma'noda ishtirokchisiga aylanib ulgurdik.

Zamon talablari va sanoat ehtiyojlaridan kelib chiqgan holda «Kompyuter grafikasi va dizayn» fani har bir soha bilan uzviy bog'lanib, unga bo'lgan ehtiyoj tobora o'shib borayotganligini kuzatish qiyin emas.

Ma'lumki, axborot almashinuvida insonning ko'rish sezgi organi yordamida qabul qilingan axborot eng samarali hisoblanadi va u xotirada ham chuqur iz qoldiradi. Tovush vositasida berilgan axborot ham ijobiy ta'sir etadi. Ammo axborot almashinuvi nafaqat so'zlar va tovushlar, balki tasvirlar, ranglar va shakllar bilan ham amalga oshiriladi. Buning yorqin dalili sifatida turli xil kitoblar, daftar va jurnallar muqovalari, ko'chalar yoqasida va binolar peshtoqida ilingan reklamalar, ommaviy axborot vositasi bo'lgan televideniye orqali uzatilayotgan turli xildagi kinolar, kliplar va boshqa ijtimoiy-madaniy ko'rsatuvlar, gazeta va internet orqali berilayotgan manbalarning naqadar did bilan ishlanganligi, uyali aloqa vositalarining platformalaridan ham ko'rishimiz mumkin. Albatta, ushbu ishlar zamirida yurtimiz iqtisodiyotini ichki va tashqi bozorda yanada mustahkamlash va xalq farovonligini ta'minlash uchun o'zining intellektual qobiliyatlarini namoyon etadigan yuksak malakali mutaxassis kadrlar tayyorlash kabi vazifalarga bog'liq ravishda oliy ta'lim muassasasining ilmiy salohiyati va moddiy-texnik ta'minlanganligi muhim ahamiyat kasb etadi. O'zbekiston Respublikasi Prezidentining «Oliy ta'lim muassasalarining moddiy-texnika bazasini mustahkamlash va yuqori malakali mutaxassislar tayyorlash sifatini tubdan yaxshilash chora-tadbirlari to'g'risida» 2011 yil 20 maydagi PQ–1533-son qarori hamda «Axborot-kommunikatsiya texnologiyalari sohasida kadrlar tayyorlash tizimini

yanada takomillashtirish chora-tadbirlari to‘g‘risida» 2013 yil 26 martdagi PQ–1942-son qarori, shuningdek, «2011–2016 yillarda oliy ta’lim muassasalarining moddiy-texnika bazasini modernizatsiya qilish va mutaxassislar tayyorlash sifatini tubdan yaxshilash Dasturi» oliy ta’lim sohasida qator yo‘nalishlar bo‘yicha faoliyat ko‘rsatish va ta’lim mazmunini takomillashtirishni talab etdi. Jumladan, o‘quv va ilmiy laboratoriyalarni lingafon kabinetlari hamda ulardagi ilmiy asbob-uskunalari, jihozlari zamon talabiga mos ravishda yangilanishini jadallashtirish, fanning eng ilg‘or yutuqlari bilan boyitilgan o‘quv adabiyotlari, zamonaviy kompyuter texnologiyalarining texnik va dasturiy vositalari bilan ta’minlash, axborot resurs markazlarining avtomatlashtirilishi va Internet tarmog‘iga chiqish imkoniyatini yaratish kabi vazifalar belgilangan. Hozirgi kunda ushbu vazifalarga bog‘liq ravishda respublikada zamonaviy axborot-kommunikatsiya texnologiyalari sohasida yangi o‘quv adabiyotlarni yaratish, axborot resurs markazlariga joylashtirish va ulardan samarali foydalanishni rivojlantirishga alohida e’tibor qaratilayotganini ko‘rish mumkin.

Ushbu vazifalarga bog‘liq ravishda mazkur o‘quv qo‘llanma yurtimiz oliy ta’lim tizimidagi bakalavr bosqichida o‘qitiladigan «Kompyuter grafikasi va dizayn» o‘quv fani mazmunini yoritishga bag‘ishlangan. Kompyuter grafikasi va dizaynning qo‘llanish ko’lami juda ham keng bo‘lib, avvalo ushbu sohaning vizualligi va dizayni diqqatga sazovordir. Grafik dizaynda o‘lcham, shakl, rang teksturasi, kompozitsiya, ko‘chirish va shriftlar muhim ahamiyatga ega. Berilgan topshiriqni mavjud grafik dasturlarda bajarish va kerakli natijaga erishish uchun shakllar, shriftlar va ularning o‘lchamlari bilan ishlash, ularga rang berishda rang modellari va tekstura haqida tasavvurga ega bo‘lish, kompozitsiyani bilish, tasvirni kompyuter ekraniga chiqarish va u bilan bog‘liq amallarni bajarish foydalanuvchidan ma’lum darajada geometrik bilimlarni talab etadi. Jumladan, obyektlarni tekislikda va fazoda almashtirish, proeksiyalash, fazoda tasvirlash, ko‘rinmas chiziq va sirtlarni olib tashlash, bo‘yash, nurning yo‘nalishini kuzatish, rang modellari haqidagi ma’lumotlar qo‘llanmada o‘z aksini topgan.

Keltirilgan nazariy ma’lumotlar asosida real obyektlarni yaratish, ikki va uch o‘lchovli grafika sohasida ilovalar yaratish

uchun keng tarqalgan amaliy dasturiy interfeyslardan biri hisoblangan OpenGL muhiti qarab chiqilgan. OpenGLda axborotlar birligi uchlar hisoblanadi va ular yordamida murakkab obyektlar quriladi. Dasturchi uchlarni yaratadi va ularni qanday birlashtirish (chiziqlar yoki ko'pburchaklar orqali) kerakligini ko'rsatadi, kamera va chiroq koordinatalari hamda parametrlarini o'rnatadi, OpenGL esa ekranda tasvirni yaratish ishi bilan shug'ullanadi. OpenGL dasturchilar uchun katta bo'lmagan uch o'lchovli sahnani qurishda juda ham qo'l keladi. Uch o'lchovli grafika algoritmlarini amalga oshirish tafsilotlari haqida o'ylashga esa hojat yo'q. Uch o'lchovli dasturlashtirish bilan shug'ullanuvchi professionallar uchun ham kutubxona foydali, chunki u asosiy mexanizmlarni namoyon etadi va belgilangan avtomatlashuvni bajaradi. OpenGL apparatga bog'liq bo'lmagan kutubxona hisoblanadi. OpenGL dan foydalanib uch o'lchovli sahnani osongina yaratish, unga teksturalar qo'yish, yorug'lik manbalari bilan yoritish, shaffoflik, tuman effektini berish, ranglarni aralashtirish, shuningdek, trafaret joylashtirish, sahna obyektlarini, kameralar va chiroqlarni belgilangan trayektoriya bo'yicha harakatlantirish, aynan animatsiya tayyorlash mumkin.

I bob. DIZAYN ASOSLARI

1.1. Kompyuter grafikasining asosiy tushunchalari

Bu bobda kompyuter grafikasining texnik jihatlari, ya'ni zamonaviy texnologiyalarda aniqroq mo'ljalni olishga va oldimizga qo'yilgan masalaning yechimini topish uchun to'g'ri usulni tanlashga hamda mustaqil ravishda yangi dasturlarni o'zlashtirishga yordam beruvchi bilimlar yoritiladi. Shuningdek, rang qanday holatda joylashgan, tasvir qanday saqlanadi, fayl formatlari nima bilan farqlanadi, boshqa ko'pgina grafik dasturlarning ishlash jarayonida sizga ixtiyoriy grafikli muharrir bilan ishlash jarayonini yengillashtirishga va tezlashtirishga imkon beradi. Mazkur bo'limga xos bo'lgan ma'lumotlar [4, 6, 7, 8, 9, 13, 16] kabi adabiyotlarda ham turlicha ketma-ketlikda va atroflicha yoritib berilgan.

Rang modellari.

Rang modeli – poligrafiyada yoki monitorning rangli kanallarida foydalanish mumkin bo'lgan bo'yoqlarning chegaralangan soni yordamida ranglar namoyish qilinadigan tizim. Turli xil rangli modellarda ranglar turlicha formulalar yordamida ifodalanadi.

Rang modelini ranglar to'plami, ya'ni rang modeli qismlari birikmasi yordamida hosil bo'ladigan palitra bilan adashtirmaslik kerak.

Yana rang modelini tasvirni kodlashtirish usuli bilan aniqlanadigan fayl formati bilan adashtirmaslik lozim (fayl formatlari haqida ushbu bobning keyingi bo'limlarida to'liq ma'lumotlar keltiriladi). Masalan, fayl probal.gif quyidagi parametrlarga ega bo'lishi mumkin: RGB – rang modeli, Web safe – palitra, rastr tasvir turi, GIF – formatli fayl.

Rang modeli chop etishga yoki tasvirni monitorda namoyish qilinishiga qarab asosiy ranglarni ajratish yoki qo'shish yordamida tus yaratishni mo'ljallaydi.

Har xil rang modellari rang poligrafiyasiga yo'naltirilgan. Ular sirtdan tasvirlangan ranglarga (spektrni ma'lum qismini ajratishdan

hosil bo'luvchi) asoslangan va shuning uchun bunday modellarda 2 rangning birikmasi ularning alohida-alohida ko'rinishidan to'qroq ko'rinish beradi. Bunday modelning texnik tatbig'ida tiniq bo'yoqlar bilan chop etiluvchi bir necha (odatda 3-4) ranglar ishlatiladi. Qog'ozga birin-ketin ranglar beriladi, ularning birlashmasi tuslar to'plamini va sifatli rangli tasvirni beradi. Chop etishda tiniq siyohlar bilan birgalikda barcha foydalanadigan tuslar bo'yoqlariga ega bo'lishi muhim. U juda noqulay va natijada chop etish sifati ma'lum darajada pasayadi. To'g'ri, bu usul ma'lum bir rangni aniq tanlashga imkon beradi, (masalan, logotipning aniq tusini namoyish qilish talab qilinganda) hamda agar ranglar to'plami chegaralangan bo'lsa, xom-ashyoni tejaydi, - masalan, faqat qora va qizil ishlatiladi.

Ranglar qo'shilmasi asosidagi rang modellari monitor ekranida tasvir demonstratsiyasi uchun va turli xil ko'rinishdagi dasturiy mahsulotlar uchun keng qo'llaniladi.

Additiv modellar boshqa modellardan farqli ravishda nur tarqatuvchi ranglar orqali ifodalanadi va ularda nurlar qo'shilmasining natijasi, alohida qaralganligidan yorqinroq. Agar siz web-saytlar yoki taqdimot turidagi mahsulot ishlab chiqarish bilan shug'ullanishni rejalashtirgan bo'lsangiz, u holda sizga ana shunday rang modellaridan foydalanishga ehtiyoj tug'iladi.

CMYK rang modeli.

Poligrafiyada asosiy professional grafik muharrirlarda standart bo'yicha taklif qilinuvchi eng ko'p tarqalgan rang modeli (Cyan, Magenta, Yellow, blacK - CMYK). Bu modelning nomi ko'pqatlamli chop etishda qo'llaniladigan 4 ta tiniq bo'yoqlardan kelib chiqqan. Nashriyotda chop etish vaqtida 4 ta bo'yoq uchun 1 ta klishedan foydalaniladi va qog'ozga ketma-ket har bir klishelarga berilib boriladi. Siyoh tiniq bo'lgani uchun ularni bir joyga berish mumkin va shu tarzda millionlab rang tuslari olinadi.

CMY rang modeli.

Bu model mutaxassislar tomonidan kam ishlatiladi. Ba'zi bir printerlarda tuslarning namoyishi uchun atigi 3 ta rangdan foydalaniladi.

3 ta asosiy ranglar qo'shilmasidan hosil bo'lgan qora rangning, to'yinganligi afsuski talab darajasida emas. Qora rangning tejalishi boshqa bo'yoqlarning ko'p sarflanishiga olib keladi, bu uydagi tez ishlovchi printerlarda yaxshigina bilinadi.

RGB rang modeli.

Bu keng tarqalgan rang modeli bo'lib, monitor ekranida tasvirni qayta tiklash uchun mo'ljallangan. Unda monitor 3 ta nur ranglari foydalaniladi: qizil, yashil, ko'k (Red, Green, Blue). Har bir nurning intensivligi 0 dan 255 gacha bo'lgan qiymatlarni qabul qilishi mumkin. Rangli kanalning intensivligi qanchalik kam bo'lsa, rang shunchalik to'q, ko'p bo'lsa, shunchalik och rang bo'ladi. Uchala rangimizning intensivligi 0 bo'lsa, qora rangni olamiz va aksincha barcha rangli kanallarimizga 255 qiymatini bersak, umuman oq rangni olishimiz mumkin.

Bu modelning asosiy kamchiligi tasvir xususiyatini kerak bo'lgan sifatli poligrafiya uchun saqlash imkoniyati yo'qligidadir. Sababi «kompyuter» modellari bilan solishtirganda CMYK rangli gamma modeli chegaralangan, ravshanligi va to'yinganligi ekranning yorug'ligi bilan ta'minlangan ko'pgina ranglarni qog'ozga olish imkonsiz. Shuning uchun dizaynerlarga kerakli model bilan darhol ishlashi va «qog'oz» grafikasida CMYK modelidan foydalanishi uchun u nimaga tasvir tayyorlayotgani haqida aniq tasavvurga ega bo'lishi lozim. Chunki bir modeldan 2-modelga o'tayotganda tasvirning sifati yomonlashadi.

HSB, HSV, HLS rang modellari

Inson ko'zi qabul qiladigan ranglarning nusxasini olishdagi harakat bu modellarning asosida yotadi. HSB (Hue, Saturation, Brightness) modeli har bir rangning tusi, to'yinganligi va yorqinligi bilan aniqlanadi. Ba'zi paytlarda u HSV (Hue, Saturation, Value) va HLS (Hue, Lightness, Saturation) deb nomlanadi. Bu modellarning ranglar to'plamidan foydalanish qulay, lekin, ko'pgina dizaynerlar RGB modeli qoniqtirgani uchun bu modellardan foydalanishmaydi.

YIQ rang modeli.

Bu model mohiyati bo'yicha taniqli NTSI Amerika televizion standartining kompyuter variantidir. Rang faqatgina yorqinligi va 2

ta xromatik qismlari orqali ifodalanadi. Bu rangli televidinie uchun qulay, lekin, juda kerakli paytdagina bu modeldan kompyuter yoki chop etish uchun foydalanish mumkin.

LAB rang modeli.

CMYK, RGB va HSB modellarining yaxshi hususiyatlari mujassamlashgan yangi LAB modeli ranglarning ekranda a'lo darajada tasvirlanishi namoyishi uchun mos bo'lganidek, 4 xil rangli tiniq bo'yoqlar orqali chop etish uchun mosdir. Bu model o'zgacha hususiyatlarga ega bo'lgani bilan dizaynerlar eski odatlarini tashlamaganlari uchun bu modeldan kam foydalanishadi. Bu model nazariy jihatdan monitorga bog'liq bo'lmagan ravishda tasvirning sifatli bo'lishini ta'minlaydi. Model yorqinligida yashil va qizil, ko'k va sariq ranglar intensivliklarining o'zaro nisbati haqidagi ma'lumotlar saqlanadi.

Kulrang tuslar (Grayscale).

Oq-qora chop etish tartibi bo'yicha rangliga qaraganda past qiymatda bo'ladi va bu bilan rangni taqsimlanish zarurati ham yo'qoladi: chop etish uchun 1 ta klishe yetarli, fayl o'lchami kichiklashadi. Bu modelda, odatda, kulrang tonini 256 gradasiyasi va 1 ta rangli kanal ishlatiladi (qora rang). Kichik o'lchamli fayllar ba'zida (juda kam hollarda) Internetdagi yarimtonli tasvirlardan foydalanganimizda qo'l keladi, bu holda tasvirni RGB ga keltirishimiz kerak. Oq-qora variantning tuslarsiz namoyishi ham bo'lishi mumkin [7].

Palitralar.

Aniq bir grafik tasvir uchun foydalanadigan biror bir rang modeli asosida tashkil qilingan ranglar to'plami palitra deyiladi. Palitrada rang qancha kam bo'lsa, shuncha kam tasvir fayli bo'ladi. palitrani mustaqil ravishda yaratish mumkin.

Palitra nima uchun kerak? Grafikli faylda har bir piksel uchun rang qiymatini berishga to'g'ri keladi. Grafikli faylning tanasida tasvirning barcha piksellari tasnifi RGB modelini tashkil qiluvchilari qiymati orqali berilishi, faylning hajmini uzluksiz oshib ketishiga olib kelishi mumkin, grafika esa shundoq ham qattiq diskda katta joy egallaydi. Kerakli ma'lumot o'lchamini kamaytirish

uchun foydalanadigan ranglar sonini kamaytirish (ixtiyoriy rang bo'lishi mumkin) rang qiymati bilan emas, balki uning aniq kod nomeri bilan saqlash va uning RGB grafik muharrirni qayta ishlatishda yoki dastur ko'rinishida sanab o'tilishi kerak. Masalan, agar palitra o'lchami 1 bit bo'lsa, (faqat oq-qora rang) u holda tasvirning har bir pikseli 1 bit joyni egallaydi (1 yoki 0 qiymat qabul qiladi). 16 bitli palitra bilan tasvir rangli bo'ladi, lekin, bo'yoqlarni yaxshi bo'lgani ma`qul. Kerakli sifatdagi rangli uzatish ta'minlanadi, qachonki har bir piksel uchun palitrada 16 mln. rang tartibini beruvchi 24 bit yuritilsa (bu tartib True color deb nomlanadi). Bunda rang odatda 3 ta raqam ketma-ketligi ko'rinishida yoziladi (16 lik sanoq sistemasida): masalan, #FFFFFF – oq rang, #000000 - qora, #FF0000 - qizil, #00FF00 – yashil, #0000FF – ko'k, #FFFF00 – sariq, ko'k-moviyrang esa - #3366CC.

Bevosita foydalaniladigan palitralar.

Mukammal tarzda qurilgan grafik muharrirlaridagi palitralardan bevosita foydalanish oson va qulay bo'ladi: Default CMYK chop etish uchun, Default RGB ekranda namoyish qilinadigan taqdimotlar uchun.

Web uchun xavfsiz palitra.

Brauzerlar namoyishi uchun xavfsiz palitrada to'htalish alohida ta'kidlanadi. Bu katta bo'lmagan palitra – hammasi bo'lib 216 ta rang – Internet Explorerda sahifalarning ochilishini tezlashtirish uchun o'rnatilgan, uni Internet tarmog'i orqali jo'natish kerak emas. Agar siz tanlagan rang xavfsiz palitra bilan to'g'ri kelmayotgan bo'lsa, u palitrada mavjud bo'lgan ranglar qo'shilmasi orqali brauzerda yaratiladi va keyinchalik o'chirib yuboriladi. Shuning uchun Internetga tasvir tayyorlayotganda ranglar buzilish xavfini kamaytirish uchun albatta bu palitradan foydalanish kerak.

Kulrang nozik turlari palitrasi.

Shuni alohida ta'kidlash kerakki, oq-qora ofset usulida chop etiladigan tasvir va hujjatlarni tayyorlashda bu palitra ideal hisoblanadi, professional verstka uchun tasvirni o'tkazish ham lozim. Bu palitralar shu nomdagi rang modeli bilan yaxshi

moslashadi. Qoida bo'yicha dizayner sifatli tasvirlarni rangli variantini tayyorlashiga to'g'ri keladi.

1 bit.

Bu palitra oq-qora tasvirni beradi, fayl o'lchami ma'nosida eng tejamkor. Palitra 2 ta oq va qora ranglardan tashkil topadi. Yuqori darajali chop etish uchun ham mos keladi, lekin, tasvir sifati ancha pasayishi mumkin. Tasvirni bu palitraga o'tkazish qora va oq orasidan o'tkazish boshlanishida aniqlanishidan iborat.

Kulrangning 256 tuslari

Kulrangning 256 nozik turlaridan iborat palitra. Bu hol uchun TIFF fayl formatini qo'llash tavsiya etiladi. Internetda kulrangning nozik turlari uncha qo'llanilmaydi. GIF formatidagi tasvir kam uchraydi.

Percent of gray

Qoralikning foizlari orqali ifodalanadigan kulrangning tuslaridan iborat palitra oldingisidan ko'ra kamroq ishlatiladi.

PANTON tizimining palitralari.

Bu palitralar keyingi rang taqsimoti uchun yaxshi mos keladi, odatdagi palitradan ko'ra CMYK tizimida kam qo'llaniladi, lekin, kerak paytda ulardan foydalanish qiyin emas. Rang taqsimlanishidan keyin oldingisidan ko'ra grafikli muharrirda yaratilgan sifatli tuslar orqali hosil bo'lgan va siz foydalanmoqchi bo'lgan chop etish jarayoniga mos keluvchi tasvir hosil bo'ladi.

- PANTONE hexachrome – bu palitrada 6 xil bo'yoq ishlatiladi (moviy rang, sariq, yashil, qora, to'q sariq va to'q qizil).
- PANTONE metallic colors – yorqin bo'lmagan rangning metallik tuslaridan iborat.
- PANTONE matching system – tizimning moslashgan ranglari. Yorqin va laklanmagan qoplamaga mo'ljallangan, yaxshisi uning sifatini aniqlash uchun muharrirda bu palitrani ko'rish.
- PANTONE process color system – uchtalik ranglar. Birinchi 2000 tuslar 2 ta bo'yoq aralashmasidan foydalanishadi, qolganlari 3 ta yoki 4 ta. Muhim tomoni, ularning ko'rsatilgan qiymat ketma-ketligini qabul qiladi.

➢ PANTONE pastel color – pastel tuslar. Yoqimli oz to‘yingan bo‘yoqlar, yorqin, xira tuslar. Bu tizimning ko‘pgina palitralar singari «bolacha» dan ko‘ra ko‘proq stilli tasvirlarga mos keladi.

Boshqa palitralar.

➢ Uniform colors – 256 ta ranglardan iborat apparatga bog‘liq bo‘lmagan bazali palitra.

➢ Trumatch colors – CMYK uchun sifatli rang taqsimlanishni ta’minlovchi palitra. Ton, to‘yinganligi va yorqinligi bo‘yicha aniq tartibda taqsimlangan ranglar.

➢ HKS colors – fiksirlangan ranglar to‘plami.

➢ Focoltone colors – CMYK sistemasi bo‘yicha 4 rang aralashmasi uchun qulay sozlovchi palitra. Tuslar sonini 1 taga kamaytirish uchun bu palitrada har bir rang bir – biridan kamida 10 % ga farq qilishi kerak. Lekin, hamma grafik muharrirlarda o‘rnatilmagan.

➢ Lab colors – Lab rangli model uchun maxsus mo‘ljallangan palitra.

➢ Dic colors – maxsus standartdagi yapon palitrasi, multiplikatsiyada ko‘p qo‘llaniladi. Ko‘pgina dizaynerlar undan foydalanishmaydi.

➢ Toyo color finder – televidenie uchun qulay palitra. Ranglarni CMYK, RGB, LAB o‘zgartirishda kerak. TOYO standartida ishlashga mo‘ljallangan.

➢ Sperta master colors – «Dyupon» firmasining tipografik bo‘yoqlari bilan chop etish uchun maxsus palitra. Bu firmaning bo‘yoqlari yuqori sifati va keng tarqalganligi bilan ajralib turadi. Bu palitra grafik qurishlarda ham ko‘p ishlatiladi.

Rangning taqsimlanishi.

Rangli poligrafiyada rangning taqsimlanish jarayoni muhim. U tipografiya uchun fotoform tayyorlashni o‘z ichiga oladi. Avval rang taqsimlangan plyonkalar tayyorlanadi – bu plyonkalarda tipografiyada foydalanadigan har bir rang bo‘yog‘i uchun alohida qilingan tasvirlar bo‘ladi. Plyonkadagi rasm odatda kulrang, lekin, u kulrang tuslarini emas, balki shu plyonkaga mos keladigan rangli kanal yorug‘ligini beradi. Plyonkalar maxsus printerlarda qilinadi,

lekin, ularning elektron ko'rinishini mustaqil tayyorlash va tipografiya yoki shunday printerlarga ega tashkilot (servisbyuro) tayyor fayllarni chop etish uchun plyonkalarni berishi mumkin.

Fotoformalar tayyorlash.

Tayyor rang taqsimlangan plyonkalar yorug'lik sezuvchi metal plastinkalar ro'parasida joylashtiriladi va ular yordamida yoriladi. So'ng yoritilgan plastinkalar ximik usul bilan singdiriladi, singdirish chuqurligi plastinka nuqtasining yoritilganligiga bog'liq, yorilish ko'p bo'lsa, singdirish chuqurligi ham ko'p bo'ladi. Bu plastinkalarni ofset bosmada kog'ozga rang tushirish uchun ishchi sirt sifatida tipografik mashinalarda ishlatish mumkin, oddiy kanselyar shtamp kabi. Ushbu plastinkalarni tayyorlash umumiy bosma narxining asosiy qismini tashkil etadi, ayniqsa, kam tirajlarda rangli bosmalar uchun har bir rangga alohida plastinkalar kerak bo'ladi. Bosma har bir rang uchun alohida bajariladi, oq-qora va kulrang tuslar uchun esa bitta plastinka yetarli.

Rang chuqurligi.

Tasvirning rang chuqurligi – 1 pikselning rangi haqida ma'lumot saqlash uchun ajratilgan bitlar soni bilan aniqlanadi. Eng quyi sifati bu rang uzatish uchun atigi 1 bit kerak bo'ladigan oq-qora rasm. Biz pikselda qanchalik ko'p bitlardan foydalansak, shuncha tuslarning umumiy soni ko'p, chuqurroq ranglarga ega bo'lamiz. Lekin, bir vaqtning o'zida tasvir faylining o'lchami kattalashib boradi. Grafik fayllarning o'lchami ba'zida True color (pikselda 3 bayt) rejimidan foydalanishga to'sqinlik qiladi, tasvir sifati pastligi tufayli sifatli chop etish imkoniyati yo'qoladi. Odatda tasvirning fayl o'lchamini kamaytirish uchun yechim bilan rang chuqurligini kamaytirishga to'g'ri keladi. Zamonaviy monitorlar rang chuqurligining barcha qiymatlarida ishlatishga qodir.

Imkon beruvchi qobiliyat.

Ixtiyoriy chiqarish qurilmasida (monitorlarda va printerlarda) rastrlash qo'llaniladi – tasvir kichik yacheykalarga bo'linadi, bir xil rang bilan to'ldiriladi, natijada rastrli to'r deb nomlanuvchi tasvirga aylanadi. Chiqarish qurilmasining sifati uning imkon beruvchi

qobiliyati – rastrli to'rning 1 dyum uzunlikdagi kesimda joylashgan nuqtalar soni bilan harakterlanadi. Imkon beruvchi qobiliyat – bu tasvirning 1 dyum uzunlikka ega kesimga monitor yoki printer tomonidan kiritiluvchi nuqtalar soni. Bu kattalik *dpi* da o'lchanadi. Tasvirning imkoni qancha yuqori bo'lsa, o'tish uzatishlari shunchalik sifatli, ranglar fayl o'lchamidan shunchalik katta bo'ladi. Ba'zida aniq tasvir uchun ma'lum bir ruxsatni qat'iy berish zarurati tug'iladi, natijada rasm masshtabi o'zgarishi mumkin, mos ravishda imkoniyat ham o'zgarishi mumkin, shunday qilib, uni majburiy berish orqali biz sifat pasayishi xavfini kamaytirishimiz mumkin. Bundan tashqari, sifat yo'qolishi va yuqori sifatli qurilmada oddiy ekran tomonidan ruxsat etilgan tasvir chop etishga ham ta'sir qiladi. Internet va ekrandagi taqdimot uchun 100 dpi, standart bo'yicha chop etish uchun 300 dpi yetarli bo'ladi. Agar siz boshqa chop etish qurilmalaridan foydalanishni rejalashtirmagan bo'lsangiz, u holda printeringizning imkoniyatidan kelib chiqib tasvir imkoniyatidan foydalanishga to'g'ri keladi.

Printer imkoniyati va tasvir imkoniyati.

Hujjatni chop etishga tayyorlash vaqtida tipografiyaning aniq chop etuvchi qurilmasining sifatini e'tiborga olishga to'g'ri keladi. Ba'zida tipografiya CDR turidagi faylni qabul qiladi va rang taqsimlashni mustaqil o'tkazadi, lekin, grafik fayllarning rang taqsimlanishini dizaynerning o'zi tayyorlashga to'g'ri keladi. Oxirgi holda printer va tasvir imkoniyatining moslashuvini to'g'ri topish kerak.

Bu yerda bizga yana bitta rastr liniaturasi deb nomlanuvchi rastrlash xarakteristikasi kerak bo'ladi: u rastr to'rdagi chiziq chastotasini ifodalaydi va *lpi* da o'lchanadi. Shuni ta'kidlash kerakki, uzunlik birligiga chop etsa bo'ladigan, rastr nuqtasi yaratish uchun bir nechtasi kerak bo'ladigan «fizik» nuqta soni bilan printer imkoniyati aniqlangani uchun imkoniyatga ko'ra doim liniatura kam bo'ladi.

16 chi deb nomlanuvchi qoida mavjud: 16 ga bo'lingan rastr liniaturasi chop etuvchi qurilmaning imkon beruvchi qobiliyatidan ko'p bo'lishi kerak emas.

Chop etiluvchi tasvir sifatini yaxshilash uchun rasmdagi yarim tonlarning umumiy sonini kamaytirish ko'pincha foydali bo'ladi – bu rastr nuqtalarini maydaroq qilishga imkon beradi. Masalan, kulrangning 256 tuslarini ishlatishda 600 dpi imkoniyatiga ega printerning yarimton chiziq chastotasi 37 lpini tashkil etadi, rastr yacheykasining o'lchami esa 16*16 nuqtalar hosil bo'ladi, bu esa tasvir sifati pastligidan dalolat beradi. Agar kulrangning 100 ta tuslari bilan chegaralansak, yarimton chiziq chastotasi 60 lpi bo'ladi va tasvir sifati yaxshilanadi.

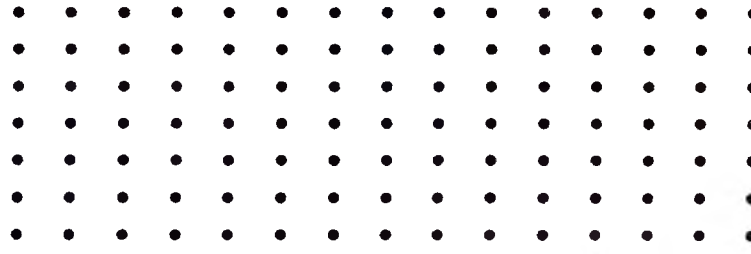
Quyidagi standart mosliklardan foydalanish tavsiya etiladi: printer imkoniyati 2400 dpi, liniatura 110 dpi, kulrang 256 tuslari; printer imkoniyati 300 dpi, kulrang 25 ta tuslari.

Tekislash.

Yuqorida aytilganidek, monitor ekranidagi chizmalar tasvir ko'rinishida bo'ladi, ya'ni piksellardan iborat to'rtburchak to'rga joylashtiriladi. Ekrandagi tasvir ko'rinishi to'rtburchakka yaqin piksellardan iborat ekran to'ri bilan tasvirdagi egri chiziqli konturlar mosligi bilan aniqlanadi. Albatta, ekran grafikasi uchun alohida ahamiyatga ega, chunki monitor piksellari ixtiyoriy holda sifatli printerning rastr nuqtalaridan ko'ra aniqroq va yirikroqdir. Agar 2 rangli sohani ajratuvchi chegara qat'iy vertikal yoki gorizontal bo'lmasa, u holda quyidagi savol tug'iladi: qanday qilib konturning egri chiziqligini yaxshiroq ko'rinishga o'tkaziladi, ya'ni uni piksellardan iborat to'rtburchak to'rga joylashtiriladi? Xuddi shunday muammolar egri chiziqli figuralarga ega tasvirni kattalashtirish mobaynida ham tug'iladi (masalan, matn yozuvlarida).

Bu muammoni yechishga yordam beruvchi usullardan biri konturlarni tekislash usulidan, ya'ni: tasvirning rangli chegarali yarimtonlari bilan to'ldiriladi, natijada konturlar mayinlashadi va tasvir bir muncha silliqlashadi. Tekislash effektining tasvir o'lchamini va uning imkoniyatini o'zgartirishda qo'llash zarurati tug'iladi. Lekin, ba'zida bu tasvirning yomonlashishiga olib keladi, masalan, mayda shrift quyuqlashib ko'rinmay ketadi. Shuning uchun har doim avval tajriba o'tkazib, natijalarni solishtirib eng yaxshi variantini tanlash lozim.

1.1-rasmda qora nuqtalar oldin mavjud bo'lgan tasvir olingan aniq bir rangga ega piksellar bilan ifodalanadi, oqlarini esa konturning oraliqlari chegarasida imkoniyatni kattalashtirishda ko'pincha to'ldirishga to'g'ri keladi. Ularning rang-barangligi rangning mavjud bo'lgan piksellari orasida tekis o'tishdek aniqlanadi.



1.1-rasm. Tasvirning alohida piksellari.

Tekislashning yana bir usuli mavjud ranglarni aralashtirish usulidir (dithering ba'zida diffuziya deb ham nomlanadi). Bu holda, agar palitrasi qirqilsa va tus oraliqlar qatori yo'qolsa, oldin bir xil rang bilan to'ldirilgan sohalar yo'qolgan palitra tuslarini aniqroq beruvchi turli xil rangli piksellar aralashmasi bilan to'ldiriladi. Bu bilan tasvir ko'pincha donadorlik (donacha donacha ko'rinishga) xarakteriga ega bo'ladi. 1.2-rasmda ko'rinib turibdiki, aralashtirish qo'llanilganda rasm chegaralari xiralashgan bo'lib qoladi.

1.2-rasm. Chapda berilgan tasvir; o'ngda ranglar aralashtirilishidan keyingi holati.

Bu metod tuslarning gradient o'tishlarini tekislashga imkon beradi va uni bosqichlilikdan xolos qiladi. Web uchun grafik tayyorlashda bu effektga e'tibor berish kerak, chunki bu ish vaqtida grafik fayllar o'lchamini qat'iy chegaralash zarur va ko'pincha tasvirni uncha katta bo'lmagan ranglar to'plami bilan GIF formatiga o'tkazishga to'g'ri keladi. Bundan tashqari, ba'zida brauzerlarning o'zi palitraning qirqilishini bajaradi. Bu web-uchun maxsus xavfsiz bo'lgan palitrani ishlatishdir.

Shrift.

Zamonaviy dizaynerlar 100 dan ortiq shriftlardan foydalanishlari mumkin. Shriftlarni bir-biridan katta farq qiluvchilari uncha ko'p emas. Lekin, ko'pgina foydalaniladigan shriftlar ko'p dasturlarning yuklanishini juda ham sekinlashtiradi. 1 betda 3-4 dan ortiq bo'lmagan shrift turlaridan foydalaniladi, yaxshisi 2 tasi bilan chegaralanish yetarli. Tajriba qilib ko'rishdan qo'rqmaslik kerak, lekin, judayam kirishib ketmaslik lozim, shuni yodda tutish kerakki asosiysi matnni o'qish va tushunishdir. Ko'pgina holatlarda matn uchun optimal bo'lgan 3 ta standart shrift turlaridan birini tanlaymiz. Times New Roman, Arial, Courer. Oq fonda qora matn oson o'qiladi.

Post Spript shriftlari.

Maxsus printerlarda vektor tasvirlarni sifatli chop etish uchun Adobe firmasi tomonidan Post Spript texnologiyasi ishlab chiqilgan: Type1 formatida shrift aynan shu texnologiyasi uchun mo'ljallangan.

Chop etishga chiqarish uchun hujjatga printerda o'rnatilgan izohlovchi bet yozuv tilida bajaradigan buyruqlar qo'yiladi.

Post Spript printerlari xonadon va ofislarda keng tarqalgan printerlardan ko'ra qimmatroq turadi.

True Type shriftlari.

Apple va Microsoft firmalari birgalikda standart printerlar va mashhur operatsion tizimlar Windows uchun True Type shriftlarini ishlab chiqishdi. Bunday shriftlarning asosiy afzallik tomoni – standart printerlarda chop etish imkoniyati Post Script texno-

logiyasidan ko'ra tezroq amalga oshadi, narxi qimmat maxsus printerlarni talab qilmaydi, True Type shrifti ko'proq ishlatiladi. Texnologiya WYSIWYG («Nimani ko'rsang, o'shani olasan») tamoyilini qo'llab quvvatlaydi – u hujjatning chop etilganligi bilan ekrandagi ko'rinishni to'liq mosligini ta'minlaydi. Lekin, bunday chop etish sifati past bo'lib qolishi mumkin, ayniqsa, katta masshtabda yoki juda kichik harflar bo'lganda.

Bu shriftlarni Post Script printerlarida chop etish zarurati tug'ilsa, formatni almashtirish va tekislash bajariladi.

Chiqarish qurilmasining kalibrovkasi.

Kalibrovka deb tasvirning rang taqsimlanishi va rang o'zlashtirishi parametrlarini ma'lum bir standartga moslashishiga aytiladi. Boshqacha aytganda, Masalan, 1 monitor yashil deb hisoblaydigan rangni ko'rsatish mumkin. (ROG255BO), lekin, boshqa monitor yashil rangni biroz boshqa tusda ko'rsatishi mumkin, undagi yorituvchi tarkibi boshqa sifatga ega. Agar monitordagi tasvirning o'lchami va rang kalibrovkasining ko'rinishi bizga yetarli bo'lsa. Ammo, rang taqsimlanishiga bu yetarli emas. Rang taqsimlanishida rang kichik chetlanish, tuslar aralashmasi deb asosiy bo'lib qolishi mumkin: tomoshabinlar fikriga ko'ra ko'p ranglar, ayniqsa, fotografiyalarda qat'iy aniqlangan tusga ega bo'lishi kerak. Chetlanish monitor konstruksiyasiga bog'liq.

Chop etuvchi qurilma kalibrovkasining qiyinchiligi shundaki, rang bo'yoqlari chop etish natijasiga katta ta'sir ko'rsatadi, chunki bo'yoqdonning bo'shash darajasiga bog'liq ravishda bo'yoq o'z tusini o'zgartirishi mumkin. Tipografiyada har bir yangi tirajni chiqarish oldidan uning o'lchami bilan solishtirib tekshirishga to'g'ri keladi, nashriyot o'zining shtatida dizayner texnologida ega.

Xuddi oldingi holatdagidek, 1 ishlab chiqaruvchi – chop etuvchi qurilmalarning o'xshashligiga va bo'yoqlarning yuqori sifatiga hamda tekshiruvchining ko'ziga ishonishiga to'g'ri keladi.

Skanerni oddiygina kalibrovka qilish, buning uchun tuslarning maxsus tekshiruvchi jadvallaridan foydalanish mumkin. Skaner qilingandan so'ng hosil bo'lgan ranglarni aniqlash uchun maxsus dasturi ishga tushiriladi. Bu jadvallarni quyosh nuridan va boshqa ta'sirlardan saqlash lozim, aks holda kalibrovka ahamiyatsiz bo'lib

qoladi. Skanerning kalibrovkasi yuqori qiymatli bo'lsa ham natija aniqligini ma'lum bir darajada qabul qilish mumkin, yana dizaynerning ko'zi asosiy sifat o'lchovi hisoblanadi.

Rastrli formatlar.

Rastrli formatlar elementar nuqtalardan tashkil topadi (piksellardan). BMP rastr formatli grafik faylda tasvir nuqtalarining koordinatalari va ularning har biri uchun rang qiymati saqlanadi. Kichik imkoniyatli fayl bo'lganda ham uning hajmi judaham katta bo'ladi. Ma'lumki, rastrli tasvirning bunday kodlashtirish usuli ko'p hollarda olib boriladi. Zarari: masalan, rasmning bir tondagi sohasi, bir xil rangdagi koordinatalar to'plami uni to'liqroq ko'rinishda yozish mumkin bo'lgan vaqtda mos keladi.

GIF formatidagi fayllarda yaqin joylashgan bir xil rangdagi nuqtalar gorizontal chiziqda guruhlanadi. Bu grafik faylning o'lchamini ma'lum darajada qisqartirishga imkon beradi. Lekin, GIF formatida hali fotografiyani muvaffaqiyatli siqishga imkon bermaydi. Chunki ko'pgina turli-tuman tuslarning gorizontal guruhlarga birlashishi samarali emas. Internet rivojlanishi bilan bu muammo ancha sezilib qoldi, chunki, tarmoq bo'ylab ko'p miqdorda rastr tasvirlar va ayniqsa, suratlar uzatiladi.

Shuning uchun tarmoq bo'ylab rastr tasvirlarni trasportirovkasi uchun JPEG formati yaratildi. Bu formatda piksellar o'lchami kattalashuvchi tasvirni siqish algoritmi amalga oshiriladi. Shunday qilib, piksellarning soni kamayadi va mos ravishda grafik faylning hajmi qisqaradi. Ko'zga uncha ko'rinmaydigan ma'lumot yo'qotiladi. Shuning uchun bunday siqishda rasm sifati amalda zarar ko'rmaydi. JPEG formatining kamchiliklaridan biri shuki, rasmni kattalashtirsak, sifati sezilarli darajada yomonlashadi – tasvir yacheykalari har xil bo'lib ketadi. Hozirgi kunda Internetda ko'p suratlar xuddi shu formatda beriladi.

Albatta, grafik fayl o'lchami uning formatiga va tasvir harakteriga katta bog'liq bo'ladi: bir xil tasvirli yirik fragmentga ega rasm GIF fotosurat JPEGga samarali qisqaradi - qoida bo'yicha GIFdan ko'ra JPEG formatida sifat pasayishi ko'proq ko'rinadi.

PNG formati boshqalarga ko'ra yangi hisoblanadi, hamda Internet uchun maxsus ishlab chiqarilgan. Bu format bitta muhim

xususiyatga ega – unda tasvirning tiniqlik niqobini saqlash mumkin. Hamma zamonaviy brauzerlar foydalanuvchilari qo'shimcha modullarni o'rnatish zaruratisiz bunday formatdagi fayllarni ko'rishi mumkin.

Bundan tashqari yana grafik fayllarning turli xil formatlari mavjud, qoida bo'yicha ma'lum bir grafik muharrirlar uchun mo'ljallangan. Masalan, Microsoft Gif Animator GIF formati rastr tasvirlarining animatsiyasini beruvchi boshqa turlarida foydalanadi: faylga bir nechta kadrlar joylashtiriladi, keyin ketma-ket namoyish qilinadi. Bunday fayllarning o'lchami qoida bo'yicha katta bo'ladi. Shuning uchun RGB rangli modeliga mo'ljallangan yuqorida sanab o'tilgan formatlardan farqli ravishda TIFF formati RGB modelidek CMYKni ham ta'minlaydi va poligrafiyada qo'llaniluvchi universal rastr format hisoblanadi. TIFF formatidagi fayl tasvirning asosiy parametrlarini to'liqroq saqlaydi. TIFF formati fayldagi tasvirning sifatini yo'qotmasdan siqishga imkon beradi va siqishning turli xil algoritmlarini qo'llasa ham bu holdagi fayl o'lchamini JPEG fayli bilan solishtirib bo'lmaydi.

Vektor formatlar.

Grafikaning vektor ko'rinishi haqida tushunchaga ega bo'lish uchun oddiy misol ko'rib o'tamiz. Faraz qilamizki, rasmda yoy bo'lsin. Rastr formatda yoy va fonning har bir nuqtalarining koordinatalari va ranglari saqlanadi. Lekin, yoy formasini yozish uchun atigi 4 ta geometrik parametrlar kerak: radius, markaz va yoyning boshlang'ich va oxirgi nuqtalarining koordinatalari, ya'ni bizga yoyning 4 ta nuqtasi kerak, 100 tasi emas.

Vektorli grafikada tasvir elementlari xuddi parametrlar to'plami bilan geometrik obyektlar ko'riladi. Biz hech qanday ma'lumotlarni yo'qotmasdan katta natija olamiz. Rasm manipulyasiyasi ham soddalashadi: masalan, yoy radiusini o'zgartirish orqali tasvir masshtabini ancha oson olish mumkin. Egri chiziqlarni matematik ifodalash uchun odatda murakkab egri chiziq konturlari yordamida ko'rsatish mumkin bo'lgan Beze egri chizig'i qo'llaniladi. Vektor grafikdagi shakllarning rangli parametrlari zalivkaning murakkab turlarini ham oson va juda aniq tasvirlash imkonini beruvchi formulalar yordamida ham beriladi.

Vektor tasvirning kamchiligi shundan iboratki, xuddi rastrli rasmdagidek ular originalni haqiqiy qilib ko'rsata olmaydi. Shuning uchun vektor grafika rasmning elementlariga aniqlikni talab qilganda, rastrlida esa agar fotosuratning tabiiyligi kerak bo'lgan hollarda ishlatiladi. Turli ilovalar o'ziga xos standartlari bilan o'zaro almashuvchi keng qo'llaniladigan vektor format deb EPSni hisoblash mumkin, lekin, amaliyotda ko'pincha vektor grafikni saqlash uchun keng tarqalgan grafik muharrirlarning sharhi, ya'ni Corel DRAW yoki Adobe Illustrator vektor muharrirlarining ishchi formatlari CDR yoki AI ishlatish qulay bo'ladi. Odatda tipografiya ular bilan ishlashni xohlaydi, muharrirning va uning versiyasini aniq tanlovi esa dizaynerlik guruhi boshqaruvining shaxsiy xohishiga bog'liq. Vektor grafika, albatta, kompyuter animatsiyasida qat'iy tatbiqini topdi. Oddiy film yaratish uchun masshtab o'zgarishini va shakl harakatlanish trayektoriyasini hisoblash kerak, keyin esa kadrlarning soni aniqlanadi. Bizga filmning hamma kadrlarini saqlashga va aloqa tarmoqlari orqali tarqatishga to'g'ri kelmaydi. Birinchi kadrni, traektoriya formulasini, so'nggi kadrning masshtab koeffitsiyentini va kadrni ko'rsatish tezligini yuborish yetarli. Bu fayl hajmida va tarmoq bo'ylab yuklanish tezligida yutuq beradi. Hozir kompyuterlar yetarli darajada tez ishlovchi va protsessorlarning hisoblovchi quvvati aloqa kanalining uzatish qobiliyatining imkoniyatini ma'lum darajada quvib o'tadi, Internet foydalanuvchilarning web-sahifa bezashga talabi borgan sari o'smoqda. Endi ularga oddiy saytlar qiziq emas, ular tez yuklanuvchi jonli dinamik grafikalarni ko'rishni xohlashadi. Bu yerda dizaynerga web-sahifalarga multfilmlarni joylashtirishga imkon beruvchi maxsus Flash texnologiyasidan foydalanishga ehtiyoj tug'iladi.

Dizaynerdan faqatgina Adobe Flash dasturidan to'g'ri foydalanish talab qilinadi.

Vektorda animatsiya grafikasi SWF formatida saqlanadi (bu format vektor film namoyishi uchun brauzerlar tomonidan ishlatiladi).

Keng tarqalgan grafik muharrirlarining sharhi.

Grafika bilan ishlovchi dasturlarning turli-tumanligini ko'rganda yosh dizayner o'zini yo'qotib qo'yishi mumkin. Kerakli

masalalarni yechish uchun ideal mos keluvchi, hamda maksimal qulay va sizga foydali ishni ta'minlovchi dastur qanday qilib tanlanadi? Biz bilamizki, dasturni to'g'ri tanlash dizaynerning oldida turgan masala atrofida aniqlanadi: chop etish uchun tasvir tayyorlash kerakmi yoki biz shriftlarni chizamizmi, yoki web-sahifa yaratmoqchimizmi yoki hammasini birgalikda qilmoqchimizmi, qolgani endi o'zimizning xohishimiz bo'yicha bo'ladi.

Microsoft Paint. Minimum imkoniyatga ega sodda grafik muharrir bo'lib, Windows operatsion tizimi bilan o'rnatiladi, jiddiy masalalarni yechishda qo'llanilmasa ham har holda eslab o'tishimizga loyiqdir. Unda web-sahifa uchun rasm tayyorlanadi va GIF ga o'tkazilganda ajoyib rasm hosil bo'ladi. Shuning uchun sodda grafik muharrirlardan o'z o'rnida foydalanish kerak.

Microsoft Photo Editor – bu muharrir asosan fotografiyalar bilan ishlash uchun mo'ljallangan. Ko'pincha Microsoft Office paketi bilan joylashtiriladi, shuning uchun ham keng tarqalgan.

Microsoft Image Composer – Microsoft firmasining grafikani qayta ishlovchi bir muncha rivojlangan dasturlaridan biri. Ishlab chiqaruvchilar fikriga ko'ra Internet uchun grafika tayyorlashda Adobe Photoshop muharririga raqobatdosh deb hisoblanadi. Microsoft mahsulotining asosiy xususiyati shundaki, uning interfeysi sodda va qulay. Bu muharrir diskda kam joy egallaydi va juda tez o'rnatiladi. Lekin, poligrafiyada foydalanadigan tasvirlar uchun asqotmaydi. Microsoft Gif Animator animatsiya grafika muharriri va web-sahifa yaratish uchun mo'ljallangan dastur Microsoft Front Page bilan birgalikda qo'yiladi. Bu dasturlarning o'ziga xos jihati, agar oldin web-saytning sahifalarida soatlab o'tirilgan bo'lsa, hozir esa ular sanoqli daqiqalarda qilinadi.

Adobe Photoshop – eng mashhur va rastr grafikada eng zamonaviy professional muharrirdir. Uning sohasi – bu skaner qilingan fotosuratlarning tayyor tasvirlarini qayta ishlashdir. So'nggi versiyalarida Web-grafika bilan ishlovchi komponentlar qo'shilgan. U Adobe firmasining boshqa dasturlari bilan birgalikda eng talabchan so'rovlarni qoniqtira oladigan dizaynerlik dasturlarining integrallasngan paketini tashkil qila oladi.

Macromedia, Adobe Flash – vektor animatsion grafikaning muharriri. Internetga joylashtiriladigan kichik o'lchamli animatsion filmlarni tayyorlash imkonini beradi.

CorelDRAW – professional grafik muharrirlari ichida e'tirof etilgan yetakchidir. Agar siz vektor muharrirlari bilan jiddiy shug'ullanmoqchi bo'lsangiz CorelDRAWni tanlashni tavsiya etamiz. Mutaxassislik mavqeingizni ko'taradi. Muharrir rastr va vektor grafikani yaratish va qayta ishlash, Web-dizayn, verstka, rangni taqsimlash, yangi shriftlarni ishlab chiqish, shtrix-kodlarni kiritish masalalarini a'lo darajada bajaradi. Poligrafiya uchun eng yaxshi dastur hisoblanadi. Qo'shimcha dasturlar, u bilan birga tarqatiladigan, animatsion vektor grafikasi bilan ishlashni ta'minlaydi. Bu muharrir ko'p sahifali hujjatlarni qo'llab quvvatlaydi. Ammo, yuqori quvvatlilik natijasi kompyuterning resurslariga talabchanlik hisoblanadi.

Shunday qilib, hozirgi vaqtda eng yaxshi grafik muharrirlar tarkibiga quyidagilarni kiritish mumkin [20]:

- Vektor grafikaning muharriri – CorelDRAW.
- Animatsion vektor grafikaning muharriri – Macromedia, Adobe Flash.
- Web-sahifa muharriri – Microsoft FrontPage.
- Rastr grafika muharriri – Adobe Photoshop.

Yuqorida sanab o'tilganlarning har biri kompyuter grafikasining biror bir sohasida yetakchi o'rinlarni egallaydi. Lekin, bu firmalarning har biri raqobatchilar bilan kurashish quroli sifatida grafik muharrirlarni ham chiqardi. Bu yerda tarafdorlar qandaydir mahsulotning oldingilaridan ham foydalanishlari hisobga olingan. Barcha firmalar rastr, vektor, animatsion grafika dizaynerlik muharrirlarining va web-sahifalarni ishlab chiquvchi vositalarining to'liq jamlanmasini taklif qilishi mumkin. Microsoft interfeysning qulaylik darajasi bo'yicha yetakchi hisoblanadi. Shuni ishonch bilan aytish mumkinki, web-sayt yaratish uchun FrontPage dasturi eng yaxshi vosita bo'ladi. Uning soddaligi, qulayligi, kirishimliligi, ofis ilovalari bilan moslashuvchanligi shu bilan qatiylashmoqdaki, ayni vaqtda Microsoft Internet Explorer eng keng tarqalgan brauzer mavqeini egallab bormoqda.

CorelDRAW muharririning interfeysi barcha uchun sodda va tushunarli bo'lib, ingliz tilida yo'l yo'riqlar va to'liq ma'lumot tizimi berib o'tilgan. CorelDRAW muharririning murakkabligi uning imkoniyatlari ko'pligidandir.

Adobe mahsuloti tushunarli yo'l yo'riqlarning deyarli yo'qligi va to'liq bo'lmagan ma'lumotnomasi bilan ajralib turadi, lokal versiyasi esa, afsuski, odatda kechikib paydo bo'ladi. Bundan tashqari, Internetda ruslashtirish uchun dasturlar to'plamini topasiz, ularning yaratuvchilarini qonuniy harakatlarini chetga sursak ham, sifatini yaxshiligi qoldiriladi. Bu firmaning mahsulotlarida, masalan CorelDRAW foydalanuvchilariga mumkin bo'lgan ba'zi bir zo'r effektlar yo'q, lekin, hisoblovchi resurslar kichik hajmni talab qiladi. Shuni ta'kidlash kerakki, Page Maker mashhur tizim verstkasi xuddi shu firmada ishlab chiqariladi.

Adobe firmasi oddiy deb nomlash mumkin bo'lgan muharrirlarni ishlab chiqaradi, ingliz tilida yo'l-yo'riqlar va tushunarli ma'lumotnoma berib o'tilgan. Adobe Flash animatsion grafikasi muharriri raqobatchilaridan kuchliroqdir. Uning murakkabligi imkoniyatlari ko'pligidan kelib chiqqan.

Grafik muharrirlarda ishlashning umumiy tamoyillari.

Ixtiyoriy grafik muharrirda ishlash odatda quyidagi ketma-ketlikda bajariladi:

1. Yangi faylni yaratish yoki mavjudini ochish.
2. Obyekt yoki sohani ajratish. Bu ajratilgan soha uchun ma'lum bir vositalarni yoki effektlarni qo'llash uchun qilinadi.
3. Effektni tashkil qilish yoki grafik panelda uni belgilovchi sichqoncha yordamida vosita aktivlash va shu vosita bilan ishlash.
4. Tasvir oynasining bo'sh joyida sichqoncha yordamida obyektdan belgilashni olib tashlash.
5. Faylni saqlash.

Ba'zida grafik muharrirlar, ayniqsa, CorelDRAW uzoq «o'ylaydi», bunday holatda sabr qilish va uning ishini to'xtatishga urinmaslik, ayniqsa, agar sizga hech narsa bilan band emasdek va diskka hech narsa yozmayotgandek tuyulsa ham kompyuterni qaytadan ishga tushirish kerak emas.

Kitobda dasturning vositalariga, buyruqlariga va boshqa elementlariga, hamda sichqoncha va klaviatura bilan ishlashga doir ba'zi maxsus terminlar ko'p uchraydi. Keyinchalik bunday savollarga to'htalmaslik uchun ko'p foydalanadigan tushunchalar bilan tanishib chiqamiz:

• **Sichqonchani bosing.** Bu bilan biz sichqonchaning chap tugmasini bir marta sekin bosishni tushunamiz. Sichqonchani ikki marta bosish – bu ketma-ket tez ikki marta bosish kerak.

• **Obyektlarni o'tkazish.** O'tkazish quyidagilar bilan ifodalanadi: biror bir obyekt sichqonchaning chap tugmasi bosilib belgilanadi, tugmani qo'yib yubormasdan sichqonchani kerakli holatga joylashtiriladi va shundan so'ng tugma qo'yib yuboriladi. Obyekt yangi holatda joylashgan bo'ladi.

• **Obyektlar guruhini ajratish va ramkali ajratish.** Obyekt guruhlarini ajratish uchun ko'p dasturlarda Shift klavishasini bosgan holda ketma-ket obyektlar ustida sinqonchani bosib ketish orqali bajariladi.

• **Atributlar paneli, parametrlar palitrasi** – hujjat oynasining yuqori qismida joylashgan faol asboblarning parametrlaridan tashkil topgan panel.

• **Qator ko'rinishi.** Ochilgan hujjat oynasining quyi qismida joylashgan va bajariladigan buyruqlarning parametrlari va hujjat haqidagi ma'lumotdan iborat.

• **Kontekst menyu.** Obyekt ustida sichqonchaning o'ng tugmasi bosilganda paydo bo'lib, u ayni paytda unga tegishli bo'lgan buyruqlar to'plamidan iborat.

• **Qaynoq klavishalar va klaviaturadagi qisqartirishlar.** Asboblarga va buyruqlarga murojaatni tezlashtiruvchi vositadir. Asbobning qaynoq klavishasi odatda uning nomi oldida ko'rsatilgan bo'ladi. Unda foydalanish uchun asboblar palitrasini faollashtirish va ko'rsatilgan klavishani bosish kerak. Qaynoq klavisha bo'yicha menyu buyruqlarini chaqirish uchun Alt klavishasini bosish va qo'yib yubormaslik talab qilinadi; buyruqlarning qaynoq klavishasi uning nomi tagiga chizilgan chiziq orqali ifodalanadi. Bundan tashqari, buyruqlarga tez murojaat qilish uchun ularga klavishalar

mosligi–menyuda buyruq nomining o‘ng tomonida ko‘rsatilgan klaviaturadagi qisqartirishlar beriladi.

• **Suzib chiquvchi yordam.** Sichqonchani asbobga yoki buyruqqa olib borganimizda paydo bo‘luvchi qator. U asbobning nomini yoki u bajaradigan funksiyalarni ko‘rsatadi.

• **OK, Cancel, Apply, Yes, No, Back, Next va boshqa tugmalar** muloqat oynalarida paydo bo‘lib, barcha dasturlarda ularning vazifasi bir xil.

Nazorat savollari:

1. Kompyuter grafikasida rang modellarining qanday turlarini bilasiz va ularning qo‘llanilish sohalarini ifodalang?
2. Palitra nima va undan qay maqsadda foydalanish mumkin?
3. Palitralarning qanday turlari mavjud va ularga izoh bering?
4. Kompyuter grafikasida shriftlarning o‘rni va ahamiyati, ularning funksiyalari?
5. Grafik muharrirlar qanday turlarga ajratiladi va ularning imkoniyatlari?
6. Grafik muharrirlarda ishlashning umumiy tamoyillari nimada?
7. Rang modellarinig turlariga ta’rif bering?
8. Poligragiya sanoatida eng ko’p qo’llaniladigan rang modellarini ayting?

Tayanch iboralar: Rang modellari, palitra, rang chuqurligi, shrift, kalibrovka, grafik muharrir.

1.2. Grafik dizayn

Grafik dizaynga kirish.

Bu bo‘limda biz dizayn haqida to‘xtalamiz. Dizaynda chiza olish qobiliyati yoki kombinatsiyalar tashkillashtirish emas – buning uchun mahorat va did yetarlidir. Birorta grafik paketlarda ishlash tavsifiga to‘htalib o‘tmaymiz. Shunday holatlarni kuzatish mumkinki, ko‘pchilik Photoshop yoki CorelDRAW dasturini o‘rganib o‘zlarini dizayner deb atashadi. Dizaynga fan sifatida qarab

o‘rganamiz. Bu bo‘limni o‘qib siz dizayner bo‘lib qo‘lmaysiz. Lekin, biz sizni dizaynning asosiy mohiyatini tushunib yetishingizga umid qilamiz: agar nimadir yaxshi qilinsa, u obyektiv sabablarga ko‘ra yaxshi qilingan, va yana ish yaratuvchi nima tufayli bunga erishdi – o‘zining qobiliyati evazigami yoki nazariy bilimlari bunga sababmi degan savollar tug‘iladi.

Biz ta’lim oluvchini shuni tushunishini xohlaymizki, badiiy talantga ega bo‘lmasdan dizayner bo‘lish mumkin. Lekin, buning uchun «nima yaxshi yoki nima yomon» qoidasini tushunish kerak.

Biz o‘lcham, shakl, rang teksturasi, ko‘chirish va shrift kabi tushunchalar haqida gaplashamiz. Kompozitsiya savollariga ham to‘htalib o‘tamiz. Tanlov va, ayniqsa, tanlovni bekor qilish nimaga tayanishini shakllantirib ko‘ramiz. Sizga ma’lumki, ko‘pincha variantlar kam bo‘lmaydi, ammo, ulardan bittasigina omadli. Biz yosh dizaynerlarning asosiy xatolari va ko‘pgina tajribali mutahassislarning xatoliklarini yoritishga harakat qilamiz.

Lekin, bu dizaynga oddiygina bir qarash xolos. Agar sizga bu qiziq tuyulsa, dizaynerning eng asosiy «qanday qilib chiroyli qilinadi?» degan savolga javob beruvchi adabiyotlardan [5, 10, 16, 19, 22, 24] hamda internet manbalaridan o’qishingiz mumkin.

O‘lcham.

O‘lcham nima, bu bizga qandaydir darajada tushunarli, biz bu haqda maktab davridagi geometriya darsidan nimalarnidir bilamiz. Lekin, inson qabul qilishi bo‘yicha o‘lcham aniq matematik ifoda tushunchasi hisoblanmaydi.

Masalan biz evkaliptaning balandligini 100 metrga yaqin deb olsak, u bizga kamlik qiladi. Lekin, buni biz 30 qavatli uy balanligiga teng desak, uning qanday daraxtligini tasavvur qilamiz.

Shunday qilib, o‘lcham – nisbiy tushunchadir. Biz 20 sm, 3 m, 5 km deb emas, balki «Miniatyur», «o‘rta», «katta», «juda katta», «ulkan» deb qabul qilamiz. Hammasi inson sezgisiga asoslangan, inson idroki esa juda moslashuvchan. Katta va kichik tushunchalar broshkada miniatyura ko‘rganimizda boshqacha, ulkan suratni ko‘rganimizda boshqacha tasavvurga ega bo‘lamiz.

Ish tarkibida aniq bir obyekt o‘lchamini tanlaganimizda, butun kompozitsiyada nima asosiy g‘oyani berishi haqida o‘ylashimiz

kerak (umuman, zamonaviy dizaynning asosiy masalasi – bu foydalanuvchiga ma'lumotni yoki emotsiyani maksimal effektiv qilib yetkazishdir) shuning uchun, uyali telefon reklamasida uyali telefonni alohida ajratish kerak deb aytamiz.

Eslatma. Kompozitsiyadagi obyekt o'lchami bilan bo'lganiga emas, balki aksinchasiga ham e'tibor qaratish mumkin: yirik detallar bilan kuchli qarama-qarshilikda bo'lmagan detalga tomoshabin e'tiborini qaratish mumkin. Bu holda aynan shu narsa asosiy ma'lumotni beradi deb, boshqalari esa fon uchun qabul qilinadi. Bu effekt 1.3-rasmda ko'rsatilgan raqamlar bilan 1 ta raqamga o'quvchi e'tiborini qaratdik.

1.3-rasm. Kompozitsiyada obyektni faqat katta hajmi bilan emas balki aksincha.

Shu sababli bitta kompozitsiyani yaxshi ishlovchi obyekt boshqa kompozitsiyada o'lchovlarni kelishtirmasdan to'g'ridan-to'g'ri o'tmasligi mumkin. Ushbu ishni bajarishda faqat ko'zga emas balki shakl, tekstura va ranglarning o'lchovga ta'sir etishini bilish kerak.

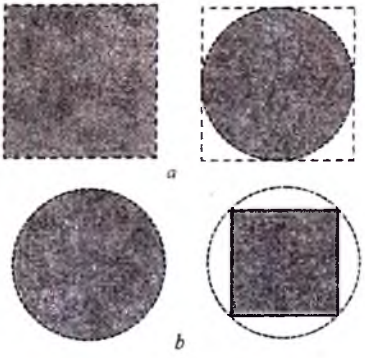
Shakl va o'lcham.

O'lcham idroki obyekt shakliga bog'liq. Bu, ayniqsa, yorug'lik interferensiyasini inson ko'zi bilan idrok qilishiga bog'liq. Ama-

liyotda asosan murakkab fraktal qurilishiga ega obyektlar o'lchamning nuqtai nazaridan baholashda qiyinchilikka uchramoqda. Albatta, agar ular nisbatan fraktallar bilan va ularning detallari boshqa obyektlar kompozitsiyasining gabaritlari bilan taqqoslanadigan bo'lsa, u holda bunday muammo kelib chiqmaydi. Lekin, ko'pincha murakkab shaklning kichik detallari obyektning o'lchamiga ta'sir qismi sifatida umuman qabul qilmaydi.

1.4-rasm. Bu figuradagi nurlar obyektning chegarasi sifatida qabul qilinmaydi.

Bu misol bizni shakl o'lchamini idrok qilish aloqasini belgilovchi kalitli tushunchaga olib keladi. Ba'zi adabiyotlarda quyuqroq shaklni aylana deb atashadi. Lekin, bizningcha bu to'g'ri emas.



1.5-rasm. Aylana va kvadrat obyektning joylashishi.
a – markada; b - tangada.

Biz shaklning ixchamligini tasavvurimizdagi chegaraning to‘ldirilmagan joylari uning sohasiga aloqasini aniqlardik. Misolda tushuntiramiz. Masalan, 2 ta shakl, aylana va kvadratni 2 ta tarqatuvchi tamg‘a va tangaga joylashtirish kerak.

Rasmda ko‘rganimizdek, marka (tamg‘a) uchun kvadrat, tanga uchun aylana ixchamroq figura hisoblanadi.

Bunday idrok qilishdan shuni esda tutish lozimki, o‘lchamni idrok qilish aynan ixchamlik orqali aniqlanadi. Quyuqroq shakl doim kattaroq ko‘rinadi. 1.5-rasmdagi markaga tegishlisidan bu juda yaxshi ko‘rinib turibdi.

Amaliyotda bu xususiyatdan quyidagi holatlarda foydalaniladi. Birorta masala mavjud deb olamiz: kompaniya logotipini «AGGW» nomi bilan vizitkada foydalanish. Agar siz kartochkada bu elementning muhimligini ko‘rsatib, hammaning e’tiborini unga qaratmoqchi bo‘lsangiz, unda logotipning ixcham shaklida to‘htalganingiz ma’qul (1.6-rasm. a). Agarda logotipga katta e’tibor qaratilmasa va uni fon tarzida qo‘llamoqchi bo‘lsangiz, u holda 1.6-rasm, b dan foydalanilgan ma’qul. Vizitkaning elementlarini bir xil o‘lchamligiga ahamiyat bering.

a *b*

1.6-rasm. Logotip formasi: a-ixcham; b-ixcham emas.

Tekstura va o‘lcham.

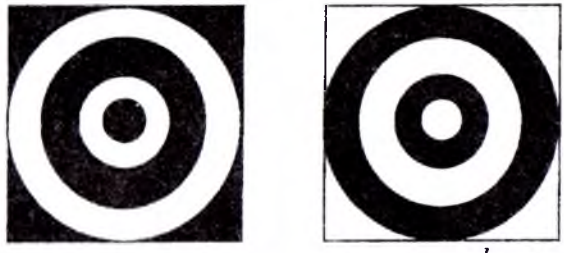
Testura qo‘llanilishi bizga yangi effektlar yaratish, shakl qiyofasini murakkablashtirish, unga ma’no berish imkoniyatlarini beradi. Lekin, tekstura obyekt o‘lchamining idrokiga ta’sir qilishi mumkin, buni esda tutish lozim.

1.7-rasmda umuman bir xil geometrik o‘lchamga ega 2 ta kvadrat tasvirlangan. Lekin, kesimlari ko‘ndalang ketgan kvadrat gorizontalnikidan ko‘ra «og‘irroq», yirikroq ko‘rinadi. Kesimlari gorizontal ketgan kvadrat esa balandroq ko‘rinadi.

1.7-rasm. Yorug‘ shakllar kattaroq ko‘rinadi: yorug‘ chiziqlarning yo‘nalishi illiyuziya yaratish qobiliyatiga ega.

Bu effekt avvaldan ma’lumdir. Eski jurnallarda to‘ladan kelgan ayollarga yo‘l-yo‘l ko‘ylaklarni kiyish tavsiya qilingan, ular qomatni biroz rostlaydi.

Amaliyotda bu narsa ko‘p qo‘llaniladi. Kerakli hollarda obyektga katta «turg‘unlik» berish uchun yorqin ifodalangan gorizontal yo‘nalishdagi tekstura ishlatiladi. Tekstura bilan yana bir qiziq effekt bog‘liq. 1.8-rasmdagi *a*-tasvir bizga uzoqlashayotgandek, *b*-tasvir yaqinlashayotgandek. Bu effekt oq rangning fonida bo‘lgan barcha predmetlarni «egish» xususiyatiga asoslangan. Buni radial gradientlar bilan ishlayotganda bilishimiz va qo‘llay olishimiz kerak, bu asosida ko‘pgina tryuklarni bajarish mumkin bo‘ladi.



a *b*

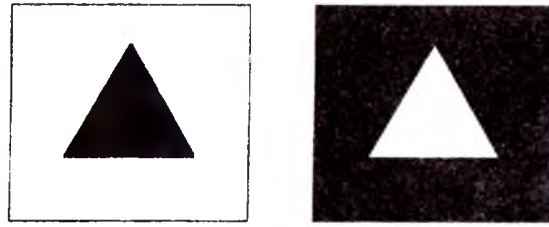
1.8-rasm. Oq va qora radial chiziqlar ketma-ketligiga asoslangan effektlar:
a-uzoqlashish, b-yaqinlashish.

Shunday qilib, tekstura obyekt tasvirida ijobiy ham, salbiy ham rol o‘ynashi mumkin (noto‘g‘ri qo‘llanilishda). Bunga katta e’tibor bering.

Rang va o'lcham.

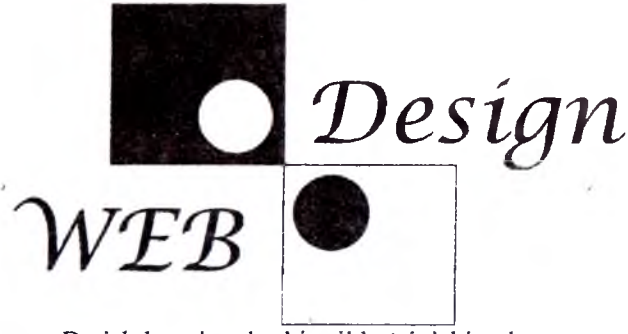
Bo'limning o'lchamga bag'ishlangan eng murakkab qismi. Uning idrokiga bog'liq tushunchalar. Odatda biz boshqa obyektlar orasida ajralib turganlarni katta deb hisoblaymiz, (ya'ni ko'rish qobiliyatiga asoslangan holda) ammo buni isbotlovchi obyektiv sabablar yo'q.

Demak, birinchi va asosiy xususiyat: yorqin fondagidan to'q obyektlardan ko'ra yorqin obyektlar to'q fonda yaxshi ajralib turadi (1.9-rasm).



1.9-rasm. To'q fonda to'q obyektlardan ko'ra yorqin obyektlar yaxshi ajralib turadi.

Bunga, albatta, katta e'tibor berishingiz kerak, agar yorqinligi bo'yicha katta farq qiluvchi biror bir obyektlarning simmetriyasini ko'rsatmoqchi bo'lsangiz, masalan, logotipda. Ish sifatsiz qilingandek tuyilmasligi uchun to'q elementni kattalashtirishga to'g'ri kelib qolishi mumkin.



1.10-rasm. Bu ishda aylanalar bir xil ko'rinishi uchun qora aylana diametrini 2% ga oshirdik.

1.10-rasmda qora aylananing radiusini 2% ga oshirdik, bo'lmasa logotipimiz biz o'ylagandek chiqmay qolishi mumkin.

Yorqin qizil bilan to'q ko'k ranglardan foydalanilganda xuddi shunday holatga duch kelamiz. Buni e'tiborga olish va ishda foydalanish lozim.

Rang.

Biz rangli dunyoda yashaymiz. Inson har kuni uydan chiqmasdan turib ham juda katta hajmda turli xil tuslarni ko'radi. Biz bunga ko'nikib qolganmiz va bu ranglar bizga, o'zimizni tutishimizga qanday ta'sir qilishiga e'tibor bermaymiz.

Biz ranglar asrida yashaymiz. Bizning hayollarimiz, emotsiyalarimiz o'z rangiga ega. Qora fikrlar, yashil zerikish, to'q sariq kayfiyat kabi tushunchalarni eslashimiz kifoya. Jismni psixologik qabul qilish uning rangiga bog'liq.

Kundalik hayotda bu unchalik katta ahamiyatga ega emas. Lekin, hozir kompyuter grafikasida qo'llanilishi haqida gap bormoqda va bu yerda nafaqat ranglar farqini, balki tuslarni ham aniq sezishimiz kerak, chunki ularni qayerda, qay holatda ishlatsak, yaxshi chiqishini bilishimiz kerak. Tasvirda to'g'ri tanlangan ranglar yaxshi ta'sir qilishi mumkin bo'lganidek, aksi ham chiqishi mumkin. Rangni almashtirish quvonch, qiziqish, zerikish, qo'rquvni keltirib chiqarishi mumkin.

Tuslar juda ko'p bo'lgani bilan har bir shaxs psixologiyasidan kelib chiqib, sevimli tuslariga ega bo'ladi. O'rtacha shaxsning psixologik portretidan kelib chiqib ranglarni tanlashimiz kerak.

Diqqat.

Qo'shimcha qiyinchilik shundan kelib chiqadiki, rang obyektiv fizik o'lcham sifatida tabiatda mavjud emas. Rangni his qilish elektromagnit to'lqinlarining obyektiv faktorlarining ta'siri ostida shakllansa ham u subyektivdir. Bundan tashqari, rangni tasvirlash uchun turli davlatlar milliy-madaniy qadriyatlari mos har xil rang modellarini ishlatishadi. Kompyuter grafikasi bilan ishlovchi bunday ixtiyoriy professional rang tasvirini yaratishning turli xil usullaridan eng yaxshisini tanlab olishiga to'g'ri keladi.

Shakldan farqli ravishda rang subyektiv tushuncha bo‘lsa ham dizayner amaliyotda qo‘llashi kerak bo‘lgan universal qonunlar mavjud.

Avval rang qanday tashkil qilinganligini bilib olamiz. Umuman olganda bu savol jiddiy, lekin, biz fiziologik tomonlariga to‘htalmasdan soddaroq qilib tushuntiramiz.

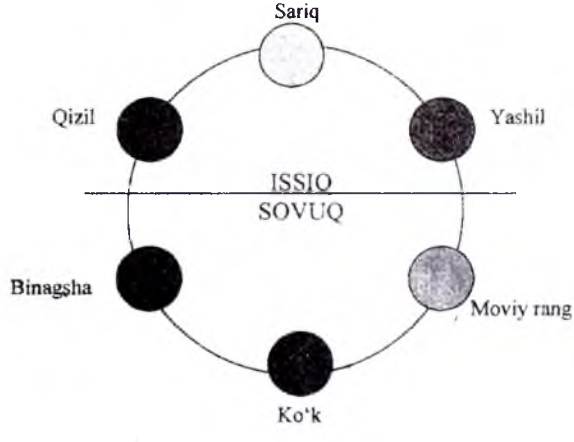
Avval rangni tashkil qiluvchilariga ajratamiz. Mavjud bo‘lgan rang modellaridan HSV modelini ko‘ramiz.

HSV tizimini 3 ta tarkibiy qismlarga ajratamiz;

- Tus - bu rang haqida bevosita ma’lumot. Buni tushuntirish qiyin, shuning uchun o‘z intuitsiyangizga ishoning.
- To‘yinganlik - bilamizki ranglar yaxshi to‘yingan yoki uncha yaxshi to‘yinmagan bo‘ladi. Xo‘jalik tilida to‘yingan rangni yorqin rang deb atashadi.
- Yorqinlik - yorqin ranglarni bir ko‘rishda aniqlash mumkin.

Bitta rangga nafaqat har xil kishilar, balki butun xalqlar ham turlicha munosabatda bo‘ladi. Masalan, oq rang yevropa xalqlarida tozalik, poklikni bildirsa, ba’zi sharq mamlakatlarida bu rang aza rangi hisoblanadi.

Asosiy ranglarning harakteristikasini berib ko‘tamiz.



1.11-rasm. Issiq va sovuq ranglarga ajratuvchi rangli aylana.

1.11-rasmda ranglar sovuq va issiq ranglarga bo‘lingan. Ma’lum darajada bu shartli bo‘linish: rang yuqori va pastki yarimsharlarning chegarasiga qanchalik yaqin bo‘lsa, uning sovuq yoki issiq rangga kirish aniqligi shunchalik kam bo‘ladi.

Qizil.

Qizil rang ko‘pchilikda olovga to‘g‘ri birikmasini uyg‘otadi. Shuning uchun uning ta’siri har xil bo‘lishi mumkin - issiqlikdan qo‘rquvgacha sezish mumkin. U ayniqsa, pulsni tezlashtiradi va ko‘z qorachig‘ini kengaytiradi, lekin, o‘zining yorqinligi tufayli ayniqsa, ko‘p miqdorda bo‘lsa, tez charchatadi (masalan, o‘tirgan xonaning rangi shunday bo‘lsa).

Toza qizil rang - bu vahima va hovliqish rangi, lekin, uning tuslari tinchlantiruvchidir. Qizil o‘ziga e’tiborni tortadi.

Bu rang ko‘pincha kompyuter grafikasida foydalaniladi, lekin, bo‘rttirib yubormaslik uchun undan ehtiyotkorlik bilan foydalanish zarur.

Qizil rang quyidagilarga to‘g‘ri keladi:

- Agressivlik, faollikni ajratish uchun. Masalan, eng zo‘r sport mashinalari, xususan, Ferrari asosan qizil rangda ishlab chiqariladi.
- Extirosni ifodalash uchun.
- Qattiqqo‘llikni ifodalash uchun.
- Boylik, dabdabani ifodalash uchun.

Sariq.

Ranglar ichida eng yorqin rangdir. U optimizmni, xursandchilik kayfiyatini ifodalash uchun ideal mos keladi. Turistik kompaniyalarning reklamasida shuning uchun ham eng ko‘p ishlatiluvchi rang hisoblanadi.

Bundan tashqari, sariq oltin rangidir. Shuning uchun u ko‘pchilikda boylik, dabdabani va omadni bildirib turadi.

To‘q sariq rang.

Issiq, pozitiv, yorqin va zamonaviy ko‘ngilni ko‘taruvchi rangdir. Zamonaviy dizaynda, ayniqsa, web-dizaynda ko‘p ishlatiluvchi ranglardan biri.

To‘q sariq rang quyidagi hollarda ishlatiladi:

➢ Zamonaviylik. O‘zini zamonaviy deb hisoblovchi ko‘p kompaniyalar o‘zining «firma» rangi deb oladi. Bu ayniqsa, uyali aloqa operatorlari orasida keng tarqalgan.

➢ Harakatchanlik.

➢ Optimizm. Yorqin plakat foni uchun to‘q sariq rang siyqasi chiqqan, lekin, ideal variant hisoblanadi.

Tajribadan kelib chiqib aytish mumkinki, to‘q sariq rang xavo rang bilan yaxshi mos tushadi.

Binafsha rang.

Rang aylanasida qizildan tortib ko‘k ranggacha ko‘rib chiqqanimizda ranglarni bir-biriga o‘tishini ko‘rishimiz mumkin, shular orasida ajoyib bo‘lgan rang binafsha rangini ko‘ramiz. Rang qabuliga bu rang juda og‘ir, chunki u tabiatda juda kam uchraydi. Binafsha rang ko‘k va moviy ranglar qatori sovuq ranglar guruhiga kiradi. Torlik xissiyotini, sohani chegaralanganligini yaratishga qodir, hamda tez charchatuvchi va faollikni pasaytiruvchi rangdir.

Binafsha rang «yer ranglariga yod rang hisoblanadi, unga qanaqadir mavhumlik mos keladi». Mistikada binafsha rangga alohida ahamiyat beriladi. Agar siz ko‘zbog‘lovchi (illyuzionist)ning kiyimlariga, buyumlariga e’tibor bergan bo‘lsangiz binafsha rangidan ko‘p foydalanilgan. Binafsha rang insonda mavhum qo‘rquvni keltirib chiqaradi.

Demak, binafsha rang quyidagilarga mos keladi:

➢ Mistik kayfiyatni yaratish uchun.

➢ Sirlilikni ko‘rsatish uchun.

Ko‘k.

Ko‘k rangli aylananing eng quyisida turuvchi sovuq rangga kiradi. U melanxolik kayfiyatni olib kelib, tinchlantiradi.

Bu rangni abadiy muzlik zonasidagi rangga kiritishimiz ham mumkin: u sovuqlik va tozalik xissini ideal beradi. Yotoqxona dizayni uchun juda mos keladi, xotirjamlikni beradi.

Toza ko‘k rangni to‘yinganligni va yorqinligini o‘zgartirish orqali biz katta tuslar gammasini olishimiz mumkin:

Ko‘k rang quyidagilarda ishlatiladi:

➢ Osoyishtalik.

➢ Tozalik. Agar e’tibor bergan bo‘lsangiz tozalovchi mahsulotlarning ko‘pchiligi ko‘k yoki moviy rangda bo‘ladi. Bu tasodifan

emas, aynan mana shu ranglar insonlarda tozalikka yetaklovchi ranglar ekanligini olimlar isbot qilishgan.

- Turg'unlik.

Yashil.

Ajoyib rang. U ham iliq, ham sovuq bo'lishi mumkin, biroq ko'pincha u to'qnashuvda bo'ladi, shuning uchun bu rangni iliq ham sovuq guruhga kiritish mumkin.

Yashil rang tinchlantiruvchi ta'sirga ega, qon bosimni tushiradi, qon yurishini normallashtiradi. Bu eng tabiiy va «tirik» rang. Dizayndagi uning asosiy roli ham aynan shundaki obyektning tabiat bilan bog'liqligini ko'rsatish. Agar logotiplarga e'tibor bersak, yashil rang asosan tabiat resurslarini qazib oladigan kompaniyalarda yoki ekologik tashkilotlarda uchratishimiz mumkin.

Agar siz zamonaviy blokbasterlarni tomosha qilganda e'tibor bergan bo'lsangiz biologik qurol, o'zga sayyora mavjudoti va boshqa biologik narsalar ko'pincha yashil (ayniqsa, yorqin yashil) rangda bo'ladi. Shuningdek bu insonlarning o'ylari bilan bog'liq.

Undan tashqari, yashil rang – jumboqli va mistik rang.

Yashil rang quyidagilarni ifodalashda yordam beradi:

- Hayotning barcha biologik ko'rinishini.
- Tabiat bilan aloqani.
- Jumboqlikni.

Moviy rang.

Moviy rang tinchlantiradi va sovutadi. Bu kabi effekt sovuq suv va muz bilan bog'lanadi. Ba'zida begonalik xissini uyg'otadi.

Asosiy ranglar: oq va qora.

Siz asosiy ranglarni ko'rib chiqqanimizda e'tibor bergan bo'lsangiz kerak, lekin, ikki alohida rangni aytib o'tmadik, qaysiki ranglar orasida hamma joyda bor bo'lgan, ayni vaqtda ular hech qayerda ko'rinmaydi, bu - oq va qora.

Oq xursandchilik rangi. Tiniq havo bilan birlashgan holda bu rang yengillik, ozodlik va vaznsizlik xissini uyg'otadi. Tomir urishini tezlatib, qorachiqni kengaytiradi. Fon yaratishda ko'pincha oq rangdan foydalanishadi.

Qora rang o'zining takrorlanmasligi bilan og'ir rang, u o'zida xafachilik, dard olib yuradi. Noqulaylik va charchash xissini uyg'otadi. Shunga qaramay, odamlar shu rangdagi kiyimlarni

tanlashadi. Ushbu vaziyatda u «mumtozlikka» qaratilgan, shu bilan birga alohida yo'nalish yaratadi. Shuningdek, bu rang u yoki bu darajada barcha rang bilan uyg'unlasha oladi. Qora ayniqsa, qizil rang bilan uyg'unlikka boy rang. Bizning udumlarga ko'ra uni motam rang deyiladi.

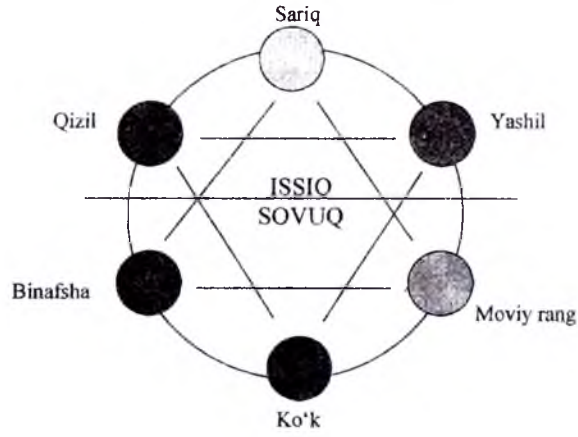
Oq rang o'zida hech qanday axborot olib yurmaydi, ammo qolgan ranglar bilan yaxshi uyg'unlashib, yanada ochroq tonlar hosil qiladi. Bu rangni poklik va aybsizlik deb hisoblashadi.

Ranglar uyg'unligi.

Ranglar uyqunligi – bu dizayndagi eng tortishuvli va bir necha ma'noli masalalardan biridir. Darhaqiqat bu yerda har kimning o'z didi haqidagi gapi kuchga ega. Shuning uchun universal qoidalar haqida gapirishga o'rin yo'q.

Biroq, ba'zida qonuniyatlarni topish mumkin. Shunday qilib, birinchi va oddiyroq tartib–yaqin rangdosh tanlash. Xuddi kiyim rangini tanlaganimizdek, web sayt uchun ham ranglarni tanlash mumkin.

To'g'risi, bunday yondashuv professional ish uchun yaramaydi. Uyg'unlasha oladigan ranglarni topishda ranglar jilosidan foydalanish qiziqarliroq.



1.12-rasm. Ranglarning o'zaro uyg'unligi.

➢ Yonma-yon ranglar yaxshi uyg'unlashadi, biroq, bunday tanlash varianti odatda zerikarli va jo'n hisoblanadi.

➢ Qarama-qarshi turgan ranglar bir-biriga nomutanosib hisoblanadi. Birgina holat – ko'k rang sariq rang bilan yaxshi ko'rinadi.

➢ Eng yaxshi tanlov – bir rang oralab 1.12-rasmdagidek, to'g'ri chiziqlar bilan tutashtirilgan. Ular o'zining ko'pchilik jilolaridan a'lo darajada uyg'unlashadi, eng asosiysi, ranglar yorqinligida katta farq bo'lmasligi.

Ahamiyatlisi shuki, oq va qora rang deyarli barcha boshqa ranglar bilan ideal uyg'unlashadi, ayniqsa, bir-biri bilan. Shuning uchun agar tanlash imkoni bo'lsa, bu ranglar bilan ish qilish har doim oson.

Lekin, baribir ranglar majmuini tanlashda asosiy mo'ljal did (kimdaki bu tug'ma bo'lsa, omadli) va tajriba (vaqt mobaynida ortirilgan) bo'lib qolishi kerak. Hech qanday ranglar nazariyasiga to'g'ri kelmaydigan ko'pgina yechimlar bor, lekin, o'z orasida sifatli va iste'dodli dizaynerlik ishi uchun namuna bo'la oladi.

Shakl.

Shakl bu har qanday obyektning muhim qismidir. Biz rang, tuzilish, hajm haqidagi axborotni o'tkazib yuborishimiz mumkin, lekin, shaklni unutmasligimiz kerak.

Har qanday dizaynerlik ishi aynan obyekt shaklini tanlashdan boshlanishi yoki ular bir nechta bo'lsa, moslashdan boshlanishi kerak.

«Shakl» so'ziga aniqlik kiritish yetarlicha mushkul vazifa. Agar biroz bo'rttirilsa, unda bu shakl barcha geometrik munosabatlar majmuidir. Umuman, shakllarning turlari ko'p, shuning uchun ularni faqatgina tuzilishi bo'yicha ajratish mumkin.

➢ Ko'pburchakli. Bunday figuralar to'g'ri chiziqlardan tuzilgan. Bular sirasiga biz uchun oddiy bo'lgan to'g'ri chiziq, uchburchak, kvadrat, yulduz kiradi, shuningdek, yanada murakkabroq figuralar ham shular sirasidandir (1.13-rasm, *a*-tasvir).

➢ Egri chiziqli. Bu silliq chiziqlarga asoslangan figuralardir (1.13-rasm, *b*-tasvir). Ularga aylana, oval, yarim aylana va boshqalar kiradi.

➢ Amorfli. Bu noaniq murakkab shakllar (1.13-rasm, *c*-tasvir). Amorf shakllar teksturalarga juda yaqin, shuning uchun vaqti-vaqti bilan ularni differensiallash qiyin emas.

1.13-rasm. Uchta tipdagi shakl:
a – ko'pburchakli, b – egri chiziqli, c – amorfli.

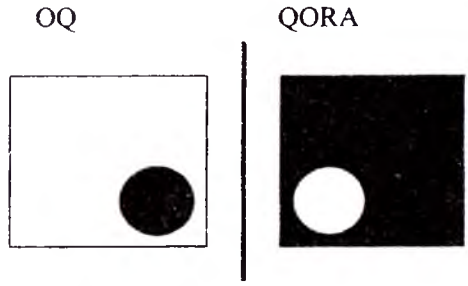
Shakl bilan ishlash – bu dizaynerlik ishining eng sodda va eng qiziq qismidir. Shakllarning mutanosibligi bo'yicha ko'pgina avlodlarning tajribasi mavjud, shuning uchun ko'pchilik shakl bilan ishlaganda (rang bilan ishlashdan farqli o'laroq) ichki tuyg'uga bo'ysunadi. Lekin, undan ham asosiy shakllar haqida bir necha fikr aytib o'tamiz.

Chiziq.

Chiziq – geometrik figuralarning eng soddasi. Nazariy jihatdan u faqat birgina o'lchamga ega – uzunlik. Biroq, biz amaliyotda shu narsaga duch kelamizki, uning qalinligi, rangi va turlariga ahamiyat berishimiz kerak (aytaylik, amaliyotda keng tarqalgan punktirli chiziq). Ba'zida chiziq va to'g'ri burchak orasidagi chegarani anglash qiyin.

Chiziqning ikkita asosiy bajaradigan vazifasi bor: boshqa obyektlarni birlashtirish va ajratish.

Ajratish – chiziqlardan foydalanishning mumtoz usulidir, qaysiki kitoblarni bezashda gullagan va gullab boryapti. Dizaynda ham u faol ishlatilib kelinmoqda.

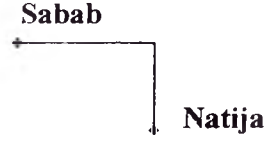


1.14-rasm. Ajratish uchun chiziqdan faydalanish.

Biroq, chizg'ichlarni (tipografik atamalardan ajratuvchi chiziqlar shunday ataladi) qo'llash oddiy va o'ta ravshan usul hisoblanadi. Agar sizga ishni sodda va tez bajarish kerak bo'lsa, u holda bu yomon variant emas, ammo ajratuvchi chiziqlar yordamida qandaydir original narsa olmoqchi bo'lsangiz, unda ko'proq mehnat qilish talab etiladi.

Aytaylik, matn bloklarini ajratish uchun fon rangidagi farqdan yoki bo'sh bloklardan foydalanish qiziqroq bo'ladi.

Chiziqlarni birlashtirish funksiyasi yanada qiziqroq. Aytish mumkinki, zamonaviy dizaynda keng tarqalgan va yorqinroq jihatlaridan biridir. Bu holatda chiziqlar axborot bloklari yoki grafik komponenti o'rtasida asosiy o'zaro harakat funksiyasini bajaradilar (1.15-rasm).



1.15-rasm. Chiziqning ulovchi funksiyasi.

Chiziq bu o'ziga xos, xohlagan vaqtda yordamga keluvchi «qutqaruvchi» chiziqdir. Eng oddiy usul bilan kompozitsiyaga kiritish mumkin bo'lgan ikkita jihat: birlik va jo'shqinlik.

Ishda chiziqlar o'ziga xos yo'naltiruvchi rolni o'ynashi kerak, unga qarab tomoshabin bir elementdan keyingisiga o'tadi. Shuning

uchun ulardan foydalanganda har doim faqatgina jozibadorlik haqida emas, balki mantiqiy asosni esdan chiqarmaslik lozim.

To'rtburchak.

To'rtburchak – darhaqiqat, dizayn uchun antiqa figuradir, ayniqsa, kompyuter dizaynida. Buning sababi oddiy. Ko'pchilik ko'rgazmalar: plakat, kitob sahifalari, deyarli barcha poligrafiya mahsulotlari, monitor ekranini aytmasa ham bo'ladi, to'g'ri burchak shakliga ega. Bunday holat ushbu figurani eng asosiy qilib beradi.

To'g'ri to'rtburchak asosida qilingan dizayn – bu eng oddiy va qulay yo'l hisoblanadi. Buning isboti sifatida qilingan ko'pgina web sahifalarni ko'rishingiz mumkin. To'gri to'rtburchak bilan ishlayotganda eng asosiysi, ularning nisbatini to'g'ri tanlash. Kvadrat ko'rinishdagi figuralar esa o'zining simmetrik ko'rinishidan kelib chiqib, dizayn sohasida «eskirgan» hisoblanadi. Oldindan to'g'ri to'rtburchak tomonlarining eng samarali nisbati ma'lum bu – oltin kesishuv. Bu nisbat 0.618 ga teng. To'g'ri, bu turli holatlarda yordam bermasligi mumkin, ammo, buni bilish kerak. Chunki oltin kesishuv asosida butun bir klassik arxitektura barpo qilingan.

Uchburchak.

Bu figura har tomonlama qulay, lekin, chiziqlar yoki to'g'ri to'rtburchak kabi ko'p qo'llanilmaydi. Buning asosiy sababi, uning boshqa figuralar bilan chiqishmasligidadir.

Uchburchak logotip asosida chiroyli ko'rinishi mumkin (1.16-rasm).

1.16-rasm. Uchburchak asosidagi logotip.

Mazkur ko‘rinishdagi logotip, ya’ni figuraning asosi bilan pastga qarashi, uning va kompaniyaning mustahkamligi to‘g‘risida aniq dalolatdir. Dunyodagi ko‘pgina firmalar uchburchakni o‘z ramzi sifatida tanlashgan. Shuningdek, bu figuraning ikkinchi o‘ziga xos xususiyati «yo‘nalish ko‘rsatish»dir. Yo‘nalish ko‘rsatish strelkasiga esa hammamiz o‘rganganmiz, bu element hammani o‘ziga jalb qiladi.

Masalan, 1.17-rasmda biz so‘zlarning ma’nosini idea.com saytiga jamladik.

1.17-rasm. Bu yerda uchburchak yo‘nalish strelkasi sifatida.

Uchburchak yordamida biz saytning sifatini ko‘rsatgan holda uni kompaniya nomiga yo‘naltirdik.

Doira.

Ko‘p sivilizatsiyalarda doira eng takomillashgan figura edi. Doira quyosh ramzi edi, doira ko‘rinishida ko‘p uylar, binolar qo‘rishgan. Hozirgi dizaynda bu figuradan keng qo‘llanilmayapti. 1.18-rasmda nashriyot saytining birinchi sahifasi ko‘rsatilgan. Bu sahifani yaratish jarayonida dasturchilar rubrikatorni doiraning chetiga joylaganligi tufayli sifatli deb topilgan.

1.18-rasm. Doira asosida qilingan web sahifa.

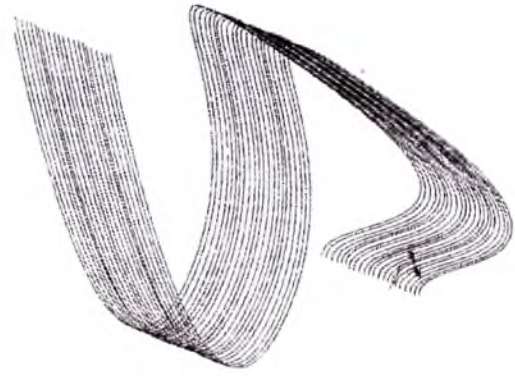
Siz o'zingiz ham doira ko'rinishda ko'p saytlar ko'rganingizni eslab ko'ring. Bu uslubdagi web sahifa qisqa vaqt ichida bajariladi, shuningdek, eng omadsiz hisoblanadi. Doira o'zining ustunligini faqat logotiplar dizaynida namoyon qilishi mumkin. Doiraning hammaga ma'lum ko'rinishi bu «O» harfidir.

1.19-rasm. Doira yordamida «O» harfini «imitatsiya»si – dizaynda bu shaklning omadsiz ishtiroki.

Bir so'z bilan aytganda agar sizning mahoratingiz yetarli bo'lmasa, bu figura bilan ishlamang.

Egri chiziqlar.

Egri chiziqlar har xil darajali egriliklarga ega to‘g‘ri chiziqlardir (matematikada bu termin ikkinchi darajali egri chiziq deyiladi) [21].



1.20-rasm. Ikkinchi darajali egri chiziqlar asosida qurilgan figura.

Zamonaviy kompyuter dizaynida bu figuralar juda ham keng qo‘llanilmoqda (ular asosan to‘g‘ri chiziqlar ishlatiladigan maqsadda foydalaniladi). Shuni aytish joizki, ularga berilish kerak emas: siz asr boshida mashhur bo‘lgan «modern» stiliga mos keluvchi ishni olishingiz mumkin bo‘ladi.

Shaklsizlik.

Albatta, shaklning yo‘qligi to‘g‘risida gapirib bo‘lmaydi, chunki u har doim mavjud. Shaklsizlik simmetriyaga qarshi chiqadi.

Shaklsizlikni quyidagilarni:

- mustaqillikni;
- ultra zamonaviylikni;
- zamonaviylikni;
- qarshilikni;
- nostandartlikni

ko‘rsatish jarayonida qo‘llash mumkin.

Shaklsizlikning ko‘rinishi shriftlar bilan mos tushadi. 1.21-rasmda shu ko‘rinishda ism yozilgan. Bu ko‘rinishda u originaldir va bu odamning xarakteridan ham dalolat beradi.

1.21-rasm. Konturlarning shaklsizligi yozuvni qiziqarli qilib ko'rsatadi.

Shaklsizlik bu kamchilik emas, balki haqiqiylikni izlashga keng imkoniyat ochib beradi.

Shriftli dizayn.

Shrift bilan ishlash qiziqarli va dizaynerlikning yengil qismi hisoblanadi. Agar siz shrift bilan ishlamoqchi bo'lsangiz shrift dizayni bo'yicha adabiyotga ega bo'lishingiz kerak bo'ladi.

Shriftlar uchta asosiy ko'rinishda bo'ladi:

- Kertma belgili (1.22-rasm, *a* tasvir);
- Kesilgan (1.22-rasm, *b* tasvir);
- Erkin stilda (1.22-rasm, *c* tasvir).

Design

a Design

b *Design*

c

1.22-rasm. Shriftning uch xil ko'rinishi:
a – kertma belgili; *b* – kesilgan (bo'lingan); *c* – erkin stilda.

Garnitura.

Shrift dizaynida birlik va kontrastlik tamoyili qo'llaniladi. Kertma belgili va kesilgan (bo'lingan) shriftlar o'zaro mosdir. Ish jarayonida bunga ishonch hosil qilish qiyinmas. Kesilgan (bo'lingan) shriftlar qo'l yozuvi bilan ham mos. Dekorativ shriftlar va kertma belgili shriftlar bir-biri bilan mos emas, shuning uchun bunday moslikka yo'l qo'ymang.

Kegl.

Yozuvlarning o‘lchamini tanlash qiyin hisoblanadi. Yozuv o‘lchamini tanlash jarayonida birinchi keladigan fikr bu yozuv qanchalik katta bo‘lsa, unga e’tibor ham ko‘p bo‘ladi. Lekin, aslida bunday emas.

O‘lchami KATTA matn inson uchun ma’lumotli funksiyasini yo‘qotadi

va DIZAYN elementi sifatida qabul qilinadi.

1.23-rasm. Matn o‘lchamining izohi.

Katta o‘lchamdagi harflar logotiplarga mos tushadi, qaysiki, dekorativ funksiyasi ma’lumotliga qaraganda muhimroq rol o‘ynaydi.

Kichik o‘lchamli shrift esa o‘zida xavfni tug‘diradi. Dizayn ishida bunday shriftni kamroq ishlatish kerak, chunki inson yozilgan narsani qiynalmasdan o‘qishi kerak. Bundan tashqari, alohida berilgan matn bloklari avtonom bo‘lishi kerak, ya’ni bo‘shliq bilan ajratilgan bo‘lishi shart.

Rang.

Rang dizayniga tegishli bo‘lgan qoidalar matnga ham tegishli. Faqat bitta narsa qo‘shimcha qilsak bo‘ladi: agar siz harflar qismini bir so‘zda ajratmoqchi bo‘lsangiz, yoki gapdagi bitta so‘zni rangdor yoki garnituradan foydalansangiz maqsadga muvofiq bo‘ladi.

Yuqoridagi harakatlar natijasi quyidagi rasmda keltirilgan.

PHOTOSHOP

PHOTOSHOP

PHOTOSHOP

1.24-rasm. Qismni ikkiga ajratish (garnitura va rang bilan) – yomon usul.

Kompozitsiya.

Moslik haqida gapirayotganda alohida e'tibor bilan eshitish kerak. Biz moslik mavzusini batafsil yoritmaymiz, chunki buning uchun maxsus kitoblar bor. Agar siz professional dizayner bo'lmoqchi bo'lsangiz ularni o'zingiz o'qib o'rganishingiz mumkin. Ammo moslik nazariyasi bilan umuman tanish bo'lmasangiz, bu qismni qunt bilan o'qib chiqishingizni tavsiya etamiz.

Moslik nazariyasi asosida biz tomondan aqliy, instinkt va refleks bilan qabul qilishimiz yotadi. Bir xil joylashgan obyektlarga barcha odamlarning reaksiyasi bir xil bo'ladi. Masalan, rasmdagi doira va to'g'ri to'rtburchakning joylashishi sizda noqulaylik tug'dirmoqda? Mualliflar bu *b*-variantligiga ishonch hosil qilmoqdalar.

a *b*

1.25-rasm. Moslikda obyektlarning turlicha joylashganligi qarama-qarshilikka olib keladi: *a*-doimiylik, *b*-noqulaylik.

Shu tariqa xohlagan moslik qarori keltirilishi kerak. Dizaynerlikda samarali natijaga erishish uchun nazariy bilimga ega bo'lish kerak.

Moslik orasida 2 tushuncha yotadi: birlik va kontrastlik.

Birlik.

Bu tushuncha o'z ichiga ko'p talablarni qamrab oladi, bular o'z navbatida aniq maqsadga erishtiradi. Birlik bu usul emas, balki dizaynerlikdagi talabni, maqsadni belgilab beradi, natijada yo logotip, yo web-sahifa bo'lsin, bularni bir butunlikda ko'rishadi. Amaliyotda esa bu narsa vositalardan keskin ravishda kamroq foydalanish kerakligini anglatadi. Ishni oddiy bo'lishidan qo'rqmang! Har doim sizning dizayningizdan osonroq dizayn bo'lmasligiga intiling. Agar bitta shrift garniturasini ishlatish mumkin bo'lsa –

bitta garniturani ishlatib, butun dizayni bitta figuraga joylashtirish imkoni bo'lsa, joylashtiring.

Ilova.

Birlik tamoyiliga asosan web-dizaynerlar amal qilmaydi. Siz GIF animatsiyalarini ko'p ko'rgan bo'lsangiz kerak, ular katta sarlavha va yorqin ranglar asosida yaratilgan. Ularning yaratuvchilari esa o'zlarini professional dizayner deb tanishtirishi va o'z xizmatlari uchun haq so'rashlari umuman noo'rinli hisoblanadi. Ularning xatolarini takrorlamasdan oddiy qilishga odatlaning.

Moslikda birlikka etishishining asosiy komponenti bu tenglikdir. Tenglik bu siz yaratgan narsa odamlarda «bu yerda nimadir etishmayapdi» degan hissiyotni tug'dirmaydi. Tenglik mosligi obyektni tenglik bilan joylashtirish evaziga emas, balki uning o'lchamlariga, rangiga, shakliga ham bog'liqdir. Masalan, web-sahifani bitta tomonidan sizda katta matn bloki bo'lsa, boshqa tomondan esa uni illyustratsiya yoki qoramtir fonni joylashtirish mumkin.

Tenglik ikkita asosiy ko'rinishda bo'ladi: formal va noformal.

Formal tenglik – optik markaz mosligi atrofini ko'zda tutadi.

1.26-rasm. Formal tenglikka simmetriya orqali erishiladi.

Formal tenglik – bir moslikda garmoniyaga erishishning oddiy yo'llaridan biri hisoblanadi. Lekin, u simmetriyani nazarda tutadi, simmetriyani esa zamonaviy dizaynerlikda ko'p qo'llash samarali

emas. Simmetriyani qo'llab yaxshi natijaga erishish lozim. Klassik san'at, xususan, arxetektura klassisizmi o'z negizi asosida formal tenglik tamoyiliga tayanadi, ya'ni: oltin kesim nisbatiga, shakllarning aniqligiga, absolyut simmetriyaga.

Formal tenglikning quyidagi sifat belgilarini ko'rsatib o'tish mumkin:

- konvervatizm;
- doimiylik;
- chidamlilik;
- qadr-qimmat.

Biroq, hozir tenglikning ikkinchi turi, ya'ni noformal ko'p qo'llanilmoqda. U formal kabi aniqliklarga ega emas, shuning uchun, asosan did va intuitsiyaga bog'liq. Noformal tenglikka ko'p yo'llar bilan erishiladi, ammo, uning mazmuni bir narsaga tayanadi: hamma narsa qilish mumkin, faqat simmetriyani qo'llash mumkin emas. Ya'ni web-sahifada yoki reklama doskasida siz katta o'lchamli figurani joylashtirgan bo'lsangiz va u boshqalarga nisbatan kattaroq bo'lsa, boshqa figuralarning rangini o'zgartirish hisobiga uning rangini kamaytirish mumkin. Shuningdek, teskarisi, katta matnning blokli o'lchamini kamaytirish lozim (1.27-rasm).

Dizayn haqida aniq qoidalar va aksiomalar asosida ilmiy yondashib o'tamiz. Berilgan ma'lumotlarni o'rganib siz darrov dizayner bo'lib qolmaysiz. Biz sizlarning asosiy narsani tushunib etishingizni xohlaymiz: nimadir yaxshi qilingan bo'lsa, bu narsa obyektiv sabablarga ko'ra shunday bo'lgan. Yaratuvchi bunga o'zining mahorati yoki nazariy bilimlari asosida erishganmi, yo'qmi bu alohida boshqa savol.

Biz sizlarga dizayner bo'lish uchun badiiy mahorat kerakmasligini tushuntirmoqchimiz. Ammo buning uchun aniq qoidalarni tushunish kerak, bu qoidalar asosida inson nima yaxshi va nima yomonligini tushunadi.

Biz o'lcham, shakl, rang, matn tuzilishi, joylashtirish va shrift kabi tushunchalar haqida gaplashamiz. Shuningdek, moslik haqida aytib o'tamiz. Tanlash va tanlamaslik nima asosida amalga oshishini tushunishga harakat qilamiz. Siz variantlar ko'p emasligiga ishonch hosil qilasiz. Biz yangi dizaynerlarning, shuningdek, o'zini haqiqiy dizayner deb hisoblaydiganlarning xatoliklarini o'rganib

chiqamiz. Ammo buning barchasi dizaynga engil qarash hisoblanadi. Agar bu sizni qiziqtirsa bu yo'nalish bo'yicha ko'p adabiyotlar mavjud.

Biz dizaynga aniq qoidalar va aksiomalar asosida ilmiy yondashamiz. Bu bobni o'rganib siz darhol dizayner bo'lib qolmaysiz. Biz sizlarga asosiy narsani: nimadir yaxshi qilingan bo'lsa, bu narsa obyektiv sabablarga ko'ra shunday bo'lgan, yaratuvchi bunga o'zining mahorati bilan birga nazariy bilimlari asosida erishganligini unutmaslik lozim.

1.27-rasm. Katta matnli blok kichik o'lchamga keltiriladi, lekin, butun figura ko'rinishida bo'ladi.

Noformal tenglikning birinchi qoidasini buzmaslik kerak - bu hamma narsani oddiylik bilan qilish degani. Variantlar ko'p biz u yer, bu yerga ozgina o'zgartirishlar qilib natijada zamonaviy dizaynga o'zgartiramiz.

Moslikning birlik uchun asosiysi barcha elementlarni bir maromda joylashtirishdir. Marom bu bo'shliq va obyektlarning mos holatda bo'lishi.

1.28-rasm. Logotip dizaynida maromning qo'llanilishi.

Ilova. Aniq maromga erishish jarayonida eng oddiy qarorni amalga oshirmang – bu simmetriya.

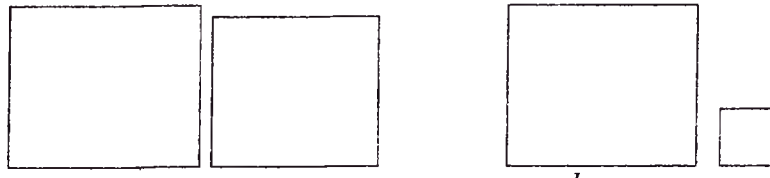
Kontrast.

Kontrast haqida biz oldin so'zlagan edik. Kontrast bu zamonaviy dizayn asosi. Professional mahorat kontrast asosida tashkil topgan.

Bir tomondan kontrast bu birlikning teskarisi. U moslikda butunlikka erishmaydi, undan farqli o'laroq, kontrastning maqsadi obyektlar o'rtasidagi farqni ko'rsatish. Ammo, bu ikkita tushunchani umuman bir-biriga to'g'ri kelmaydi, deb ta'kidlash noo'rin. Kontrast va birlik o'rtasidagi oltin kesishuvga intilish kerak.

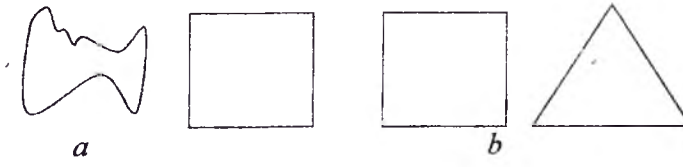
Endi esa moslikda kontrastning xilma-xilligini ko'rib chiqamiz:

• **O'lcham.** Kontrastga etishishning eng yaxshi yo'li – bir xil geometrik figuralar o'rtasida o'lchamlar farqini ko'rsatish. Ammo, shuni esda tutish kerakki, agar bu farq unchalik katta bo'lmasa, kuzatuvchi uni dizayner qarori sifatida emas, balki uning xatosi deb tushunishi mumkin.



1.29-rasm. Kontrast. *a*-figuralar orasidagi farq kamligi xato sifatida tushuniladi; *b* – faqat uni kattalashtirish.

• **Shakl.** Shakl mosligining birligini ko'rsatish uchun qo'llash samarali. Egri chiziqli va to'g'ri chiziqli figuralar o'rtasida katta va qo'pol kontrast mavjud, bunda esa ko'pchilik professionallar mahorat bilan ishini amalga oshira olmaydilar. Agar figuralar o'zaro o'xshash bo'lsa, (kvadrat, uchburchak) ular tinib oxiriga yetmaganligini ko'rsatadi.



1.30-rasm. Kontrast: *a* – egri va to'g'ri chiziqli figuralar kontrasti qo'pol; *b* – bir chipli figuralar farqi esa katta emas.

• **Rang.** To‘la kontrastga ranglar farqi bilan yetib bo‘lmaydi. Ammo, to‘ldirish sifatida, bu yaxshi variant.

• **Shrift.** Shriftlar o‘ziga xos xususiyatlari bilan aniq kontrast shaklini namoyon qiladi.

Shuning uchun, garnituralar o‘rtasidagi farq dizaynda katta rol o‘ynaydi. Bunda ehtiyot bo‘lish kerak.

Moslik kontrasti butunligi – bu barcha aytib o‘tilgan kontrastlar birikmasi. Bunda, albatta, meyorga amal qilib ish yuritish kerak. Masalan, elementga o‘zining rangi mos bo‘lsa, uni yana qandaydir rang bilan ajratish kerak emas.

Nazorat savollari:

1. Grafik dizayn nima va u bilan ishlovchi grafik paketlarni tavsiflang?
2. Grafik dizaynda o‘lchamning turlari va obyektlar o‘lchamini ifodalang?
3. Grafik dizaynda ranglar, ularning tavsifi, ranglar uyg‘unligini izohlang?
4. Qanday ranglar issiq turga va qandaylari sovuq turga tegishli?
5. Shakllar, ularning turlari va o‘ziga xos xususiyatlarini tavsiflang?
6. Kompozitsiya grafik dizaynda qanday ahamiyat kasb etadi?
7. Formal tenglik nima va u qanday sifat belgilarini o‘zida ifodalaydi?
8. Moslikda kontrastning xilma-xilligiga ta’rif bering?

Tayanch iboralar: Dizayn, o‘lcham, shakl, tekstura, rang, ranglar uyg‘unligi, shaklsizlik, garnitura, kegl, kompozitsiya, simmetriya, kontrast.

II bob. KOMPYUTER GRAFIKASI NAZARIYASI

Kompyuter monitoridagi tasvir (rasm) bilan bog'liq bo'lgan axborotni qayta ishlash uchta asosiy yo'nalishga ajratiladi: tasvirni o'rnatish yoki aniqlash, tasvirni qayta ishlash va kompyuter (mashina) grafikasi [4].

Tasvirni o'rganishning asosiy vazifasi bu, mavjud bo'lgan obrazni (tasvirni) formal, tushunarli (aniq) bo'lgan belgilar tiliga o'tkazish. Bu holda qaralayotgan tasvir abstrakt tasavvurga aylantiriladi, ya'ni sonlarga, maxsus belgilar yoki grafiklar to'plamiga o'tkaziladi. Buni quyidagicha yozish mumkin:

COMPUTER VISION:

- input- tasvir (rasm);
- output-belgi (matn) va uning tahlili.

Tasvirni qayta ishlashda kiruvchi va chiquvchi ma'lumotlar-tasvirlar. Masalan: tasvirdagi ayrim elementlarni olib tashlash (ovoz, rang, ...) yoki qo'shish, uning hajmini o'zgartirish va hokazo. Ya'ni uni quyidagicha yozish mumkin:

IMAGE PROCESSING

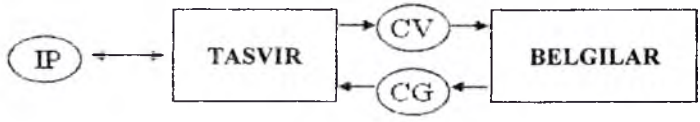
- input – tasvir (o'zgartirilmagan);
- output - tasvir (o'zgartirilgan).

Kompyuter (mashina) grafikasi dastlabki, ya'ni kiruvchi axborotni (noma'lum tabiatga ega) tasvir ko'rinishiga olib keladi. Masalan: ekspert ma'lumotlarni grafik, diagramma yoki boshqa shakllarga vizuallashtirish. Bundan tashqari, shakllarni almashtirish, harakatlantirish, virtual tasavvurga yaqinlashtirish. Kompyuter grafikasini quyidagicha tasvirlash mumkin:

COMPUTER GRAPHICS

- input – belgilar, belgili tavsif;
- output – tasvir.

Ularning o'rtasida keskin chegara yo'q va umumiy sxemada quyidagicha tasvirlash mumkin:



Tasvirni (dastlab matn, formula so‘ng oddiy rasm) shaxsiy kompyuter ekranida chiqarish kompyuter grafikasining rivojlanishida birinchi qadam bo‘ldi. Qisqa vaqt (50-yillardan boshlab) ichida kompyuter grafikasi tezkor rivojlandi va o‘zining e’tiborini ikki asosiy yo‘nalishga qaratdi: tasvirga yetarlicha tasavvur (reallik) va harakat (dinamika) berish, va ularni birlashtirish.

Tasvirni kompyuter ekraniga chiqarish va u bilan bog‘liq amallarni bajarish foydalanuvchidan ma’lum darajada geometrik bilimlarni talab qiladi. Geometrik tushunchalar, formulalar, faktlar, (birinchi navbatda ikki va uch o‘lchovga tegishli) kompyuter grafikasida o‘ziga xos maxsus o‘rinni egallaydi. Geometrik yondashish, tasavvur va fikrlar hisoblash texnikasining imkoniyatlarini doimo tezkor kengayishi bilan birgalikda kompyuter grafikasining jiddiy rivojlanishi yo‘lida va ko‘p sohalarda keng ishlatilishiga manba bo‘ldi. Ayrim hollarda oddiy, elementar geometrik metodikalar katta geometrik masalalarni yechish bosqichlarida rivojlanishni sezilarli darajada ta’minlaydi.

Ikki va uch o‘lchovli geometrik almashtirishlarni mashina grafikasida qo‘llanilishini [4, 6, 7, 8, 9, 13] adabiyotlari asosida ko‘rib chiqamiz.

2.1. Tekislik va fazodagi almashtirishlar

Tekislikdagi (ikki o‘lchovli) almashtirishlar.

Ikki o‘lchovli barcha narsalarni kompyuter grafikasida 2D (2-dimension) belgisi bilan ifodalash (kiritilgan) qabul qilingan [8].

Faraz qilamizki tekislikda to‘g‘ri chiziqli koordinatalar sistemasi kiritilgan (berilgan) bo‘lsin. Unda har qanday M nuqtaning koordinatasini aniqlash uchun ikki juft (x, y) sonlari olinadi.

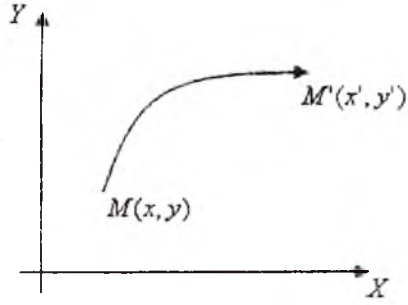
Ushbu tekislikda yana bitta to‘g‘ri chiziqli koordinatalar sistemasini kiritgan holda M nuqta uchun yangi mos juft (x', y') koordinatalarni hosil qilamiz.

Tekislikda bitta to‘g‘ri chiziqli koordinatalar sistemasidan boshqasiga o‘tish quyidagi tenglamalar orqali amalga oshiriladi:

$$\begin{cases} x' = \alpha x + \beta y + \lambda, \\ y' = \gamma x + \sigma y + \mu, \end{cases} \quad \begin{vmatrix} \alpha & \beta \\ \gamma & \sigma \end{vmatrix} \neq 0. \tag{1}$$

Bu yerda $\alpha, \beta, \gamma, \sigma, \lambda, \mu$ - ixtiyoriy sonlar.

Boshqa tomondan qaraganda, agar biz nuqta o‘zgarib koordinatalar sistemasi o‘zgarmas deb qabul qilsak, u holda (1) formulalar $M(x,y)$ nuqtani $M'(x',y')$ nuqtaga almashtirishini ifodalaydi (2.1-rasm).



2.1-rasm. Tekislikdagi (ikki o‘lchovli) almashtirish.

(1) formulalarni nuqtani almashtirishni ifodalaydi deb qabul qilamiz.

Almashtirish formulalaridagi koeffitsiyentlarning geometrik ma’nosini o‘rganish uchun berilgan koordinatalar sistemasini to‘g‘ri burchakli dekart koordinatalar sistemasi deb hisoblash qulay.

Ikki o‘lchovli almashtirishlarning xususiy hollarini ko‘ramiz.

1. Ko‘chish.

$M(x,y)$ nuqtani $M'(x',y')$ nuqtaga ko‘chish berilgan λ va μ ko‘chish konstantalari vektorining koordinatalariga qo‘shish orqali amalga oshiriladi

$$x' = x + \lambda,$$
$$y' = y + \mu.$$

2. Masshtablash. Cho'zish (siqish).

Koordinatalar o'qlari bo'yicha cho'zish (yoki siqish) ko'paytirish orqali ifodalanadi:

$$x' = \alpha x,$$
$$y' = \delta y,$$

$\alpha > 0, \delta > 0$ - masshtablash koeffitsiyentlari bo'lib mos ravishda X va Y bo'yicha cho'zish va siqishni bildiradi.

Agar $\alpha > 1, \delta > 1$ bo'lsa, koordinata o'qlari bo'yicha cho'zish va $\alpha < 1, \delta < 1$ bo'lsa, siqish ta'minlanadi.

Cho'zish (siqish) almashtirishlarini vektor-matritsa shaklida quyidagicha yozish mumkin:

$$(x', y') = (x, y)\begin{pmatrix} \alpha & 0 \\ 0 & \delta \end{pmatrix}$$

3. Burish.

Burish quyidagi formula orqali beriladi:

$$x' = x\cos\varphi - y\sin\varphi,$$
$$y' = x\sin\varphi + y\cos\varphi,$$

Bu yerda koordinatalar sistemasining boshlang'ich nuqtasi bo'ylab soat yo'nalishiga nisbatan teskari φ burchakka burish bajariladi.

Vektor-matritsa shaklida burishni quyidagicha yozish mumkin:

$$(x', y') = (x, y)\begin{pmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{pmatrix}$$

4. Akslantirish.

Tekislikda akslantirish koordinatalar sistemasining o'qlariga nisbatan bajariladi. Abtsissa o'qiga nisbatan akslantirish quyidagicha ifodalanadi:

$$x' = x,$$
$$y' = -y.$$

Vektor-matritsa shaklida esa:

$$(x', y') = (x, y)\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Ordinata o'qiga nisbatan akslantirish quyidagicha ifodalanadi:

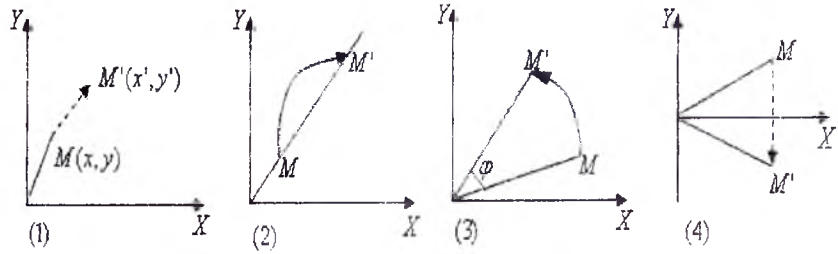
$$x' = -x;$$
$$y' = y.$$

Vektor-matritsa shaklida:

$$(x', y') = (x, y)\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Almashtirishlarni yuqoridagi berilgan to'rtta xususiy holatining berilishidan maqsad:

- har qaysi almashtirish oddiy va tushunarli geometrik ma'noga ega;
- ixtiyoriy almashtirishni ularni ketma-ket bajarish (superpozitsiyasi) orqali ifodalash mumkin.



2.2-rasm. Tekislikdagi (ikki o'lchovli) almashtirishlar.

Ammo, keyingi masalalarni ko'rish uchun to'rtta oddiy almashtirishlarni ham (ko'chirishni hisobga olgan holda) matritsa shaklida ifodalash kerak.

Nuqtaning bir jinsli koordinatalari.
Tekislikdagi almashtirishlarni vektor-matritsa shaklida ifodalash.

Faraz qilamiz, tekislikda $M(x,y)$ nuqta berilgan bo'lsin. Ixtiyoriy x_1, x_2, x_3 (bir vaqtda noldan farqli sonlar M nuqtaning bir jinsli koordinatalari deb ataladi, agarda:

$$\frac{x_1}{x_3} = x, \quad \frac{x_2}{x_3} = y.$$

Ya'ni ixtiyoriy $h \neq 0$ kupaytiruvchi uchun $-M(hx, hy, h)$.

Kompyuter grafikasi masalasini ishlash jarayonida ixtiyoriy $M(x,y)$ nuqtaning bir jinsli koordinatalari quyidagicha kiritiladi:

$M(x,y,1)$, ya'ni $h=1$.

Ko'rish mumkinki, (1) almashtirish formulalarni bir jinsli koordinatalarda quyidagicha ifodalash mumkin:

$$(x', y', 1) = (x, y, 1)\begin{pmatrix} \alpha & \gamma & 0 \\ \beta & \sigma & 0 \\ \lambda & \mu & 1 \end{pmatrix} \qquad (2)$$

Ikki o'lchovli almashtirishlarning xususiy hollari, ya'ni 1, 2, 3, 4 uchun mos matritsalarni yozib chiqamiz:

1. Ko'chish matritsasi (translation):

$$K = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \lambda & \mu & 1 \end{pmatrix}.$$

2. Masshtablash (cho'zish, siqish) matritsasi (dilatation):

$$M = \begin{pmatrix} \alpha & 0 & 0 \\ 0 & \delta & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

3. Burish matritsasi (rotation):

$$B = \begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

4. Akslantirish matritsasi (reflection):

$$A=\begin{pmatrix}1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & 1\end{pmatrix}.$$

Ixtiyoriy almashtirishlarning matritsasini yuqorida keltirilgan *K, M, B, A* matritsalarni ketma-ket (superpozitsiya qoidasi asosida) ko'paytirish orqali hosil qilish mumkin. Ular oddiy almashtirishlarning bajarilishiga qarab mos ravishda ko'paytiriladi.

Misol: *ABC* uchburchakni $A(x,y)$ uchiga nisbatan φ burchakka burish almashtirishining matritsasini quring.

1-qadam. $A(x,y)$ nuqtani koordinatalar sistemasi boshiga $O(0,0)$ nuqtaga, ya'ni $(-x,-y)$ vektoriga ko'chishi:

$$K_{-A}=\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ -x & -y & 1\end{pmatrix}.$$

2-qadam. Koordinatalar sistemasi boshiga nisbatan φ burchakka burish:

$$B_{\varphi}=\begin{pmatrix}\cos\varphi & \sin\varphi & 0\\ -\sin\varphi & \cos\varphi & 0\\ 0 & 0 & 1\end{pmatrix}$$

3-qadam. *A* nuqtani dastlabki holatiga qaytarish uchun (x,y) vektorga ko'chish:

$$K_{A}=\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ x & y & 1\end{pmatrix}$$

Keltirilgan tartibda almashtirish matritsalarini ketma-ket ko'paytiramiz:

$$B_A=(K_{-A}\times B_{\varphi})\times K_A.$$

Natijada matritsa ko'rinishidagi almashtirishni quyidagi ko'rinishda olamiz:

$$(x',y',1)=(x,y,1)\times\begin{pmatrix}\cos\varphi & \sin\varphi & 0\\ -\sin\varphi & \cos\varphi & 0\\ -x\cos\varphi+y\sin\varphi+x & -x\sin\varphi-y\cos\varphi+y & 1\end{pmatrix}.$$

E'tibor berilsa, barcha almashtirishlarning matritsalari determinantlari noldan farqli.

Fazodagi (uch o'lchovli) almashtirishlar

Fazodagi, ya'ni uch o'lchovli (3D, 3-dimension) almashtirishlarni ko'ramiz va ularni bir jinsli koordinatalarni kiritgan holda qaraymiz.

Ikki o'lchovli holdagidek, nuqtani fazoda aniqlovchi uchta koordinatasini (x,y,z) to'rtta bir jinsli koordinatalarga almashtiramiz $(x,y,z,1)$ yoki umumiy hol uchun $(hx,hy,hz,h),h\neq 0$. Bu yerda ham h-ko'paytiruvchi va $h=1$ deb olamiz .

Keltirilgan bir jinsli koordinatalar uch o'lchovli almashtirishlarni matritsalar orqali yozish imkonini beradi.

Ixtiyoriy almashtirish uch o'lchovli fazoda ko'chish, masshtablash (cho'zish, siqish), burish va akslantirishlarni superpozitsiyasi orqali aniqlanishi mumkin. Shuning uchun birinchi navbatda ushbu akslantirishlarning matritsalarini ko'ramiz.

Ma'lumki, qaralayotgan holatda matritsalarning o'lchovi to'rtga teng.

1. Ko'chish. Fazoda ko'chish ham ko'chish vektori bilan beriladi, ya'ni ko'chish vektorining λ,μ,ν koeffitsiyentlari bilan ifodalanadi:

$$K=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ \lambda & \mu & \nu & 1\end{pmatrix}$$

bu yerda (λ,μ,ν)– ko'chirish vektori .

2. Cho'zish (siqish). Fazoda ham masshtablash (cho'zish, siqish) masshtablash koeffitsiyentlari bilan ifodalanadi:

$$Ch = \begin{pmatrix} \alpha & 0 & 0 & 0 \\ 0 & \beta & 0 & 0 \\ 0 & 0 & \gamma & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

bu yerda $\alpha > 1$ $(1 > \alpha > 0)$ - abtsissa o'qi bo'ylab cho'zish (siqish) koeffitsiyenti,

$\beta > 1$ $(1 > \beta > 0)$ - ordinata o'qi bo'ylab cho'zish (siqish) koeffitsiyenti,

$\gamma > 1$ $(1 > \gamma > 0)$ - applikata o'qi bo'ylab cho'zish (siqish) koeffitsiyenti.

3. Burish. Fazoda burish koordinata sistemasining koordinatalariga nisbatan amalga oshiriladi va burish burchagi bilan beriladi.

Absissa o'qi bo'ylab φ burchakka burish matritsasi:

$$B_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi & 0 \\ 0 & -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Ordinata o'qi bo'ylab ψ burchakka burish matritsasi:

$$B_y = \begin{pmatrix} \cos\psi & 0 & -\sin\psi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\psi & 0 & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Applikata o'qi bo'ylab θ burchakka burish matritsasi:

$$B_z = \begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

4. Akslantirish. Fazoda akslantirish koordinata sistemasining tekisliklariga nisbatan amalga oshiriladi.

XY tekisligiga nisbatan akslantirish matritsasi:

$$A_{XY} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

YZ tekisligiga nisbatan akslantirish matritsasi:

$$A_{YZ} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

ZX tekisligiga nisbatan akslantirish matritsasi:

$$A_{ZX} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Bu yerda shuni aytish joizki, barcha matritsalarning determinantlari noldan farqli, geometrik almashtirishdan keyin geometrik obyekt o'lchamini yo'qotmaydi.

Fazodagi barcha almashtirishlarni keltirilgan oddiy to'rtta almashtirishlar ketma-ket bajarilishi (superpozisiya) orqali amalga oshirish mumkin.

Ixtiyoriy fazodagi almashtirishning matritsasi quyidagi ko'rinishga ega:

$$M = \begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 & 0 \\ \beta_1 & \beta_2 & \beta_3 & 0 \\ \gamma_1 & \gamma_2 & \gamma_3 & 0 \\ \lambda & \mu & \nu & 1 \end{pmatrix}.$$

Agar biror bir geometrik obyekt n-ta nuqtalardan iborat bo'lsa (ya'ni n-ta nuqta orqali berilgan bo'lsa), u holda almashtirish matritsasi M aniqlangandan so'ng, berilgan nuqtalarning $V_i(x_i, y_i, z_i), (i = \overline{1,n})$ matritsasini hosil qilamiz, so'ngra ko'paytirish amalini bajaramiz:

$$V' = V \times M = \begin{pmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ \dots & \dots & \dots & \dots \\ x_n & y_n & z_n & 1 \end{pmatrix} \times M.$$

Natijada geometrik almashtirishdan so‘ng gosil bo‘lgan obyektning nuqtalari matritsasini olamiz.

Platon jismlari (ko‘pyoqliklar).

Barcha yoqlari to‘g‘ri ko‘pburchaklar va barcha uchlariga tegishli burchaklar o‘zaro teng bo‘lgan qavariq ko‘pyoqliklar muntazam ko‘pyoqliklar (Platon jismlari) deb ataladi [4, 13].

Beshta muntazam ko‘pyoqliklar mavjud (Yevklid buni isbotlagan): to‘g‘ri tetraedr, geksaedr (kub), oktaedr, dodekaedr, ikosaedr. Ularning o‘ziga xos asosiy xususiyatlari:

Nomi	Yoqlari soni (Yo)	Qirralari soni (Q)	Uchlari soni (U)
Tetraedr	4	6	4
Geksaedr	6	12	8
Oktaedr	8	12	6
Dodekaedr	12	30	20
Ikosoedr	20	30	12

Yoqlar, Qirralar va Uchlar soni o‘zaro quyidagi Eyler formulasi bilan bog‘liq:

$$Yo+U=Q+2.$$

Ko‘pyoqliklarni qurish algoritmlarini ko‘ramiz. Buning uchun ularning uchlarini topish etarli.

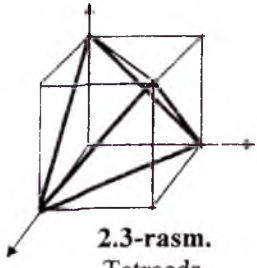
Geksaedrni (kub) qurish qiyinchilik tug‘dirmaydi. Masalan faraz qilamizki, kubning bitta uchi koordinata sistemasining boshida yotsin. Unga qo‘shni bo‘lgan uchlari esa koordinata sistemasining o‘qlarida bo‘lsin. Unda kubning yana uchta uchlari mos ravishda koordinata sistemisining tekisliklarida yotadi va so‘nggi uchi fazoda

joylashadi. Agarda kubning qirrasi d ga teng deb olsak, u holda uning koordinatalari quyidagicha topiladi:

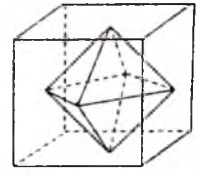
$V_1 = (0,0,0),$
$V_2 = (d,0,0),$
$V_3 = (0,d,0),$
$V_4 = (0,d,0),$
$V_5 = (d,d,0),$
$V_6 = (0,d,d),$
$V_7 = (d,0,d),$
$V_8 = (d,d,d),$

Tetraedrni qurish algoritmi: Tetraedrning qirralari kubning qarama-qarshi yoqlaridagi ayqashgan diagonallari bo'ladi.

Oktaedrni qurishda quyidagi xossadan foydalanamiz: oktaedrning uchlari kub yoqlarining markazlariga (og'irlik) mos keladi, ya`ni mos yoq uchlarining o'rta arifmetik qiymatlari.

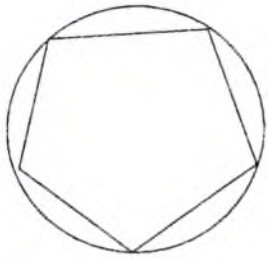


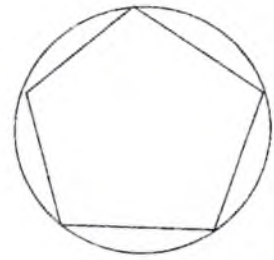
2.3-rasm.
Tetraedr.



2.4-rasm.
Oktaedr.

Ikosaedrni qurishni ko'ramiz. Z o'qida $z = \pm 0.5$ markazli, $r = 1$ radiusli va XY tekisligiga parallel ikkita aylana o'tkazamiz. Har bir aylanani beshta teng bo'lakka bo'lib, ularni rasmda ko'rsatilgan tartibga mos birlashtiramiz. Ikkita aylanalarning nuqtalarini ketma-ket birlashtiramiz va ikosaedrning yoqlarini tashkil qiluvchi o'nta muntazam uchburchakni hosil qilamiz. Qolgan yoqlari uchun $z = \pm\sqrt{5}/2$ nuqtalarini olamiz va mos aylanalarning nuqtalari bilan tutashtiramiz. Bunda yana o'nta yoq hosil bo'ladi.





Dodekaedrning uchlari ikosaedr yoqlarining og'irlik markazlari bo'ladi.

Nazorat savollari:

1. Axborotlarni qayta ishlashning asosiy yo'nalishlari hisoblangan tasvirni tanlash, qayta ishlash va kompyuter grafikasiga izoh bering?
2. Rastr, vektor va fraktal grafikaga izoh bering?
3. Rastr va vektor grafika dasturlari haqida ma'lumot bering?
4. Tekislikdagi almashtirishlar deganda nima tushuniladi?
5. Ikki o'lchovli almashtirishlarning xususiy hollarini ifodalang?
6. Nuqtaning bir jinsli koordinatalari nima va ular qanday vazifalarda qo'llaniladi?
7. Fazodagi (uch o'lchovli) almashtirishlar deganda nima tushuniladi?
8. Platon jismlari haqida umumiy ma'lumot bering?

Tayanch iboralar: kompyuter grafikasi, tasvirni tanlash, tasvirni qayta ishlash, rastr, vektor va fraktal grafika, ko'chish, burish, akslantirish, masshtablash.

2.2. Proeksiyalar. Ularning turlari

Umuman olganda, *proeksiya* (*proeksiyalash*) deb n o'lchovli koordinatalar sistemasida berilgan nuqta(lar)ni n dan kam bo'lgan o'lchovli koordinatalar sistemasidagi nuqta(lar)ga, geometrik

almashtirishlarga aytiladi. Xususiy holda, kompyuter grafikasida 3 o'lchovlidan 2 o'lchovliga proeksiyalashni ko'ramiz.

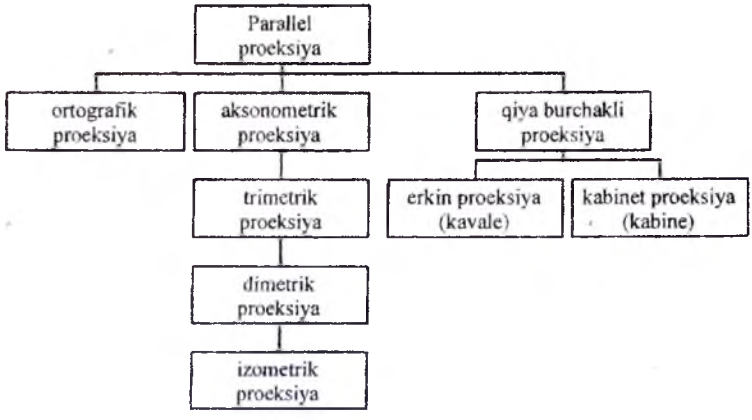
Uch o'lchovli *obyektning* proeksiyasi proeksiya nurlari orqali quriladi va ular *proektorlar* deyiladi, ular *proeksiya markazi* (nuqta)dan chiqib obyektning har bir nuqtasidan o'tadi va *proeksiya (tasvir) tekisligidan* kesib o'tib unda obyektning *proeksiyasini* yasaydilar.

Kompyuter grafikasida proeksiyalarni bir necha turlari ishlatiladi. Ulardan amaliyotda ko'p ishlatiladigani va asosiylari bu *parallel* va *markaziy* (*perspektiv*) proeksiyalar. Ular proeksiya markazi va proeksiya tekisligi orasidagi masofa orqali farqlanadi, ya'ni parallel proeksiyada ushbu masofa cheksiz.

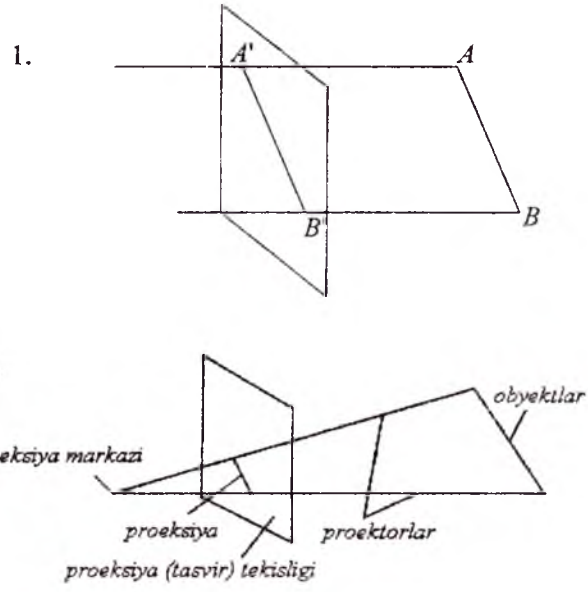
Aytish joizki, markaziy proeksiyalar vizual taassurotni vujudga keltiradi, ya'ni fotografiya sistemalari yoki odamning ko'rish sistemasi kabi bo'ladi va ma'lum darajada reallik darajasiga erishiladi. Bu effekt (taassurot) perspektiv qisqartirish deyiladi. Parallel proeksiyalarda tasvirning realligi kamroq, ammo obyektning ayrim haqiqiy o'lchovlari va paralelligi saqlanadi. Ular, asosan, muhandislik grafikasida ishlatiladi.

Har turdagi proeksiyalar o'z navbatida proeksiya tekisligi, proektorlar va koordinatalar sistemasining joylashishiga qarab bir necha turlarga bo'linadi.

1. Parallel proeksiyalar:



2. Markaziy (perspektiv) proeksiyalar:
Bir nuqtali, ikki nuqtali va uch nuqtali.



2.5-rasm. Parallel va markaziy proeksiyalar.

Proeksiyalashda bir jinsli koordinatalardan va to‘rtinchi tartibdagi geometrik almashtirish matritsalaridan foydalanish geometrik masalalarni yechishda ko‘pgina qulayliklarni beradi.

Yuqorida aytib o‘tilgan proeksiyalarni qarab chiqamiz.

Ortografik proeksiyalarda proeksiya tekisligi biror-bir koordinatalar tekisligi bilan ustma-ust yoki parallel bo‘ladi. deb olinadi, proektorlar esa unga perpendikulyar bo‘ladi, ya’ni koordinata o‘qlariga nisbatan parallel bo‘ladi.

Abtsissa X o‘qi bo‘ylab YZ tekisligiga proeksiyalash matritsasi quyidagicha:

$$M_X = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Agar proeksiya tekisligi koordinatalar tekisligiga parallel bo‘lsa, M_X matritsasini ko‘chish matritsasiga ko‘paytirish kerak:

$$M_x \times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & 0 & 0 & 1 \end{pmatrix}.$$

Shu kabi ordinata va applikata o‘qlari bo‘ylab proeksiyalarni quyidagicha yozish mumkin:

$$M_Y = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & q & 0 & 1 \end{pmatrix},$$

$$M_z = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & r & 1 \end{pmatrix}.$$

Bu yerda p, q, r – mos ravishda proeksiya tekisligidan koordinatalar sistemasining tekisliklarigacha bo‘lgan masofa.

Boshqacha aytganda: obyektning oldi, yon va ustidan ko‘rinishi.

Eslatma. Barcha matritsalarning determinantlari nolga teng.

Aksonometrik proeksiyalarda proeksiya tekisligi koordinata sistemasining tekisliklari bilan usta-ust tushmaydi va proektorlar proeksiya tekisligiga perpendikulyar bo‘ladi.

Proeksiya tekisligi va koordinatalar o‘qlari yo‘nalishiga qarab aksonometrik proeksiya uchta sinfga bo‘linadi :

- *Trimetriya*, ya’ni proeksiya tekisligining normal vektori koordinatalar o‘qlari bilan o‘zaro har xil burchaklarni tashkil qiladi,
- *Dimetriya*, ikkita burchaklari o‘zaro teng,
- *Izometriya*, barcha burchaklar o‘zaro teng.

Proeksiyalash matritsasi ordinat o‘qi bo‘ylab ψ burchakka, abtsissa o‘qi bo‘ylab φ burchakka burish, so‘ng applikata o‘qi bo‘ylab ortografik proeksiyalash orqali hosil qilinadi.

$$M=\begin{pmatrix}\cos\psi & 0 & -\sin\psi & 0\\ 0 & 1 & 0 & 0\\ \sin\psi & 0 & \cos\psi & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\times\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & \cos\varphi & \sin\varphi & 0\\ 0 & -\sin\varphi & \cos\varphi & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\times\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}\cos\psi & \sin\varphi\sin\psi & 0 & 0\\ 0 & \cos\varphi & 0 & 0\\ \sin\psi & -\sin\varphi\cos\varphi & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

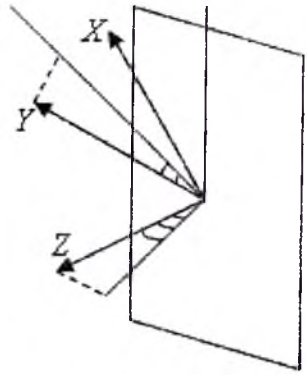
Bu yerda X,Y,Z o‘qlari bo‘ylab bazis vektori quyidagicha o‘zgaradi.

$$(1,0,0,1)\cdot M=(\cos\psi,\sin\varphi\sin\psi,0,1),$$
$$(0,1,0,1)\cdot M=(0,\cos\varphi,0,1),$$
$$(0.0,1,1)\cdot M=(\sin\psi,-\sin\varphi,\cos\psi,0,1).$$

Aytish joizki dimetriya holida:

$$\sin^2\psi=tg^2\varphi.$$

Izometriyada esa:

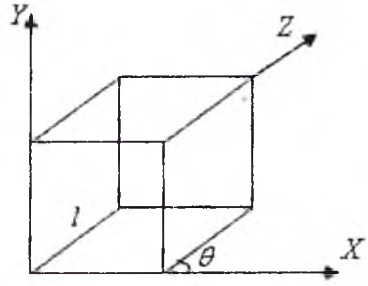


2.6-rasm. Oksonometrik proeksiya.

$$\sin^2\varphi=\frac{1}{3},\sin^2\psi=\frac{1}{2}.$$

Qiya burchakli proeksiyalashda proektorlar proeksiya tekisligiga perpendikulyar emas.

Proeksiyalash matritsasi bu holda quyidagi ko‘rinishga ega:



2.7-rasm. Kubning qiya burchakli proeksiyasi.

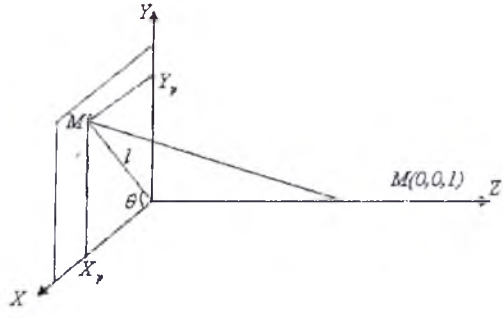
$$K = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \alpha & \beta & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Qiya burchakli proeksiyalar ikkita sinfga ajratiladi: Kavale (erkin) proeksiya va Kabinet proeksiyasi.

Kavale proeksiyasida: $\alpha = \beta = \cos\frac{\pi}{4}$.

Kabinet proeksiyasida: $\alpha = \beta = \frac{1}{2}\cos\frac{\pi}{4}$.

o‘qi bo‘ylab yo‘nalgan ortning (birlik vektor) qiya burchakli proeksiyasini ko‘ramiz.



2.8-rasm. Qiya burchakli proeksiya.

Bu biz ko‘rayotgan $M(0,0,1) \to M'(l\cos\theta, l\sin\theta, 0)$.

Umumiy holda $M(x,y,z)$ nuqtani qiya burchakli akslantirishni quyidagicha ifodalash mumkin:

$$X_p = X + Z(l\cos\theta), \quad Y_p = Y + Z(l\sin\theta).$$

Proeksiyalash matritsa-K ni quyidagicha yozish mumkin:

$$K = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ l\cos\theta & l\sin\theta & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Kavale : $l = 1, \theta = 45^0$,

Kabine: $l = \frac{1}{2}, \theta = 45^0$

Markaziy proeksiyalarni ko‘ramiz.

Faraz qilamizki, proeksiya markazi Z o‘qida yotib $C(0,0,c)$ nuqtada joylashgan bo‘lsin. Proeksiya tekisligi XY tekisligi bilan ustma-ust joylashgan bo‘lsin. Proeksiya markazidan proeksiya tekisligigacha masofa $d = c$ ga teng. Fazoda $M(x,y,z)$ nuqta olamiz va uni markaz bilan $C(0,0,c)$ tutashtiruvchi to‘g‘ri chiziq o‘tkazamiz. Ushbu to‘g‘ri chiziqning parametrik tenglamasini tuzamiz:

$x' = xt, y' = yt, z' = c + (z - c)t$. $z' = 0$ shartiga ko‘ra $t = \frac{1}{1 - \frac{z}{c}}$ topamiz va bundan foydalangan holda $M(x,y,z)$ proeksiyasining koordinatalarini topamiz :

$$x' = \frac{1}{1 - \frac{z}{c}} x,$$

Ushbu natijani proeksiya markazi $C(0,0,c)$ va $M(x,y,z)$ nuqtani tutashtiruvchi to‘g‘ri chiziq tenglamasidan ham olish mumkin, ya’ni

$$\frac{x' - x_1}{x_2 - x_1} = \frac{y' - y_1}{y_2 - y_1} = \frac{z' - z_1}{z_2 - z_1}, \qquad (x_1, y_1, z_1) = (0,0,c), \qquad (x_2, y_2, z_2) = (x, y, z).$$

Bir nuqtali markaziy proeksiyalashning matritsasi quyidagicha:

$$P=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&-\frac{1}{c}\\0&0&0&1\end{pmatrix}$$

tekshirib ko‘ramiz: $(x,y,z,1)\cdot P=\left(x,y,0,1-\frac{z}{c}\right)$.

Bir jinsli koordinatalarning xossalaridan foydalangan holda (ya’ni $\frac{1}{1-\frac{z}{c}}$ ga ko‘paytiramiz)

$$M'\left(\frac{x}{1-\frac{z}{c}},\frac{y}{1-\frac{z}{c}},0,1\right).$$

Ushbu akslantirishga mos almashtirish matritsasi quydagicha:

$$P_A=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&-\frac{1}{c}\\0&0&0&1\end{pmatrix},\qquad P_A\cdot\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&0&1\end{pmatrix}=P.$$

Umuman olganda, markaz nuqtasi uchta bo‘lishi mumkin va bu holda almashtirish matritsasi:

$$P_T=\begin{pmatrix}1&0&0&-\frac{1}{a}\\0&1&0&-\frac{1}{b}\\0&0&1&-\frac{1}{c}\\0&0&0&1\end{pmatrix}.$$

Mos ravishda x,y,z o‘qlariga parallel to‘g‘ri chiziqlar dastalari $(1,0,0,0)$, $(0,1,0,0)$, $(0,0,1,0)$ quyidagi markazli $\left(1,0,0,-\frac{1}{a}\right)$, $\left(0,1,0,-\frac{1}{b}\right)$ va $\left(0,0,1,-\frac{1}{c}\right)$ to‘g‘ri chiziqlar dastasiga o‘tadi.

Ularni quyidagicha ham tasvirlash mumkin, mos ravishda: $(-a,0,0,1)$, $(0,-b,0,1)$ va $(0,0,-c,1)$ bular bosh tutashish nuqtalarini aniqlaydi.

Nazorat savollari:

1. Proeksiyadan nima va qanday maqsadda foydalaniladi, uning ta'rifini bering?
2. Parallel va markaziy proeksiyalarni bir-biridan qanday farqlash mumkin?
3. Muhandislik grafikasida proeksiyaning qanday turidan foydalaniladi?
4. Parallel proeksiya qanday turlarga bo'linadi?
5. Ortografik proeksiyalarda proeksiya koordinata o'qlariga nisbatan qanday joylashadi?
6. Aksonometrik proeksiya qanday turlarga bo'linadi va uning koordinata o'qlariga nisbatan joylashishi?
7. Qiyaburchakli proeksiyalashda proektorlar proeksiya tekisligiga nisbatan qanday joylashadi?
8. Markaziy proeksiyalash haqida umumiy ma'lumot bering?

Tayanch iboralar: Proeksiya, proeksiya markazi, proeksiya tekisligi, parallel va markaziy proeksiya, ortografik, aksonometrik, qiya burchakli proeksiyalash.

2.3. Fazoviy shakllarni tasvirlash

Poligonal setkalar.

Kompyuter grafikasida fazodagi uch o'lchovli obyektlar sirtlarini tasvirlashning ikkita usuli keng tarqalgan: Poligonal setkalar va bikubik parametrik bo'laklar.

Poligonal setka bu fazoviy obyektni tasvirlovchi o'zaro bog'liq balandliklar, qirralar va yoqlar (ko'pburchaklar) to'plami. Nuqtalar (uchlar) qirralar bilan tutashtiriladi, ko'pburchaklar esa uchlar va qirralar bilan ifodalanadi [4, 13].

Poligonal setkalar qurishning 3 xil usuli mavjud:

1. Ko'pburchaklarni oshkora berish.

Har bir ko'pburchak uning uchlari koordinatalari bilan beriladi, ya'ni

$$P=((X_1,Y_1,Z_1),(X_2,Y_2,Z_2),\dots(X_n,Y_n,Z_n)).$$

Uchburchakni ifodalovchi (aniqlovchi) uchlar ketma-ket saqlanadi va qirralar bilan tutashtiriladi, shu jumladan oxirgi va birinchi uchlar ham.

Har bir alohida ko'pburchak uchun bu usul albatta qulay, hamma umumiy uchlarn koordinatalarini takroran saqlash evaziga poliganal setka xotirada ko'p joyni egallaydi.

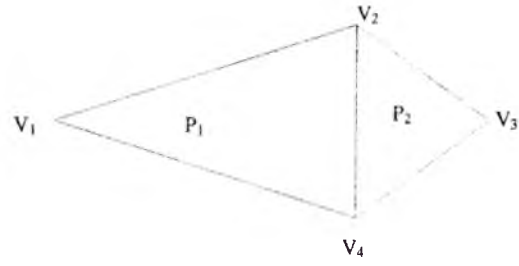
2. Ko'pburchaklarni uchlar ro'yxatidagi ko'rsatkichlari orqali ifadalash.

Bu holda poligonal setkaning har bir tuguni uchlar ro'yxatida bir marta saqlanadi:

$$V=((X_1,Y_1, Z_1), (X_2, Y_2, Z_2),...., (X_n,Y_n, Z_n)).$$

Ko'pburchak uchlar ro'yxatidagi (indeks) ko'rsatkichlari orqali beriladi. Ko'pburchakning har bir uchi bir marta saqlanadi va bu xotira hajmini tejashga olib keladi. Ammo, umumiy qirralar ikki martada chiziladi. Misol:

$$V = (V_1, V_2, V_3, V_4) = ((X_1, Y_1, Z_1), \ldots,(X_4, Y_4, Z_4)).$$



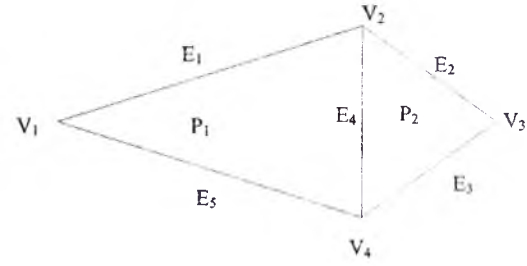
2.9- rasm. Poligonlarni tasvirlash.

3. Qirralarni oshkora berishi.

Bu holda ko'pburchak qirralar ro'yxatidagi ko'rsatkichlar to'plami orqali beriladi. Qirralar ro'yxatida har bir qirra bir marta uchraydi va har bir qirra ro'yxatda uchlari (ikkita) va mos ko'pburchaklar (1 yoki 2 ta) orqali ifodalanadi. Ya'ni har bir ko'pburchak quyidagicha, $R=(E_1, \ldots, E_n)$, va har bir qirra

quyidagicha $E=(V_1, V_2, R_1, R_2)$ ifodalanadi. Agar qirra bitta ko'pburchakka tegishli bo'lsa, u holda R_1 yoki R_2 bo'sh to'plam.

Qirralarni oshkora berishda poniganal setka hamma qirralarni chizish orqali beriladi va umumiy qirralar qayta chizilmaydi. Misol:



2.10-rasm. Poligonlarni tasvirlash.

$V=(V_1, V_2, V_3, V_4) = ((X_1, Y_1, Z_1), \ldots, (X_4, Y_4, Z_4))$.
$P_1=(E_1,E_4,E_5)$ $P_2=(E_2,E_3,E_4)$
$E_1=(V_1, V_2, P_1, 0)$, $E_2=(V_2, V_3, P_2, 0)$, $E_3=(V_3, V_4, P_2, 0)$.
$E_4=(V_4, V_2, P_1, P_2)$, $E_5=(V_4, V_1, P_1, 0)$.

Geometrik splaynlar.

Splayn egri chiziqlari.

Kompyuter grafikasida parametrik kubik (3 chi darajali) egri chiziqlar ishlatiladi.

Parametrik ko'rinishda berilgan γ egri chizig'i deb x,y,z koordinatalari

$$x = x(t),\ y = y(t),\ z = z(t),\quad a \le t \le b, \tag{1}$$

munosabatlar bilan aniqlanuvchi $M(x,y,z)$ nuqtalar to'plamiga aytiladi, bu yerda $x(t), y(t), z(t)$ – $[a,b]$ kesmada uzluksiz formulalar:

$u=t-a/b-a$ almashtirish orqali $[a,b]$ kesmani $[0;1]$ kesmaga olib kelishi mumkin. Vektor ko'rinishda (1) chi tenglamani quyidagicha yozish mumkin:

Vektor forma $r = r(t) = (x(t), y(t), z(t)),\ 0 \le t \le 1.$ (2)

Parametrik kub (3) darajali egri chiziqning tenglamasini quyidagicha ko'rinishda yozamiz.

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x$$

$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y, 0 \le t \le 1 \qquad (3)$$
$$z(t) = a_z t^3 + b_z t^2 + c_z t + d_z$$

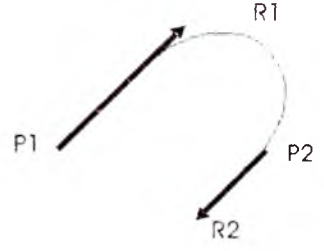
Ermit egri chizig'i.

Ermit egri chizig'i boshlang'ich va oxirgi nuqtalari P_1 va P_2 va ushbu nuqtalardagi egri chiziqqa urunma vektorining yo'nalishlari bilan R_1 va R_2 beriladi (2.11-rasm).

Egri chiziqni noma`lum koeffitsiyentlarini aniqlash uchun (3) tenglamaning birinchi tenglamasini ko'ramiz va uni quyidagi ko'rinishda yozib olamiz.

$x(t) = at^3 + bt^2 + ct + d$ yoki $x(t) = (t^3, t^2, t, 1)(a,b,c,d)$ yoki

$$x(t) = T \cdot X. \quad T = (t^3, t^2\ t, 1), \ X = (a,b,c,d). \qquad (4)$$



2.11-rasm. Ermit egri chizig'i.

(4) chi ifoda differensiallangandan so`ng:

$$X^1(t) = (3t^2, 2t, 1, 0) \cdot X. \qquad (5)$$

Berilgan shartlarni va (4),(5) ni hisobga olgan holda:

$X(0)=P_{1x}=(0,0,0,1) \cdot X$

$X(1)= P_{2x}=(1,1,1,1) \cdot X$ yoki $\begin{pmatrix} P_1 \\ P_2 \\ R_3 \\ R_4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{pmatrix} \cdot X$

$X(0)= R_{1x}=(0.0,1,0) \cdot X$

$X(1)= R_{2x}=(3,2.1,0) \cdot X.$

Qidirilayotgan X ni topish uchun teskari matritsani hisoblash kerak:

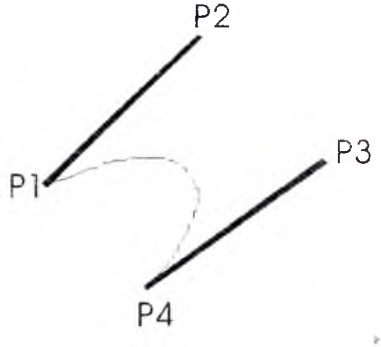
$$X = \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{pmatrix} = M_e \cdot P_e \qquad (6)$$

Hosil bo'lgan M_e matritsasi va R_x ermit geometrik vektori deb ataladi.

Beze egri chizig'i.

Agar Ermit egri chizig'i $P_e=(P_1,P_2,P_1,P_2)$ bilan berilsa, Beze egri chizig'i P_1,P_2,P_3,P_4, nuqtalar yoki $P_b=(P_1,P_2,P_3,P_4)$, orqali beriladi. P_e Ermit geometrik matritsalari va P_b Beze geometrik matritsalari o'zaro quyidagi munosabatlar bilan bog'liq:

$$P_e = \begin{pmatrix} P_1 \\ P_2 \\ R_1 \\ R_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{pmatrix} \cdot \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{pmatrix} = M \cdot P_b.$$



2.12-rasm. Beze egri chizig'i.

Ermit matritsasini M_e M matritsaga ko'paytirib Beze matritsasini olamiz:

$$M_b = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

P_1, P_2, P_1, P_2 nuqtalari bilan beriluvchi Beze egri chizig'i vektor parametrik tenglamasi: $r(t) = (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t)P_3 + t^3 P_4$.

Yoki matritsa kurinishda: $r(t)=T \cdot M_e \cdot P_e = T \cdot (M_e \cdot M) \cdot P_b = T \cdot M_b \cdot P_b$, $0 \le t \le 1$.

$P_0, P_i, \ldots P_m$ nuqtalar bilan aniqlanuvchi Beze egri chizig'i:

1. C – uzluksiz bo'lishi uchun uning uchta P_{3i-1}, P_{3i+1} – nuqtalarning har biri bitta to'g'ri chiziqda yotishi kerak:

2. C – uzluksiz va berk bo'lishi uchun birinchi va oxirgi nuqtasi ustma-ust tushib, P_{m-1}, $P_m = P_0$, P_1 – nuqtalari bitta to'g'ri chiziqda yotishi kerak.

3. Umumiy holda Beze egri chizig'ini quyidagi ko'rinishda yozish mumkin.

4. $r(t) = \sum_{i=0}^{m} C_m^i t^i (1-t)^{m-i} P_i, \quad 0 \le t \le 1.$

5. P_i, $i=0$ egri chiziqni aniqlovchi nuqtalar: $r(t) = \sum_{i=0}^{m} C_m^i t^i (1-t)^{m-i} = (t + (1-t))^m$.

6. $C_m^i t^i (1-t)^{m-i}$ funksional koeffitsiyenlar, ya'ni universal Bershteyn ko'p hollari har doim manfiy emas va ularning yig'indisi doim 1 ga teng.

B-splayn egri chizig'i.

P_1, P_2, P_3, va P_4 nuqtalari bilan aniqlanuvchi B-splayn kubik egri chizig'i matritsa ko'rinishdagi tenglamasi quyidagi ko'rinishda:

$$r(t) = T \cdot M_s \cdot P$$

$$r(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}, T = (t^3, t^2, t, 1)$$

$$P=(P_1, P_2, P_3, P_4), \quad M_0 = \frac{1}{6} \cdot \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

B-splayn egri chizig‘i bazis matritsasi yoki vektori parametrik ko‘rinishda:

$$r(t)=\frac{(1-t)^3}{6}P_1+\frac{3t^3-6t^2+4}{6}P_2+\frac{-3t^3+3t^2+3t+1}{6}P_3+\frac{t^3}{6}P_4.$$

Kubik B-splayn egri chizig‘i uzluksiz, bundan tashqari, birinchi va ikkichi hosilalari uzluksiz. $P_1,P_2,P_3,$ va P_4 nuqtalari (4) qavariq ko‘pburchak uchlarini (5), qavariq ko‘pyoqlikning uchlarini tashkil qiladi va egri chiziq uning ichida yotadi.

B-splayn egri chizigi berk bo‘lishi uchun uchta nuqta qo‘shilishi etarli:

$$P_{m+1}=P_0,\quad P_{m+2}=P_1,\quad P_{m+3}=P_2.$$

Agar B-splayn egri chizig‘ining uchta ko‘shni nuqtasi $P_i,P_{i+1},$ P_{i+2} bitta to‘g‘ri chiziqda yotsa, egri chiziq to‘g‘ri chiziqqa urinib o‘tadi.

P_1,P_2,P_1,P_2 nuqtalari bilan beriluvchi Beze egri chizig‘i vektor parametrik tenglamasi: $r(t)=(1-t)^3P_1+3t(t-1)^2P_2+3t^2(1-t)P_3+t^3P_4$

yoki matritsa ko‘rinishda $r(t)=T*M_Б*P_Б \quad 0\le t\le 1.$

$P_0,P_i,\ldots P_m$ nuqtalar bilan aniqlanuvchi Beze egri chizig‘i:

1) C^1 – uzluksiz bo‘lishi uchun uning uchta $P_{3i-1},\ P_{3i+1}$ – nuqtalarining har biri bitta to‘g‘ri chiziqda yotishi kerak yoki $\overline{P_{3i-1}P_{3i}}=k\overline{P_{3i}P_{3i+1}}$.

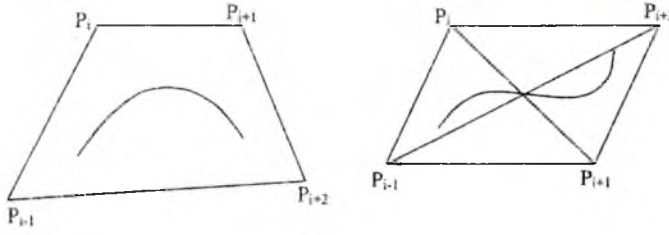
2) C^1 – uzluksiz va berk (yopiq) bo‘lishi uchun birinchi va oxirigi nuqtasi ustma-ust tushib $P_{m-1},\ P_m=P_0,\ P_1$ nuqtalari bitta to‘g‘ri chiziqda yotishi kerak.

3) C^2– uzluksiz bo‘lishi uchun $P_{3i-2},P_{3i-1},P_{3i},P_{3i+1},P_{3i+2},\ (i\ge 1)$ nuqtalari bitta tekislikda yotishi kerak.

Umumiy holda Beze egri chizig‘ini quyidagi ko‘rinishda yozish mumkin:

$$r(t)=\sum_{i=0}^{m}C_m^it^i(1-t)^{m-i}Pi,\quad 0\le t\le 1$$

bu yerda: $P_i,\ i=\overline{0,m}$ egri chiziqni aniqlovchi nuqtalari; $r(t)=\sum_{i=0}^{m}C_m^it^i(1-t)^{m-i}=(t+(1-t))^m=1;\ C_m^it^i(1-t)^{m-i}$ funksional koeffitsiyentlar, ya’ni universal Bershteyn ko‘phadlari, ular har doim manfiy emas va ularning yigindisi doim 1 ga teng.



2.13-rasm. B-splayn egri chizig'i.

Splayn sirtlari.

Kompyuter grafikasida bikubik splayn sirtlari keng ishlatiladi. Xususan, Beze va B-splayn sirtlari.

Beze kubik sirtlari fazoda 16 ta nuqta bilan aniqlanadi:

$$P_{ij},\ i=1,2,3,4,\ \ j=1,2,3,4$$

Parametrik tenglamasi quydagi ko'rinishga ega:

$$r(s,t)=\sum_{i=1}^{4}\sum_{j=1}^{4}C_3^{i-1}C_3^{j-1}S^{i-1}(1-S)^{4-i}t^{j-1}(1-t)^{4-j}P_{ij}$$

bu yerda: $0\le s\le 1,\ 0\le t\le 1,\ r(s,t)=(x(s,t),y(s,t),z(s,t))$
yoki quyidagi qo'rinishda:

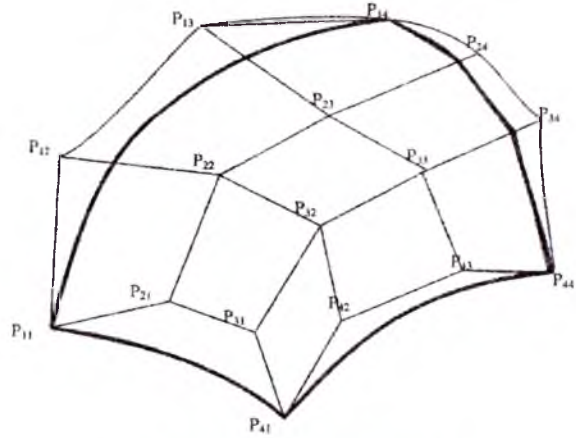
$$X(s,t)=SM_bP_xM_b^tT^t$$
$$Y(s,t)=SM_bP_yM_b^tT^t$$
$$Z(s,t)=SM_zP_xM_b^tT^t$$

bu yerda: $S=(s^3,s^2,s,1),\ T=(t^3,t^2,t,1).$
M_b- Beze matritsasi.

$$P_x=\begin{pmatrix}P_{11} & P_{12} & P_{13} & P_{14}\\ P_{21} & P_{22} & P_{23} & P_{24}\\ P_{31} & P_{32} & P_{33} & P_{34}\\ P_{41} & P_{42} & P_{43} & P_{44}\end{pmatrix}$$

$P_Y P_Z$ mos sirtni aniqlovchi *u, z* koordinatalari matritsalari.

Beze sirtining xossalari:
1. Sirt qavariq kubikda yotadi;
2. Sirt silliq (uzluksiz);
3. P_{11}, P_{14}, P_{41}, P_{44} nuqtalarga tayanadi.



2.14-rasm. Beze splayn sirti.

B-splayn sirti tenglamasini quydagicha bo‘sh bo‘yash usullari:

$$X(s,t) = SM_B P_x M_B^t T^t$$
$$Y(s,t) = SM_B P_y M_B^t T^t$$
$$Z(s,t) = SM_B P_z M_B^t T^t$$

Nazorat savollari:

1. Poligonal setka nima?
2. Poligonal setkani berish usullarini tavsiflang?
3. Ixtiyoriy obyektning qurilishini poligonal setka orqali tushuntiring?
4. Geometrik splaynlar. Splayn egri chiziqlariga izoh bering?
5. Ermit splayn egri chizig‘ining tavsifini bering?

6. Beze splayn egri chizig'ining tavsifini bering?
7. B-splayn egri chizig'ining tavsifini bering?
8. Splayn sirtlariga izoh bering?

Tayanch iboralar: Poligonal setka, ko'pburchak, uchlar, yoqlar, qirralar, Splayn egri chiziqlari, Ermit egri chizig'i, Beze egri chizig'i, B-splayn egri chizig'i, Splayn sirtlari.

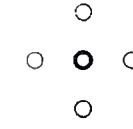
2.4. Rastr grafikasining algoritmlari

Ko'pgina grafik qurilmalar rastrli, ya'ni tasvirni piksellar (rastr) to'g'ri burchakli matritsasi (butun sonlardan tuzilgan setka) ko'rinishda ifodalaydi. Shu sababli rastr algoritmlariga zaruriyat tug'iladi. Ammo, aytish joizki ko'pgina grafik kutubxonalarda (modul) yetarlicha oddiy rastr algoritmlari mavjud [4, 8, 13].

Rastr (grafikasida) setkasida asosiy tushunchalardan biri bu bog'lanishlik, ya'ni rastr chizig'ining ikki qo'shni (yonma-yon joylashgan) piksellarning bog'lanish imkonyati. Savol: qachon (x_1, y_1)va (x_2, y_2) piksellar qo'shni deb hisoblanadi?

To'rt bog'lanishlik. Piksellar qo'shni deyiladi agar ularning yoki x - koordinatalari yoki y - kordinatalari, birga (1) farq qilsa, ya'ni:

$$|x_1 - x_2| + |y_1 - y_2| \le 1.$$



Sakkiz bog'lanishlik. Piksellar qo'shni deyiladi agar ularning x - va y - koordinatalari birdan ko'pga farq qilmasa, ya'ni:

$$|x_1 - x_2| + |y_1 - y_2| \le 1.$$

to'rt bog'lanishlik tushunchasi sakkiz bog'lanishdan kuchliroq, ya'ni ikkita to'rt bog'lanishlik piksellar har doim sakkiz bog'lanishlik, teskarisi har doim o'rinli emas.

Rastr setkasida ixtiyoriy egri chiziq $P_1, P_2, ..., P_n$ piksellar guruhi orqali ifodalanadi, bu yerda ixtiyoriy ikkita P_i va P_{i+1} qo'shni piksellar.

Yuqorida keltirilgan ta'riflarga ko'ra egri chiziq to'rt bog'lanishlik va sakkiz bog'lanishlik bo'lishi mumkin.

Brezenxeym algoritmi. Kesmaning rastr tasviri.

(X_1, Y_1) va (X_2, Y_2) nuqtalarini tutashtiruvchi kesmaning rastr tasvirini ko'rish masalasini ko'ramiz.

Faraz qilamizki $0 \le y_1 \le y_2 \le x_1 \le x_2$.

Berilgan ikki nuqtadan o'tuvchi to'g'ri chiziq tenglamasini tuzamiz:

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1},$$

Unda kesma quyidagi tenglama bilan beriladi:

$$y = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1), \quad x \in [x_1, x_2].$$

yoki: $y = ky + b$, bu yerda $k = \frac{y_2 - y_1}{x_2 - x_1}$, $b = y_1 - kx_1$.

$$d_i = 2dyx_{i-1} - 2dxy_{i-1} + 2dy - dx,$$

keyingi qadamga, ya'ni $i+1$:

$$d_{i+1} = 2dyx_i - 2dxy_i + 2dy - dx$$

d_{i+1} dan d_i ayiramiz va $x_i - x_{i-1} = 1$ ni hisobga olgan holda:

$$d_{i+1} = d_i + 2dy - 2dx(y_i - y_{i-1})$$

So'ng, agar $d_i < 0$ bo'lsa s_i tanlanadi, u holda $y_i = y_{i-1}$ va $d_{i+1} = d_i + 2dy$.

Aks holda, ya'ni $d_i \ge 0$ bo'lsa s_i tanlanadi, va u holda $y_i - y_{i-1} = 1$.

$$d_{i+1} = d_i + 2(dy - dx).$$

Shunday qilib biz d_{i+1} ni d_i ning qiymati orqali hisoblash va s_i, T_i nuqtalarni tanlash uchun interativ usulni hosil qildik. Boshlang'ich holatda $d_i=2dy-dx$ $(x_0, y_0)=(0,0)$ ni hisobga olgan holda $i=1$ da topiladi.

Brezenxeym algoritmi uchun dastur quyidagicha ifodalanadi:

```
#include <iostream.h>
#include <graphics.h>
#include <conio.h>

void brezline(int x1,int y1,int x2,int y2,int c)
{
int dx,dy,d,d1,d2,x,y;
dx=x2-x1;
dy=y2-y1;
d=2*dy-dx;
d1=2*dy;
d2=2*(dy-dx);
x=x1;
y=y1;
putpixel(x,y,c);
while (x<x2)
{
x=x++;
if (d<0) d=d+d1; else
{
y=y++;
d=d+d2;
}
putpixel(x,y,c);
}
}
void main()
int gd=0,gm;
initgraph(&gd,&gm,"c:\\borlandc\\bgi");
brezline(100,100,200,200,10);
getch();
```

```
closegraph();
}
```

Sohani bo'yash (rang berish).

Komp'yuter grafikasida soha 2 ta usul bilan berilishi mumkin:

1. Sohani tashkil etuvchi tashqi nuqtalari bilan, ya'ni sohani ichida yotuvchi har bir piksel biror bir rang (oldcolor) bilan beriladi (chegaradagi piksellar bu qiymatga ega emas).

2. Soha chegarasi bilan berilishi mumkin, ya'ni chegaradagi piksellar biror bir rang bilan (bcolor) beriladi (chegara ichidagi piksellar bu qiymatga ega emas). Shu sababli sohani bo'yash. algoritmlari ikki turga bo'linadi.

Bundan tashqari, 4 va 8 bog'lanishlik sohalar uchun algoritmlar mavjud. Ichki oldcolor rang bilan berilgan yangi newcolor rang bilan 4 bog'lanishlik sohani bo'yash oddiy rekursiya algoritmini keltiramiz:

```
function fill4 (int x, int y,int newcolor,int oldcolor)
{
if (getpixel(x,y)==oldcolor)
{
putpixel (x,y,newcolor);
fill4 (x,y-1,newcolor,oldcolor);
fill4 (x,y+1,newcolor,oldcolor);
fill4 (x-1,y,newcolor,oldcolor);
fill4 (x+1,y,newcolor,oldcolor);
 } }
function fill4 (int x, int y,int newcolor,int oldcolor)
{
if (getpixel(x,y)==oldcolor)
{
putpixel (x,y,newcolor);
fill4 (x,y-1,newcolor,oldcolor);
fill4 (x,y+1,newcolor,oldcolor);
fill4 (x-1,y,newcolor,oldcolor);
fill4 (x+1,y,newcolor,oldcolor);
}
}
```

Bu yerda, (*x,y*) ixtiyoriy soha ichida yotuvchi nuqta, oldcolor qiymatiga ega piksel.

Chegaradagi rangi bilan berilgan (bcolor) sohani bo'yash algoritmi quyidagicha:

```
#include <iostream.h>
#include <graphics.h>
#include <conio.h>
#include <dos.h>
void floodfill (int x,int y,char BorderColor,char Newcolor)
{
if (getpixel(x,y)!=BorderColor)
{
 if(getpixel(x,y)!=Newcolor)
 {
putpixel (x,y,Newcolor);
floodfill (x-1,y,BorderColor,Newcolor);
delay (10);
floodfill (x+1,y,BorderColor,Newcolor);
delay (10);
floodfill (x,y-1,BorderColor,Newcolor);
delay (10);
floodfill (x,y+1,BorderColor,Newcolor);
}
}
}
void main()
{int gd=0,gm,errorcode;
initgraph(&gd,&gm,"c:\\borlandc\\bgi");
circle (300,300,20);
floodfill (300,300,15,10);
getch();
closegraph();
}
```

Bu yerda, (*x,y*) - sohaning ichida yotuvchi biror bir nuqta (piksel), newcolor-bo'yash rangi.

Keltirilgan algoritmlarni 8 bog'lanishlik sohalarga 4 ta yo'nalishni 8 ta yo'nalishga almashtirish orqali osongina o'tkazish mumkin.

Sazerland-Koxen algoritmi. Kesmaning kesilishi.

Kompyuter ekraniga chiqarish kerak bo'lgan tasvirni biror berilgan chegara bo'yicha kesilishi keng qo'llaniladi. Ko'p hollarda chegara sifatida to'g'ri to'rtburchakli soha ishlatiladi, xususan, kompyuter ekrani.

Kesmani biror bir to'rtburchakli soha bilan kesilish oddiy va qulay algoritmini ko'rib chiqamiz.

Faraz qilamizki bizga (x_1,y_1) va (x_2,y_2) nuqtalari bilan kesma berilgan bo'lsin. To'g'ri burchakli to'rtburchak esa quyidagi qiymatlar bilan berilgan bo'lsin:

$$x_{min}, y_{min}, x_{max}, y_{max}.$$

Xususiy holni ko'ramiz, ya'ni kesmaning bir uchi to'g'ri to'rtburchakli sohani ichida, ikkinchisi esa tashqarida joylashgan bo'lsin. Aynan shu holat bizni qiziqtiradi. Bu yerda kesmani soha chegarasi bilan kesilish nuqtasi $M(x,u)$ ni topish kerak.

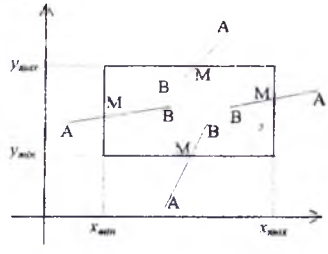
Faraz qilamizki (x_1, y_1) nuqta to'g'ri burchakli to'rtburchak tashqarisida, (x_2, y_2) nuqta esa soha ichida yotsin.

Ushbu masalani yechishda (x_1, y_1) va (x_2, y_2) nuqtalaridan o'tuvchi

$$\frac{x-x_2}{x_1-x_2}=\frac{y-y_2}{y_1-y_2}$$

to'g'ri chiziq tenglamasidan foydalanamiz.

Qaralayotgan masalada, ya'ni kesilish nuqtasi $M(x,u)$ ni aniqlash jarayonida quyidagi hollar bo'lishi mumkin:



2.15-rasm. Kesmani to'g'ri burchakli to'rtburchak bilan kesilish hollari.

1. Agar $x_1 < x_{min}$ u holda $x = x_{min}$, $\frac{x_{min} - x_2}{x_1 - x_2} = \frac{y - y_2}{y_1 - y_2} \Rightarrow y = (y_1 - y_2)\frac{x_{min} - x_2}{x_1 - x_2} + y_2$.

2. Agar $y_1 < y_{min}$ u holda $y = y_{min}$, $\frac{x - x_2}{x_1 - x_2} = \frac{y_{min} - y_2}{y_1 - y_2} \Rightarrow x = (x_1 - x_2)\frac{y_{min} - y_2}{y_1 - y_2} + x_2$.

3. Agar $x_1 > x_{max}$ u holda

$x = x_{max}$, $\frac{x_{max} - x_2}{x_1 - x_2} = \frac{y - y_2}{y_1 - y_2} \Rightarrow y = (y_1 - y_2)\frac{x_{max} - x_2}{x_1 - x_2} + y_2$.

4. Agar $y_1 > y_{max}$ u holda

$y = y_{max}$, $\frac{x - x_2}{x_1 - x_2} = \frac{y_{max} - y_2}{y_1 - y_2} \Rightarrow x = (x_1 - x_2)\frac{y_{max} - y_2}{y_1 - y_2} + x_2$.

Bu yerda $M(x,y)$ biz qidirayotgan nuqtaning koordinatalari, ya'ni soha bilan kesilgandan so'ng kesma (x_1,y_1) va (x_2, y_2) nuqtalari orqali ifodalanadi.

Nazorat savollari:

1. Rastr grafikasida bog'lanishlik tushunchasini izohlang?
2. Bog'lanishlik qanday turlarga bo'linadi va ularning farqi?
3. Brezenxeym algoritmini misollar orqali tushunturing?
4. Kesmaning rastr tasviriga misol keltiring?
5. Sohani bo'yash usullarini misollar orqali berilishini tavsiflang?
6. Komp'yuter grafikasida soha necha hil usulda beriladi?
7. Kesmaning kesilishi. Sazerland-Koxen algoritmini tavsiflang?
8. Kesmaning kesilishiga oid misollar keltiring?

Tayanch iboralar: To'rt bog'lanishlik, sakkiz bog'lanishlik, kesmaning rastr tasviri, sohani bo'yash, kesmaning kesilishi.

2.5. Ko'rinmas chiziq va sirtlarni olib tashlash

Biror bir uch o'lchovli obyektni ikki o'lchovli tekislikda (komp'yuter ekranida) qurish uchun avvalo uning qaysi qismlari ko'rinarli, qaysi qismlari ko'rinmas, ya'ni obyektning boshqa yoqlari bilan yopiqligini aniqlash kerak. Proeksiyalashda markaziy yoki paralel proeksiyalash ishlatiladi [4, 13].

Proeksiyalashda proektorlar obyektning har bir nuqtasidan o'tadi. Proeksiyalash yo'nalishi bo'yicha tasvir tekisligiga yaqinroq masofadagi nuqtalar ko'rinadigan hisoblanadi.

Sodda ko'ringanligiga qaramay ushbu masalani yechish ancha qiyinchiliklarga va ayrim hollarda biroz hisob kitoblarga olib keladi. Ushbu masalani yechishda komp'yuter grafikasida ikkita asosiy yondashuv mavjud:

1. Proeksiyalash yo'nalishi bo'yicha tasvir tekisligiga yaqinroq masofada joylashgan obyektning nuqtalarini aniqlash. Bunda displeyning rastr xossalaridan foydalaniladi.

2. Obyektlarni yoki obyekt qismlarini o'zaro taqqoslab obyektlarni yoki obyekt qismlarini ko'rinarliligini aniqlash.

Bu ikki yondashuvni o'zaro ichiga oluvchi algoritmlar ham mavjud.

Ko'rinmas yoqlarni ajratish.

Har bir yoqlari uchun tashqi birlik normal vektori *n* berilgan ko'p yoqlikni ko'ramiz.

Agar yoqning normal vektori *n* va proeksiyalash yo'nalishini beruvchi vektor *l* o'rtasidagi burchak o'tmas bo'lsa, u holda qaralayotgan yoq ko'rinmaydi va ko'rinmas yoq deb ataladi. Agar mos bo'lgan burchak o'tkir bo'lsa, u holda qaralayotgan yoq ko'rinadigan yoq deyiladi. Parallel proeksiyalashda burchakka qo'yiladigan shartni quyidagicha yozish mumkin:

$$(n,l) = (n_1 l_1 + n_2 l_2 + n_3 l_3) \le 0.$$

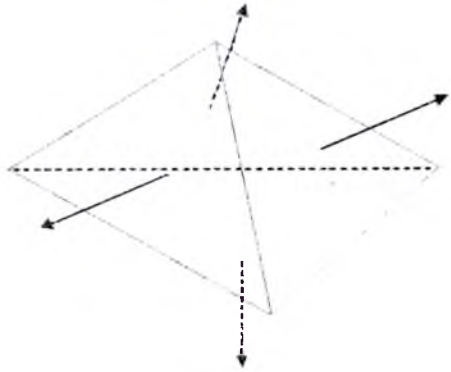
Ushbu shart bajarilsa yoq ko'rinmas.

Yoqning ixtiyoriy *R* nuqtasi markazi C nuqtada joylashgan markaziy proeksiyalashning yo'nalish vektori quyidagicha topiladi:

$$L=C-P.$$

Va so'ng yoqning ixtiyoriy *R* nuqtasi uchun shart tekshiriladi.

$$(n,l) \le 0.$$



2.16-rasm. Normallarning ko‘rinishi.

Ko‘rinmas chiziqlarni (qirralarni) olib tashlash. Robert algoritmi.

Qavariq ko‘pburchaklardan tuzilgan obyektning ko‘rinmas qirralarinini olib tashlash Robert algoritmi hisoblanadi. Ushbu algoritmni keltirib o‘tamiz.

Dastlab ikkita aniqlovchi yoqlarni ko‘rinmas bo‘lgan qirralari olib tashlanadi. Keyingi qadamlarda qolgan qirralar har bir yoqlar bilan yopiqlikka tekshiriladi. Bunda uchta holat mavjud va ular alohida tekshiriladi:

1. Yoq qirrani yopmaydi, bu holda qirra olib tashlanmaydi.
2. Yoq qirrani to‘liq yopadi, bu holda qirra olib tashlanadi.
3. Yoq qirrani qisman yopadi, bu holda qirra bir necha bo‘laklarga bo‘linadi. Qirra ko‘rilgan qirralar ro‘yxatiga qirraning yoq bilan yopilmaydigan qismlari qo‘yiladi.

2.17-rasm. Kesmani ko‘pburchak bilan opilish holari.

Ko'rinmas yoqlarni chiqarib yuborish. Z bufer usuli.

Ko'rinmas chiziq va sirtlarni olib tashlash algoritmlaridan biri bu Z bufer usuli hisoblanadi.

Bu usul bir xil yondashishga to'g'ri keladi va har bir nuqta bilan alohida ishlanadi. Tasvir tekisligidagi har bir nuqtaga (pikselga) (x,y) rangdan tashqari u xotirada saqlanadi. Dastlab uni (chuqurlik) $+\infty$ teng deb hisoblaymiz. Ixtiyoriy yoqni tasvir tekistligiga tasvirlash uchun uning har bir pikseli uchun Z chuqurligi hisoblanadi. Agar u dastlabki chuqurlikdan kichik bo'lsa, bu qiymat Z buferiga kiritiladi va eski qiymati chiqarib yuboriladi. Z buferidagi piksellar ekranda chiqariladi. Qo'shni piksellarning Z chuqurligini hisoblashda Brezenxeym algoritimidan foydalanish tavsiya etiladi. Aytish joizki Z koordinatsiya qiymati obyektlarning yorug'ligini berish yoki ularni umuman olib tashlashda keng qo'llaniladi.

Tartiblash algoritmlari.

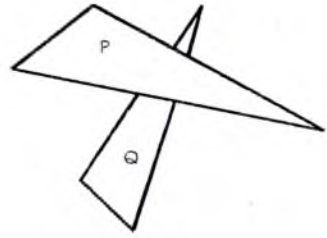
Chuqurligi bo'yicha tartiblash usuli.

Yoqlarni tartiblashning eng oddiy algoritmi bu ularning proeksiyalash yo'nalishi bo'yicha tasvir tekisligigacha bo'lgan minimal masofa bo'yicha tartiblash hisoblanadi, qaysiki, ularni yaqinlashish tartibida chiqarish maqsadida.

OZ o'qi bo'yicha parallel proeksiyalashni ko'ramiz. Faraz qilamizki bizga R va Q yoqlar berilgan bo'lsin. Ularni tasvir tekisligida (kompyuter ekranida) tartiblangan holda chiqarish uchun 5 ta shartni tekshirish tavsiya etiladi. Ularni tekshirish murakkabligi oshishi tartibida keltiramiz:

1. OX o'qidagi yoqlar proeksiyalari kesishadimi?
2. OY o'qidagi ularning proeksiyalari kesishadimi?
3. R yoki Q yoqidan o'tuvchi tekislikdan nisbatan koordinatalar boshi yotadigan tomonida yotmaydi.
4. Q yoki P yoqidan o'tuvchi tekislikga nisbatan koordinatalar boshi yotadigan tomonida yotadi.
5. Yoqlarning tasvir tekisligidagi proeksiyalari o'zaro kesishadi.

Agar keltirilgan shartlardan birortasi inkor bo'lsa R yoki Q yoqiga nisbatan tasvir tekisligida yaqinroq joylashadi va quyidagicha tasvirlanadi:



2.18-rasm. Poligonlarning tartiblanishi.

Varnok algoritmi.

Varnok algoritmi tasvir tekisligini to'rt qismga bo'lishga asoslangan va har bir qismi uchun algoritm oson yechiladi.

Ekran to'rt qismga bo'linadi. Agar qism eng yaqin yoq proeksiyasi bilan to'liq yopilsa, yoki birorta ham yoqning proeksiyasi bilan yopilmasa, unda masala yopiladi, ya'ni to'liq bo'yaladi yoki chetlashtiriladi. Agar ikkala shart ham bajarilmasa u holda qism yana to'rt qismga bo'linadi va shartlar tekshiriladi. Ushbu jarayon qismning o'lchovi bir pikseldan kichik bo'lgunga qadar bajariladi.

Nazorat savollari:

1. Ko'rinmas chiziq va sirtlarni olib tashlashga bo'lgan asosiy yondashuvlar.
2. Ko'rinmas yoqlarni ajratish metodi.
3. Ko'rinmas qirralarni olib tashlashda Robert algoritmining qo'llanilishi.
4. Ko'rinmas yoqlarni chiqarib yuborishda Z bufer usulining qo'llanilishi.
5. Tartiblash algoritmlarining tavsifi.
6. Varnok algoritmi nimaga asoslanadi?

Tayanch iboralar: Ko'rinmas yoqlar, ko'rinmas chiziqlar, Z bufer, tartiblash algoritmi, Varnok algoritmi.

2.6. Nurning oddiy geometrik obyektlar bilan kesilishi

Nurning oddiy geometrik obyektlar bilan kesilish nuqtalarini topishning qulay algoritmlari komp'yuter grafikasida juda ko'p qo'llaniladi.

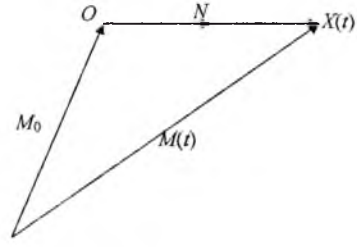
Nurning sfera, tekislik va to'g'ri to'rtburchakli parallelepiped bilan kesilish nuqtalarini topishning qulay algoritmlarini ko'rib chiqamiz.

M_0= (x_0,y_0,z_0) nuqtadan chiquvchi N=(l,m,n) yo'nalishli vektor bilan ifodalanuvchi nurning vektor-parametrik tenglamasi quyidagi ko'rinishda

$$M(t) = M_0 + N \cdot t, \quad t > 0,$$

Yoki koordinatali parametrik tenglama orqali ifodalanadi:

$$\begin{cases} x = x_0 + l \cdot t, \\ y = y_0 + m \cdot t, \quad (t > 0). \\ z = z_0 + n \cdot t, \end{cases} \tag{7}$$



2.19-rasm. Nur.

Agar N-birlik vektor bo'lsa,

$$l^2 + m^2 + n^2 = 1$$

bu holda t parametri oddiy geometrik manoga ega bo'ladi: t ning qiymati nurning boshlang'ich nuqtasi O dan $X(t)$ nuqtasigacha bo'lgan masofaga teng.

Ixtiyoriy vektorni birlik vektorga olib kelish uchun uning har bir koordinatasini uning uzunligiga bo'lish kerak.

Sferaning nur bilan kesilishi.

$C(x_c, y_c, z_c)$ markazi va r-radiusi bilan beriluvchi sferaning tenglamasi quyidagicha ifodalanadi:

$$(x-x_c)^2+(y-y_c)^2+(z-z_c)^2=r^2. \tag{8}$$

Sferani (7) ifoda bilan berilgan nur orqali kesilish nuqtalarini aniqlash uchun x,y,z qiymatlarini (8) tenglamaga qo'yamiz:

$$(x_0+lt-x_c)^2+(y_0+mt-y_c)^2+(z_0+nt-z_c)^2=r^2$$

soddalashtirib quyidagi kvadrat tenglamaga kelamiz:

$$at^2+2bt+c=0, \tag{9}$$

bu yerda:

$$a=l^2+m^2+n^2.$$

(N-birlik vektori uchun);

$$b=l(x_0-x_c)+m(y_0-y_c)+n(z_0-z_c),$$
$$c=(x_0-x_c)^2+(y_0-y_c)^2+(z_0-z_c)^2-r^2.$$

Kvadrat tenglamaning echimlari

$$t_{\pm}=-b\pm\sqrt{b^2-c},\ (a=1).$$

Agar $D=b^2-c<0$ bo'lsa , u holda nur sferani kesmaydi.

Aks holda, ya'ni $D\geq 0$ bo'lsa

$$t^*_-=-b\pm\sqrt{b^2-c},\ \ t^*_+=-b\pm\sqrt{b^2-c}$$

topiladi, va $t^*_{\pm}>0$ tekshiriladi.

Nurni boshiga yaqinroq kesilish nuqtasini topish uchun ulardan kichikroq qiymati aniqlanadi, ya'ni

$$t^*=\min(t^*_-,t^*_+)$$

Nuqtaning koordinatalarini topish uchun nurning tenglamasidan foydalanamiz

$$\begin{cases} x^* = x_0 - lt^*, \\ y^* = y_0 + mt^*, \\ z^* = z_0 + nt^*, \end{cases}$$

va $M^*(x^*, y^*, z^*)$ nuqtasi topiladi. Sferaning ushbu nuqtadagi birlik normal vektori quydagicha topiladi:

$$N = \frac{1}{r}(x^* - x_C, y^* - y_C, z^* - z_C).$$

Tekislikning nur bilan kesilishi.

Faraz qilamizki tekislik umumiy tenglamasi bilan berilgan bo'lsin

$$ax+by+cz+d=0. \qquad (10)$$

Bu yerda $N(a,b,c)$ - tekislikning normal vektori. Agar N vektor birlik, $a^2+b^2+c^2=1$ bo'lsa, d tekislikdan (0,0,0) markazgacha bo'lgan masofa.

(10) tenglamada x,y,z qiymatlarini nurning tenglamasi orqali ifodalasak , u holda t ga nisbatan chiziqli tenglamani olamiz, ya'ni

$$a(x_0 + lt) + b(y_0 + mt) + c(z_0 + nt) + d = 0.$$

Bu yerda

$$t^* = -((ax_0 + by_0 + cz_0 + d))/(al + bm + cn),$$

yoki vektor ko'rinishida

$$t^* = (-((A_0, M) + d))/(A, N).$$

Agar skalyar ko'paytma $(A,N) = al + bm + cn = 0$ bo'lsa, nur tekislikka parallel va uni kesmaydi.

Aks holda $(A,N) \neq 0$ va $t^* > 0$ bo'lsa, kesilish nuqtasini quyidagicha topamiz.

$$\begin{cases} x^* = x_0 + lt^*, \\ y^* = y_0 + mt^*, \\ z^* = z_0 + nt^*, \end{cases}$$

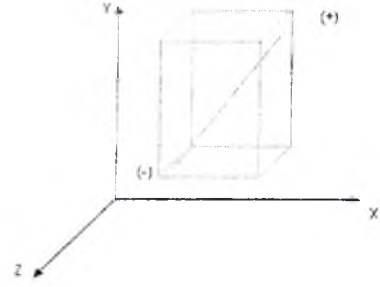
va $M^*(x^*,y^*,z^*)$ nuqtasi topiladi.

Ushbu nuqtada normal vektor quyidagicha olinadi:

A (agar $(A,N)<0$), yoki -A (agar $(A,N)>0$).

Parallelepipedning nur bilan kesilishi.

To'g'ri burchakli va tamonlari koordinata tekisliklariga parallel bo'lgan parallelepipedni ko'ramiz. U holda uning ixtiyoriy birorta diagonaliga tegishli uchlari orqali ifodalash mumkin, ya'ni $(x_-,y_-,z_-),(x_+,y_+,z_+)$, bu yerda $x_-<x_+$, $y_-<y_+$, $z_-<z_+$.



2.20-rasm. Parallelopiped.

$M_0=(x_0,y_0,z_0)$ nuqtadan chiquvchi va $N=(l,m,n)$ yo'nalishli vektori, bu yerda $(l^2+m^2+n^2=1)$, orqali beriluvchi nurni ko'rib chiqamiz.

Parallelopipedning qarama-qarshi tomonlari koordinata tekisliklarga parallel. Dastlab, YZ tekisligiga parallel tomonlarini ko'ramiz, ya'ni $x=x_-$ va $x=x_+$ tekisliklarini.

Agar $l=0$ bo'lsa, nur tekisliklarga parallel va $x_0<x_-$ yoki $x_0>x_+$ holda parallelopipedni kesmaydi. Agar $l\neq 0$ bo'lsa, nur tekisliklarga parallel emas va

$$t_{1x}=(x_- - x_0)/l,\quad t_{2x}=(x_+ - x_0)/l.$$

hisoblanadi va ularda eng kichik yoki kattasi belgilanadi:

$$t_{near}=\min(t_{1x},t_{2x}),\ t_{far}=\max(t_{1x},t_{2x})$$

Shu kabi XY va XZ tekisliklarga parallel tomonlari tekshiriladi va $0<t_{near}<t_{far}$ yoki $0<t_{far}$ bo'lsa nur paralellepipedni kesadi.

Nazorat savollari:

1. Komp'yuter grafikasida nurning geometrik obyektlar bilan kesilishi.
2. Sferaning nur bilan kesilishi tavsifi.
3. Sferaning nur bilan kesilishiga oid misol keltiring?
4. Tekislikning nur bilan kesilishi tavsifi.
5. Tekislikning nur bilan kesilishiga oid misol keltiring?
6. Parallelopipedning nur bilan kesilishi tavsifi.
7. Parallelopipedni nur bilan kesilishiga oid misol keltiring?

Tayanch iboralar: Nurning kesilishi, sferaning nur bilan kesilishi, tekislikning nur bilan kesilishi, parallelepipedning nur bilan kesilishi.

2.7. Bo'yash. Bo'yash usullari

Real tasvirlarni yaratishning keyingi qadami bu qurilgan obyektlarni chegaralovchi sirtlarni bo'yash masalasini yechish. Bo'yash ko'rinmas chiziq va sirtlarni olib tashlashdan so'ngi tartibda bajariladi. Bo'yashning bir necha oddiy usullarini (modellarini) ko'ramiz.

Yorug'lik nuqtasidan sirtga tushuvchi yorug'lik energiyasi singishi, qaytishi (aks etish) yoki o'tkazib yuborishi mumkin. Singigan, qaytgan yoki o'tkazib yuborilgan energiya miqdori yorug'lik to'lqinining uzunligiga bog'liq.

Diffuzion qaytish. Yorug'likning barcha yo'nalishlar bo'yicha tekis tarqalishi. Qaytgan yorug'likning xossalari yorug'lik manbasining shakli, yo'nalishiga va yana yoritilayotgan sirtning joylashishiga va uning xossalariga bog'liq.

Ideal tarqatuvchidan nuqtaviy manbaning yorug'ligi Lamberning kosinuslar qonuniga asosan qaytariladi.

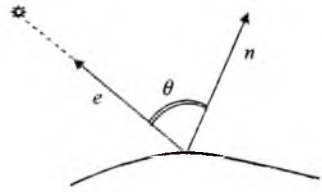
$$I = I_e * k_d * \cos\theta, \qquad 0 \le \theta \le \frac{\pi}{2} \tag{11}$$

Bu yerda I-qaytgan yorug'likning intensivligi;

I_e – nuqtaviy manbaning intensivligi;

k_d – diffuzion qaytishning koeffitsiyenti (const, $0 \le K_d \le 1$);

θ – yorug'lik manbasiga e va sirtning (tashqi) normali n yo'nalishlari o'rtasidagi burchak.



2.21-rasm. Nurning qaytishi.

Yorug'lik manbasiga e va sirtning (tashqi) normali n yo'nalishlarining birlik vektorlaridan foydalangan holda (11) munosabatni quyidagi ko'rinishda yozish mumkin:

$$I = I_e \cdot K_d \cdot (n \cdot e). \tag{12}$$

Real sahna obyektlariga bundan tashqari obyektlarning qaytishi egriligiga mos sochilgan (tekis tarqalgan) yorug'lik tushadi. Bu holda intensivlik quyidagicha hisoblanadi:

$$I = I_a \cdot k_a + I_e K_d \cos\theta, \tag{13}$$

bu yerda I_a - sochilgan yorug'likning intensivligi; k_a – sochilgan yorug'likning diffuzion koeffitsiyenti (*const*, $0 \le k_a \le 1$,).

Albatta, yorug'likning intensivligi obyektdan yorug'lik manbasigacha bo'lgan masofa d, ga bog'liq. Buni hisobga olgan holda quyidagi yoritish modelini olamiz:

$$I = I_a \cdot k_a + \frac{I_e k_d \cos\theta}{d + K} \cdot \cos\theta, \tag{14}$$

bu yerda K- ixtiyoriy konstanta.

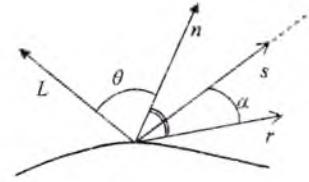
Simmetrik aks obyektning (tashqi) sirti yaltiroq bo'lganda hosil bo'ladi. Simmetrik akslangan yorug'likning intensivligi tushish burchagi to'lqinning uzunligi va moddaning xossasiga bog'liq. Simmetrik aksning fizik xossalari juda murakkab, shu sababli komp'yuter grafikasida oddiy modellardan foydalaniladi.

Fong modeli.

$$I_f = I_e k_s \cos^p \alpha, \tag{15}$$

bu yerda k_s – eksperimental doimiy (yorug'likning simmetrik akslanuvchi qismi); α- aks nuri r va S orasidagi burchak; R-yorug'likning fazoviy taqsimotining aproksimasiyalovchi darajasi.

(14) va (15) larni birlashtirgan holda obyekt sirtining nuqtalari intensivligini hisoblovchi yoritish modelini olamiz.



2.22-rasm. Nurning qaytishi.

$$I = I_a k_a + \frac{I_e}{d+K}(k_d \cos\theta + k_S \cos^p \alpha). \tag{16}$$

Tashqi normal vektori n, yorug'lik manbasiga L, aks nuri r, kuzatish s yo'nalishlari vektorlarining birlik vektorlaridan foydalangan holda (16) ni quyidagi ko'rinishda yozish mumkin:

$$I = I_a k_a + \frac{I_e}{d+K}(k_d(n*l=L) + k_S(r*s)^p). \tag{17}$$

Rangli tasvirni hosil qilish uchun, har bir asosiy ranglar uchun (red-qizil, green-yashil, blue-ko'k) bo'yash funksiyasini topish zarur.

Simmetrik aksda doimiy R_s har bir keltirilgan rang uchun bir xil hisoblanadi. Agar nuqtaviy manbalar soni bir nechta bo'lsa (m), unda yoritish modeli quyidagicha aniqlanadi:

$$I = I_a k_a + \sum_{j=1}^{m} \frac{I_{sj}}{d+K}(k_d \cos\theta_j + k_s \cos^{R_s}\alpha_j). \qquad (18)$$

Guro usuli.

Bu usul uchlarning yorug'liklari aniqligiga asoslangan holda ularning qiymatlarini bir chiziqli interpolyasiyalash orqali butun yoqning yorug'lik qiymatlarini topishga asoslangan. Qavariq to'rtburchakli yoqni quramiz. Faraz qilamizki V_1, V_2, V_3, V_4 uchlarida mos I_{V1}, I_{V2}, I_{V3}, I_{V4} intensivliklar berilgan. Yoqda ixtiyoriy W nuqtasini olamiz. Ushbu nuqtalardan o'tuvchi gorizontal to'g'ri chiziqni o'tkazib yoqning chegarasi bilan kesishish nuqtalarini belgilaymiz U va V.

Faraz qilamizki intensivlik kesmada chiziqli o'zgaradi, ya'ni:

$$I_w = (1-t)I_u + tI_v$$

bu yerda: $t = \frac{|UW|}{|UV|}$, $0 \le t \le 1$.

Shu kabi U va V nuqtalardagi intensivliklarni yozamiz, ya'ni ular yoqni uchlarining intensivliklari orqali ifodalanadilar.

$$I_u = (1-U)I_{V4} + UI_{V1}$$
$$I_V = (1-V)I_{V1} + UI_{V2}$$

Bu yerda

$$U = \frac{|V_4 U|}{|V_1 V_4|},\ 0 \le U \le 1$$
$$V = \frac{|V_1 V|}{|V_1 V_2|},\ 0 \le V \le 1.$$

Fong usuli.

Fong usuli har bir nuqtada normal vektorni hisoblashdan iborat, so'ng qaralayotgan nuqtadagi yorug'lik intensivligi (16) chi formulaga asosan hisoblanadi. Bu yerda interpolyasiya sxemasi Guro bo'yash interpolyasiyasiga o'xshaydi.

W nuqtaning normal vektorini n_w topish uchun ushbu nuqtadan gorizoantal to'g'ri chiziqni o'tkazamiz va yoqning qirrasini

kesuvchi nuqtalarning U va V normal vektorlaridan foydalanilgan holda topamiz:

$$n_w = \frac{(1-t)n_u + tn_v}{(1-t)n_u + tn_v} \quad \text{bu yerda } t = \frac{|uw|}{|uv|},\ 0 \le t \le 1.$$

U va V nuqtalarda normal vektorlarni topish uchun mos qirralarning vektorlaridan foydalanamiz.

$$N_u = (1-u)nv_4 + unv_1$$
$$N_v = (1-v)nv_1 + vnv_2$$

Bu yerda:

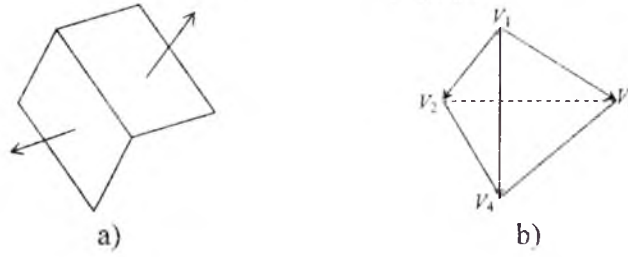
$$u = \frac{|v_4 u|}{|v_4 v_1|},\quad v = \frac{|v_1 v|}{|v_1 v_2|}$$

Fonga usuli orqali bo'yashda tasvir Guro usuliga nisbatan realroq bo'ladi, ammo, hisob-kitoblar sezilarli darajada ko'p hajmni talab qiladi.

Qirralarda va uchlarda normallarni aniqlash.

Faraz qilamizki bitta nuqtada tutashuvchi qirralar yotadigan tekisliklar tenglamalari quyidagicha:

$$A_i x + B_i y + C_i z + D_i = 0,\ i = 1,\ldots,m.$$



2.23-rasm. Qirra va uchlar yo'nalishini aniqlash.

Ushbu tekisliklarning normal vektorlari mos ravishda:

$$(A_i, B_i, C_i),\ i = 1,\ldots,m.$$

Agar ular tashqi normal vektorni bermasa ularning koordinatalarini ishorasini o'zgartirishi kifoya. Taqribiy normal vektorning yo'nalishini aniqlovchi vektor quyidagicha topiladi:

$$(A,B,C)=\sum_{i=1}^{m}(A_i,B_i,C_i)$$

2.23-rasmda ko‘rsatilgan V_1 uning tashqi normal vektorini topish uchun mos vektor ko‘paytmalar yig‘indisi hisoblanadi:

$$V_1V_2 * V_1V_3 \qquad V_1V_3 * V_1V_4 \qquad V_1V_4 * V_1V_2$$

ya’ni

$$n_{1+1} = V_1V_2 * V_1V_3 + V_1V_3 * V_1V_4 + V_1V_4 * V_1V_2.$$

Nazorat savollari:

1. Yorug‘likning diffuzion qaytishini tavsiflang?
2. Diffuzion qaytishning koeffitsiyenti deganda nima tushuniladi?
3. Yorug‘likning intensivligi nima?
4. Yorug‘lik manbasi deganda nimani tushunasiz?
5. Simmetrik aks qanday vaziyatda hosil bo’ladi?
6. Fong modelini tushuntiring?
7. Guro va Fong usullarini tavsiflang?
8. Qirralarda va uchlarda normallarni aniqlash usullari?

Tayanch iboralar: Bo‘yash, diffuzion qaytish, yorug‘lik manbasi, manbaning intensivligi, diffuzion koeffitsiyent, simmetrik aks.

2.8. Nurning yo‘nalishini kuzatish usullari (trace - kuzatish)

Real tasvirlarni ko‘rishda nurning yo‘nalishini kuzatish usullari, uning qaytish (aks etish) va sinish effektlarini hisobga olgan holda keng qo‘llaniladi.

Bunda yorug‘lik manbasidan to‘g‘ri chiziqli trayektoriya bo‘yicha biror bir obyektga tushadi. Obyektga tushgan yorug‘lik nuri sinib obyektning ichiga o’tishi yoki qaytishi mumkin. Obyektdan qaytgan nur yana to‘g‘ri chiziqli tarqalib keyingi obyektga tegishi mumkin va hokazo. Oxirida nurlarning bir qismi nazoratchi ko‘ziga tushib unda tasvirni hosil qiladi. Agar nazoratchi ko‘zi oldida tasvir tekisligini joylashtirsak uni kesib o‘tgan nurlar unda

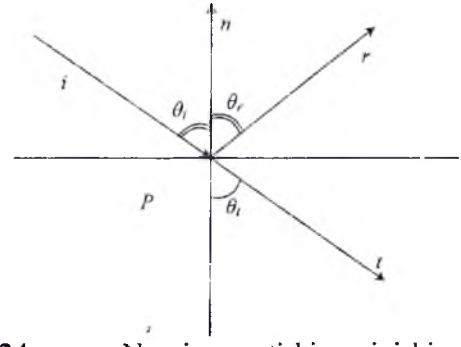
tasvirini paydo qiladi. Yuqorida aytib o'tilgan jarayon nurlarning to'g'ri yo'nalishini kuzatish deyiladi. Bu holda yorug'lik manbasidan tarqalgan ko'pgina nurlar obyektlarga tushmasligi yoki nazoratchi ko'ziga etib kelmasligi mumkin.

Shuning uchun kompyuter grafikasida faqat foydali nurlar, ya'ni nazoratchi ko'ziga tushuvchi nurlarni hisobga olish tavsiya qilinadi. Shu sababli nurlarning yo'nalishini kuzatish teskari bajariladi, ya'ni yorug'lik manbasidan emas, nazoratchi ko'zidan boshlab biror bir obyektning nur bilan kesilish nuqtasiga qadar.

Yuqorida aytib o'tilgan jarayon nurning yo'nalishini teskari kuzatish, yoki shunchaki nurning yo'nalishini kuzatish deyiladi. Aynan shu usulni quyida ko'ramiz:

Nurning yo'nalishini kuzatish usulida obyektning ixtiyoriy nuqtasining yorug'ligi va berilgan yo'nalishda undan qaytuvchi yorug'lik energiyasi qismini aniqlash masalasi ko'riladi. Bu energiya ikki qismdan tashkil topadi – bevosita (dastlabki) yorug'lik, ya'ni yorug'lik manbasidan bevosita olinuvchi energiya va ikkinchi yorug'lik, ya'ni boshqa obyektlardan keluvchi energiya. Bu bo'lish nisbiy.

Ma'lumki, bevosita yorug'lik tasvirga katta hissa qo'shadi.



2.24-rasm. Nurning qaytishi va sinishi.

Simmetrik qaytish.

Berilgan *i* yo'nalish bo'yicha *P* nuqtaga tushgan nur *r* yo'nalish bo'yicha qaytadi va quyidagi qonun bilan aniqlanadi: nurning yo'nalishini aniqlovchi *i* vektor, qaytgan nurning yo'nalishini

aniqlovchi r vektori va P nuqtaning normal vektori n bitta tekislikda yotadi. Normal vektorga nisbatan nurning tushish burchagini Q_i va qaytish burchagi Q_r deb belgilaymiz.

Faraz qilamizki, i,n,r vektorlar birlik (normal) vektorlar bo'lsin. Yuqorida keltirilgan shartlarga ko'ra r vektori i va n vektorlari chiziqli kombinatsiyasiga teng:

$$r = \alpha i + \beta n, \tag{19}$$

tushish va qaytish burchagi o'zaro teng, ya'ni $Q_i = Q_r$.
Bunga ko'ra:

$$(-i,n) = Cos\theta_i = Cos\theta_r = (r,n), \tag{20}$$

bu yerdan quyidagi ifodani olamiz:

$$r = i - 2(i,n) \cdot n.$$

Isbot:

$(r,n)=(-i,n) \Rightarrow (r,n)= - (i,n) \Rightarrow (r,n)=(i,n)-2(i,n)-|(n,n)=1| \Rightarrow$
$\Rightarrow (r,n)=(i,n)-2(i.n)(n,n) \Rightarrow (r,n)=(i-2(i,n)*n,n) \Rightarrow$
$\Rightarrow r=i-2(i,n)* n.$
(1) ifoda bilan berilgan r vektor – birlik vektor.

Isbot:

$$||r||^2 =(r,r)=(i-2(i,n)n,i-2(n,i)n)=(i,i)-2(n,i)(i,n)-$$
$$2(n,i)(n,i)++4(n,i)\,(n,i)\,(n,n)=$$
$$|=(i,i)=1,(n,n)=1|=1-4(n,i)^2+4(n,i)^2=1.$$

Diffuzion qaytish.

Diffuzion qaytish Lambert qonuni orqali ifodalanadi va unga asosan tushayotgan yorug'lik hamma yo'nalishlarga bir xil intensivlik bilan tarqaladi. Tushayotgan nurning qaytish yo'nalishini aniqlab bo'lmaydi, barcha yo'nalishlar bir xil va nuqtaning yorug'ligi (i,n) ga proporsional.

Ideal sinish.

i vektori yo'nalishi bo'yicha *R* nuqtaga tushuvchi nur ikkinchi sohaga *t* yo'nalish bo'yicha sinadi. Sinish Sneliusning qonuniga bo'ysunadi va burchaklari uchun quyidagi ifoda bilan beriladi.

$$\eta_i \cdot Cos\theta_i = \eta_t \cdot Cos\theta_t, \qquad (21)$$

bu yerda: η_i - (*i*) nur tushuvchi muhitning sinish koeffitsiyenti; η_i – (*t*) nur sinuvchi muhitning (sohaning) sinish koeffitsiyenti; θ_i – tushuvchi nur (*i*) va *R* nuqtaning normal vektori (*n*) orasidagi burchak; θ_t - qaytgan nur (*t*) va *R* nuqtaning normal vektori (*n*) yo'nalish orasidagi burchak.

Sneliusning qonuniga asosan *i,n* va *t* vektorlari bitta tekislikda yotadi, ya'ni:

$$t = \alpha_i + \beta_n, \qquad (22)$$

bu yerda α va β qiymatlarini topish uchun (1) ifodani quyidagi ko'rinishda yozamiz:

$$\eta \cdot Sin\theta_i = Sin\theta_t$$

bu yerda: $\eta = \frac{\eta_i}{\eta_t}$.

Kvadratga ko'taramiz:

$$\eta^2 Sin^2\theta_i = \eta^2 Sin^2\theta_t$$

yoki

$$\eta^2(1 - Cos^2\theta_i) = 1 - Cos^2\theta_t$$

$$Cos\theta_i = (-i,n), \quad Cos\theta_t = (-t,n)$$

larni hisobga olgan holda

$$\eta^2(1 - (i,n)^2) = 1 - (-t,n)^2$$

(2) tenglamani hisobga olgan holda

$$\eta^2(1 - (i,n)^2) = 1 - (\alpha i + \beta n, n)^2$$

bundan

$$\alpha^2(i,n)^2 + 2\alpha\beta(i,n) + \beta^2 = 1 + \eta^2((i,n)^2 - 1), \qquad (3)$$

t vektorning normalligi shartiga ko‘ra:

$$\| t \|^2 = (t,t) = \alpha^2 + 2\alpha\beta(i,n) + \beta^2 = 1 . \qquad (4)$$

(4) ifodani (3) ifodadan ayiramiz.

$$\alpha^2((i,n)^2 - 1) = \eta^2((i,n)^2 - 1)$$

bundan, $\alpha = \pm\eta$.

Fizik nuqtai nazarga ko‘ra

$$\alpha = \eta .$$

β- ning qiymatini topish uchun (4) ifodadan foydalanamiz; $\alpha = \eta$ hisobga olgan holda

$$\beta^2 + 2\beta\eta(i,n) + \eta^2 - 1 = 0 ,$$

diskriminant $D = 4(1 + \eta^2((i,n)^2 - 1))$.

Tenglamaning yechimi

$$\beta = -\eta(i,n) \pm \sqrt{(1 + \eta^2((i,n)^2 - 1)} .$$

Buni hisobga olgan holda t vektori quyidagicha ifodalanadi:

$$t = \eta i + (\eta c_i - \sqrt{(1 + \eta^2(c_i^2 - 1)) * n}) ,$$

bu yerda, $c_i = \cos\theta_i = (-i,n) = -(i,n)$.

Agar $1 + \eta^2(c_i^2 - 1) < 0$ bo‘lsa hamma yorug‘lik energiyasi chegarada aks etadi va sinish bo‘lmaydi.

Diffuzion sinish.

Diffuzion sinish, diffuzion qaytish kabi, ya’ni singan yorug‘lik barcha yo‘nalishlar t, $(t,n) < 0$ bo‘yicha bir xil intensivliklar bilan tarqaladi.

Energiya taqsimoti.

Yorug‘likning qaytishi va sinishida energiya taqsimotini ko‘ramiz. Fizika kursidan ma’lumki qaytgan energiya hissasi (ulushi) Frenel koeffitsiyentlari orqali beriladi.

$$F_r(\lambda,\theta) = \frac{1}{2}\left(\left(\frac{Cos\theta_i - \eta Cos\theta_t}{Cos\theta_i + \eta Cos\theta_t}\right)^2 + \left(\frac{\eta Cos\theta_i - Cos\theta_t}{\eta Cos\theta_i + Cos\theta_t}\right)^2\right),$$

bu yerda: λ - to‘lqinning uzunligi.

Ushbu formula yarim o'tkazgichlar uchun o'rinli va uni boshqa ko'rinishda ham yozish mumkin:

$$F_r(\lambda,\theta)=\frac{1}{2}(\frac{c-g}{c+g})(1+(\frac{c(c+g)-1}{c(c-g)-1})^2),$$

bu yerda: $c=Cos\theta_i$; $g=\sqrt{(\eta^2+c^2-1)}=\eta Cos\theta_t$.

O'tkazuvchilar uchun odatda quyidagi formulalardan foydalaniladi:

$$F_2(\lambda,\theta)=\frac{1}{2}((\frac{(\eta_t^2+k_t^2)\cos^2\theta_i-2\eta_t\cos\theta_i+1}{(\eta_t^2+k_t^2)\cos^2\theta_i+2\eta_t\cos\theta_i+1})^2+(\frac{(\eta_t^2+k_t^2)-2\eta_t\cos\theta_i-\cos^2\theta_\perp}{(\eta_t^2+k_t^2)+2\eta_t\cos\theta_i+\cos^2\theta_i})^2),$$

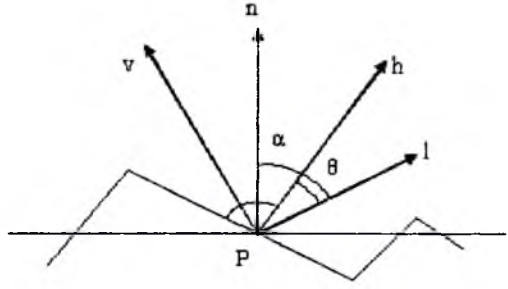
bu yerda: k_t - singish koeffitsiyenti.

Yuqorida keltirilgan barcha holatlar ideallashtirilgan. Aslida ideal qaytuvchi va ideal silliq tekisliklar yo'q.

Nur yo'nalishini kuzatishning asosiy modeli

Odatda amaliyotda obyektning tashqi taqsimot qonuni berilgan tasodifiy yo'naltirilgan tekis ideal mikroyoqlardan tashkil topadi deb hisoblanadi.

Faraz qilamizki: n - sirtning R nuqtadagi normal vektori, ya'ni o'rta tekislik normal vektori; h - mikroyoqning normal vektori; α - ular orasidagi burchak: $\alpha=arccos(n,h)$.



2.25-rasm. Nurning tarqalishi.

Sirt α - ning tasodifiy qiymati taqsimot zichligini beruvchi funksiya $D(\alpha)$ orqali ifodalanadi. $D(\alpha)$ uchun bir nechta model mavjud:

Gauss taqsimoti

$$D(\alpha) = ce^{-(\frac{\alpha}{m})^2}.$$

Bekmen taqsimoti

$$D(\alpha) = \frac{1}{4\pi m^2 \cos^4 \alpha} e^{-(\pi \frac{\alpha}{m})^2}.$$

Bu yerda: m - sirtning tekis emasligini harakterlovchi daraja, m ning qiymati kichik bo'lgan sari sirt shuncha tekisroq bo'ladi.

R nuqtaga l yo'nalishi bo'ycha tushayotgan yorug'lik nurining aks etishini ko'ramiz.

V yo'nalishi bo'yicha aks etuvchi yorug'lik energiyasi qismatini topish uchun quyidagi ifoda kerak bo'ladi:

$$h = \frac{l+V}{\|l+V\|}.$$

Mikroyoqdan aks etuvchi energiya ulushi Frenel $F_r(\lambda, \theta)$ koeffitsiyentlari bilan aniqlanadi, bu yerda

$$\theta = \arccos(h, v) = \arccos(h, l),$$

h, v, l - vektorlari birlik vektorlar.

Qo'shni yoqlarning soya qiluvchi ta'siri, odatda, quyidagi funksiya orqali ifodalanadi:

$$G = mm(l, \frac{2(n,h)(n,v)}{(v,h)}, \frac{2(n,h)(n,l)}{(v,h)}).$$

Mikroyoqlar to'plamidan tashkil topuvchi sirt uchun bizni qiziqtiruvchi energiya ulushi juft yo'nalgan akslanish funksiyasi orqali ifodalanadi, ya'ni (Bidirectional Reflection Distribution Function):

$$BRDF(l, v, \lambda) = \frac{F_\lambda(\lambda, \theta) D(\alpha) G(n, v, l)}{(n,l)(n,v)}.$$

Bu formula hisoblashlar uchun murakkab, shuning uchun oddiyroq formuladan foydalaniladi:

$$BRDF(l, v, \lambda) = (n, h)^k F_r(\lambda, \theta). \quad (23)$$

Umuman olganda *BRDF* funksiyasi simmetrik shartga bo'ysunadi:

$$BRDF(l, v) = BRDF(v, l).$$

Frenel koeffitsiyentlari tasvirning realligiga sezilarli ta`sir ko'rsatishiga qaramay amaliyotda ular juda kam ishlatiladi va (23) formula o'rnida soddaroq formuladan foydalaniladi:

$$BRDF(l, v, \lambda) = (n, h)^k$$

Yana shunday sabablardan biri, formulada ishlatiladigan to'lqin uzunligi λ haqida aniq ma'lumot yo'qligi.

Berilgan yo'nalish bo'yicha R nuqtadan qaytuvchi energiya quyidagicha topiladi:

$$I(\lambda) = K_a I_a(\lambda) C(\lambda) + K_d C(\lambda) \sum_i I_{ei}(\lambda)(n, l_i) + K_r \sum_i I_{ei}(\lambda) \frac{F_r(\lambda, \theta) D(\lambda) G}{(n, l_i)(n, v)} +$$
$$+ K_r I_r(\lambda) F_r(\lambda, \theta_r) e^{-\beta_\lambda d_r} + K_t I_t(\lambda, \theta_t)(1 - F_r(\lambda, \theta_t) e^{-\beta_\lambda d_t})$$

$$I(\lambda) = K_a I_a(\lambda) C(\lambda) + K_d C(\lambda) \sum_i I_{ei}(\lambda)(n, l_i) + K_r \sum_i \frac{I_{ei}(\lambda) F_r(\lambda, \theta) D(\lambda) G}{(n, l_i)(n, v)} + K_3 I_r(\lambda) F_r(\lambda, \theta_r) e^{-\beta_\lambda d_r}$$
$$+ K_t I_t(\lambda, \theta_t)(1 - F_r(\lambda, \theta_t) e^{-\beta_\lambda d_t})$$

bu yerda: $I_a(\lambda)$, $I_{ei}(\lambda)$, $I_r(\lambda)$, $I_t(\lambda)$ - mos ravishda i – chi yorug'lik manbasi, fon yorug'ligi, aks etuvchi nur bo'yicha yo'nalgan yorug'lik, singan nur bo'yicha yo'nalgan yorug'lik intensivligi;

$S(\lambda)$ - P nuqtaning rangi;

K_a, K_d, K_r - fon, diffuzion, aks yorug'lik koeffitsiyentlari;

K_t – signal turining ulushi;

l_i – P nuqtaning i-chi yorug'lik manbasiga bo'lgan yo'nalishning birlik vektori;

θ_r, θ_t - aks va sinish burchaklari;

β_r, β_t - aks va sinish nurlarining bo‘shashish koeffitsiyentlari.

Keltirilgan model fizik nuqtai nazardan qaralganda korrekt bo‘lgani bilan, hisoblashlar uchun juda murakkab. Yorug‘likning oddiy modeli:

$$I(\lambda)=K_a I_a(\lambda)C(\lambda) + K_d C(\lambda)\sum I_{li}(\lambda)(n,l_i) + K_s\sum I_{li}(\lambda)(n,h_i)^P$$

$$I(\lambda) = K_a I_a(\lambda)C(\lambda) + K_d C(\lambda)\sum I_{li}(\lambda)(n,l_i) + K_S\sum I_{li}(\lambda)(n,h_i)^P .$$

Metallar uchun oxirgi hadda rang $C(\lambda)$ qo‘shiladi.

Nazorat savollari:

1. Nurni yo‘nalishni kuzatish usullari qanday vaziyatlarda qo’llaniladi?
2. Nurlarning to‘g‘ri yo‘nalishini kuzatish deb nimaga aytiladi?
3. Simmetrik qaytishni tushuntiring?
4. Diffuzion qaytishni tushuntiring?
5. Ideal sinish deganda nima tushuniladi?
6. Diffuzion sinishga misol keltiring?
7. Energiya taqsimoti nima, unga izoh bering?
8. Nur yo‘nalishini kuzatishning asosiy modeli?

Tayanch iboralar: Simmetrik qaytish, ideal sinish, diffuzion sinish, energiya taqsimoti, nurning tarqalishi.

2.9. Rang. Rang modellari

Rang tushunchasi odam (odamning ko‘zi) yorug‘likni qanday qabul qilishi bilan bog‘liq.

Yorug‘likni o‘z navbatida ikki xil tushunish mumkin – har-xil energiyali zarrachalarning oqimi (u holda rangni zarrachalarning energiyasi aniqlaydi), yoki elektromagnit to‘lqinlarning oqimi (bu holda rang to‘lqin uzunligi λ orqali aniqlanadi). Ko‘rinadigan yorug‘lik bu 400-700 nm (nanometr) gacha bo‘lgan to‘lqin uzunligiga ega elektromagnit to‘lqinlar.

Rang odamning ko‘zida tug‘iladi. Inson ko‘zi yorug‘likni qanday qabul qilishini ko‘ramiz.

Ko‘zning «setchatkasi» fotoreseptorga ega – «kolbochki». Ular tor (ensiz) spektral egri chiziqlar bilan tavsiflanadi va rang ta’sirchanligiga ega. Ular («kolbochki») uch xil bo‘ladi – uzun, o‘rta va qisqa to‘lqinlar ta’sirchanligiga javob beruvchi. Ular («kolbochki») tomonidan beriladigan qiymat spektral funksiya $I(\lambda)$ bilan ta’sirchanlik funksiyasini integrallash natijasini beradi.

2.26-rasmda uch xil tipdagi «kolbochyok»lar uchun ta’sirchanlik funksiyalarining grafiklari keltirilgan.

Shunday qilib, inson ko‘zi spektral funksiya uchun $I(\lambda)$, mos ravishda uchta sonni qabul qilandi (R, G, B) va ular quyidagi formula yordamida hisoblanadi:

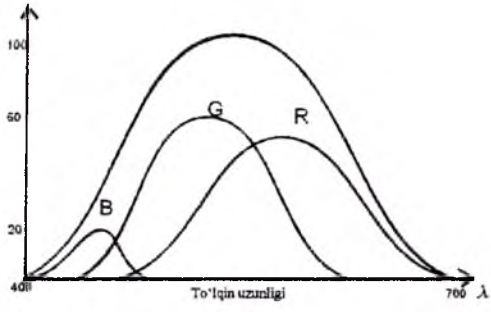
$$R = \int I(\lambda) P_R(\lambda) d\lambda,$$

$$G = \int I(\lambda) P_G(\lambda) d\lambda,$$

$$B = \int I(\lambda) P_B(\lambda) d\lambda.$$

Bu yerda: $P_R(\lambda)$, $P_G(\lambda)$, $P_B(\lambda)$ – mos ravishda har-xil tipdagi «kolbochyok» larning vazniy ta’sirchanlik funksiyalari.

Inson ko‘zining umumiy ta’sirchanligi uchun javob beruvchi egri chiziqlar grafigini olishda uchta egri chiziqlar grafiklari o‘zaro yig‘iladi.



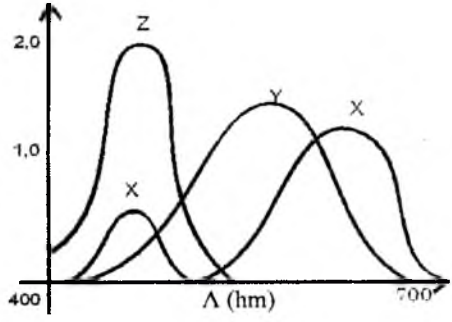
2.26-rasm. Ko‘zning nisbiy ta’sirchanligi.

Aslida ayrim grafiklar manfiy qiymatlarni ham qabul qilishlari mumkin. 1931 yilda Yoritish (Yorug‘lik) bo‘yicha Xalqaro Komissiya (YoXK)si (CIE-Comission Internationale de L’Eclairage) gipotetik ideal kuzatuvchi standart egri chiziqlarni qabul qildi. Ular

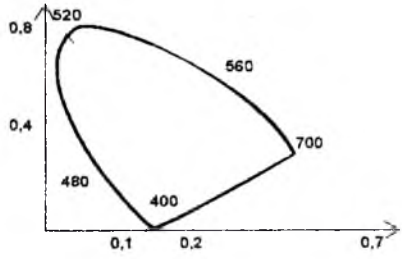
missiya (YoXK)si (CIE-Comission Internationale de L'Eclairage) gipotetik ideal kuzatuvchi standart egri chiziqlarni qabul qildi. Ular yordamida *XYZ* rang modeli quriladi, bunda *x,y,z* asosiy ranglar. *X,Y,Z* ning qiymatlari quyidagi munosabatlar orqali ifodalanadi:

$$X = \int I(\lambda)x(\lambda)d\lambda,$$
$$Y = \int I(\lambda)y(\lambda)d\lambda,$$
$$Z = \int I(\lambda)z(\lambda)d\lambda.$$

Ushbu uchta sonlar orqali inson ko'zi qabul qiladigan ixtiyoriy rangni bir qiymatli ifodalash mumkin.



2.27-rasm. Rang bo'yicha tenglashtirish koeffitsiyenti.



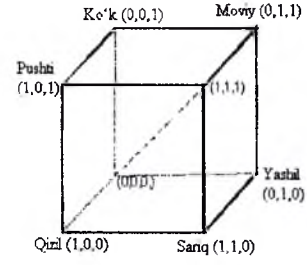
2.28-rasm. Yoritish bo'yicha halqaro komissiyaning rang grafigi.

Aytish joizki *Y* rangi uchun javob beruvchi energiya taqsimoti egri chizig'i inson ko'zining yorug'likka bo'lgan ta'sirchanlik spektral egri chizig'i bilan ustma-ust tushadi.

RGB rang modeli.

RGB (red-qizil, green-yashil, blue-ko'k) rang modeli eng oddiy deb hisoblanadi.

Bu rang modeli additiv, ya'ni biror bir kerakli rangni hosil qilish uchun uning asosiy ranglari yig'iladi. Bu tizim orqali ifodalanuvchi ranglar birligi kubni tashkil qiladi (ya'ni uning ichida yotadi).



2.29-rasm. RGB rangli kub.

Kubning bosh diagonali, ya'ni barcha asosiy ranglar miqdori barobar, kulranglarni beradi, ya'ni qoradan (0,0,0) oq (1,1,1) ranggacha.

CIE XYZ sistemasidan RGB sistemasiga o'tish uchun quyidagi munosabatlardan foydalaniladi:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 3{,}240479 & -1{,}537156 & -0{,}498535 \\ -0{,}969256 & 1{,}875992 & 0{,}041556 \\ 0{,}055648 & -0{,}204043 & 1{,}057311 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}.$$

Agar biror bir rangni RGB tizimi orqali ifodalab bo'lmasa, u holda uning biror bir asosiy rangi manfiy (<0) yoki birdan katta (>1).

Teskari almashtrishni topish uchun teskari matritsadan foydalaniladi.

RGB rang modeli bir qator video qurilmalarda ishlatiladi, xususan rangli televizion monitorlarda.

CMY rang modeli.

Rangli bosmaga chiqaruvchi qurilmalarda CMY (cyan-moviy; magenta-pushti; yellow-sariq) rang modeli tez-tez ishlatiladi. Moviy, pushti va sariq asosiy ranglar qizil, yashil va ko'k ranglarni to'ldiruvchi bo'ladilar.

CMY rang modeli - subtraktiv, ya'ni biror kerakli bo'lgan rangni hosil qilish uchun asosiy ranglar oq rangdan ajraladi.

Moviy rang qog'ozga tushirilayotgan qizil rang to'liq yutiladi, ya'ni moviy rang tushayotgan oq rangdan (qizil, yashil va ko'k

ranglarning yig'indisi) qizil rangni ayirib tashlaydi. Pushti rang yashil rangni yutadi, sariq rang esa – ko'k rangni. Moviy va sariq ranglar bilan bo'yalgan sirt qizil va ko'k ranglarni yutib faqat yashil rangni qoldiradi.

Moviy, sariq va pushti ranglar qizil, yashil va ko'k ranglarni yutib, natijada qora rangni qoldiradi.

Yuqorida keltirilgan munosabatlarni quyidagi formula orqali ifodalaymiz:

$$\begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} R \\ G \\ B \end{pmatrix}.$$

RGB rang modelidan CMY modeliga o'tish quyidagi munosabatlar orqali bajariladi:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} G \\ M \\ Y \end{pmatrix}.$$

Ayrim sabablarga ko'ra, qora rangni hosil qilish uchun uchta asosiy ranglardan foydalanish noqulay. Shu sababli CMY modelining asosiy ranglariga qora (black) qo'shiladi va natijada CMYK rang modeli hosil qilinadi.

CMY rang modelidan CMYK modeliga o'tish uchun quyidagi munosabatlardan foydalanamiz:

$$\begin{aligned} K &= \min(C, M, Y), \\ C &= C - K, \\ M &= M - K, \\ Y &= Y - K. \end{aligned}$$

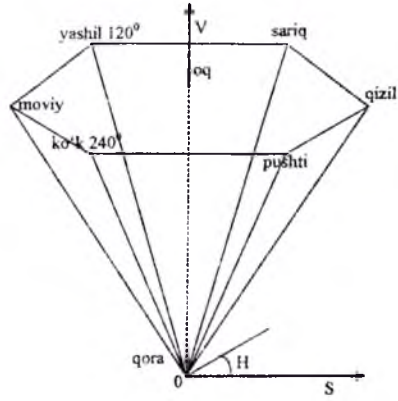
YIQ rang modeli.

Televideniyeda YIQ rang modelidan keng foydalaniladi. Bu model RGB rang modelining bir varianti hisoblanadi, efirga uzatish samaradorligini oshirish maqsadida ishlatiladi va oq-qora televideniye bilan ishlashni ta'minlaydi.

RGB rang modelidan YIQ modeliga o'tish uchun quyidagi munosabatdan foydalaniladi:

$$\begin{pmatrix} Y \\ I \\ Q \end{pmatrix} = \begin{pmatrix} 0{,}30 & 0{,}59 & 0{,}11 \\ 0{,}60 & -2{,}28 & -0{,}32 \\ 0{,}21 & -0{,}52 & 0{,}31 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

Teskari almashtirishlar uchun teskari matritsadan foydalaniladi.



2.30-rasm. HSV olti qirrali konussimon rang modeli.

HSV rang modeli.

Yuqorida keltirilgan RGB, CMY(K) va YIQ rang modellari qurilmalar uchun mo'ljallangan va inson tomonidan rang berish uchun noqulay.

HSV (Hue-ton, saturation-to'yinganlik, value-yorug'lik qiymati) rang modeli foydalanuvchiga mo'ljallangan bo'lib, rangni ton, to'yinganlik, yorug'lik qiymati kabi tushunchalar orqali berish imkonini beradi.

HSV modelida silindrik koordinatalar sistemasi ishlatiladi, barcha ifodalanuvchi ranglar esa olti qirrali konusni tashkil qiladi (2.31-rasm).

Konusning asosi yorug' ranglarga mos keladi (V=1). OV o'qi kulranglarga mos keladi (S=0), bu holda ya'ni S=0 bo'lganda H ning qiymati aniqlanmagan bo'ladi (HUE_UNDEFGIUNED).

Ton N burchak gradusi bilan o'lchanadi, 0^0 ga qizil rang mos keladi, 120^0 ga esa yashil rang va x.k.

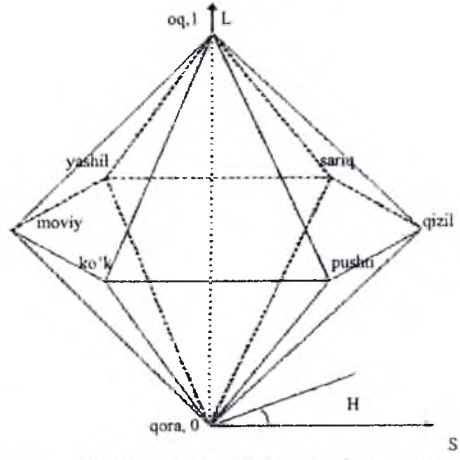
HSV rang modelini ifodalovchi asosiy qiymatlar mos ravishda quyidagicha o'zgaradi:

$$0^0 \le H \le 360^0, \qquad 0 \le S \le 1, \qquad 0 \le V \le 1.$$

oq rangga S=0, V=1 mos keladi, qora rangga esa V=0.

HLS rang modeli.

HLS (Hue–ton, Lightness-yorug'lik, Saturation-to'yinganlik) HSV modelining modifikatsiyasi bo'lib, u orqali ifodalanuvchi ranglar ikkita olti qirrali va asoslari birlashtirilgan konusni tashkil qiladi. Oq rang yuqoriga siljitilgan (2.32-rasm).



2.31-rasm. HLS olti qirrali konussimon rang modeli.

Nazorat savollari:

1. Rang tushunchasiga ta'rif bering?
2. Inson ko'zi spektral funksiya uchun qanday ranglarni qabul qiladi?
3. RGB rang modelining o'ziga xos jihatlari.
4. CMYK rang modelini izohlang.
5. RGB rang modelidan CMYK rang modeliga o'tish.
6. YIQ rang modelini izohlang.
7. RGB rang modelidan YIQ rang modeliga o'tish.
8. HSV va HLS rang modellari.

Tayanch iboralar: Rang, RGB modeli, CMY modeli, YIQ modeli, HSV modeli, HLS modeli.

III bob. OPENGL GRAFIK KUTUBXONASI

3.1. OpenGL grafik kutubxonasi bilan tanishish

OpenGL - ikki va uch o'lchovli grafika sohasida ilovalar yaratish uchun ancha keng tarqalgan amaliy dasturiy interfeys (API – Application Programming Interface) lardan biri hisoblanadi.

OpenGL (Open Graphics Library – ochiq grafik kutubxona) standarti turli xil platformalarda amalga oshirish uchun kerakli, apparatga bog'liq bo'lmagan interfeys sifatida dasturiy ta'minot ishlab chiqarish sohasidagi yetuk firma tomonidan 1992 yilda ishlab chiqilgan va tasdiqlangan. Standartning asosi bo'lib Silicon Graphics Inc firmasi tomonidan ishlab chiqilgan IRIS GL kutubxonasi hisoblanadi [1, 2, 3, 12, 15].

Kutubxona 120 ga yaqin turli buyruqlardan iborat bo'lib, dasturchi operatsiyalar va obyektlarni tayinlashda hamda interaktiv grafik ilovalarni yozishda foydalanadi. Bugungi kunda OpenGL grafik tizimini ko'pgina apparatli va dasturiy platformalarni quvvatlaydi. Ushbu tizim Windows va Linux operatsion tizimlarida ishlovchilar uchun ochiq.

OpenGL kutubxonasining o'ziga xos xususiyatlari, ya'ni ushbu grafik standartning rivojlanishi va keng tarqalishini ta'minlovchi jihatlar, quyidagilar hisoblanadi:

- *Barqarorlik.* Standartga o'zgartirish va qo'shimchalar kiritish, oldingi dasturiy ta'minotlarda ishlab chiqilganlarning mosligini saqlab qolish ko'rinishida amalga oshiriladi.
- *Ishonchlilik va uzatuvchanlik.* OpenGL dan foydalanayotgan ilova, axborotlar tasvirlanishini tashkil qilish va foydalanilayotgan operatsion tizim turiga bog'liq bo'lmagan holda bir hil vizual natijani kafolatlaydi. Bundan tashqari, ushbu ilovalar ShK larda qanday bajarilsa, xuddi shunday ishchi stansiya va superkompyuterlarda ham bajariladi.
- *Qo'llashning osonligi.* OpenGL standarti mukammal o'ylangan tuzilmaga va tushunarli interfeysga ega bo'lib, boshqa grafik

kutubxonalardan foydalanishga nisbatan dastur kodi kamroq bo'lgan samarali ilovalarni yaratish imkonini beradi. Turli qurilmalar bilan moslikni ta'minlash uchun zaruriy funksiyalar kutubxona darajasida tashkillashtirilgan bo'lib, ilovalar ishlab chiqishni ancha osonlashtiradi.

Nazorat savollari:

1. OpenGL kutubxonasini yaratishdan ko'zlangan maqsad nima?
2. OpenGL kutubxonasi asoschisi?
3. Kutubxonada mavjud buyruqlar soni va ular qanday vazifalarni bajaradi?
4. OpenGL kutubxonasining o'ziga xos jihatlarini keltiring?
5. OpenGL kutubxonasidan qanday operatsion tizimlarda foydalanish mumkin?
6. Turli qurilmalar bilan ishlash qanday tashkillashtirilgan?

Tayanch iboralar: OpenGL, kutubxona, barqarorlik, ishonchlilik va uzatuvchanlik, qo'llashning osonligi, moslashish.

3.2. OpenGL asoslari

Asosiy imkoniyatlar

OpenGL imkoniyatlarini biz uning kutubxonasidagi funksiyalar orqali ta'riflaymiz. Barcha funksiyalarni beshta toifaga ajratish mumkin:

- *Primitivlarni tavsiflash funksiyasi* grafik nimtizimni aks ettirishga qodir bo'lgan, ierarxiyaning quyi darajasidagi obyektlarni belgilaydi. OpenGLda primitivlar sifatida nuqta, chiziq, ko'pburchak va boshqalar nazarda tutiladi.
- *Ranglar manbasini tavsiflash funksiyasi* uch o'lchovli sahnada joylashgan, ranglar manbasi parametri va holatini tavsiflash uchun xizmat qiladi.
- *Atributlarni tayinlash funksiyasi.* Atributlarni tayinlash yordamida dasturchi aks etadigan obyekt ekranda qanday ko'rinishga kelishini belgilaydi. Boshqacha so'z bilan aytganda, agar

ekranda *nima* hosil bo'lishi primitivlar yordamida belgilansa, unda atributlar ekranga chiqarish *usulini* belgilaydi. Atributlar sifatida OpenGL rang, material xususiyati, tekstura, yorug'lik parametrlarini berish imkonini beradi.

- *Vizuallashtirish funksiyasi* virtual fazoda kuzatuvchi (kamera obyektivi parametri) holatini belgilash imkonini beradi. Ushbu parametrlarni bilish, tizim nafaqat tasvirni to'g'ri qurishi, balki kuzatuv maydonidan tashqari bo'lgan obyektlarni ajratib qo'yishi ham mumkin.
- *Geometrik o'zgartirish funksiyalari* to'plami dasturchiga obyektlarni turli xil o'zgartirish – burish, ko'chirish, masshtablashtirishni bajarish imkonini beradi.

Shunday ekan, OpenGL qo'shimcha operatsiyalarni ham bajara olishi mumkin, masalan, chiziqlar va sirtlarni qurishda splaynlardan foydalanish, tasvirning ko'rinmas qismlarini olib tashlash, piksellar darajasidagi tasvirlar bilan ishlash va b.

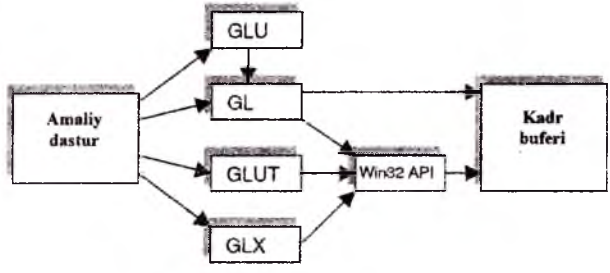
OpenGL interfeysi

OpenGL kutubxonalar to'plamidan tarkib topgan. Barcha bazaviy funksiyalar asosiy kutubxonada saqlanadi, belgilab qo'yish uchun keyinchalik biz *GL* qisqartmasidan foydalanamiz. Bundan tashqari, OpenGL o'zida bir qancha qo'shimcha kutubxonalarni qamrab oladi [2].

Ulardan birinchisi – *GL (GLU – GL Utility) utilit kutubxonasi.* Ushbu kutubxonaning barcha funksiyalari GL ning bazaviy funksiyalari orqali belgilanadi, masalan, keng tarqalgan geometrik primitivlar to'plami (kub, shar, silindr, disk), splaynlar qurish funksiyalari, matritsalar ustida qo'shimcha operatsiyalarni amalga oshirish (amallar bajarish) va x.k.

OpenGL da oynalar bilan ishlash yoki foydalanuvchi ma'lumot kiritishi uchun zarur hech qanday maxsus buyruqlar mavjud emas. Shu sabab foydalanuvchi bilan ishlash uchun va oynalar tizimi yordamida ma'lumotlarni ko'rsatish uchun maxsus uzatuvchi kutubxonalar yaratilgan, ya'ni tez-tez ishlatiladigan funksiyalarning foydalanuvchilar bilan o'zaro aloqasini ta'minlash uchun va oynali quyi tizimlar yordamida axborotlarni aks ettirish uchun. *GLUT (GL Utility Toolkit)* kutubxonasi bugungi kunda keng tarqalgan.

Umuman olganda, GLUT kutubxonasi OpenGL tarkibiga kirmaydi. Ammo, de facto barcha distributlari tarkibiga kiradi va turli platformalar uchun ishlay oladi. GLUT OpenGL ilovalarini yaratish uchun minimal darajadagi kerakli funksiyalar to'plamidan tashkil topgan. Funksional jihatdan GLX kutubxonasi nisbatan kamroq ommaviylashgan. Ushbu qo'llanmaga asosan GLUT kutubxonasi ko'riladi.



3.1-rasm. OpenGL kutubxonalarining tashkil etilishi.

Bundan tashqari, odatda, oynalar tizimi bilan ishlovchi funksiyalar uning amaliy dastur interfeysiga kiradi. Demak, OpenGLning ishlashini ta'minlovchi funksiyalar Win32 API va X Windowlar tarkibida mavjud. Yuqorida keltirilgan rasmda, Windows tizimida ishlaydigan kutubxonalar tizimlarini tashkil qilish sxemasi keltirilgan. OpenGLning boshqa versiyalarida ham tashkil qilish shunga o'xshash amalga oshiriladi.

OpenGL tuzilishi

OpenGL funksiyalari mijoz-server texnologiyasi asosida yaratilgan. Ilovalar mijoz vazifasini bajaradi – ular buyruqlarni ishlab chiqadilar, server OpenGL esa ularni qayta ishlaydi va bajaradi. Server o'zi mijoz joylashgan kompyutyerda joylashgan bo'lishi ham mumkin (masalan, dinamik yuklanuvchi kutubxona ko'rinishida – DLL), yoki boshqa kompyuterda (ushbu holda mashinalar o'rtasida ma'lumot almashinish uchun maxsus protokol qo'llaniladi).

GL bir qancha tanlangan rejimlar bo'yicha grafik *primitivlarni* kadr buferida chizadi va qayta ishlaydi. Har bir primitiv – bir nuqta,

kesma, ko'pburchak va boshqalar bo'lishi mumkin. Har bir rejim boshqasiga bog'liq bo'lmagan holda o'zgartirilishi mumkin. Primitivlarni aniqlash, rejimlarni tanlash va boshqa amallar kutubxona funksiyalarini chaqirish ko'rinishidagi *buyruqlar* orqali tavsiflanadi.

Primitivlar bir yoki bir nechta *uchlar* (vertex) to'plami yordamida aniqlanadi. Uch nuqta orqali ifodalanadi, kesma oxiri yoki ko'pburchak uchi. Har bir uch atributlar deb ataluvchi ma'lum bir ma'lumotlarga ega (koordinatalar, rang, normal, tekstura koordinatalari va x.k.). Ko'p hollarda uchlar bir biriga bog'liqsiz holda qayta ishlanadi.

OpenGL grafik tizimi tuzilishi jihatidan grafik obyektlarni ketma-ket qayta ishlashning bir necha bosqichlaridan iborat konveyer hisoblanadi.

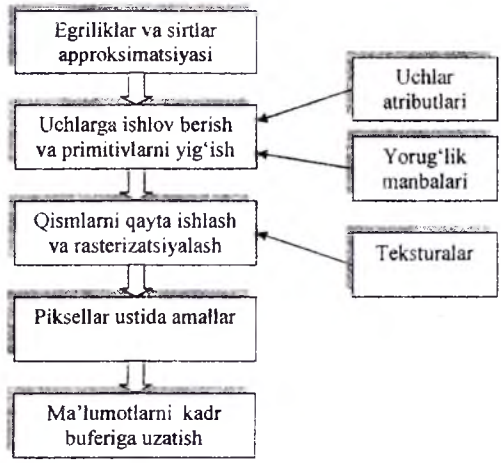
OpenGL buyruqlari har doim ularning kelishi tartibida amalga oshiriladi, biroq, ularning hosil qilinish effektidan oldin to'htalishlar bo'lib qolishi mumkin. Ko'p hollarda OpenGL aniq interfeysni taqdim etadi, ya'ni obyektning aniqlanishi uning kadr buferida vizualizatsiya bo'lishiga olib keladi.

Dasturchilar nuqtai nazarida, OpenGL – grafik qurilmalardan foydalanishni boshqarib turadigan buyruqlar to'plamidir. Agarda qurilma faqatgina adreslangan kadr buferidan tarkib topgan bo'lsa, u holda OpenGL markaziy prosessor resurslaridan foydalanish yordamida to'liq amalga oshirilishi kerak. Odatda grafik qurilma turli darajadagi tezlikni ta'minlaydi: chiziq va ko'pburchaklarni chiqarishdan geometrik ma'lumotlar ustida turli amallarni qo'llab murakkab grafik usullarni uskunaviy amalga oshirilishigacha.

OpenGL qurilma va foydalanuvchi darajalari o'rtasidagi qatlam hisoblanib, qurilmalar imkoniyatlaridan foydalangan holda turli platformalarda yagona interfeysni taqdim etish imkonini beradi.

Bundan tashqari, OpenGLni chekli avtomat sifatida qarash mumkin, uning holati joriy normal, rang, tekstura koordinatalari va boshqa atributlar, alomatlar qiymatlari hamda maxsus o'zgaruvchilarning ko'pgina qiymatlari bilan belgilanadi. Ushbu axborotlarning barchasi ekranga chiqariladigan figuralarni qurish uchun uchlar koordinatasi grafik tizimiga kirishda ishlatiladi.

Holatlarni almashtirish funksiyalarni chaqirish uchun ishlatiladigan buyruqlar orqali amalga oshiriladi.



3.2-rasm. OpenGL konveyerining funksionallashuvi.

Buyruqlar sintaksisi

GL buyruqlarining belgilanishi gl.h faylida joylashgan, uni qo'shish uchun quyidagini yozish zarur bo'ladi:

#include <gl/gl.h>

GLU kutubxonasi bilan ishlash uchun glu.h faylini qo'shish zarur. Ushbu kutubxona versiyalari qoida sifatida dasturlash tizimlari distributivlariga avtomatik o'rnatiladi, masalan, Microsoft Visual C++, DevC++ yoki Borland C++ .

Standart kutubxonalardan farqli ravishda, GLUT paketini alohida o'rnatish va qo'shish zarur. OpenGL bilan ishlash uchun dasturlash muhitini sozlash to'g'risidagi batafsil ma'lumot ilova C da berilgan.

GL kutubxonasining barcha buyruqlari (prosedura va funksiyalar) gl old qo'shimchasi bilan boshlanadi, barcha o'zgarmaslar - GL_ old qo'shimchasi bilan boshlanadi. GLU va GLUT

kutubxonalarining tegishli buyruqlari va o‘zgarmaslari glu (GLU_) va glut (GLUT_) old qo‘shimchalariga ega bo‘ladi.

Bundan tashqari, buyruqlar nomiga parametrlar soni va turi to‘g‘risidagi ma’lumotlarni o‘zida saqlaydigan suffikslar ham kiradi. OpenGL da buyruqlarning to‘liq nomi quyidagi ko‘rinishga ega:

type **glCommand_name[1 2 3 4][b s i f d ub us ui][v]**
(type1 *arg1*,...,typeN *argN*)

Nomlar bir qancha qismlardan tashkil topadi:

gl bu funksiya ko‘rsatilgan kutubxona nomi: OpenGL ning bazaviy funksiyalari uchun, GL, GLU, GLUT, GLAUX kutubxona funksiyalari, bular mos ravishda gl, glu, glut, aux.

Command_name buyruqlar nomi (proseduralar yoki funksiyalar)

[1 2 3 4] buyruqlar argumentlari soni

[b s i f d ub us ui] argument turi: b – GLbyte (C\C++ da char singari), i – GLint (butun), f – GLfloat (kasrli), s – GLshort (qisqa butun). d – GLdouble (ikkili aniqlikdagi kasrli), ub – GLubyte (belgisiz bayt), us – GLushort (belgisiz qisqa butun), ui – GLuint (belgisiz butun).

[v] ushbu belgining mavjudligi funksiya parametrlari sifatida belgilar massiviga yo‘nalish ishlatilishini ko‘rsatadi.

Kvadrat qavs ichidagi belgilar ba’zi nomlarda ishlatilmaydi. Masalan, glVertex2i() buyrug‘i GL kutubxonasida ko‘rsatilgan va parametrlari sifatida ikkita butun sonni ishlatadi, glColor3fv() buyrug‘i esa uchta haqiqiy sondan iborat massivga ko‘rsatkich parametr sifatida ishlatiladi.

Ilovaga misol

OpenGL dan foydalanib dastur tuzishda, dastlab tasvirni ko‘rsatuvchi oyna aniqlab olinadi. Shundan so‘ng OpenGL konteksi (mijoz) yaratiladi va shu oyna bilan bog‘lanadi. Keyingi qadamlarda dasturchi erkin holda OpenGL API buyruqlari va operatsiyalaridan foydalanishi mumkin.

Quyida GLUT kutubxonasidan foydalanib yozilgan kichik bir dastur matni keltirilgan.

Ushbu dastur oyna markazida qizil rangli kvadrat chizadi. Shu kichik dastur orqali ham OpenGL yordamida dastur tuzishning mohiyatini tushunib olish mumkin.

```
#include <stdlib.h>
/* GLUT kutubxonasini bog'lash*/
/* oynaning dastlabki eni va balandligi (o'lchamlari)*/
GLint Width = 512, Height = 512;
/* kubning hajmi */
const int CubeSize = 200;
/* ushbu funksiya ekranga chiqarilayotgan barcha
ma'lumotlarni boshqaradi */
void Display(void)
{
int left, right, top, bottom;
left  = (Width - CubeSize) / 2;
right = left + CubeSize;
bottom = (Height - CubeSize) / 2;
top = bottom + CubeSize;
glClearColor(0, 0, 0, 1);
glClear(GL_COLOR_BUFFER_BIT);
glColor3ub(255,0,0);
glBegin(GL_QUADS);
glVertex2f(left,bottom);
glVertex2f(left,top);
glVertex2f(right,top);
glVertex2f(right,bottom);
glEnd();
glFinish();
}

/* ushbu funksiya oyna o'lchami o'zgargan holatini chaqirish
uchun chaqiriladi */
void Reshape(GLint w, GLint h)
{
Width = w;          Height = h;
```

```
    /* ko'rsatish sohasi o'lchamini o'rnatamiz */
    glViewport(0, 0, w, h);     // x, y, balandlik, kenglik

    /* ortografik proeksiya */
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0, w, 0, h, -1.0, 1.0); // chap, o'ng, quyi, yuqori, yaqin,
uzoq
    }

    /* klaviaturadan kelgan ma'lumotlarni qayta ishlovchi funksiya
*/
    void Keyboard( unsigned char key, int x, int y )
    {
    #define ESCAPE '\033'
    if( key == ESCAPE )
    exit(0);            }

    /* ilovaning asosiy sikli */
    main(int argc, char *argv[])
    {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGB);
    glutInitWindowSize(Width, Height);
    glutInitWindowPosition(10,10);
    glutCreateWindow("Qizil maydonga misol");
    glutDisplayFunc(Display);
    glutReshapeFunc(Reshape);
    glutKeyboardFunc(Keyboard);
    glutMainLoop();
    }
```

Ushbu dastur hajmi kichik bo'lishiga qaramay OpenGL va GLUT kutubxonalarini qo'llaydigan, istalgan tizimda ishlaydigan va kompilyatsiya qilinadigan to'liq tugallangan dastur.

GLUT kutubxonasi foydalanuvchi bilan teskari aloqa (*callback function*) deb nomlanuvchi funksiya yordamida o'zaro aloqani

quvvatlaydi. Agarda foydalanuvchi sichqonchani siljitsa, klaviatura tugmasini bossa yoki oyna o'lchamini o'zgartirsa, hodisa yuz beradi va foydalanuvchining tegishli (hodisalarga ishlov berish (teskari aloqaga ega funksiya)) funksiyasi chaqiriladi.

Yuqoridagi misolda keltirilgan main funksiyasini batafsil ko'rib chiqamiz. U uchta qismdan tarkib topgan – OpenGL chizishi kerak bo'lgan oynani initsiallash (faollashtirish), teskari aloqali funksiyalarni va hodisalarga ishlov berishning asosiy davrini sozlash.

Oynani initsiallash (faollashtirish) kadrning tegishli buferlari, boshlang'ich holati va oyna o'lchami, shuningdek, oyna sarlavhasini sozlashdan tarkib topadi.

glutInit (&argc, argv) funksiyasi GLUT kutubxonasining o'zini dastlabki initsiallashuvga tayyorlaydi.

glutInitDisplayMode (GLUT_RGB) buyrug'i kadr buferini initsiallashtiradi va to'liq rangli (palitrasiz) RGB rejimiga to'g'rilaydi.

glutInitWindowSize (Width, Height) oynaning boshlang'ich o'lchamini tayinlash uchun ishlatiladi.

glutCreateWindow ("Qizil maydonga misol") oyna sarlavhasini kiritadi va oynani ekranda vizuallashtiradi.

Shunda quyidagi buyruqlar:

glutDisplayFunc (Display);

glutReshapeFunc (Reshape);

glutKeyboardFunc (Keyboard);

Display(), Reshape() va Keyboard() funksiyalari oynani qayta chizish, oyna o'lchamlarini o'zgartirish, klaviatura klavishini bosganda chaqiriladigan funksiyalar sifatida qayd qiladi.

Barcha hodisalarni nazorat qilish va kerakli funksiyalarni chaqirish glutMainLoop () funksiyasining cheksiz davri ichida bajariladi.

Ko'rishimiz mumkinki, GLUT kutubxonasi OpenGL tarkibiga kirmaydi, ya'ni OpenGL va oynali nimtizimning orasidagi keltirilgan eng kichik interfeysdir. OpenGL ilovasi aniq bir platformalar uchun maxsus API (Win32, X Window va b.) yordamida yozilishi mumkin, ular esa o'z navbatida yanada kengroq imkoniyatlarni taqdim etadi. GLUT kutubxonasi bilan ishlash *A* ilovada batafsil berilgan.

OpenGL buyruqlarini chaqirish hodisalarni qayta ishlovchida sodir bo‘ladi. Ular keyingi bo‘limlarda kengroq qarab chiqiladi. Hozir esa, ekranda chizishga javob beradigan kodga ega bo‘lgan Display funksiyasiga e’tibor qaratamiz.

Display funksiyasiga tegishli quyidagi buyruqlar ketma-ketligi:

```
glClearColor(0, 0, 0, 1);
glClear(GL_COLOR_BUFFER_BIT);

glColor3ub(255,0,0);
glBegin(GL_QUADS);
glVertex2f(left,bottom);
glVertex2f(left,top);
glVertex2f(right,top);
glVertex2f(right,bottom);
glEnd();
```

oynani tozalaydi va rang hamda to‘rtta burchak uchlari koordinatalarini berib ekranga kvadrat chiqaradi.

D.1 ilovasida sichqoncha tugmasini bosganda ekranda har xil rangdagi tasodifiy to‘g‘ri to‘rtburchakni chizadigan yana bir murakkab bo‘lmagan dastur keltirilgan.

Nazorat savollari:

1. Standart grafik kutubxonalar yaratish zarurati nimadan iborat?
2. OpenGL kutubxonasi funksiyalarini tavsiflang?
3. Konveyerni tashkil qilish va OpenGL kutubxonasi tuzilishini qisqacha ifodalang?
4. Kutubxona buyruqlari (funksiyalari) toifalarini keltiring?
5. OpenGL kutubxonasiga qanday kutubxonalar qo’shimcha o’rnatiladi?
6. Faqatgina parametrlar turi bilan farq qiluvchi OpenGL buyruqlarining turlicha variantlari nima uchun kerak?
7. Nima sababdan OpenGLni tashkil etish ko‘pincha chekli avtomat bilan qiyoslanadi?

Tayanch iboralar: Primitivlar, ranglar manbasi, atributlarni tayinlash, vizuallashtirish, geometrik o'zgartirish, OpenGL ilovalari, GL, GLU, GLUT, GLX, konveyer.

3.3. Geometrik obyektlarni chizish

Tasvirlarni yangilash jarayoni

Qoida sifatida, OpenGLni ishlatadigan dasturning vazifasi uch o'lchamli tasvirni qayta ishlash va kadr buferida interaktiv tasvirlash hisoblanadi. Tasvir kuzatuvchining joriy holatini aniqlaydigan uch o'lchamli obyektlar to'plami, yorug'lik manbai va virtual kameradan tashkil topadi.

Odatda OpenGL ilovasi uzluksiz siklda oynada tasvirni yangilash funksiyasini chaqiradi. Ushbu funksiyada OpenGLning asosiy buyruqlarini chaqirish joriy qilingan. Agar GLUT kutubxonasi ishlatilsa, unda bu glutDisplayFunc() chaqiruvi bilan qayd qilingan teskari aloqali funksiya bo'ladi. GLUT bu funksiyani oyna tarkibini qaytadan chizish kerakligi to'g'risida operatsion tizim ilovaga ma'lumot berganda bu funksiyani chaqiradi. Hosil qilinadigan tasvir ham statik, ham animatsiya ko'rinishida bo'lishi mumkin, ya'ni vaqt bo'yicha o'zgaradigan parametrlarga bog'liq bo'lganda. Bu holatda yangilash funksiyasini mustaqil ravishda chaqirish maqsadga muvofiq. Misol uchun, glutPostRedisplay () buyrug'i yordamida. Yanada batafsil ma'lumot *A* ilovada keltirilgan.

Tasvirni yangilashning tipik funksiyasi qanday vazifalarni bajarishini ko'rib chiqamiz. Qoida sifatida, u uch bosqichdan tarkib topadi:

1. OpenGL buferlarini tozalash;
2. Kuzatuvchining holatini o'rnatish;
3. Geometrik obyektlarni chizish va o'zgartirish.

Buferni tozalash quyidagi buyruqlar yordamida amalga oshiriladi:

void glClearColor (clampf *r*, clampf *g*, clampf *b*, clampf *a*) // *(clamp-fiksator)*

void glClear (bitfield *buf*) *///(bitfield - bitovoe pole)*

glClearColor buyrug'i rangni o'rnatadi, ya'ni kadr buferi to'ldiriladi. Buyruqning dastlabki uchta parametri ranglarning R, G

va B komponentlarini tayinlaydi va ular [0,1] kesma oralig'ida bo'lishi kerak. To'rtinchi parametr alfa kompetentani beradi. Qoidaga ko'ra, u 1 ga teng. Shart berilmaganda rang – qora (0,0,0,1).

glClear buyrug'i buferni tozalaydi, *buf* parametri tegishli bufer o'zgarmaslari kombinatsiyasini belgilaydi. Odatdagi dastur bufer rangi va chuqurligini tozalash uchun

glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

buyrug'ini chaqiradi.

Kuzatuvchi holatini o'rnatish va uch o'lchovli obyektlarni o'zgartirish (burish, ko'chirish va b.), almashtirish matritsalarini tayinlash yordamida nazorat qilinadi. Obyektlarni o'zgartirish va virtual kamera holatini sozlash keyingi bo'limlarda keltirilgan.

Diqqatimizni sahnada joylashgan obyektlar tavsifini OpenGL ga uzatishga qaratamiz. OpenGL ning har bir obyekti primitivlar to'plamidan tashkil topgan.

Uchlar va primitivlar

Uchlar (nuqta) OpenGLning grafik primitivlari hisoblanadi va nuqtani, kesmaning oxiri, ko'pburchakning burchagi va boshqalarni tavsiflaydi. Qolgan barcha primitivlar uchlarni tayinlash orqali shakllantiriladi. Masalan, kesma ikkita uchlar orqali ifodalanadi.

Har bir uch bilan birga uning *atributlari* ham mavjud. Asosiy atribut sifatida uchning fazodagi holati, uchning rangi va vektor normallarini olish mumkin.

Uchlarning fazodagi holati

Uchlarning holati ikki, uch yoki to'rt o'lchovli (bir jinsli koordinatalar) fazoda uning koordinatalarini tayinlash orqali belgilanadi. Bu glVertex* buyrug'ini bir qancha variantlari yordamida amalga oshiriladi:

void **glVertex [2 3 4][s i f d]** (type *coords*)

void **glVertex[2 3 4][s i f d]v** (type **coords*)

Har bir buyruq uchlarning to'rtta koordinatasini tayinlaydi: x, y, z, w. glVertex2* buyrug'i x va y qiymatlarini qabul qiladi. Bunday holatda z koordinatasi 0 ga teng, w koordinatasi – 1 ga teng.

Vertex3* buyrug'i x, y, z koordinatalarini qabul qiladi va w koordinatasiga 1 qiymatini kiritadi.

Vertex4* barcha – to'rtta koordinatalarni tayinlash imkonini beradi.

Uchlar rangi, normallar va teksturlar koordinatalarini birlashtirish uchun mavjud ma'lumotlarning joriy qiymatlari ishlatiladi. Ushbu qiymatlar ixtiyoriy vaqtda tegishli buyruqni chaqirish yordamida o'zgartirilishi mumkin.

Uchlar rangi

Uchlarning joriy rangini tayinlash uchun quyidagi buyruqlar ishlatiladi:

void **glColor[3 4][b s i f]** (GLtype *components*)

void **glColor[3 4][b s i f]v** (GLtype *components*)

Dastlabki uchta parametr rangning R, G, B komponentlarini tayinlaydi, oxirgi parametr notiniqlik koeffitsiyenti (alfa-qism deb ataluvchi)ni belgilaydi. Agarda buyruqning nomlanishida 'f' (float) tipi ko'rsatilgan bo'lsa, unda barcha parametrlar qiymatlari [0,1] kesma oralig'ida joylashishi kerak. Bunday holda alfa-qism qiymati 1.0 ga tenglashtirilishi o'rnatiladi. 'ub' (unsigned byte) tipi qiymat [0,255] kesmada joylashishi kerakligini ko'zda tutadi.

Uchlarga turli xil ranglarni tayinlash mumkin, agarda mos rejim yoqilgan bo'lsa. Unda primitiv sirti bo'ylab ranglarning chiziqli interpolyatsiyasi amalga oshiriladi.

Interpolyatsiya jarayonini boshqarish uchun quyidagi buyruqdan foydalaniladi

void **glShadeModel** (GLenum *mode*)

chaqiruv **GL_SMOOTH** parametr bilan berilsa, interpolyatsiya qo'shiladi (tanlangan parametr asosida), **GL_FLAT** parametr berilganda esa ajratadi, o'chiradi.

Normal

Normallarni ifodalash uchun quyidagi buyruqlardan foydalaniladi

void **glNormal3[b s i f d]** (type *coords*)

void **glNormal3[b s i f d]v** (type *coords*)

Yorug'lik hisobini to'g'ri olish uchun normal vektori yagona uzunlikka ega bo'lishi zarur. glEnable(GL_NORMALIZE) buyrug'i maxsus buyruqlarni qo'shishi mumkin, bunda tayinlanayotgan normal avtomatik normallashadi.

Avtomatik normallash rejimi modelli kengayish-torayish almashtirilishi ishlatilganda yoqilgan bo'lishi kerak, chunki bu holda normal uzunligi model-matritsasiga ko'paytirilganda o'zgaradi. Ammo, ushbu rejimni joriy qilish OpenGLning vizuallashtirish mexanizmi ishini sekinlashtiradi, xuddi shunday vektorlarni normallash hisoblashning sezilarli qiyinchiliklariga olib keladi (kvadrat ildiz olish va b.). Shuning uchun birdaniga yagona normalni berish ma'qul.

Eslatib o'tish kerakki,

void **glEnable** (GLenum *mode*)

void **glDisable** (GLenum *mode*)

buyruqlari OpenGL konveyerining u yoki bu ish rejimini o'chirilishi yoki yoqilishini ta'minlaydi. Ushbu buyruqlar ko'p hollarda qo'llaniladi va ularning parametrlari har bir aniq holatlar uchun qaraladi.

Operatorli qavslar glBegin / glEnd

Biz yuqorida bitta uchning atributlarini tayinlashni ko'rib chiqdik. Ammo, grafik primitiv atributlarini tayinlash uchun, uchning bitta koordinatasi etarli emas. Ushbu uchlarni zaruriy xususiyatini belgilovchi bir butunlikka birlashtirish kerak. Buning uchun OpenGLda operatorli qavs deb ataluvchi maxsus buyruqlarni chaqirishda xizmat qiluvchilardan foydalaniladi. Primitivlarni belgilash yoki primitivlar ketma-ketligi

void **glBegin** (GLenum *mode*);

void **glEnd** (void);

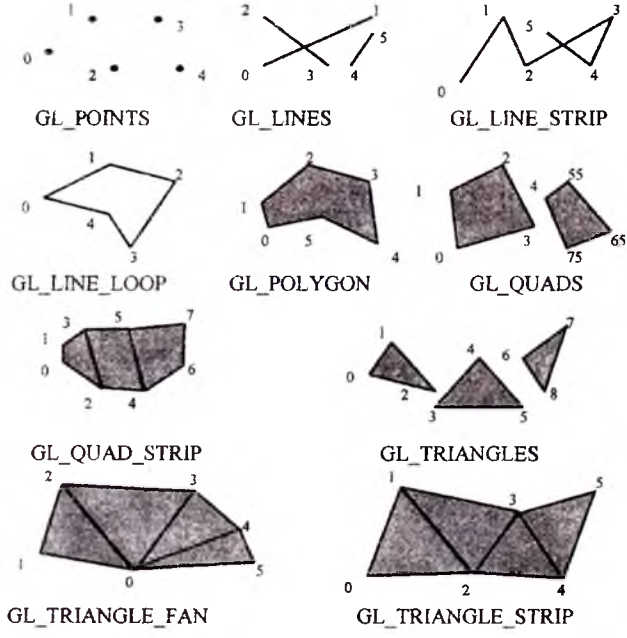
buyruqlarini chaqirish oralig'ida olib boriladi.

Mode parametri primitivning tipini belgilaydi, qaysiki ichkarida tayinlanadi va quyidagi qiymatlarni qabul qilishi mumkin:

GL_POINTS har bir uchlar bir qancha nuqtalar koordinatalarini belgilaydi.

GL_LINES har bir alohida uchlar juftligi kesmani belgilaydi. Agarda toq sonli uchlar ko'rsatilsa, u holda oxirgi uch e'tiborga olinmaydi.

GL_LINE_STRIP har bir keyingi uch oldingisi bilan birgalikda kesmani belgilaydi



3.3-rasm. OpenGL primitivlari.

GL_LINE_LOOP oldingi primitivdan farqli jihati shuki, oxirgi kesma so'ngi va birinchi uchni berk to'g'ri chiziq timsolida belgilaydi.

GL_TRIANGLES har bir alohida uchta uch uchburchakni belgilaydi. Agarda uchta sonli uchlar karrali berilmasa, unda oxirgi uchlar e'tiborga olinmaydi.

GL_TRIANGLE_STRIP har bir keyingi uch oldingi ikkitasi bilan birgalikda uchburchakni belgilaydi.

GL_TRIANGLE_FAN uchburchaklar birinchi uch va har bir keyingi juft uchlar bilan beriladi (juftliklar kesishmaydi).

GL_QUADS har bir alohida to'rtta uch to'rtburchakni belgilaydi; agarda to'rtta sonli uchlar karrali berilmasa, unda oxirgi uchlar e'tiborga olinmaydi.

GL_QUAD_STRIP *n* nomerli to'rtburchak 2n-1, 2n, 2n+2, 2n+1 nomerli uchlar bilan belgilanadi.

GL_POLYGON qavariq ko'pburchakning uchlarini ketma-ket berish.

Masalan, uchlari turli rangda bo'lgan uchburchakni chizish uchun, quyidagilarni yozish etarli:

```
GLfloat BlueCol[3] = {0,0,1};
glBegin(GL_TRIANGLES);
glColor3f(1.0, 0.0, 0.0);          /* qizil */
glVertex3f(10.0, 10.0, 0.0);
glColor3ub(0,255,0);       /* yashil */
glVertex2i(100, 100);
glColor3fv(BlueCol);       /* ko'k */
glVertex3f(100.0, 100.0, 0.0);
glEnd();
```

Qoida sifatida, primitivlarning har xil turlari turlicha platformalarda turlicha vizuallashish tezligiga ega. Unumdorlikni oshirishda, serverga uzatish uchun kam sonli axborotlarni talab etuvchi, GL_TRIANGLE_STRIP, GL_QUAD_STRIP, GL_TRIAGLE_FAN singari primitivlardan foydalanish afzalroqdir.

Mazkur ko'pburchaklarni tayinlashdan tashqari, ularni ekranda akslantirish usullarini ham berish mumkin.

Ammo dastlab old va orqa yoq tushunchalarini belgilab olish lozim.

Yoq deganda ko'pburchakning tomonlaridan biri tushuniladi, va odatda soat strelkasiga teskari aylanuvchi uchlar joylashgan tomon old hisoblanadi. Old yoqning uchlari aylanish yo'nalishini quyidagi buyruqni chaqirish orqali o'zgartirishi mumkin

void **glFrontFace** (GLenum *mode*)

mode parametrining qiymati GL_CW (clockwise) ga teng, qiymatni odatdagi holatiga qaytarish uchun GL_CCW (counter-clockwise) ni ko'rsatish kifoya.

Ko'pburchakni tasvirlash usulini o'zgartirish uchun quyidagi buyruq ishlatiladi

void **glPolygonMode** (GLenum *face*, Glenum *mode*)

mode parametri ko'pburchak qanday akslanishini belgilaydi, *face* parametri ko'pburchakning tipini o'rnatishda qo'llaniladi va quyidagi qiymatlar ham qabul qilinishi mumkin:

GL_FRONT	ko'rinadigan tomon (old yoq) uchun
GL_BACK	ko'rinmas tomon (orqa yoq) uchun
GL_FRONT_AND_BACK	barcha tomon (yoq)lar uchun

mode parametri quyidagilarga teng bo'lishi mumkin:

GL_POINT ko'pburchakning faqat uchlari tasvirlanadi.

GL_LINE ko'pburchaklar qismlar to'plamida ifodalanadi.

GL_FILL yorug'likni hisobga olib ko'pburchaklar joriy rang bilan bo'yab chiqiladi, va bu odatdagi rejim sifatida qaraladi.

Shu bilan birga, ekranda qanday chegara tasvirlanishini ko'rsatish ham mumkin. Buning uchun dastlab **glEnable** (GL_CULL_FACE) buyrug'ini chaqirish rejimi o'rnatilishi kerak, shundan so'ng quyidagi buyruqlar yordamida tasvirlanuvchi tomon tipi tanlanadi:

void **glCullFace** (GLenum *mode*)

Funksiya **GL_FRONT** parametri bilan chaqirilsa, tasvirdan barcha ko'rinadigan tomonlar olib tashlanadi, **GL_BACK** – parametr bilan esa teskarisi (tanlov asosida o'rnatiladi).

Ko'rilgan standart primitivlardan tashqari GLU va GLUT kutubxonalarida yanada murakkabroq figuralar mavjud, jumladan, sfera, silindr, disk (GLU da) va sfera, kub, konus, tor, tetraedr, dodekaedr, ikosaedr, oktaedr va choynak (GLUT da). Teksturalarni avtomatik joylashtirish faqat GLU kutubxonasi figuralari uchun ko'rilgan (OpenGL da tekstura yaratish 3.6 paragrafda ko'riladi).

Misalan, sfera yoki silindr chizish uchun, avvalo quyidagi:

GLUquadricObj* **gluNewQuadric** (void)

buyruq yordamida maxsus tipdagi GLUquadricObj obyekti yaratiladi.

Undan so‘ng kerakli buyruq chaqiriladi:

void **gluSphere** (GLUquadricObj * *qobj*, GLdouble *radius*, GLint *slices*, GLint *stacks*)

void **gluCylinder** (GLUquadricObj * *qobj*,
GLdouble *baseRadius*,
GLdouble *topRadius*,
GLdouble *height*, GLint *slices*,
GLint *stacks*)

bu yerda *slices* parametri z o‘qi atrofida siniq egri chiziqlar sonini beradi, *stacks* – esa z o‘qi atrofi bo‘ylab bo‘linishlar sonini beradi.

Primitivlarni qurishning ushbu va boshqa buyruqlari haqida yanada aniqroq ma’lumotlar *B* ilovada keltirilgan.

Displeyli ro‘yxatlar

Agar biz bitta buyruqlar guruhiga bir necha marotaba murojaat qilsak, unda ularni displey ro‘yxati (display list) ga birlashtirish va zarur bo‘lganda uni chaqirish mumkin. Yangi displey ro‘yxatini hosil qilish uchun unga kirishi kerak bo‘lgan barcha buyruqlarni quyidagi operator qavslari orasiga joylashtirish kerak:

void **glNewList** (GLuint *list*, GLenum *mode*)
void **glEndList** ( )

Ro‘yxatlarni ajratish uchun butun musbat sonlardan foydalaniladi, berilgan ro‘yxat *list* parametri bilan beriladi, *mode* parametri esa ro‘yxatga kirgan buyruqni qayta ishlash rejimini tavsiflaydi:

GL_COMPILE buyruq bajarilmasdan oldin ro‘yxatga yoziladi

GL_COMPILE_AND_EXECUTE buyruqlar oldin bajarilib, keyin ro‘yxatga yoziladi

Ro'yxat tuzilgandan keyin, *list* parametrda kerakli ro'yxatning identifikatorini ko'rsatib, uni quyidagi buyruq bilan chaqirib olish mumkin:

void **glCallList** (GLuint *list*)

Birdaniga bir necha ro'yxatni chaqirib olish uchun, quyidagi buyruqdan foydalanish mumkin:

void **glCallLists** (GLsizei *n*, GLenum *type*, const GLvoid **lists*)

Bu buyruq identifikatorlari *lists* massivida bo'lgan *n* ro'yxatlarni chaqirib oladi. *lists* massivi elementlarining turi *type* parametrda ko'rsatiladi. Bular quyidagi turlar bo'lishi mumkin: **GL_BYTE**, **GL_UNSIGNED_BYTE**, **GL_ShORT**, **GL_INT**, **GL_UNSIGNED_INT.**

Ro'yxatlarni o'chirish uchun

void **glDeleteLists** (GLint *list*, GLsizei *range*)

buyrug'i ishlatiladi. Bu buyruq ID identifikatorlarga ega bo'lgan *list* ≤ ID ≤ *list*+*range*-1 diapazonida ro'yxatlarni o'chiradi.

Masalan:

```
glNewList(1, GL_COMPILE);
glBegin(GL_TRIANGLES);
glVertex3f(1.0f, 1.0f, 1.0f);
glVertex3f(10.0f, 1.0f, 1.0f);
glVertex3f(10.0f, 10.0f, 1.0f);
glEnd();
glEndList()
...
glCallList(1);
```

Displey ro'yxatlari optimal, kompilyatsiya qilingan holatda server xotirasida saqlanadi. Bu shakl primitivlarni maksimal tezlikda chizishga imkon beradi. Shu bilan birga, katta hajmli ma'lumotlar xotirani ko'p qismini egallaydilar, bu o'z navbatida, unumdorlikning pasayishiga olib keladi. Bunday katta hajmlarni (bir necha o'n ming primitivlar) uchlar massivlari bilan chizish qulay.

Uchlar massivlari

Agar uchlar soni ko'p bo'lsa va ularning har birini glVertex*() buyrug'i bilan chaqirmaslik uchun void **glVertexPointer** (GLint *size*, GLenum *type*, GLsizei *stride*, void* *ptr)* buyrug'idan foydalanib, massivlarga birlashtirish qulayroq. Bu buyruq uchlarning koordinatalari va saqlash uslubini belgilaydi. Bu yerda *size* uchlarning koordinatalar sonini ko'rsatadi (2,3,4 bo'lishi mumkin), type ma'lumotlarning turini belgilaydi (**GL_ShORT, GL_INT, GL_FLOAT, GL_DOUBLE**). Ba'zan bitta massivda boshqa uchlarning atributlarini saqlash qulayroq, shunda *stride* parametri bitta uchning koordinatasidan ikkinchisining koordinatasigacha ko'chishini belgilaydi. Agar *stride* nolga teng bo'lsa, bu degani koordinatalar ketma ket joylashgan. *Ptr* parametrida ma'lumotlar joylashgan manzil ko'rsatiladi.

Shunga o'xshab, quyidagi buyruqlardan foydalanib normal, ranglar va uchlarni boshqa atributlarining massivlarini aniqlash mumkin:

void **glNormalPointer** (GLenum *type*, GLsizei *stride*, void **pointer*)

void **glColorPointer** (GLint *size*, GLenum *type*, GLsizei *stride*, void **pointer*)

Bu massivlarni keyinchalik ham ishlatish uchun quyidagi buyruqni chaqirish kerak:

void **glEnableClientState** (GLenum *array*)

shu buyruq bilan keyingi parametrlar qo'llaniladi:

GL_VERTEX_ARRAY, GL_NORMAL_ARRAY, GL_COLOR_ARRAY.

Massiv bilan ishlashni tugatgandan so'ng void **glDisableClientState** (GLenum *array*) buyrug'ini qo'llaymiz.

Massivning ichini qurish uchun

void **glArrayElement** (GLint *index*)

buyrug'i ishlatiladi.

Bu buyruq *index* nomerga ega massiv elementlaridan foydalanib uchlarning atributlarini OpenGL ga yuboradi. Uning o'rniga odatta

void **glDrawArrays** (GLenum *mode*, GLint *first*, GLsizei *count*)

buyrug'i ishlatiladi. U *mode* parametri bilan aniqlanadigan *count* primitivlarni chizadi. Bunda u *first* dan *first+count*-1 gacha indeksli massiv elementlaridan foydalanadi. Bu glArrayElement() buyrug'ini chaqirish bilan teng.

Agarda uch bir necha massivga kirsa, uning koordinatalarini qaytarish o'rniga massivda indeksini qo'llash qulayroq.

Buning uchun

void **glDrawElements** (GLenum *mode*, GLsizei *count*. GLenum *type*, void **indices*)

buyrug'i ishlatiladi, bu yerda *indices*-uchlarning raqamlar massivi. *Type* bu massivning elementlarining turini belgilaydi **GL_UNSIGNED_BYTE**, **GL_UNSIGNED_ShORT**, **GL_UNSIGNED_INT**, *count* esa ularning sonini belgilaydi.

Massivlarning qo'llanishi OpenGL serverига ma'lumotlarning jo'natilishini optimal holatga keltiradi va shu bilan birga uch o'lchamli sahnani chizishni tezlashtiradi. Bunday uslub primitivlarni aniqlash uchun juda tez va katta hajmli ma'lumotlarni vizualizatsiyalash uchun juda qulay.

Nazorat savollari:

1. Teskari chaqiruv funksiyasi nima va OpenGL bilan ishlash uchun teskari chaqiruv funksiyasidan qanday foydalanish mumkin?
2. Tasvirlarni yangilash funksiyasi nima uchun kerak va qanday ish bajaradi?
3. OpenGL da primitiv degani nima?
4. Atribut nima? OpenGL da sizga ma'lum bo'lgan uchlar atributlarini keltiring?
5. OpenGL dagi atomar primitiv deb nimaga aytiladi? Qanday turdagi primitivlarni bilasiz?
6. glEnable/glDisable buyruqlari OpenGL da nima uchun ishlatiladi?
7. Operatorli qavslar nima va ular OpenGL da qanday maqsadda ishlatiladi?
8. Displey ro'yxatlari nima? Ro'yxatni qanday belgilash va uni tasvirlashga qanday chaqirish mumkin?

9. Massivlar bilan ishlashni tushuntirib bering va displey ro'yxatlaridan farqlanishini ayting?

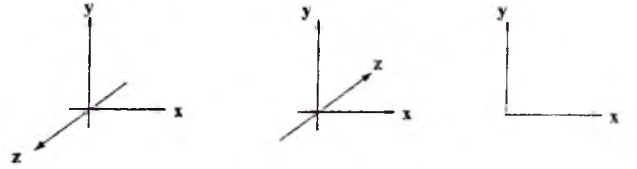
10. glDrawElements () buyrug'ining ishlashini tushuntiring?

Tayanch iboralar: Tasvirni yangilash, virtual kamera, buferni tozalash, kuzatuvchining holati, geometrik obyektlar chizish, uchlarning fazodagi holati, uchlar rangi, uchlar massivlari, primitivlar.

3.4. Obyektlarni o'zgartirish

OpenGL da asosan uchta koordinalar sistemasidan foydalaniladi: chap tomonlama, o'ng tomonlama va oynali. Birinchi ikkita sistema uch o'lchovli hisoblanadi va bir-biridan *z* o'qining yo'nalishi bilan farqlanadi: o'ng tomonlamada u kuzatuvchiga yo'naltiriladi, chap tomonlamada esa ekran orqasiga yo'naltiriladi. *x* o'qi kuzatuvchiga nisbatan o'ng tomonga, *y* o'qi yuqoriga yo'naladi.

Chap tomonli sistema gluPerspective(), glOrtho() buyruqlari parametrlari qiymatlarini berish uchun ishlatiladi. O'ng tomonli koordinatalar sistemasi qolgan barcha holatlar uchun ishlatiladi. Uch o'lchovli axborotlarni tasvirlash ikki o'lchovli oynali koordinatalar sistemasida olib boriladi.



a) o'ng tomonlama sistema b) chap tomonlama sistema c) oynali sistema

3.4-rasm. OpenGL da koordinatalar sistemasi.

Shuni alohida ta'kidlash lozimki, OpenGL matritsalar bilan murakkab hisoblashlarni olib borish yo'lida o'ng va chap koordinatalar sistemasini modellashtirish imkonini beradi. OpenGL ning asosiy koordinatalar sistemasi o'ng tomonlama sistema hisoblanadi.

Matritsalar bilan ishlash.

OpenGLda sahna obyektlarini turlicha o'zgartirishni ko'rsatish uchun matritsalar ustida operatsiyalardan foydalaniladi, buning uchun matritsalar uch turga ajratiladi: modelli-tasvir, proeksiyalash matritsasi va teksturalash matritsasi. Ularning barchasi 4x4 o'lchamga ega. Tasvirli matritsa obyektni parallel ko'chirish, masshtabni o'zgartirish va burish singari eng yuqori koordinatalarda o'zgarishni belgilaydi. Proeksiyalash matritsasi, uch o'lchovli obyektning ekran tekisligiga (oynali koordinatalarda) proyeksiyalanishini belgilaydi, teksturalash matritsasi obyekt teksturalar bilan qoplanishini belgilaydi.

Koordinatani matritsaga ko'paytirish, koordinatani belgilovchi (qoida sifatida, bu buyruq glVertex*) OpenGLning tegishli buyrug'ini chaqirish vaqtida sodir bo'ladi.

Matritsani qanday o'zgartirish kerakligiga qarab zaruriy buyruqlar ishlatiladi:

void **glMatrixMode** (GLenum *mode*)

mode parametrli qiymatga ega bo'lgan chaqiruv **GL_MODELVIEW, GL_PROJECTION,** yoki **GL_TEXTURE** larga teng bo'lib, mos holda modelli tasvir matritsasi, proeksiyalash matritsasi yoki teksturalash matritsalari bilan ishlash rejimini qamrab oladi. U yoki bu tipdagi matritsani tayinlash buyrug'ini chaqirish uchun, avvalo tegishli rejimni o'rnatib olish zarur.

Joriy tipdagi matritsa elementlarini aniqlash uchun quyidagi buyruq chaqiriladi:

void **glLoadMatrix [f d]** (GLtype **m*)

bu yerda *m* buyruqning nomlanishiga mos holda float yoki double tipidagi 16 elementli massivni bildiradi. Bunday holda dastlab unda matritsaning birinchi ustuni, so'ngra ikkinchi, uchinchi va to'rtinchi ustuni yozilishi kerak.

void **glLoadIdentity** (void) buyrug'i joriy matritsani birlik matritsaga o'zgartiradi.

Ko'p hollarda joriy matritsa elementlarini saqlash talab qilinadi, buning uchun quyidagi buyruqlar ishlatiladi.

void **glPushMatrix** (void)

void **glPopMatrix** (void)

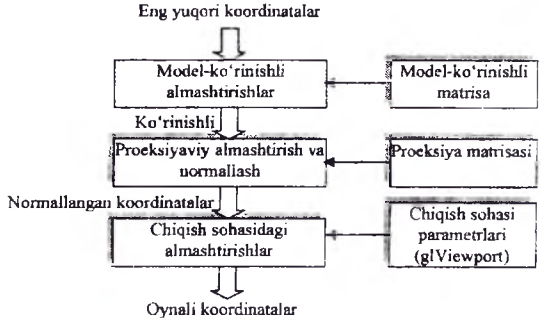
Ular joriy matritsani birlashmasidan yozib oladi va qayta tiklaydi, bunda har bir matritsa o'zining birlashmasiga ega. Model-ko'rinishli matritsalar uchun uning chuqurligi minimum 32 ga teng, boshqalar uchun minimum 2 ga teng.

Joriy matritsani boshqa matritsaga ko'paytirish uchun quyidagi buyruq ishlatiladi:

void **glMultMatrix [f d]** (GLtype **m*)

bunda *m* parametr 4x4 o'lchamli matritsani berishi kerak. Agar joriy matritsani *M* deb, yuboriladigan matritsani *T* deb belgilasak, unda glMultMatrix buyrug'i bajarilganda *M* * *T* joriy matritsa bo'lib qoladi. Biroq u yoki bu turdagi matritsani o'zgartirish uchun, qiymati jihatidan kerakli matritsani hosil qilib joriy matritsaga ko'paytiradigan maxsus buyruqlarni qo'llash qulay.

Umuman olganda, sahnaning uch o'lchamli obyektlarini ilova oynasida tasvirlash uchun rasmda ko'rsatilgan ketma-ketlik bajariladi.



3.5-rasm. OpenGL da koordinatalarni almashtirish.

Esda saqlang: OpenGL da obyektlar va kamerani almashtirishlar koordinata vektorlarini matritsaga ko'paytirish orqali amalga oshiriladi. Bunda ko'paytirish joriy matritsaga nisbatan glVertex* buyrug'i yordamida koordinatalar aniqlangan vaqtda amalga oshiriladi.

Modelli-tasvir o'zgartirishlar.

Modelli-tasvir o'zgartirishlarini koordinata o'qlari bo'ylab ko'chirish, burish va masshtabni o'zgartirish deb hisoblaymiz.

Ushbu operatsiyalarni bajarish uchun obyektning har bir uchi bilan tegishli matritsani ko'paytirish va ushbu uchlarning o'zgartirilgan koordinatalarini olish yetarli:

$$(x', y', z', 1)^T = M \cdot (x, y, z)^T.$$

Bu yerda M – modelli-tasvir o'zgartirish matritsasi. Matritsaning o'zi quyidagi buyruqlar yordamida yaratilishi mumkin:

void **glTranslate [f d]** (GLtype *x*, GLtype *y*, GLtype *z*)

void **glRotate [f d]** (GLtype *angle*, GLtype *x*, GLtype *y*, GLtype z)

void **glScale [f d]** (GLtype *x*, GLtype *y*, GLtype *z*)

glTranlsate*() obyektni ko'chirishga tayyorlaydi, o'zining parametri qiymatlarini uning uchlari koordinatalariga qo'shadi.

glRotate*() buyrug'i obyektni (*x*,*y*,*z*) vektori atrofida *angle* (graduslarda o'lchanadi) burchagi ostida soat strelkasiga teskari burilishini bajaradi.

glScale*() buyrug'i (*x*,*y*,*z*) vektorlar o'qi bo'yicha obyektning masshtablanishini bajaradi.

Obyekt holatini o'zgartirishdan tashqari, kuzatuvchi holatini o'zgartirish zarurati ham tug'iladi. Buni quyidagi buyruq yordamida bajarish mumkin:

void **gluLookAt** (GLdouble *eyex*, GLdouble *eyey*,
GLdouble *eyez*, GLdouble *centerx*,
GLdouble *centery*, GLdouble *centerz*,
GLdouble *upx*, GLdouble *upy*,
GLdouble *upz*)

bu yerda (*eyex,eyey,eyez*) nuqtasi kuzatuvchi nuqtasini belgilaydi, (*centerx, centery, centerz*) chiqarish sohasi markazida proyeksiyalanuvchi sahna markazini beradi, (*upx,upy,upz*) vektori esa kamera burilishini aniqlab, *y* o'qining musbat yo'nalishini belgilaydi. Misol uchun, agarda kamerani burish talab qilinmasa, (0,1,0) qiymati beriladi va sahnani (0,-1,0) qiymati bilan burish amalga oshiriladi.

Qisqacha aytganda, bu buyruq sahna obyektlarini ko'chirish va burishni amalga oshiradi, lekin, parametrlarni bunday ko'rinishda berish qulayroq. Shuni ta'kidlash joizki, gluLookAt() buyrug'ini

chaqirish obyektlarni almashtirishdan oldin, ya'ni model ko'rinishli matritsa birlik matritsasiga teng bo'lganda ma'noga ega.

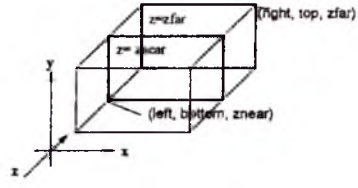
Eslab qoling: umumiy holda OpenGL da matritsaviy almashtirishlarni teskari tartibda yozib borish kerak. Misol uchun, agar siz avval obyektni burib, keyin ko'chirmoqchi bo'lsangiz, oldin glTranslate() buyrug'ini, keyin esa glRotate() buyrug'ini chaqirasiz. Bundan keyin esa obyektning o'zini belgilaysiz.

Proeksiyalar.

OpenGL da ortografik (parallel) va perspektiv (markaziy) proyeksiyalarni tayinlash uchun standart buyruqlar mavjud. Proyeksiyalashning birinchi turi quyidagi buyruq orqali ifodalanadi:

void **glOrtho** (GLdouble *left*, GLdouble *right*,
GLdouble *bottom*, GLdouble *top*,
GLdouble *near*, GLdouble *far*)

void **gluOrtho2D** (GLdouble *left*, GLdouble *right*, GLdouble *bottom*, GLdouble *top*)



3.6-rasm. Ortografik proeksiya.

Birinchi buyruq chap tomonlama koordinatalar tizimida proyeksiya matritsasini hosil qiladi. Buyruq parametrlari chiqarish oynasining chapki past va o'ng yuqori burchaklariga javob beruvchi (*left, bottom, znear*) va (*right, top, zfar*) nuqtalarni beradi. *near* va *far* parametrlari yaqin va uzoq kesishish tekisliklari orasidagi masofani yo'qolish tartibida (0,0,0) nuqtasidan boshlab belgilaydi va ular manfiy bo'lishi mumkin.

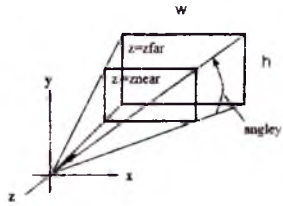
Ikkinchi buyruqda, birinchisidan farqli ravishda *near* va *far* qiymatlari mos ravishda -1 va 1 ga teng qilib belgilanadi. Bu hol OpenGL ikki o'lchamli obyektlarni chizish uchun qo'llanilganda

qulay. Bunda cho'qqilarning holatini glVertex2*() buyrug'i yordamida berish mumkin.

Perspektiv proyeksiyalash quyidagi buyruq bilan beriladi:

void **gluPerspective** (GLdouble *angley*, GLdouble *aspect*, GLdouble *znear*, GLdouble *zfar*)

qaysiki chap tomonli koordinatalar sistemasida kesik konus ko'rinishini beradi. *angley* parametri *y* o'qi bo'yicha graduslarda ko'rish burchagini belgilaydi va 0 dan 180 gacha oraliqda graduslarni aniqlaydi. *x* o'qi bo'ylab ko'rish burchagi *aspect* parametri bilan beriladi. *x* o'qi bo'ylab ko'rish burchagi *aspect* parametri bilan, odatda chiqarish sohasi tomonlari orasidagi munosabat sifatida beriladi.



3.7-rasm. Perspektiv proyeksiya.

zfar va *znear* parametrlari kuzatuvchidan kesishish chuqurligi bo'ylab kesilgan tekisliklarigacha masofani belgilaydi, ular musbat bo'lishi kerak. *zfar/znear* orasidagi munosabat qanchalik katta bo'lsa, bufer chuqurligida unga yaqin joylashgan tekislik shunchalik yomon farqlanadi, chunki shart berilmaganda unga 0 dan 1 gacha oraliqda chuqurlikning «siqilgan» sonlari yoziladi.

Proyeksiyalarning matritsasini berishdan oldin, glMatrixMode (GL_PROJECTION) buyrug'i yordamida kerakli matritsa bilan ishlash rejimini yoqishni hamda glLoadIdentity() ni chaqirib joriy sozlashlarni bekor qilishni unutmaslik lozim.

Misol uchun:

```
/* ortografik proeksiya */
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(0, w, 0, h, -1.0, 1.0);
```

Chiqarish sohasi.

Proyeksiya matritsasi qo'llanilgandan so'ng, keyingi almashtirish kirishiga kesilgan (clipped) koordinatalar deb ataluvchilar beriladi. Keyin esa uchlarning normallangan koordinatalari quyidagi formula yordamida topiladi:

$$(x_n, y_n, z_n)^T = \left(\frac{x_c}{w_c}, \frac{y_c}{w_c}, \frac{z_c}{w_c}\right)^T.$$

Chiqarish sohasi koordinatalar oynaviy tizimida to'g'ri to'rtburchakni aks ettiradi, uning o'lchamlari esa quyidagi buyruq yordamida beriladi:

void **glViewPort** (GLint *x*, GLint *y*, GLint *width*, GLint *height*).

Barcha parametrlarning qiymatlari piksellarda beriladi va koordinatalar tizimining oynaviy tizimida chap pastki burchak koordinatalari (*x*,*y*) bilan chiqarish sohasining kengligi va balandligini aniqlaydi. Koordinatalar oynaviy tizimining o'lchamlari ilova oynasining joriy o'lchamlari bilan aniqlanadi, (0,0) nuqtasi oynaning chap pastki burchagida joylashgan.

glViewPort() buyrug'i parametrlarini qo'llab OpenGL chiqarish sohasi markazining (o_x,o_y) oynali koordinatalarini quyidagi $o_x=x+width/2$, $o_y=y+height/2$ formulalar bilan aniqlaydi.

Aytaylik, $p_x=width$, $p_y=height$ bo'lsin, unda har bir uchning oynaviy koordinatalarini topish mumkin:

$$(x_w, y_w, z_w)^T = \left(\left(\frac{p_x}{2}\right)\cdot x_n + o_x, \left(\frac{p_y}{2}\right)\cdot y_n + o_y, \left(\frac{f-n}{2}\right)\cdot z_n + \frac{(n+f)}{2}\right)^T.$$

Bunda *n* va *f* musbat butun qiymatlari oynada nuqtaning minimal maksimal chuqurligini beradi va shart berilmaganda, mos ravishda 0 va 1 ga teng. Har bir nuqtaning chuqurligi maxsus chuqurlik buferiga (z-bufer) yozib boriladi va u ko'rinmas chiziq va sirtlarni o'chirish uchun ishlatiladi. *n* va *f* qiymatlarini quyidagi funksiyalar yordamida o'rnatish mumkin.

void **glDepthRange** (GLclampd *n*, GLclampd *f*)

glViewPort() buyrug'i odatda, foydalanuvchi oyna o'lchamlarini o'zgartirganda chaqiriladigan glutReshapeFunc () buyrug'i bilan qayd qilingan funksiyada ishlatiladi.

Nazorat savollari:

1. OpenGL da qanday koordinatalar sistemasi ishlatiladi?
2. OpenGL da matritsali o'zgartirish ko'rinishlarini keltiring?
3. OpenGL da obyektlar qanday qilib o'zgartiriladi?
4. OpenGL da proyeksiyalar qanday tayinlanadi?
5. Matritsali stek deganda nimani tushunasiz?
6. OpenGL da kuzatuvchi holatini o'zgartirish usullarini keltiring?
7. glTranslate(), glRotate() va glScale() buyruqlarini chaqirishning qanday ketma-ketligi gluLookAt(0, 0, -10, 10, 0, 0, 0, -1, 0) buyrug'iga mos keladi?
8. Proyeksiyalarni tayinlash uchun qanday standart buyruqlarni bilasiz?

Tayanch iboralar: Koordinatalar sistemasi, matritsalar ustida amallar, modelli tasvir, proyeksiyalash matritsasi, teksturalash matritsasi, proeksiyalash, chiqarish sohasi.

3.5. Materiallar va yorug'lik

Haqiqiy tasvirlarni hosil qilish uchun obyektning xossalarini hamda u joylashgan muhit xossalarini ham aniqlash zarur. Xossalarning birinchi guruhi obyekt yasalgan materialning parametrlari, uning yuzasiga teksturani qo'yish usullari, obyektning shaffoflik darajasini o'z ichiga oladi. Ikkinchi guruhga yorug'lik manbalarining soni va xususiyatlarini, muhitning shaffoflik darajasi hamda yoritish modelini kiritish mumkin. Bu xossalarning barchasini OpenGL ning mos buyruqlarini chaqirgan holda tayinlash mumkin.

Yorug'lik modeli.

OpenGL da yoritish modeli ishlatiladi va unga muvofiq nuqta rangi bir necha faktorlar bilan belgilanadi: material va tekstura xossalari, shu nuqtadagi normal kattaligi hamda yorug'lik manbai

va kuzatuvchi holati bilan. Nuqtadagi yorug'likni aniq hisoblash uchun birlik normallardan foydalanish kerak, lekin, glScale*() turidagi buyruqlar normallar uzunligini o'zgartirishi mumkin.

Yorug'likning global parametrlarini berish uchun

void **glLightModel [i f]** (GLenum *pname*, GLenum *param*)

void **glLightModel[i f]v** (GLenum *pname*, const GLtype **params*)

buyruqlaridan foydalaniladi.

pname argumenti yorug'lik modelining qaysi parametri sozlanishini va quyidagi qiymatlarni qabul qilishi mumkinligini belgilaydi:

GL_LIGHT_MODEL_LOCAL_VIEWER *param* parametri bul toifasiga tegishli bo'lishi kerak va kuzatuvchining holatini belgilaydi. Agar u GL_FALSE ga teng bo'lsa, unda tasvir yo'nalishi ko'rinishli koordinatalardagi holatiga bog'liq bo'lmagan holda, $-z$ o'qiga parallel bo'lib hisoblanadi. Agar u GL_TRUE ga teng bo'lsa, unda kuzatuvchi ko'rinishli koordinatalar tizimining boshida joylashgan. Bu yorug'lik sifatini oshirishi mumkin, lekin uni hisoblashni qiyinlashtiradi. Shart qo'yilmaganda qiymat GL_FALSE bo'ladi.

GL_LIGHT_MODEL_TWO_SIDE *param* parametri bul toifasiga tegishli bo'lishi kerak hamda yuza va teskari yoqlar uchun yorug'lik hisoblash rejimini boshqaradi. Agar u GL_FALSE ga teng bo'lsa, unda yoritilganlik faqat yuza yoqlari uchun hisoblanadi. Agarda u GL_TRUE ga teng bo'lsa, hisoblash teskari yoqlar uchun ham amalga oshiriladi. Shart qo'yilmaganda qiymat GL_FALSE ga teng.

GL_LIGHT_MODEL_AMBIENT *params* parametri to'rtta butun yoki haqiqiy sonlarga ega bo'lishi kerak, bu sonlar hattoki ma'lum bir yorug'lik manbalari bo'lmaganda ham fonning yorug'ligi rangini belgilaydi. Shart qo'yilmaganda qiymatlar: (0.2, 0.2, 0.2,1.0) ga teng.

Materiallar tasnifi.

Joriy materialning parametrlarini berish uchun quyidagi buyruqlardan foydalaniladi:

void **glMaterial [i f]** (GLenum *face*, GLenum *pname*, GLtype *param*)

void **glMaterial[i f]v** (GLenum *face*, GLenum *pname*, GLtype **params*)

Ular yordamida materialning ko'zguli, diffuziyali, tarqoq ranglarini, hamda agar obyekt nurlansa, nurning tarqalish intensivligini va ko'zguli akslantirishni aniqlash mumkin. *Param* qiymati bilan aynan qaysi parametr tavsiflanishi *pname* qiymatiga bog'liq:

GL_AMBIENT *params* parametri RGBA ranglarining to'rtta butun yoki haqiqiy qiymatlariga ega bo'lishi kerak, ular esa o'z navbatida materialning tarqoq rangini belgilaydi (materialning soyadagi rangi). Shart qo'yilmaganda qiymatlar (0.2, 0.2, 0.2, 1.0) ga teng bo'ladi.

GL_DIFFUSE *params* parametri RGBA ranglarining to'rtta butun yoki haqiqiy qiymatlariga ega bo'lishi kerak, ular esa o'z navbatida materialning diffuziyali rangini aniqlaydi. Shart qo'yilmaganda qiymatlar (0.8, 0.8, 0.8, 1.0) ga teng.

GL_SPECULAR *params* parametri RGBA ranglarining to'rtta butun yoki haqiqiy qiymatlariga ega bo'lishi kerak, ular esa o'z navbatida materialning ko'zguli rangini aniqlaydi. Shart qo'yilmaganda qiymatlar (0.0, 0.0, 0.0, 1.0) ga teng.

GL_SHININESS *params* parametri 0 dan 128 gacha oraliqda bitta butun yoki haqiqiy qiymatga ega bo'lishi kerak va u materialning ko'zguli akslanish darajasini aniqlaydi. Shart qo'yilmaganda qiymat 0 ga teng.

GL_EMISSION *params* parametri RGBA ranglarining to'rtta butun yoki haqiqiy qiymatlariga ega bo'lishi kerak, ular esa o'z navbatida materialdan tarqalayotgan yorug'lik intensivligini aniqlaydi. Shart qo'yilmaganda qiymatlar (0.0, 0.0, 0.0, 1.0) ga teng.

GL_AMBIENT_AND_DIFFUSE glMaterial*() buyrug'ini pname **GL_AMBIENT** va **GL_DIFFUSE** qiymati hamda *params* bir xil qiymatlari bilan ikki marta chaqirishga ekvivalent.

Bundan kelib chiqadiki, glMaterial[i f]() buyrug'ini chaqirish faqat materialning ko'zguli akslanish darajasi o'rnatilishi uchungina

mumkin bo'ladi. glMaterial[i f]v() buyrug'i qolgan parametrlarni berish uchun ishlatiladi.

face parametri bu material beradigan yoqlar turini aniqlaydi va **GL_FRONT, GL_BACK** yoki **GL_FRONT_AND_BACK** qiymatlarini qabul qilishi mumkin.

Agar sahnada obyektlarning materiallari faqat bitta parametr bilan farqlansa, glEnable() ni **GL_COLOR_MATERIAL** parametri bilan chaqirib kerakli rejimni o'rnatish tavsiya etiladi, keyin esa quyidagi buyruqdan foydalaniladi:

void **glColorMaterial** (GLenum *face*, GLenum *pname*)

bu yerda, *face* parametri yuqorida ko'rsatilgan ma'noga ega, *pname* parametri esa barcha sanab o'tilgan qiymatlarni qabul qilishi mumkin. Bundan keyin *pname* aniq obyektning materiali xossalari yordamida olingan qiymatlar glColor*() buyrug'ini chaqirish orqali o'rnatiladi va bu ko'p resurslarni talab qiladigan glMaterial*() buyrug'i chaqirilishining oldini oladi hamda dastur samaradorligini oshiradi.

Material xossalarini aniqlashga misol:

```
float mat_dif[]={0.8,0.8,0.8};
float mat_amb[] = {0.2, 0.2, 0.2};
float mat_spec[] = {0.6, 0.6, 0.6};
float shininess = 0.7 * 128;
...
glMaterialfv (GL_FRONT_AND_BACK,GL_AMBIENT, mat_amb);
glMaterialfv (GL_FRONT_AND_BACK,GL_DIFFUSE, mat_dif);
glMaterialfv (GL_FRONT_AND_BACK,GL_SPECULAR, mat_spec);
glMaterialf (GL_FRONT,GL_ShININESS, shininess);
```

Yorug'lik manbalarining tavsifi.

Obyekt materialining xossalarini aniqlash faqat sahnada yorug'lik manbalari bo'lgandagina o'rinli. Aks holda barcha obyektlar qora rangda (materialning tarqoq rangida) bo'ladi.

Sahnaga yorug'lik manbaini quyidagi buyruq yordamida qo'shish mumkin:

void **glLight [i f]** (GLenum *light*, GLenum *pname*, GLfloat *param*)
void **glLight[i f]** (GLenum *light*, GLenum *pname*, GLfloat **params*)

light parametri o'z-o'zidan yorug'lik manbaini belgilaydi. U **GL_LIGHTi** ko'rinishdagi maxsus belgili nomlar to'plami bilan tanlanadi, *i* bu yerda 0 dan odatda sakkizdan katta bo'lmaydigan **GL_MAX_LIGHT** o'zgarmaslari orasida yotadi.

pname va *params* parametrlari glMaterial*() buyrug'iga o'xshash ma'noga ega. *pname* parametri qiymatlarini qarab chiqamiz:

GL_SPOT_EXPONENT *param* parametri yorug'lik intensivligi tarqalishini beradigan 0 dan 128 gacha butun yoki haqiqiy qiymatga ega bo'lishi kerak. Bu parametr yorug'lik manbaining fokuslanish darajasini tavsiflaydi. Shart qo'yilmaganda qiymat 0 ga teng (tarqoq yorug'lik).

GL_SPOT_CUTOFF *param* parametri yorug'lik tarqalishining maksimal burchagini aniqlaydi. Bu parametrning qiymati manba tomonidan yaratiladigan konus ko'rinishdagi yorug'lik oqimi uchidagi burchakning yarmiga teng. Shart qo'yilmaganda qiymat 180 (tarqoq yorug'lik) ga teng.

GL_AMBIENT *params* parametri, fondagi yoritilganlikni aniqlaydigan, RGBA ranglarining to'rtta butun yoki haqiqiy qiymatlariga ega bo'lishi kerak. Shart qo'yilmaganda qiymatlar (0.0, 0.0, 0.0, 1.0) ga teng.

GL_DIFFUSE *params* parametri, diffuziyali yoritilishni aniqlaydigan, RGBA ranglarining to'rtta butun yoki haqiqiy qiymatlariga ega bo'lishi kerak. Shart qo'yilmaganda qiymatlar **GL_LIGHT0** uchun (1.0, 1.0, 1.0, 1.0) ga, qolganlari uchun (0.0, 0.0, 0.0, 1.0) ga teng.

GL_SPECULAR *params* parametri, ko'zguli akslanishni aniqlaydigan, RGBA ranglarining to'rtta butun yoki haqiqiy qiymatlariga ega bo'lishi kerak. Shart qo'yilmaganda qiymatlar

GL_LIGHT0 uchun (1.0, 1.0, 1.0, 1.0) ga va qolganlari uchun (0.0, 0.0, 0.0, 1.0) ga teng.

GL_POSITION *params* parametri, yorug'lik manbaining joyini aniqlaydigan to'rtta butun yoki haqiqiy qiymatlarga ega bo'lishi kerak. Agar w komponentining qiymati 0.0 ga teng bo'lsa, unda manba cheksiz uzoqda deb hisoblanadi va yoritilganlikni hisoblashda (x,y,z) nuqtaga nisbatan yo'nalish hisobga olinadi, aks holda manba (x,y,z,w) nuqtada joylashgan deb hisoblanadi. Birinchi holatda manbadan uzoqlashganda yorug'lik kamayishi kuzatilmaydi, ya'ni manba cheksiz uzoqda deb olinadi. Shart qo'yilmaganda qiymatlar (0.0, 0.0, 1.0, 0.0) ga teng.

GL_SPOT_DIRECTION *params* parametri yorug'lik yo'nalishini aniqlaydigan to'rtta butun yoki haqiqiy qiymatlarga ega bo'lishi kerak. Shart qo'yilmaganda qiymatlar (0.0, 0.0, -1.0, 1.0) ga teng. Manbaning bu tavsifi **GL_SPOT_CUTOFF** qiymati 180 dan farqli bo'lganda ma'noga ega.

GL_CONSTANT_ATTENUATION,
GL_LINEAR_ATTENUATION,
GL_QUADRATIC_ATTENUATION

params parametri manbadan uzoqlashtirilganda yorug'lik intensivligining kamayishini aniqlaydigan uchta koeffitsiyentdan bittasini beradi. Faqat manfiy bo'lmagan qiymatlar o'rinli. Agar manba yo'naltirilmagan bo'lsa, unda kamayish quyidagi summaga teskari proporsional:

$$att_{constant}+att_{linear}*d+ att_{quadratic} *d^2,$$

bu yerda d – yorug'lik manbai va u yoritadigan uch orasidagi masofa, $att_{constant}$, att_{linear} i $att_{quadratic}$ mos ravishda

GL_CONSTANT_ATTENUATION,
GL_LINEAR_ATTENUATION va
GL_QUADRATIC_ATTENUATION konstantalari bilan berilgan parametrlarga teng. Shart qo'yilmaganda parametrlar uchlik bilan beriladi (1, 0, 0) va yorug'lik kamayishi sodir bo'lmaydi.

Manba holati o'zgarganda quyidagi omilni hisobga olish zarur: OpenGL da yorug'lik manbalari ko'p hollarda ko'pburchaklar va nuqtalarga o'xshagan obyektlar hisoblanadi. Ularga nisbatan OpenGL da koordinatalar qayta ishlanishining asosiy qoidalari

o'rinli, ya'ni fazodagi holatni tavsiflaydigan parametrlar joriy model-ko'rinishli matritsa bilan obyekt o'zgartirilayotganda, ya'ni OpenGL ning mos buyruqlari chaqirilganda almashtiriladi. Shunday qilib, yorug'lik manbaini sahna obyekti yoki kamera bilan bir vaqtda o'zgartirib, uni shu obyektga bog'lash mumkin. Yoki aksincha, boshqa obyektlar ko'chayotganda, joyida qoladigan statsionar yorug'lik manbaini tuzish mumkin.

Umumiy qoida quyida keltirilgan:

Agar yorug'lik manbai holati virtual kamera holatini belgilashdan oldin glLight*() buyrug'i bilan berilgan bo'lsa, unda manba koordinatalari (0,0,0) kuzatish nuqtasida joylashgan deb hisoblanadi va shunga muvofiq yorug'lik manbai holati kuzatuvchi holatiga nisbatan aniqlanadi.

Agar holat kamera holati va obyektning model-ko'rinishli matritsasi almashtirishlari orasidagi belgilanish orqali o'rnatilsa, unda u qo'zg'almas holatga keltiriladi, ya'ni bu holda yorug'lik manbai holati eng yuqori koordinatalarda beriladi.

Yorug'likdan foydalanish uchun dastlab mos rejim glEnable (GL_LIGHTNING) buyrug'ini chaqirish orqali o'rnatiladi, keyin esa kerakli manba glEnable(GL_LIGHTi) buyrug'i bilan yoqiladi.

Yana bir bor e'tiborimizni shunga qaratishimiz kerakki, yorug'lik o'chirilganda uchning rangi glColor*() buyrug'i bilan beriladigan joriy rangga teng. Yorug'lik o'chirilganda uchning rangi yorug'lik manbalari, normallar va material to'g'risidagi ma'lumotlardan kelib chiqib hisoblanadi.

Yorug'lik o'chirilganda vizualizatsiya tezroq amalga oshadi, lekin, bu holda ilovaning o'zi uchlarning ranglarini hisoblashi kerak bo'ladi.

Tuman effektini hosil qilish.

Yakunda OpenGL ning qiziqarli va ko'p ishlatiladigan imkoniyati – tuman effektini hosil qilishni ko'rib chiqamiz. Sahnaning yengil tumanlanishi haqqoniy effektni hosil qiladi hamda sahnada uzoqlashgan obyektlar bo'lganda hosil bo'ladigan ba'zi artefaktlarni qisman yashirishi mumkin.

OpenGL da tuman sahnadagi obyektlarning rangini ularning chuqurligiga, ya'ni kuzatish nuqtasigacha bo'lgan masofaga bog'liq

ravishda o'zgartirish orqali amalga oshiradi. Rangning o'zgarishi OpenGLni amalga oshirishga bog'liq ravishda rasterizatsiya pog'onasida har bir piksel uchun yoki primitivlarining uchlari uchun sodir bo'ladi. Bu jarayonni qisman boshqarish mumkin.

Tumanlash effektini yoqish uchun **glEnable(GL_FOG)** buyrug'ini chaqirish zarur.

Uchdagi tuman intensivligini hisoblash usulini quyidagi buyruqlar yordamida aniqlash mumkin:

void **glFog [if]** (enum *pname*, T *param*);
void **glFog[if]v** (enum *pname*, T *params*);

pname argumenti quyidagi qiymatlarni qabul qilishi mumkin:

GL_FOG_MODE *param* argumenti nuqtadagi tuman intensivligi hisoblanadigan formulani aniqlaydi.

Bu holda *param* quyidagi qiymatlarni qabul qilishi mumkin:

GL_EXP Intensivlik $f=\exp(-d*z)$ formula bilan topiladi

GL_EXP2 Intensivlik $f=\exp(-(d*z)^2)$ formula bilan topiladi

GL_LINEAR Intensivlik $f=e-z/e-s$ formula bilan topiladi

Bu yerda z – tuman intensivligi hisoblanadigan uchdan kuzatish nuqtasigacha bo'lgan masofa.

d,e,s koeffitsiyentlari *pname* argumentining quyidagi qiymatlari bilan beriladi:

GL_FOG_DENSITY *param d* koeffitsiyentini aniqlaydi
GL_FOG_START *param s* koeffitsiyentini aniqlaydi
GL_FOG_END *param e* koeffitsiyentini aniqlaydi

Tuman rangi **GL_FOG_COLOR** ga teng bo'lgan *pname* argumenti yordamida beriladi. Bu holda *params* – 4 ta rang komponentiga ega massivga ko'rsatkich.

Bu effektni qo'llashga misol keltiramiz:

```
GLfloat FogColor[4]={0.5,0.5,0.5,1};
glEnable(GL_FOG);
glFogi(GL_FOG_MODE,GL_LINEAR);
glFogf(GL_FOG_START,20.0);
glFogf(GL_FOG_END,100.0);
glFogfv(GL_FOG_COLOR,FogColor);
```

Nazorat savollari:

1. Yorug'likning lokal va cheksiz uzoqlashgan manbalari o'rtasidagi farqni tushuntiring?
2. glColorMaterial buyrug'i nima uchun xizmat qiladi?
3. Yorug'lik manbasi har doim kuzatuvchi holati nuqtasida bo'lishi uchun uning holati qanday o'rnatiladi?
4. Yorug'lik manbasining fiksirlangan holati qanday o'rnatiladi?
5. Obyektning lokal koordinatalariga nisbatan manba holatini o'rnatish mumkinmi?
6. Yorug'likning konusli manbasi qanday beriladi?
7. Yorug'lik manbai holati kuzatuvchi holatiga nisbatan qanday aniqlanadi?
8. Agar sahna yoritilgan ammo yorug'lik manbai yo'q bo'lsa, unda obyektlar qanday rangga ega bo'ladi?

Tayanch iboralar: Yorug'lik modeli, material, yorug'lik manbalari, yorug'lik intensivligi, diffuziyali yoritilish, ko'zguli akslanish, tuman effekti.

3.6. Teksturalash

Tekstura deganda obyektga ma'lum bir usulda qo'yish kerak bo'lgan qandaydir tasvir tushuniladi.

Sahna obyektlari yuzasiga tekstura qo'yilganda uning haqqoniyligi oshadi, lekin, shuni ham hisobga olish lozimki, bu jarayon ortiqcha hisob-kitobni talab qiladi.

Tekstura bilan ishlash uchun quyidagi harakatlar ketma-ketligini amalga oshirish kerak:

1. Tasvirni tanlash va uni kerakli formatga keltirish;
2. Tasvirni OpenGL ga uzatish;
3. Tekstura obyektga qanday qo'yilishini va u bilan qanday munosabatda bo'lishini aniqlab olish;
4. Teksturani obyekt bilan bog'lash.

Teksturani tayorlash.

Teksturadan foydalanish uchun, oldin xotiraga kerakli tasvirni joylashtirish va uni OpenGL ga uzatish kerak.

Grafikaviy ma'lumotlarni fayldan o'qishni va ularni almashtirishni qo'lda bajarish mumkin. Shu bilan birga GLAUX (undan foydalanish uchun qo'shimcha ravishda glaux.lib ni bog'lash kerak bo'ladi) kutubxonasiga kiruvchi muhim amallarni o'zi bajaradigan funksiyadan foydalanish mumkin. Bu funksiya

AUX_RGBImageRec* **auxDIBImageLoad** (const char **file*)

bu yerda *file* – *.bmp yoki *.dib kengaytmali fayl nomi. Funksiya ko'rsatkichni qayta ishlangan ma'lumotlar saqlanadigan xotira sohasiga qaytardi.

Xotirada tekstura obrazini hosil qilishda quyidagi talablarni hisobga olish kerak.

Birinchidan, teksturaning o'lchamlari, ham gorizontal bo'yicha, ham vertikal bo'yicha o'zida ikkining darajalarini aks ettirishi lozim. Bu talab teksturani tekstura xotirasida kompakt holda qo'yilishini ko'zda tutadi va uning samarali qo'llanilishini ta'minlaydi. Faqat shunday teksturalar bilan ishlash noqulay, shuning uchun joylashtirilgandan keyin ularni almashtirish kerak. Tekstura o'lchamlarini quyidagi buyruq yordamida o'zgartirish mumkin:

void **gluScaleImage** (GLenum *format*, GLint *widthin*,
GL *heightin*, GLenum *typein*,
const void **datain*,
GLint *widthout*,
GLint *heightout*, GLenum *typeout*,
void **dataout*)

Format parametrining qiymati sifatida odatda ma'lumotlarni saqlash formatini belgilaydigan **GL_RGB** yoki **GL_RGBA** qiymati ishlatiladi. *widthin*, *heightin*, *widhtout*, *heightout* parametrlari kiruvchi va chiquvchi tasvirlarining o'lchamlarini aniqlaydi, *typein* va *typeout* yordamida esa *datain* va *dataout* manzillarida joylashgan massiv elementlarining turi beriladi. Odatiy holga o'xshash bu **GL_UNSIGNED_BYTE, GL_ShORT, GL_INT** va boshqa turlar bo'lishi mumkin. Funksiya o'z ishining natijasini *dataout* parametrini ko'rsatadigan xotira sohasiga kiritadi.

Ikkinchidan, obyekt rasterizatsiyadan keyin o'lchamlari jihatidan unga qo'yiladigan teksturadan kichkina bo'lib qolishi holini ko'zda tutish kerak. Obyekt qanchalik kichkina bo'lsa, unga qo'yiladigan tekstura ham shunchalik kichkina bo'lishi kerak, shuning uchun ham *teksturani detalizatsiyalash pog'onalari* degan tushuncha kiritiladi (mipmap). Detalizatsiyaning har bir pog'onasi originalning ikki baravar kichiklashtirilgan nusxasi sifatidagi qandaydir tasvirni beradi. Bunday yondashuv obyektga tekstura qo'yishni osonlashtiradi. Masalan, $2m$x$2n$ o'lchamdagi tasvir uchun detalizatsiyaning turli pog'onalariga mos keluvchi max(m,n)+1 kichiklashtirilgan tasvirlarni qurish mumkin.

OpenGL ning ichki xotirasida tekstura obrazini yaratishning bu ikki pog'onasini quyidagi buyruqlar yordamida olib borish mumkin:

void **gluBuild2DMipmaps** (GLenum *target*, GLint *components*,
GLint *width*, GLint *height*,
GLenum *format*, GLenum *type*,
const void **data*)

bu yerda *target* parametri **GL_TEXTURE_2D** ga teng bo'lishi kerak. *components* parametri teksturaning rang komponentlarini aniqlaydi va quyidagi asosiy qiymatlarni qabul qilishi mumkin:

GL_LUMINANCE bitta komponent – yorqinlik (tekstura monoxrom bo'ladi)

GL_RGB qizil, ko'k, yashil

GL_RGBA barcha komponentlar.

width, *height*, *data* parametrlari teksturaning joylashuvi va o'lchamlarini aniqlaydi, *format* va *type* esa gluScaleImage() buyrug'iga o'xshash ma'noga ega.

Bu buyruq amalga oshrilgandan keyin, tekstura OpenGL ning ichki xotirasiga nusxalanadi va shuning uchun joriy tasvir band qilib turgan xotirani bo'shatish mumkin.

OpenGL da bir o'lchamli, ya'ni 1xN teksturalardan foydalanishga ruxsat berilgan, lekin, buni har doim *target* sifatida **GL_TEXTURE_1D** o'zgarmaslarini berib ko'rsatib o'tish kerak. Bir o'lchamli teksturalarning kerakliligi shubha ostida bo'lganligi sababli ularga batafsil to'htalib o'tmaymiz.

Sahnada bir necha teksturalar qo'llanilganda OpenGL da tasvirlar ro'yxatini (teksturali obyektlar) yaratishni eslatuvchi amal bajariladi. Avval

void **glGenTextures** (GLsizei *n*, GLuint* *textures*)

buyrug'i yordamida *textures* massiviga yoziladigan teksturalar *n* identifikatorlarini hosil qilish kerak. Keyingi teksturaning xossalarini aniqlashni boshlashdan oldin uni quyidagi buyruqni chaqirib joriy qilish kerak:

void **glBindTexture** (GLenum *target*, GLuint *texture*)

bu yerda *target* **GL_TEXTURE_1D** yoki **GL_TEXTURE_2D** qiymatlarini qabul qilishi mumkin, *texture* parametri esa keyingi buyruqlar taalluqli bo'lgan tekstura identifikatoriga teng bo'lishi kerak. Chizish jarayonida joriy teksturani bir qancha identifikatorlar bilan bajarish uchun *target* va *texture* qiymatlariga mos glBindTexture() buyrug'ini takroran chaqirish yetarli. Shu tarzda, glBindTexture() buyrug'i *texture* identifikatori bilan teksturalar yaratish rejimini kiritadi, agarda bunday tekstura yoki uning bajarilish rejimi yaratilmagan bo'lsa, ushbu teksturani joriy etib tayinlaydi.

Chunki barcha apparaturalar ham katta hajmdagi teksturalardan foydalana olmaydi, teksturalar o'lchamini 256x256 yoki 512x512 pikselgacha cheklash maqsadga muvofiq. Eslatib o'tamiz, katta bo'lmagan teksturalardan foydalanish dastur samaradorligini oshiradi.

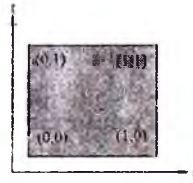
Obyektlarga teksturalar qo'yish.

Teksturalar qo'yishda, yuqorida ta'kidlanganidek, teksturaning o'lchami u qo'yilayotgan obyektning oynasi o'lchamidan farq qilishi holatlarini hisobga olish kerak. Bunda tasvirni cho'zish, siqish, ushbu o'zgartirishlar qanday olib borilishi, qurilayotgan tasvir sifatiga jiddiy ta'sir qilishi mumkin. Teksturada nuqta holatini aniqlash uchun koordinatalarning parametrik tizimi (s,t) dan foydalaniladi, *s* va *t* qiymatlari [0,1] oraliqda yotadi (3.8. rasm).

Teksturaning turli parametrlarini o'zgartirish uchun quyidagi buyruqlardan foydalaniladi:

void **glTexParameter [i f]** (GLenum *target*, GLenum *pname*, GLenum *param*)

void **glTexParameter[i f]v** (GLenum *target*, GLenum *pname*, Glenum* *params*)



3.8-rasm. Tekstura koordinatalari.

Shunda *target* **GL_TEXTURE_1D** yoki **GL_TEXTURE_2D** qiymatlarini qabul qilishi mumkin, *prame* qaysi xususiyatni almashtirishimizni belgilaydi, *param* yoki *params* yordamida yangi qiymat o'rnatiladi. *pname* ning mumkin bo'lgan qiymatlari:

GL_TEXTURE_MIN_FILTER *param* parametri teksturalarni siqish uchun ishlatiladigan funksiyani belgilaydi. **GL_NEAREST** qiymatida bitta (yaqin) va **GL_LINEAR** qiymatida to'rtta yaqin tekstura elementlari ishlatiladi. Shart kiritilmagandagi qiymati: **GL_LINEAR**.

GL_TEXTURE_MAG_FILTER *param* parametri teksturani kattalashtirish (cho'zish) uchun ishlatiladigan funksiyani belgilaydi. **GL_NEAREST** qiymatida bitta (yaqin) tekstura elementi ishlatiladi. Shart kiritilmagandagi qiymati: **GL_LINEAR.**

GL_TEXTURE_WRAP_S *param* parametri *s* koordinata qiymatini o'rnatadi, agar u [0,1] kesma oralig'ida bo'lmasa. **GL_REPEAT** qiymatida *s* ning butun qismi olib tashlanadi va natijada tasvir yuza bo'yicha ko'payadi. **GL_CLAMP** qiymatida chetki qiymatlar ishlatiladi: 0 yoki 1, agar obyekt ustiga bitta tasvir tushirilsa, qo'llash qulay bo'ladi. Shart kiritilmagandagi qiymati: **GL_REPEAT.**

GL_TEXTURE_WRAP_T faqat *t* koordinatasi uchun oldingi qiymatga o'xshash.

GL_NEAREST rejimidan foydalanish teksturani to'ldirish (ustiga qo'yish) tezligini oshiradi, ammo bunda sifat pasayadi. Nega

deganda **GL_LINEAR** ga qaraganda unda interpolyatsiya ishlatilmaydi.

Teksturaning obyekt bajarilgan material bilan o'zaro ta'sirini aniqlash uchun quyidagi buyruqlar ishlatiladi.

void **glTexEnv [i f]** (GLenum *target*, GLenum *pname*, GLtype *param*)

void **glTexEnv[i f]v** (GLenum *target*, GLenum *pname*, GLtype **params*)

target parametri **GL_TEXTURE_ENV** ga teng bo'lishi kerak, *pname* sifatida ko'p ishlatiladigan **GL_TEXTURE_ENV_MODE** faqat bitta qiymatini ko'rib chiqamiz.

Tez-tez ishlatiladigan *param* parametri qiymatlari:

GL_MODULATE natijaviy rang ustida joylashgan rang nuqtasi va tekstura unga mos nuqtasini ko'paytirish natijasida topiladi.

GL_REPLACE natijaviy rang sifatida teksturadagi rang nuqtasi ishlatiladi.

Quyidagi dastur, teksturani yaratishga bo'lgan umumiy yondashuvni namoyon etadi:

```
/* bizga zarur bo'lgan teksturalar soni */
#define NUM_TEXTURES 10
/* tekstura identifikatorlari */
int TextureIDs[NUM_TEXTURES];
/* teksturalar ko'rinishi */
AUX_RGBImageRec *pImage;
...
/* 1) tekstura identifikatorlarini olish */
glGenTextures(NUM_TEXTURES,TextureIDs);
/* 2) parametrlarni modifikatsiyalash uchun teksturani tanlash
*/
glBindTexture(TextureIDs[i]); /* 0<=i<NUM_TEXTURES*/
/* 3) teksturani yuklaymiz. Tekstura o'lchami – 2 bosqich*/
pImage=dibImageLoad("texture.bmp");
if (Texture!=NULL)
{
```

```
    /* 4) teksturani OpenGL ga uzatamiz va parametrlarini
beramiz */
    /* bayt bo'yicha to'g'rilaymiz */
    glPixelStorei(GL_UNPACK_ALIGNMENT,1);
    gluBuildMipmaps(GL_TEXTURE_2D,GL_RGB, pImage-
>sizeX,
pImage->sizeY, GL_RGB, GL_UNSIGNED_BYTE,
pImage->data);
    glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MAG_F
ILTER,
(float)GL_LINEAR);
    glTexParameterf(GL_TEXTURE_2D,
GL_TEXTURE_MIN_FILTER,
(float)GL_LINEAR);
    glTexParameterf(GL_TEXTURE_2D,
GL_TEXTURE_WRAP_S,
(float)GL_REPEAT);
    glTexParameterf(GL_TEXTURE_2D,
GL_TEXTURE_WRAP_T,
(float)GL_REPEAT);
    glTexEnvf(GL_TEXTURE_ENV,
GL_TEXTURE_ENV_MODE,
(float)GL_REPLACE);
    /* 5) dastlabki tasvirni o'chiramiz.*/
    free(Texture);
    }else Error();
```

Teksturali koordinata.

Teksturani obyekt ustiga qo'shish uchun obyekt yuzasidagi nuqta bilan teksturaning nuqtasi o'rtasidagi moslikni aniqlash kerak. Bu moslikni ikki usul bilan berish mumkin: maxsus funksiyaning tasviriy parametrlarni berib, har bir uchlar uchun alohida yoki barcha uchlar uchun barobar.

Birinchi usul buyruqlar yordamida qo'llaniladi:

void **glTexCoord [1 2 3 4][s i f d]** (type *coord*)

void **glTexCoord[1 2 3 4][s i f d]**v (type **coord*)

Hozirgi vaqtda ko'pincha teksturalar koordinatalarini beruvchi **glTexCoord2*(type s, type t)** ko'rinishdagi buyruqlari ishlatiladi. Teksturaning hozirgi paytdagi koordinatalar tushunchasi hozirdagi norma tushunchasiga o'xshash bo'lib, uchlar atributi hisoblanadi. Ammo hatto kub uchun teksturaning mos koordinatalarini topish yetarlicha murakkab mashg'ulot hisoblanadi, shuning uchun GLU kutubxonasida doira, silindr va disk singari primitivlarni qurish buyruqlari, shuningdek, ularga teksturani qo'yish ko'zda tutilgan. Buning uchun quyidagi

void **gluQuadricTexture** (GLUquadricObj* *quadObject*, GLboolean *textureCoords*)

buyruqni **GL_TRUE** ga teng bo'lgan *textureCoords* parametri bilan chaqirish yetarli. Shunda joriy tekstura primitivlar ustiga avtomatik qo'yiladi.

Ikkinchi usul buyruqlar yordamida qo'llaniladi.

void **glTexGen [i f d]** (GLenum *coord*, GLenum *pname*, GLtype *param*)

void **glTexGen[i f d]v** (GLenum *coord*, GLenum *pname*, const GLtype **params*)

Coord parametri qaysi koordinataga formula berilishini belgilaydi va **GL_S, GL_T** qiymatini o'zlashtirishi mumkin; *pname* keyingi qiymatlardan biriga teng bo'lishi mumkin:

GL_TEXTURE_GEN_MODE teksturani olish uchun funksiyani belgilaydi.

Bu holatda *param* argumenti qiymatlarni qabul qiladi:

GL_OBJECT_LINEAR tekstura koordinatalarining mos qiymati tekislikkacha bo'lgan masofani, *pname* qiymati yordamida beriluvchi **GL_OBJECT_PLANE** belgilaydi. Formula quyidagi ko'rinishda bo'ladi: $g=x*xp+y*yp+z*zp+w*wp$, g–tekstura koordinatasi (s yoki p), x, y, z, w – nuqtalar koordinatasi. xp, yp, zp, wp – tekislik tenglamasi koeffitsiyentlari. Formulada obyekt kordinatalari ishlatiladi.

GL_EYE_LINEAR oldingi qiymatga o'xshash, faqat formulada koordinatalar ishlatiladi, ya'ni obyektning tekstura

koordinatalari ushbu holatda mazkur obyektning joylashishiga bog'liq bo'ladi.

GL_SPHERE_MAP obyektning sirtidan aks etish imkonini beradi. Tekstura obyekt atrofida xuddi «aylanayotganga» o'xshaydi. Ushbu usul uchun koordinatalar to'ri ishlatiladi va kerakli normalar beriladi.

GL_OBJECT_PLANE tekislik berishga imkon beradi, ungacha bo'lgan masofani inersiya yordamida ishlatiladi, agar **GL_OBJECT_LINEAR** tartibi o'rnatilgan bo'lsa. Bu vaziyatda *params* parametri tekislik tenglamasining to'rtta koeffitsiyenti massivga ko'rsatgich hisoblanadi.

GL_EYE_PLANE Oldingi qiymatga o'xshash. **GL_EYE_LINEAR** rejimi uchun tekislikni berish imkoniyatini beradi.

Tekstura koordinatalarini tayinlashning avtomatik rejimini o'rnatish uchun glEnable buyrug'ini **GL_TEXTURE_GEN_S** yoki **GL_TEXTURE_GEN_P** parametrlari bilan chaqirish kerak.

Animatsiya va tekstura olinishidan foydalanilgan dastur D.3 ilovada keltirilgan.

Nazorat savollari:

1. Tekstura o'zi nima va qanday maqsadda ishlatiladi?
2. Tekstura bilan ishlash ketma-ketligini tushuntiring?
3. Tekstura koordinatalari nima va ularni obyekt uchun qanday berish mumkin?
4. Agar tekstura o'zida devorda osib qo'yiladigan suratni aks ettiradigan vaziyatda material (GL_MODULATE, GL_REPLACE) bilan o'zaro ta'sirlashishning qanday usulidan foydalanish zarur?
5. Obyektga tekstura berish buyrug'i qanday ifodalanadi?
6. OpenGL da tekstura koordinatalarini generatsiyalash usullaridan sizga ma'lum bo'lganlarini keltiring?
7. Teksturani batafsil tekshirish bosqichi (mip-mapping) nima uchun ishlatiladi?
8. Teksturani filtratsiyalash rejimi nima va OpenGL da ular qanday beriladi?

Tayanch iboralar: Tekstura, tekstura obrazi, tekstura parametrlari, detallashtirish, teksturalar qo'yish, teksturali koordinata.

3.7. Piksellar ustida amallar

Uchlar koordinatalarini o'zgartirish bo'yicha barcha amallarni olib borilgandan so'ng rang va boshqalarni hisoblashda OpenGL *rasterizatsiyalash* bosqichiga o'tadi. Bu bosqichda teksturani qo'yish, tuman effektlarini qo'yish, barcha primitivlarni rasterizasiyalash ishlari amalga oshiriladi. Ushbu jarayonning natijasi har bir primitiv uchun kadr buferi sohasida egallangan, ushbu sohaning har bir pikseliga rang va chuqurlik qiymati yozilishi hisoblanadi.

OpenGL ushbu axborotni kadr buferida yangilangan ma'lumotlarni yozish uchun ishlatadi. Buning uchun OpenGL nafaqat alohida qayta ishlov berish piksellar konveyeriga ega, shuningdek, unda qo'shimcha turli vazifalarni bajaradigan bufer ham mavjud. Bu dasturlovchiga eng quyi darajada vizual jarayonni nazorat qilish imkonini beradi.

OpenGL grafik kutubxonasi quyidagi buferlar ishini ta'minlaydi:

- rangning bir qancha buferlari
- chuqurlik buferi
- to'plovchi bufer (akkumulyator)
- niqob buferi

Rang buferlari guruhida kadr buferi ham mavjud, lekin, bunday bufer bir qancha bo'lishi mumkin. Ikkitali buferizatsiyalashdan foydalanganda ishchi (front) va fonli (back) bufer haqida gap boradi. Qoida sifatida fonli buferda dastur tasvirni hosil qiladi, keyin uni bir yo'la ishchi buferiga ko'chiradi. Ekranda faqatgina rang buferlari haqidagi axborot namoyon bo'lishi mumkin.

Chuqurlik buferi ko'rinmas sirtlarni o'chirish uchun ishlatiladi va ular bilan to'g'ridan-to'g'ri ishlash kamdan-kam talab qilinadi.

To'plovchi bufer har xil operatsiyalar uchun ishlatiladi. U haqda to'liq ma'lumot 3.3-bo'limda keltirilgan.

Niqob buferi piksellar niqobini (trafaretlarni) shakllantirish uchun ishlatiladi. Umumiy massivlardan kerakli ekranga uzatish lozim deb topilgan piksellarni qirqib beradi.

Tasvirlarni aralashtirish. Shaffoflik

Har xil shaffof obyektlar - oyna, shaffof idishlar va boshqalar ko'pincha amaliyotda (hayotda) tez-tez uchraydi, shuning uchun interfaol grafikada ham shunday obyektlarni tashkil qilish kerak. OpenGL dasturchiga yarim shaffof obyektlar bilan ishlash mexanizmini taqdim etadi va buning qisqacha ta'rifi ushbu bo'limda beriladi.

Shaffoflik ranglarni aralashtirishning maxsus tartibi bilan amalga oshiriladi (blending). Aralashtirish algoritmi allaqachon buferda saqlanayotgan mos piksellar rangi bilan kirish piksellari (ya'ni bufer kadrida joylashgan «nomzodlar») deb ataluvchi ranglarni kombinatsiya qiladi. Aralashtirish uchun rangning to'rtinchi qismi - alfa qism ishlatiladi, shuning uchun bu tartib alfa-aralashtirish deb ham ataladi. Dastur asosiy ranglar bilan qancha tez ishlasa, xuddi shunday alfa qism bilan katta tezlikda ishlaydi, ya'ni primitivning har bir uchi yoki har bir piksel uchun tezkorlik qiymatini beradi.

Rejim glEnable(GL_BLEND) buyrug'i yordamida o'rnatiladi.

Aralashtirish parametrlarini quyidagi buyruq yordamida aniqlash mumkin:

void **glBlendFunc** (enum *src*,enum *dst*)

src parametri pikselning dastlabki rangi k_1 koeffitsiyentini qanday olish mumkinligini belgilasa, *dst* parametri bufer kadridagi rang uchun k_2 koeffitsiyentini olish usulini beradi.

Natijaviy rangni olish uchun quyidagi formula ishlatiladi: $res=s_{src}*k_1+c_{dst}*k_2$, bu yerda s_{src} – dastlabki piksel rangi, c_{dst} – bufer kadri ichidagi piksel rangi (res, k_1, k_1, s_{src}, c_{dst} – to'rt qismli RGBA-vektori).

Tez-tez ishlatiladigan src va dst argumentlarining qiymatlarini keltiramiz.

GL_SRC_ALPHA	$k=(A_s,A_s,A_s,A_s)$
GL_SRC_ONE_MINUS_ALPHA	$k=(1,1,1,1)-(A_s,A_s,A_s,A_s)$
GL_DST_COLOR	$k=(R_d,G_d,B_d)$
GL_ONE_MINUS_DST_COLOR	$k=(1,1,1,1)-(R_d,G_d,B_d,A_d)$
GL_DST_ALPHA	$k=(A_d,A_d,A_d,A_d)$
GL_DST_ONE_MINUS_ALPHA	$k=(1,1,1,1)-(A_d,A_d,A_d,A_d)$

GL_SRC_COLOR $k=(R_s,G_s,B_s)$
GL_ONE_MINUS_SRC_COLOR $k=(1,1,1,1)-(R_s,G_s,B_s,A_s)$

Misol:

Shaffof obyektlarning natijasini amalga oshirishni taqdim etamiz. Shaffoflik koeffitsiyenti rangning alfa-qismi bilan beriladi. 1 – shaffof bo'lmagan obyekt deylik, 0 mutlaqo shaffof, ya'ni ko'rinmaydigan deb hisoblaymiz. Amalga oshirish uchun quyidagi kod xizmat qiladi:

```
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA,GL_SRC_ONE_MINUS_ALP
HA);
```

Masalan, yarim shaffof uchburchakni quyidagi ko'rinishda berish mumkin:

```
glColor3f(1.0, 0.0, 0.0, 0.5);
glBegin(GL_TRIANGLES);
glVertex3f(0.0, 0.0, 0.0);
glVertex3f(1.0, 0.0, 0.0);
glVertex3f(1.0, 1.0, 0.0);
glEnd();
```

Agar sahnada bir-birini yopib turadigan bir qancha shaffof obyektlar bo'lsa, aniq natijani faqatgina quyidagi shartlar bajarilgan hollarda kafolatlash mumkin:

- Barcha shaffof obyektlar shaffof bo'lmagan obyektlardan keyin chiqadi.
- Obyektlar shaffof bo'lib chiqishida chuqurlik qisqarishi bo'yicha tartiblangan bo'lishi kerak. Bu degani, kuzatuvdan uzoqlashganlar birinchi bo'lib chiqa boshlaydi.

OpenGL da buyruqlarga kelish tartibiga qarab ishlov beriladi, shu sababli keltirilgan talablarni amalga oshirish uchun tegishli tartibda glVertex*() buyrug'ini chaqirish va kerakli joyga qo'yish yetarli, lekin, bu ham umumiy holda noto'g'ri.

To'plovchi bufer.

To'plovchi bufer (*accumulation buffer*) – bu OpenGL qo'shimcha buferlaridan biri. Unda vizuallashtirilgan tasvirni aniqlash

mumkin, maxsus piksel operatsiyalarini ishlatgan holda to'plovchi bufer keng qo'llaniladi.

Tasvir quyidagi buyruqni tanlash orqali buferdan olinadi:

void **glReadBuffer** (enum *buf*)

buf argumenti o'qish uchun buferni aniqlaydi. *buf* qiymati **GL_BACK**, **GL_FRONT** larga teng bo'lib, o'qish uchun mos buferlarni aniqlaydi. Ekranlashtirilmagan buferni **GL_BACK** bosh piksellar sifatida beradi. **GL_FRONT** – natijaning joriy tarkibiy oynasi. Buyruq qiymatga ega bo'ladi, agar dublikatlangan (ikkinchi nusxa) buferizasiyasi bo'lsa. Aks holda faqat bitta bufer ishlatiladi, chiqarish oynaga mos (jiddiy aytganda, OpenGL ga qo'shimcha bir to'plam buferi bor. Ular sterio tasvirda ishlatiladi, lekin, biz ularni hozir ko'rib chiqmaymiz).

To'plovchi bufer qo'shimcha rang buferi hisoblanadi. U ko'rinish natijalari uchun to'g'ridan to'g'ri ishlatilmaydi, lekin, ular rang buferlaridan biriga natijadan so'ng qo'shiladi. Quyida keltirilgan, har xil amallarni qo'llab, buferda tasvirni yig'ish mumkin.

Keyin hosil bo'lgan tasvirni to'plovchi buferdan bitta rang buferiga quyidagi buyruq yordamida yozuv bo'yicha o'tkazish mumkin.

void **glDrawBuffer** (enum *buf)*

buf qiymati **glReadBuffer** buyrug'idagi mos argument qiymatiga o'xshash.

To'plovchi bufer bilan barcha amallar quyidagi buyruq bilan nazorat qilinadi

void **glAccum** (enum *op*,GLfloat *value*)

op argumenti piksellar ustida amallarni belgilaydi va quyidagi qiymatlarni qabul qiladi:

GL_LOAD Piksel o'qish uchun tanlangan buferdan olinadi. Uning qiymati *value* ga ko'paytiriladi va to'plovchi buferga kiritiladi.

GL_ACCUM Oldingiga o'xshash, ammo, qiymatlar ko'paytirilgandan keyin hosil bo'lganlar buferda mavjud bo'lganlar bilan qo'shiladi.

GL_MULT Bu amal buferdagi har bir pikselni *value* ga ko'paytiradi.

GL_ADD Oldingiga o‘xshash, faqat ko‘paytirish o‘rniga qo‘shish ishlatiladi.

GL_RETURN Tasvir to‘plovchi buferdan yozish uchun buferga ko‘chiriladi. Undan oldin har bir piksel *value* ga ko‘paytiriladi.

Aytish joizki to‘plovchi buferni ishlatish uchun glEnable buyrug‘ini chaqirish shart emas. Faqatgina buferning o‘zini initsializatsiyalash, faollashtirish yetarli.

Rasterizatsiyalashdagi (bo‘laklash) xatoliklarni bartaraf qilish uchun to‘plovchi buferdan foydalanishga oid misol 3.8 bo‘limda keltirilgan.

Niqob buferi.

Kadr buferiga piksellarni chiqarayotganda ayrim hollarda ba’zi piksellarni chiqarish zaruriyati tug‘iladi, ya’ni tasvirga trafaret (niqob) qo‘yiladi. Buning uchun OpenGL niqob buferini taqdim etadi (stencil buffer). Niqob qo‘yishdan tashqari, bu bufer ancha qiziqarli imkoniyatlar namoyon etadi.

Pikselni kadr buferiga joylashtirishdan oldin, OpenGL ning vizuallashtirish mexanizmi berilgan qiymatlar va niqob buferdagi qiymatlar o‘rtasida solishishtirish amalini (testlash) bajarish imkonini beradi. Agar test o‘tkazilsa, piksel kadr buferida chiziladi.

Solishtirish mexanizmi juda ham moslashuvchan va u quyidagi buyruqlar tomonidan nazorat qilinadi:

void **glStencilFunc** (enum *func*, int *ref*, uint *mask*)

void **glStencilOp** (enum *sfail*, enum *dpfail*, enum *dppass*)

glStencilFunc buyrug‘ining *ref* argumenti solishtirish uchun qiymat beradi. U 0 dan 2^s-1 gacha bo‘lgan qiymatlarni qabul qiladi. s – niqob buferida nuqtadagi bitlar soni.

func argumenti yordamida solishtirish funksiyasi beriladi. U quyidagi qiymatlarni qabul qilishi mumkin:

GL_NEVER test hech qachon o‘tkazilmaydi, ya’ni har doim *false* qaytadi.

GL_ALWAYS test har doim o‘tkaziladi.

GL_LESS, GL_LEQUAL, GL_EQUAL,

GL_GEQUAL, GL_GREATE, GL_NOTEQUAL test o‘tkaziladi, agar *ref* mos holda niqob buferdagi qiymatdan kam bo‘lsa, kam yoki teng, teng, ko‘p, ko‘p yoki teng, yoki teng emas.

Mask argumenti qiymatlar uchun niqob tayyorlaydi, ya'ni bu test uchun natijada quyidagi formulaga ega bo'lamiz: ((*ref* AND *mask*) *op* (svalue AND *mask*)).

glStencilOp buyrug'i niqob buferidagi piksellar ustidan harakat uchun berilgan testning ijobiy yoki salbiy natijasini ko'rsatish uchun ishlatiladi.

sfail argumenti testning natijasi salbiy bo'lgan holatda harakatni belgilaydi va quyidagi qiymatlarni qabul qilishi mumkin:

GL_KEEP, GL_ZERO, GL_REPLACE,
GL_INCR, GL_DECR, GL_INVERT niqob buferidagi qiymatlarni mos holda saqlaydi, uni 0 ga tushiradi, berilgan qiymat (*ref*) ga o'zgartiradi, ko'paytiradi, kamaytiradi yoki bitlarga bo'ladi.

dpfail argumenti z-buferi chuqurligida testning natijasi salbiy bo'lgan hollarda harakatni belgilaydi, *dppass* argumenti esa ushbu testning natijasi ijobiy bo'lgan hollarda harakatni belgilaydi. Argumentlar *sfail* argumentidagi qiymatlarni qabul qiladi.

Niqoblashni yoqish uchun glEnable(GL_STENCIL_TEST) buyrug'ini bajarish, ishlatish kerak. Niqob buferi soya tushishi, akslanish. bir rasmdan boshqa rasmga tekis o'tish va boshqalar singari maxsus effektlarni yaratishda foydalaniladi.

Rasterizatsiyalarni boshqarish.

Primitivlarni rasterizatsiyalashni bajarish usullarini glHint (target, mode) buyrug'i bilan qisman tartibga solish mumkin, bu yerda *target* – nazorat qilinayotgan harakat ko'rinishi bo'lib, quyidagi qiymatlardan birini qabul qiladi.

GL_FOG_HINT – tuman qo'yishda hisoblashning aniqligi. Hisoblash piksellar (yuqori aniqlikda) bo'yicha bajarilishi mumkin yoki faqat uchlarda. Agar OpenGL piksellar bo'yicha hisoblashni amalga oshirilishini ta'minlamasa, unda faqat uchlar bo'yicha hisoblash bajarilishi amalga oshiriladi.

GL_LINE_SMOOTH_HINT – sifatlarni to'g'ri boshqarish. *mode* qiymatida **GL_NICEST** teng bo'lib, to'g'rilikda ko'p sonli piksellar hisobiga to'g'rilik pog'onasi kamayadi.

GL_PERSPECTIVE_CORRECTION_HINT – teksturalar qo'yish va ranglarni hisoblashda koordinatalar interpolyatsiyasining

aniqligi. Agar OpenGL **GL_NICEST** rejimini quvvatlamasa, unda koordinatalarning chiziqli interpolyatsiyasi amalga oshadi.

GL_POINT_SMOOTH_HINT – nuqtalar sifatini boshqarish. **Mode** parametr qiymatida **GL_NICEST** ga teng bo'lganda, nuqtalar doira sifatida chiziladi.

GL_POLYGON_SMOOTH_HINT – ko'pburchak tomonlarini chiqarish sifatini boshqarish.

mode parametri quyidagi shaklda interpolyatsiyalanadi:

GL_FASTEST – ancha tez algoritm ishlatiladi.

GL_NICEST – eng yaxshi sifatni ta'minlaydigan algoritm ishlatiladi.

GL_DONT_CARE - algoritmni tanlash amalga oshirishga bog'liq.

Shuni ta'kidlash lozimki, dasturchi glHint() buyrug'i bilan primitivlarni rasterizatsiyalashning u yoki bu jihatlariga nisbatan faqatgina o'zining xohishidagini belgilashi mumkin. OpenGL ni aniq amalga oshirilishi ushbu o'rnatishni e'tiborsiz qoldirishga haqli.

Ahamiyat bering, glHint() buyrug'ini glBegin()/glEnd() operatorli qavslari o'rtasida chaqirish mumkin emas.

Nazorat savollari:

1. OpenGL da qanday tasvir buferlaridan foydalaniladi?
2. glBlendFunc buyrug'idan nima uchun foydalaniladi?
3. OpenGL da tasvir buferlaridan nima uchun foydalaniladi?
4. Shaffof obyektlarga nimalar kiradi?
5. To'plovchi bufer nima uchun ishlatiladi? U bilan bog'liq bo'lgan misol keltiring.
6. OpenGL da natijaviy tasvirga qanday qilib niqobni qo'yish mumkin?
7. glHint() buyrug'i nima uchun qo'llaniladi, tushuntiring.
8. glHint(GL_FOG_HINT, GL_DONT_CARE) buyrug'ini bajarishning qanday effektlari bor?

Tayanch iboralar: Rasterizatsiyalash, kadr buferi, piksellar konveyeri, rang buferi, chuqurlik buferi, to'plovchi bufer, niqob buferi, shaffoflik.

3.8. OpenGL da ishlash usullari

Ushbu paragrafda biz OpenGL yordamida, to‘g‘ridan-to‘g‘ri quvvatlanishi kutubxona standartida mavjud bo‘lmagan qiziqarli vizual effektlarni yaratishni ko‘rib chiqamiz.

Pog‘onalilikni bartaraf qilish.

Pog‘onalilikni bartaraf qilish (*antialiasing*) masalasidan boshlaymiz. Pog‘onalilik effekti (*aliasing*) buferning chekli (qoida sifatida, katta bo‘lmagan) ruxsati hisobiga kadr buferida primitivlarni rasterizatsiyalashdagi xatoliklar natijasida yuzaga keladi. Ushbu muammoni hal qilish bo‘yicha bir necha yondashishlar mavjud. Masalan, olingan tasvirga filtrlashni qo‘llash mumkin. Shuningdek, ushbu effektni, har bir primitiv ko‘rinishini tekislash, rasterizatsiyalash bosqichida bartaraf qilish mumkin. Bu yerda biz barcha sahnani butun holi uchun o‘xshash umumiy dalillar bilan bartaraf qilish imkonini beruvchi usulni ko‘rib chiqamiz.

Har bir kadr uchun sahnani bir necha marta chizish zarur, har bir o‘tishda kamera boshlang‘ich holatiga nisbatan ozroq suriladi. Kameralarning joylashishi, masalan, doira hosil qilish mumkin. Agar kamerani surilishi nisbatan kam bo‘lsa, unda diskretizatsiya xatolari har xil bo‘lishi mumkin, va, olingan tasvirni o‘rtacha ko‘rinishga olib kelguncha biz tekis tasvirni olamiz.

Kuzatuvchi joyini ko‘chirish osonroq, lekin, undan oldin ko‘chirish joyini shunday hisoblash kerakki, ekranga uzatilgan koordinatalar qiymati, piksellar yarmidan oshmasligi lozim.

Barcha hosil bo‘lgan tasvirlarni to‘plovchi buferda $1/n$ koeffitsiyenti bilan saqlaymiz, bu yerda n-har bir kadr uchun o‘tish yo‘laklari soni. Bunday yo‘laklar qancha ko‘p bo‘lsa, unumdorlik shuncha pasayadi, lekin, natija yaxshilanadi.

```
for(i=0; i<samples_count;++i)
/* odatda samples_count 5 dan 10 gacha bo'lgan chegarada
yotadi */
{
ShiftCamera(i); /* kamerani ko'chiramiz */
RenderScene();
if (i==0)
/* birinchi iteratsiyada tasvirni yuklaymiz */
```

```
glAccum(GL_LOAD,1/(float)samples_count);
else
/* avval mavjud bo'lganlarga qo'shamiz */
glAccum(GL_ACCUM,1/(float)samples_count);
}
/*Olingan natijani orqaga boshlang'ich buferga yozamiz */
glAccum(GL_RETURN,1.0);
```

Shuni aytish joizki, barcha sahna uchun pog'onalilikni bartaraf qilish, qoida sifatida vizuallashtirish unumdorligining pasayishiga jiddiy ta'siri bilan bog'liq. Negaki, har bir sahna bir necha marotaba chiziladi. Zamonaviy tezlashtiruvchilar, odatda, tasvirni qayta diskretizatsiyalash deb atashga asoslangan boshqa usullarni apparatli amalga oshiradi.

Soyalarni qurish.

OpenGL da bazaviy buyruqlar darajasida soyalarni ko'rishning ichki ta'minlanishi mavjud emas. Buning ahamiyatli bosqichi shundaki, ularni qurishda ko'pgina algoritmlarning mavjudligi OpenGL funksiyalari yordamida amalga oshiriladi. Soyalarning borligi uch o'lchovli tasvirning realligiga kuchli ta'sir o'tkazadi, shuning uchun ularni qurishga bo'lgan yondashuvlardan birini ko'rib chiqamiz.

Soyalarni qurish uchun mo'ljallangan ko'pgina algoritmlar perspektiv proeksiyalashning turlangan tamoyillarini ishlatadi. Bu yerda eng oddiy usullardan biri ko'rib chiqiladi. Uning yordamida uch o'lchovli obyektni tekislikka tushirib soya hosil qilish mumkin.

Umumiy yondashuv shunday: obyektning barcha nuqtalari uchun ularning proeksiyasi, ba'zi berilgan tekislikdagi yorug'lik manbasida joylashgan ushbu nuqta va nuqtani bog'lovchi vektorga parallel deb topiladi. Shundan so'ng berilgan tekislikda butunligicha yotuvchi yangi obyektga ega bo'lamiz. Ushbu obyekt dastlabki soya hisoblanadi.

Ushbu usulning matematik asosini ko'rib chiqamiz:

Berilgan bo'lsin:

P – uch o'lchovli fazoda soyani tushiruvchi nuqta.

L – ushbu nuqtaga yorituvchi yorug'lik manbasi holati.

S=a(**L**-**P**)-**P** - nuqta, unga P nuqtasi o‘z soyasini tashlaydi, bu yerda a – parametr.

Taxmin qilish mumkinki, soya $z = 0$ tekislikka tushadi. Bunday vaziyatda $a=z_p/(z_l-z_p)$.

Demak,

$x_s = (x_p z_l - z_l z_p) / (z_l - z_p)$,

$y_s = (y_p z_l - y_l z_p) / (z_l - z_p)$,

$z_s = 0$.

Bir jinsli koordinatalarni kiritamiz:

$x_s=x_s'/w_s'$

$y_s=y_s'/w_s'$

$z_s=0$

$w_s'=z_l-z_p$

Bu yerda S koordinatalari matritsani quyidagi ko‘rinishda ko‘paytirish orqali hosil qilinishi mumkin:

$$[x_s' \quad y_s' \quad 0 \quad w_s'] = [x_s \quad y_s \quad z_s \quad 1] \begin{bmatrix} z_l & 0 & 0 & 0 \\ 0 & z_l & 0 & 0 \\ -x_l & -y_l & 0 & -1 \\ 0 & 0 & 0 & z_l \end{bmatrix}$$

Algoritm ixtiyoriy tekislikka tushadigan soyani hisoblashi uchun, bir jinsli koordinatalarda berilgan **S** va **P** o‘rtasidagi chiziqda ixtiyoriy nuqtani ko‘rib chiqamiz:

$$G = \begin{bmatrix} x_n \\ y_n \\ z_n \\ d \end{bmatrix}$$

a**P**+b**L**, a va b – skalyar parametrlar.

Quyidagi matritsa normal koordinatalari orqali tekislikni beradi:

Ushbu R nuqta orqali yorug‘lik manbasidan o‘tkaziluvchi nur (nuqta), G tekislikni kesib o‘tadi, quyidagi tenglamani qanoatlantiruvchi a va b parametrlar bilan belgilanadi:

(a**P**+b**L**)**G** = 0

Bu yerdan quyidagiga ega bo‘lamiz: a(**PG**) + b(**LG**) = 0.

Ushbu tenglamani a = **(LG)**, b = - **(PG)** qanoatlantiradi.

Demak, topilgan nuqtaning koordinatalari **S=(LG)P-(PG)L**. Matritsa ko'paytmasining assotsiativligidan foydalanib, quyidagiga ega bo'lamiz:

S = P[(LG)I - GL], bu yerda **I** – birlik matritsa.

(LG)I - GL matritsasi ixtiyoriy tekislikda soyaga ega bo'lish uchun foydalaniladi.

OpenGL yordamida ushbu usulni amalga oshirishning ba'zi bir jihatlarini ko'rib chiqamiz.

Taxmin qilish mumkinki, floorShadow matritsasi oldin biz tomonimizdan **(LG)I - GL** formulasidan olingan edi. Quyidagi kod yordamida berilgan tekislik uchun soyani qurish mumkin:

```
/* ranglarni aralashtirishdan (blending) foydalanib soyani
yarim shaffof ko'rinishga olib kelamiz*/
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA,GL_ONE_MINUS_SRC_ALP
HA);
glDisable(GL_LIGHTING);
glColor4f(0.0, 0.0, 0.0, 0.5);
glPushMatrix();
/* soyani proeksiyalaymiz */
glMultMatrixf((GLfloat *) floorShadow);
/* proeksiyada sahnani vizuallashtiramiz */
RenderGeometry();
glPopMatrix();
glEnable(GL_LIGHTING);
glDisable(GL_BLEND);
/* Odatdagi rejimda sahnani vizuallashtiramiz */
RenderGeometry();
```

floorShadow matritsasi quyidagi funksiya yordamida (*) tenglamasidan olinishi mumkin:

```
/* parametrlar: plane – tekislik tenglamasi koeffitsiyentlari
lightpos – yorug'lik manbasi koordinatlarini
qaytaradi: matrix – natijaviy matritsa */
void shadowmatrix(GLfloat matrix[4][4], GLfloat plane[4],
GLfloat lightpos[4])
```

```
{
GLfloat dot;
dot = plane[0] * lightpos[0] +plane[1] * lightpos[1] +plane[2] *
lightpos[2] +plane[3] * lightpos[3];

matrix[0][0] = dot - lightpos[0] * plane[0];
matrix[1][0] = 0.f - lightpos[0] * plane[1];
matrix[2][0] = 0.f - lightpos[0] * plane[2];
matrix[3][0] = 0.f - lightpos[0] * plane[3];

matrix[0][1] = 0.f - lightpos[1] * plane[0];
matrix[1][1] = dot - lightpos[1] * plane[1];
matrix[2][1] = 0.f - lightpos[1] * plane[2];
matrix[3][1] = 0.f - lightpos[1] * plane[3];

matrix[0][2] = 0.f - lightpos[2] * plane[0];
matrix[1][2] = 0.f - lightpos[2] * plane[1];
matrix[2][2] = dot - lightpos[2] * plane[2];
matrix[3][2] = 0.f - lightpos[2] * plane[3];

matrix[0][3] = 0.f - lightpos[3] * plane[0];
matrix[1][3] = 0.f - lightpos[3] * plane[1];
matrix[2][3] = 0.f - lightpos[3] * plane[2];
matrix[3][3] = dot - lightpos[3] * plane[3];
}
```

Ko‘rishimiz mumkinki, shunday ko‘rinishda qurilgan soyalar ham bir qancha kamchiliklarga ega.

- Ta’riflangan algoritm tekislik cheksizligini ko‘zda tutadi va soyani chegara bo‘yicha kesmaydi. Masalan, agar qandaydir obyekt o‘z soyasini stolga tushursa, u chegara bo‘yicha kesilmaydi, va buning ustiga, stolning yon sirtiga «buriladi».
- Ayrim joylarda soya «Ikkilik» aralashmasi» (reblending) effekti bo‘lib kuzatilishi mumkin, ya’ni uchburchaklar proeksiyasi bir-birini qoplaydigan joyda qora dog‘lar kuzatilishi mumkin.

▫ Tekisliklar soni ortgani sari algoritmlar ham murakkablashib boradi, nega deganda, har bir tekislik uchun alohida sahna quriladi, hattoki soyani chegarada kesilish muammosi yechilsa ham.

▫ Ko'pincha soyalar noaniq chegaralarga ega, berilgan algoritmda esa u aniq chegaraga ega. Undan qisman qochish esa, bir necha yorug'lik soyasini hisoblash, bir joyda va keyinchalik natijasini aralashtirganda kelib chiqadi.

Birinchi va ikkinchi muammonining yechimi mavjud. Uning uchun niqob buferi ishlatiladi (3.7. paragrafga qarang).

Demak, kesish vazifasining geometrik natijasi (ushbu holatda, soya) ba'zi ixtiyoriy soha chegarasi bo'yicha va «Ikkilik aralashmasi» dan qochish hisoblanadi. Niqob buferi yordamida yechimning umumiy algoritmi quyidagicha:

1. Niqob buferini 0 qiymati bilan tozalaymiz.
2. Kesilgan doirani tasvirlaymiz, niqob buferiga 1 qiymatini o'rnatamiz.
3. Niqob buferida 1 joylashgan bo'lsa, o'sha atrofda soyani yasaymiz. Agar test muvaffaqiyatli o'tsa, bu sohada 2 qiymatini o'rnatamiz.

Endi bu bosqichlarni batafsil ko'rib chiqamiz.

1.

```
/* niqob buferini tozalaymiz */
glClearStencil(0x0);
/* test qo'shamiz*/
glEnable(GL_STENCIL_TEST);
```

2.

```
/* shart doim bajariladi va buferdagi qiymat 1 ga teng bo'ladi */
glStencilFunc (GL_ALWAYS, 0x1, 0xffffffff);
/* har qanday holatda niqob buferidagi qiymatni o'zgartiramiz */
glStencilOp (GL_REPLACE, GL_REPLACE, GL_REPLACE);
/* keyinchalik soya kesilishi bo'yicha geometriyani chiqaramiz */
RenderPlane();
```

3.

```
/* agar niqob buferidagi qiymat 1 ga teng bo'lsa, test rostlikni
beradi va shart bajarildi */
```

```
glStencilFunc (GL_EQUAL, 0x1, 0xffffffff);
/* agar soya allaqachon aniqlangan bo'lsa, buferdagi qiymat
2 ga teng */
glStencilOp (GL_KEEP, GL_KEEP, GL_INCR);
/* soyani chiqaramiz */
RenderShadow();
```

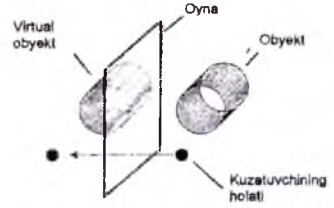
Ochiq aytganda, niqoblashtirishni ishlatganda ham z-bufer bilan bog'liq bo'lgan ayrim muammolar qoladi. Qisman, ayrim soyalar doirasi ko'rinmaydigan bo'lib qolishi mumkin. Bu muammoning yechimini topish uchun tekislik ta'riflanadigan tenglamani o'zgartirish yordamida soyani tekislik ustida ko'tarish mumkin. Boshqa usullarni ta'riflash ushbu qo'llanma doirasidan tashqaridir.

Obyektning ko'zgudagi aksi.

Bu bo'limda biz tekis obyektlarni akslantirish algoritmlarini ko'rib chiqamiz. Bunday aks qaytarishlar tasvir ko'rinishida katta axamiyatga ega va ularni oson amalga oshirish mumkin.

Algoritm ko'zgu bilan 2 ta to'liq sahnani jamlaydi: «haqiqiy» va «virtual» ko'zgu orqasida joylashgan. Demak, aksni chizish jarayoni ikki qismdan iborat: 1. Har daqiqa sahnaning vizualligi hamda 2. Virtuallarni qurilishi va vizuallashi har bir «haqiqiy» obyekt uchun uning ikkinchisi – aksi ko'riladi, ya'ni kuzatuvchi ko'zguda ko'riladi.

Misol uchun, devorda osilib turgan ko'zguli xonani tasavvur qilamiz. Xona va obyektlar huddi, agar ko'zgu shisha bo'lsa, ko'rinadigan, uning orqasida ham huddi shunaqa ko'zgu bor. Ularning tekislikka nisbatan simmetrik – aksi berilgan, ko'zgu tepasidan o'tkazilgan.



3.9-rasm. Ko'zgudagi tasvir.

Tekislik aksini yaratish algoritmning soddalashtirilgan varianti quyidagi qadamlardan iborat:

1. Sahnani har doimgidek chizamiz, ammo ko'zgu obyektlarisiz.
2. Niqob buferi ishlatgan holda, ekranga ko'zgu proeksiyasining keyingi natijalarini cheklaymiz.
3. Ko'zgu tekisligiga nisbatan akslangan sahnani vizuallashtiramiz. Shunda niqob buferi ko'zgu obyekti proeksiyasi shaklining natijasini cheklash imkonini beradi.

Bu harakat ketma-ketligi akslantirishning ishonchli samarasini olish imkonini beradi.

Bosqichlarni batafsil kurib chiqamiz.

Dastlab sahnani odatdagidek chizish kerak. Bu bosqichga batafsil to'htalmaymiz. Chizishdan oldin bevosita OpenGL buferi buferini tozalashni unutmaslik lozim:

```
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_B
IT|
GL_STENCIL_BUFFER_BIT);
```

Sahnani vizuallashtirish vaqtida obyektlarni chizmagan ma'qul, keyinchalik ular ko'zguda tasvirdek bo'lib qoladi.

Ikkinchi bosqichda ekranga ko'zgudagi obyekt proyeksiyasining keyingi natijalarini cheklash zarur.

Buning uchun niqob buferini to'g'rilaymiz va ko'zgu yasaymiz.

```
glEnable(GL_STENCIL_TEST);
/* shart doim bajariladi va buferdagi qiymat 1 ga teng bo'ladi */
glStencilFunc(GL_ALWAYS, 1, 0);
glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE);

RenderMirrorObject();
```

Natijada biz quyidagiga ega bo'ldik:

- Kadr buferida ko'zgu sohasidan tashqarida bo'lgan aniq chizilgan sahna;
- Ko'zgu sohasida (biz akslantirishni ko'rmoqchi bo'lgan joyda) niqob buferi qiymati 1 ga teng.

Uchinchi bosqichda ko'zgudagi obyekt tekisligiga nisbatan aks etuvchi sahnani chizish zarur.

Dastlab akslantirish matritsasini to‘g‘rilab olamiz. Akslantirish matritsasi tekislikka nisbatan butun geometriyani ko‘zguda akslantirishi lozim, ko‘zgu – aksi yotgan obyektni ko‘rsatishi kerak. Uni topish mumkin, masalan, quyidagi funksiyalar orqali (mashq sifatida mustaqil ushbu matritsani olishga harakat qiling).

```
void
reflectionmatrix(GLfloat reflection_matrix[4][4],
GLfloat plane_point[3],
Glfloat plane_normal[3])
{
GLfloat* p;
GLfloat* v;
float pv;

GLfloat* p = (Glfloat*)plane_point;
Glfloat* v = (Glfloat*)plane_normal;
float pv = p[0]*v[0]+p[1]*v[1]+p[2]*v[2];

reflection_matrix[0][0] = 1 - 2 * v[0] * v[0];
reflection_matrix[1][0] = - 2 * v[0] * v[1];
reflection_matrix[2][0] = - 2 * v[0] * v[2];
reflection_matrix[3][0] = 2 * pv * v[0];

reflection_matrix[0][1] = - 2 * v[0] * v[1];
reflection_matrix[1][1] = 1- 2 * v[1] * v[1];
reflection_matrix[2][1] = - 2 * v[1] * v[2];
reflection_matrix[3][1] = 2 * pv * v[1];

reflection_matrix[0][2] = - 2 * v[0] * v[2];
reflection_matrix[1][2] = - 2 * v[1] * v[2];
reflection_matrix[2][2] = 1 - 2 * v[2] * v[2];
reflection_matrix[3][2] = 2 * pv * v[2];

reflection_matrix[0][3] = 0;
reflection_matrix[1][3] = 0;
reflection_matrix[2][3] = 0;
reflection_matrix[3][3] = 1;
```

```
}

va sahnani yana bir marta chizamiz (ko'zgu obyektlarisiz)
glPushMatrix();
glMultMatrixf((float *)reflection_matrix);
RenderScene();
GlPopMatrix();

Nihoyat, niqoblashni o'chiramiz
glDisable(GL_STENCIL_TEST);
```

Shundan so'ng yana bir marta ko'zgudagi obyektni chiqarish mumkin, masalan, alfa-aralash bilan ko'zguni xiralashtirish effektini yaratish uchun va boshqalar.

E'tibor bering, keltirilgan metod aniq ishlaydi, agar sahnada ko'zgudagi obyektdan boshqa obyekt bo'lmasa. Shuning uchun ushbu algoritmning harakat ketma-ketligi bilan farqlanuvchi va geometriyada turlicha cheklovga ega bo'lgan har xil turlari mavjud.

Nazorat savollari:

1. Tasvirning pog'onalilik effekti natijasida nima yuz beradi?
2. Pog'onalilikni bartaraf etish algoritmini misol orqali tushuntiring?
3. Nima uchun OpenGL da soyalarni qurish ichki tomondan quvvatlanmaydi?
4. Soyalarni qurishda uchraydigan kamchiliklarni keltiring?
5. Ko'zgudagi obyektlarni vizuallashtirishning taklif etilgan metodini qisqacha yozing.
6. Ko'zguda sahnadagi boshqa obyektlar kuzatilsa u nima sababdan ishlamaydi?
7. Bunday holda nima tasvirlanadi?
8. Ushbu cheklanishdan qanday chiqish haqida o'ylab ko'ring?

Tayanch iboralar: Vizual effekt, pog'onalilik, pog'onalilik effekti, soyalar, ko'zgudagi aks.

3.9. Dasturni optimallashtirish

Ilovani tashkil etish.

Bir qarashda OpenGLga asoslangan ilovaning unumdorligi, birinchi navbatda OpenGLning o'z kutubxonasini amalga oshirish unumdorligini belgilanishi bo'lib ko'rinishi mumkin. Bu to'g'ri, ammo umumiy ilovani tashkil etish juda ham muhim.

Yuqori darajali optimallash.

Odatda OpenGL ostida ishlovchi dasturlardan interaktiv tezlikdagi yuqori sifatni vizuallashtirish talab etiladi. Lekin, qoida sifatida, u yoki boshqasiga birdan erishishning imkoni yo'q. Shunday ekan, sifat va unumdorlik o'rtasida o'zaro kelishuvni qidirish lozim. Ko'pgina turlicha yondashuvlar mavjud bo'lsa-da, ammo ularning batafsil tavsifi ushbu qo'llanma doirasidan tashqarida hisoblanadi. Faqatgina bir nechta misollarni keltiramiz:

- Animatsiya vaqtida past sifatli sahna geometriyasini tasvirlash mumkin, to'xtash vaqtida esa uni eng yuqori sifatda ko'rsatish mumkin. Interaktiv burish vaqtida (masalan, sichqoncha tugmasini bosishda) primitivlar sonini qisqartirish bilan modelni vizuallashtirish. Statistik tasvirlarni chizishda modelni to'liq tasvirlash.
- Obyektlar kuzatuvchidan uzoqda joylashadi va quyi murakkablikdagi model ko'rinishida taqdim etilishi mumkin. Bu esa OpenGL konveyerining barcha pog'onalari vazifalarini sezilarli darajada pasaytiradi. Kuzatuv maydonidan to'liqligicha tashqarida yotgan obyektlar, ko'rish piramidasida ularning oddiy hajmlari (sfera yoki kub) chegaralanishining tushishini nazorat qilish yordamida OpenGL konveyeriga uzatishsiz samarali kesilishi mumkin.
- Animatsiya vaqtida soxta ishlov berish, suzuvchi zalivka, teksturalarni o'chirib qo'yish mumkin. Xuddi shunday, bularning barchasini statistik tasvirlarni ko'rsatishda yoqish mumkin. Ushbu yondashuv OpenGLning apparatli quvvatlanmagan tizimi uchun ayniqsa, samaralidir.

Past darajali optimallash.

OpenGL yordamida tasvirlanadigan obyektlar, ayrim ma'lumotlar tuzilmasida saqlanadi. Bunday tuzilmalardan bir turi boshqasiga nisbatan foydalanish ancha samarali va vizuallashtirish tezligini belgilaydi.

OpenGL konveyeriga tez va samarali uzatish mumkin bo'lgan, ma'lumotlar tuzilmasidan foydalanish maqsadga muvofiq. Masalan, agarda biz uchburchak massivini tasvirlamoqchi bo'lsak, ushbu massivga ko'rsatkichlardan foydalanish uni qismlar bo'yicha OpenGL ga uzatishga qaraganda ancha samaraliroq.

Misol:

Tasavvur qilamiz, biz yer xaritasini chizishni amalga oshiruvchi ilovani yozayapmiz. Ma'lumotlar bazasidagi qismlardan biri – shaharlarning nomi, uzunligi va kengligi bo'yicha ro'yxati. Ma'lumotlarning mos tuzilmasi quyidagicha bo'lishi mumkin:

```
struct city
{
float latitute, longitude; /* shaharning holati */
char *name;              /* nomi */
int large_flag;     /* 0 = kichik, 1 = katta */
};
```

Shaharlar ro'yxati shunday massiv tuzilmasida saqlanishi mumkin. Faraz qilamiz, biz yozuvlar bilan turli o'lchamdagi nuqta ko'rinishida xaritada shahar chizish funksiyasini yozayapmiz:

```
void draw_cities( int n, struct city citylist[] )
{
int i;
for (i=0; i < n; i++)
{
if (citylist[i].large_flag)
glPointSize( 4.0 );
else
glPointSize( 2.0 );

glBegin( GL_POINTS );
glVertex2f( citylist[i].longitude,
```

```
citylist[i].latitude );
glEnd();
/* shahar nomini chizamiz */
DrawText(citylist[i].longitude,
citylist[i].latitude,
citylist[i].name);
}
}
```

Ushbu amalga oshirish quyidagi sabablarga ko‘ra muvaffaqiyatsiz:

- glPointSize davrning har bir interatsiyasi uchun chaqiriladi.
- glBegin va glEnd o‘rtasida faqat bitta nuqta chiziladi.
- uchlar qulay bo‘lmagan formatda belgilanadi.

Quyida ancha ratsional yechim keltirilgan:

```
void draw_cities( int n, struct city citylist[] )
{
int i;
/* dastlab kichkina nuqtalar chizamiz */
glPointSize( 2.0 );
glBegin( GL_POINTS );
for (i=0; i < n ;i++)
{
if (citylist[i].large_flag==0) {
glVertex2f( citylist[i].longitude,
citylist[i].latitude );
}
}
glEnd();
/* ikkinchi navbatda katta nuqtalar chizamiz */
glPointSize( 4.0 );
glBegin( GL_POINTS );
for (i=0; i < n ;i++)
{
if (citylist[i].large_flag==1)
{
glVertex2f( citylist[i].longitude,
citylist[i].latitude );
```

```
}
}
glEnd();
/* so‘ngra shaharlar nomini chizamiz */
for (i=0; i < n ;i++)
{
DrawText(citylist[i].longitude,
citylist[i].latitude,
citylist[i].name);
}
}
```

Bunday amalga oshirishda biz glPointSize buyrug‘ini ikki marta chaqiramiz va uchlar sonini glBegin va glEnd oralig‘ida oshiramiz.

Biroq, optimallashtirish uchun yana yo‘llar qoladi. Agar biz ma’lumotlar tuzilmasini almashtirsak, unda nuqtalarni chizish samaradorligini yanada oshirishimiz mumkin. Masalan:

```
struct city_list
{
int num_cities; /* ro‘yxatdagi shaharlar soni */
float *position;/* shahar koordinatalari */
char **name;    /* shaharlar nomiga ko‘rsatkich */
float size;     /* shaharni bildiruvchi nuqta o‘lchami */
};
```

Endi turli o‘lchamdagi shahar turlicha ro‘yxatda saqlanadi. Nuqtaning holati dinamik massivda alohida saqlanadi. Qayta tashkil etishdan so‘ng biz glBegin/glEnd ichida shartli operator zaruriyatini chiqarib tashlaymiz va optimallashtirish uchun uchlar massividan foydalanish imkoniga ega bo‘lamiz. Natijada biz qurgan funksiya quyidagi ko‘rinishga keladi:

```
void draw_cities( struct city_list *list )
{
int i;

/* nuqtani chizamiz */
glPointSize( list->size );
```

```
glVertexPointer( 2, GL_FLOAT, 0,
list->num_cities,
list->position );
glDrawArrays( GL_POINTS, 0, list->num_cities );
/* shahar nomini chizamiz */
for (i=0; i < list->num_cities ;i++)
{
DrawText(citylist[i].longitude,
itylist[i].latitude
citylist[i].name);
}
}
```

OpenGL da chaqirishlarni optimallashtirish.

OpenGL unumdorligini oshirishning ko'pgina imkoniyatlari mavjud. Bundan tashqari, optimallashtirishga bo'lgan turlicha yondashuvlar apparatli va dasturiy vizuallashtirishda turlicha effektlar bilan olib boriladi. Masalan, ranglar interpolyatsiyasi apparat ta'minotisiz juda qimmatli operatsiya bo'lishi mumkin, apparatli vizuallashtirishda ushlanish deyarli bo'lmaydi.

Quyidagi uslublarning har biridan keyin, simvollardan biri ko'rsatilgan va aniq tizim uchun ushbu uslubning ahamiyatini anglatuvchi kvadrat qavslar kuzatiladi:

- [A] – OpenGLning apparatli ta'minot tizimi uchun afzal
- [D] – dasturiy amalga oshirish uchun afzal
- [barchasi] – ehtimol barcha amalga oshirishlar uchun afzal

OpenGL da ma'lumotlarni uzatish

Biz quyida OpenGLda primitivlar haqidagi ma'lumotlarni uzatishda vaqtni minimallashtirish usullarini ko'rib chiqamiz.

Bog'langan primitivlardan foydalaning.

GL_LINES, **GL_LINE_LOOP**, **GL_TRIANGLE_STRIP**, **GL_TRIANGLE_FAN**, va **GL_QUAD_STRIP** singari bog'langan primitivlar, alohida chiziq yoki ko'pburchakka qaraganda kam uchlarni belgilash uchun talab etiladi. Bu OpenGLga uzatilayotgan ma'lumotlar sonini kamaytiradi. [barchasi]

Uchlar massivlaridan foydalaning.

Ko'pgina arxitekturalarda ko'p sonli chaqirishlarni (glVertex/glColor/glNormal) almashtirish uchlar massivi mexanizmida juda ham yutuqli bo'lishi mumkin. [barchasi]

Indekslangan primitivlardan foydalaning.

Ayrim hollarda hattoki bog'langan primitivlardan **GL_TRIANGLE_STRIP (GL_QUAD_STRIP)** foydalanganda ham uchlar nusxalanadi.

OpenGLda nusxalarni uzatmaslik uchun, shunday vazifalarni ko'paytiruvchi, glDrawElements() buyrug'idan foydalaning. (3.3. paragrafga qarang) [barchasi]

Kerakli massivlarni bitta buyruq bilan bering.

Buyruqni ishlatish o'rniga

glVertexPointer/glColorPointer/glNormalPointer bitta buyruqdan foydalanish ma'qul

void glInterleavedArrays (Glint format, Glsizei stride, void * ptr);

xuddi shunday, tuzilmaga ega bo'lsa,

```
typedef struct tag_VERTEX_DATA
{
float color[4];
float normal[3];
float vertex[3];
}VERTEX_DATA;
VERTEX_DATA * pData;
```

unda parametrlarni quyidagi buyruq yordamida uzatish mumkin:

glInterleavedArrays (GL_C4F_N3F_V3F, 0, pData);

bunda, birinchi to'rtta float rangga tegishli, keyingi uchta float normalga, va oxirgi uchta float uchlar koordinatalarini beradi. Buyruqning batafsil tavsifini OpenGL tasnifidan ko'ring. [barchasi]

Uchlar haqida ma'lumotlarni xotirada ketma-ket saqlang.

Ma'lumotlarni xotirada ketma-ket saqlanishi, asosiy xotira va grafik quyitizim o'rtasida almashinuv tezligini oshiradi. [A]

glVertex, glColor, glNormal i glTexCoord **vektorli versiyalaridan foydalaning.**

glVertex*(), glColor*() va boshqa funksiyalar, argument sifatida ko'rsatkichlarni (masalan, glVertex3fv(v)) qabul qilib, mos versiyalariga glVertex3f(x,y,z) nisbatan tez ishlashi mumkin. [barchasi]

Primitivlar murakkabligini kamaytiring.

Ko'pgina hollarda katta tekisliklarni keragidan ortiqcha qismlarga bo'lib tashlamaslik uchun ogoh bo'ling. Masalan, sifat va unumdorlikning eng yaxshi o'zaro nisbatini belgilash uchun GLU primitivlari bilan tajriba o'tkazing. Teksturalangan obyektlar, masalan, geometriyaning o'rtacha murakkabligi bilan sifatli tasvirlanishi mumkin.

Displey ro'yxatidan foydalaning.

Tez-tez chaqirilgan obyektlar uchun displey ro'yxatidan foydalaning. Displey ro'yxati grafik quyitizimida saqlanishi mumkin va demak, asosiy xotiradan ma'lumotlarni tez-tez olib o'tilishini olib tashlash mumkin. [A]

Uchlarning kerak bo'lmagan atributlarini ko'rsatmang.

Agar yorug'lik o'chirilgan bo'lsa, gl Normal buyrug'ini chaqirmang. Agar tekstura ishlatilmayotgan bo'lsa, glTexCoord buyrug'ini chaqirmang, va b. [barchasi]

glBegin/glEnd orasidagi ortiqcha kodlar sonini kamaytiring.

high-end tizimlarida maksimal unumdorlik, uchlar haqidagi axborot grafik qiyitizimga maksimal darajada tez uzatilishi uchun muhim. glBegin/glEnd o'rtasida ortiqcha kodlardan yiroqlashing.

Muvaffaqiyatsiz yechimga misol:

```
glBegin(GL_TRIANGLE_STRIP);
for (i=0; i < n; i++)
{
if (lighting)
{
glNormal3fv(norm[i]);
}
glVertex3fv(vert[i]);
```

```
}
glEnd();
```

Bu konstruksiya shunisi bilan yomonki, biz lighting o'zgaruvchisini har bir uch oldidan tekshiramiz. Bunday holatdan qochish mumkin, kodning qismlangan nusxasi hisobiga:

```
if (lighting)
{
glBegin(GL_TRIANGLE_STRIP);
for (i=0; i < n ;i++)
{
glNormal3fv(norm[i]);
glVertex3fv(vert[i]);
}
glEnd();
}
else
{
glBegin(GL_TRIANGLE_STRIP);
for (i=0; i < n ;i++)
{
glVertex3fv(vert[i]);
}
glEnd();
}
```

O'zgartirish

O'zgartirish o'zida, oyna koordinatalari, kesish, yorug'lik va boshqalarni berishda ishlatiladigan glVertex*() bilan ko'rsatiluvchi koordinatalar orqali uchlarning o'zgarishini namoyon etadi.

Yorug'lik

- Yorug'likning lokal manbalaridan foydalanishdan qochish, ya'ni manba koordinatalari (*x,y,z,*0) shaklida bo'lishi kerak. [D]
- Yorug'likning nuqtasimon manbasidan foydalanishdan qochish. [barchasi]
- Ikki tomonlama yoritishdan (two-sided lighting) foydalanishdan qochish. [barchasi]

- Yorug'lik va material parametrlaridagi manfiy koeffitsiyentlardan foydalanishdan qochish. [D]
- Yorug'likning lokal modelidan foydalanishdan qochish. [barchasi]
- **GL_ShININESS** materiali parametrining tez-tez almashishidan qochish. [D]
- OpenGLning ba'zi amalga oshirishlari yorug'likning bitta manbasi holi uchun optimallashtirilgan. [A, D]
- Yorug'likni oldindan hisoblab chiqish imkonini ko'rib chiqish. Normal o'rniga uchga rang berish orqali yorug'lik effektiga ega bo'lish mumkin. [barchasi]

Zarur bo'lmagan hollarda, normal vektorlarini normallashtirishni o'chirish.

glEnable/Disable(GL_NORMALIZE) buyrug'i normallardan foydalanishdan oldin normal vektorlarini normallashtirishni boshqaradi. Agarda siz glScale buyrug'idan foydalanmasangiz, unda normallashtirishni boshqa effektlarsiz o'chirish mumkin. Odatda bu band o'chirilgan bo'ladi. [barchasi]

Bog'langan primitivlardan foydalaning.

GL_LINES, **GL_LINE_LOOP**, **GL_TRIANGLE_STRIP**, **GL_TRIANGLE_FAN** va **GL_QUAD_STRIP** kabi bog'langan primitivlar OpenGL konveyeri ishini kamaytiradi, shuningdek, grafik quyitizimga uzatiluvchi ma'lumotlar sonini qisqartiradi.

Rasterizatsiya

Rasterizatsiya ko'pincha OpenGLni dasturiy amalga oshirishning cheklangan joyi hisoblanadi.

Zaruriyat bo'lmaganda ranglar interpolyatsiyasini o'chiring.

Ranglar interpolyatsiyasi odatda yoqilgan bo'ladi. Tekis soya tashlash rangning to'rtta qismi interpolyatsiyasini talab etmaydi va qoida sifatida, OpenGLda dasturiy amalga oshirishda tezroq. Apparatli amalga oshirish odatda soya tashlashning ikkala ko'rinishini ham bir xil tezlikda bajaradi. O'chirish uchun glShadeModel(GL_FLAT) buyrug'idan foydalaning. [D]

Zaruriyat bo'lmaganda chuqurlikka bo'lgan testni o'chiring.

Fon berilgan obyektlar, masalan, chuqurlashtirish testsiz chizilishi ham mumkin, agarda ular birinchi bo'lib vizuallashtiriiganda. [barchasi]

Poligonlarning teskari yoqlarini kesishdan foydalaning.

Yopiq obyektlar teskari yoqlarni glEnable(GL_CULL_FACE) kesish rejimini o'rnatish bilan chizilishi mumkin. Ba'zida bu ko'pburchaklarning yarmigacha olib tashlash imkonini beradi, ularni rasterizatsiya qilmasdan. [barchasi]

Piksellar bilan ortiqcha amallardan voz kecbing.

Rasterizatsiya bosqichida niqoblashtirish, alfa-aralashtirish va boshqa piksel bo'yicha operatsiyalar ko'p vaqt oladi. Foydalanilmayotgan barcha operatsiyalarni o'chiring. [barchasi]

Oyna o'lchami yoki ekran kengligini qisqartiring.

Rasterizatsiyalash vaqtini qisqartirishning oddiy usuli – chiziladigan piksellar sonini qisqartirish. Agar oynaning kichik o'lchami yoki ekranning kichik kengaytmasi qabul qilingan bo'lsa, unda bu rasterizatsiyalash tezligini oshirish uchun yaxshi yo'l hisoblanadi. [barchasi]

Teksturalash

Tekstura qo'yish dasturiy va apparatli amalga oshirishlarda muhim operatsiya hisoblanadi.

Tasvirlarni saqlashning samaradorli formatlaridan foydalaning.

GL_UNSIGNED_BYTE formati, odatda, OpenGLga teksturalarni uzatish uchun ancha mos keladi.

Displey ro'yxati yoki tekstura obyektlarida teksturalarni birlashtiring.

Agarda siz bir qancha teksturadan foydalanayotgan bo'lsangiz bu juda muhim va grafik quyitizimga videoxotirada teksturalarni joylashtirishni samarali boshqarish imkonini beradi. [barchasi]

Katta o'lchamdagi teksturalarni ishlatmang.

Katta bo'lmagan teksturalarga tez ishlov beriladi va xotirada kam joy egallaydi, shu jumladan, grafik quyitizimiga barcha teksturani saqlash imkonini beradi. [barchasi]

Katta bo'lmagan teksturalarni bittaga birlashtiring.

Agar siz bir qancha kichik teksturalarni ishlatadigan bo'lsangiz, ularni bitta katta o'lchamdagiga birlashtirish va kerakli tekstura ustida ishlash uchun tekstura koordinatalarini o'zgartirish mumkin. Bu teksturaning o'zgarishi sonini qisqartirish imkonini beradi.

Animatsiyalashtirilgan teksturalar.

Agar siz animatsiyalashtirilgan teksturadan foydalanmoqchi bo'lsangiz, tekstura ko'rinishini yangilamaslik uchun glTexImage 2D buyrug'idan foydalanmang. Buning o'rniga glTexSubImage2D yoki glTexCopyTexSubImage2D dan foydalaning.

Buferlarni tozalash

Rang, chuqurlik, niqob buferlari va to'plovchi buferni tozalash ko'p vaqt talab qilishi mumkin, ayniqsa, OpenGLni dasturiy amalga oshirishda. Bu bo'limda ushbu operatsiyani optimallashtirishga yordam berishi mumkin bo'lgan ba'zi usullar berilgan.

glClear **buyrug'idan ehtiyotkorlik bilan foydalaning.** [barchasi]

Kerakli barcha buferlarni glClear buyrug'i yordamida tozalang.

Noto'g'ri:

```
glClear(GL_COLOR_BUFFER_BIT);
if (stenciling)  /* niqob buferini tozalash kerakmi? */
{
glClear(GL_STENCIL_BUFFER_BIT);
}
```

To'g'ri:

```
if (stenciling) /* niqob buferini tozalash kerakmi? */
{
glClear(GL_COLOR_BUFFER_BIT
STENCIL_BUFFER_BIT);
}
else
{
glClear(GL_COLOR_BUFFER_BIT);
}
```

Och ranglar uzatilishini (dithering) o'chiring.

Buferni tozalashdan oldin och ranglar uzatilishini o'chiring. Odatda och ranglar o'rnatilgan va usiz tozalash o'rtasidagi farq sezilmaydi. [D]

Kichkina sohalarni tozalashda qaychilardan (scissors) foydalaning.

Agar siz butun buferni tozalamoqchi bo'lsangiz, berilgan soha bo'yicha tozalash chegarasi uchun glScissor() buyrug'idan foydalaning. [barchasi]

Rang buferini to'liqligicha tozalamang.

Agar sizning sahnangiz oynaning bir qismini egallasa, rangning butun buferini tozalashning hojati yo'q.

glClearDepth (d) buyrug'idan foydalanmang, bu yerda d!=1.0

Ayrim dasturiy amalga oshirishlar buferni 1.0. chuqurlikda tozalash uchun optimallashtirilgan. [D]

Har xil

Dastur yozish vaqtida GL xatolarini tekshiring. [barchasi]

OpenGL funksiyalaridan birini chaqirish vaqtida xatolikka yo'l qo'yilmaganligini tekshirish uchun glGetError() buyrug'ini chaqiring. Qoida sifatida, xatolik OpenGL buyruqlarining noto'g'ri parametrlari yoki buyruqlarning noto'g'ri ketma-ketligi hisobiga yuzaga keladi. Kodning yakuniy versiyasi uchun ushbu tekshirishni o'chiring, bu ishni sezilarli darajada sekinlashtiradi. Tekshirish uchun, masalan, shunday makrosdan foydalanish mumkin:

```
#include <assert.h>
#define ChECK_GL \
assert(glGetError() != GL_NO_ERROR);
```

Uni shunday ishlatish mumkin:

```
glBegin(GL_TRIANGLES);
glVertex3f(1,1,1);
glEnd();
ChECK_GL
```

glMaterial o'rniga glColorMaterial buyrug'idan foydalaning.

Agar sahnada obyektlar materiali hech bo‘lmaganda bitta parametr bo‘yicha farq qilsa, glColorMaterial buyrug‘i glMaterial buyrug‘iga nisbatan tez ishlaydi. [barchasi]

OpenGL holatida o‘zgarishlar sonini minimallashtirish.

OpenGL holatini o‘zgartiruvchi buyruqlar (glEnable/glDisable/ glBindTexture va boshq.), yaxlitlikni takroriy ichki tekshiradi, qo‘shimcha ma’lumotlar tuzilmasini yaratadi va boshqalarni chaqiradi, bu esa to‘xtalishlarga olib keladi. [barchasi]

glPolygonMode buyrug‘idan foydalanmaslikka harakat qiling.

Agar sizga ko‘pgina rangsiz ko‘pburchaklilarni chizish kerak bo‘lsa, primitivlar chizish rejimini o‘zgartirish o‘rniga glBegin buyrug‘ini **GL_POINTS**, **GL_LINES**, **GL_LINE_LOOP** yoki **GL_LINE_STRIP** bilan ishlating, bu esa tezroq bo‘lishi mumkin. [barchasi]

Albatta, bu tavsiyalar OpenGL ilovasini optimallashtirish bo‘yicha faqat imkoniyatlarning kichik qismini o‘zida qamraydi. Shunga qaramasdan, ulardan to‘g‘ri foydalanish sizning dasturingiz ishini jiddiy ravishda tezlashtirishi mumkin.

Nazorat savollari:

1. OpenGL ilovalarini yuqori darajali optimallashtirish usullaridan sizga ma’lum bo‘lganlarini keltiring?
2. OpenGLni past darajali optimallash deganda nima tushuniladi?
3. OpenGL unumdorligini oshirish imkoniyatlari.
4. Nima uchun bog‘langan primitivlardan foydalanish afzal ko‘riladi?
5. Quyidagi ikki buyruqdan qaysi biri OpenGLni tez ishlashini ta’minlaydi?

glVertex3f(1,1,1) yoki float vct[3] = {1,1,1}; glVertex3fv(vct)

Tayanch iboralar: Yuqori darajali optimallash, past darajali optimallash, chaqirishlarni optimallashtirish, teksturalash.

ILOVA A. GLUT - ILOVASINING TUZILISHI

GLUT kutubxonasi yordamida konsol ilovalar tuzilishini hosil qilishni qarab chiqamiz. Bu kutubxona plotformadan qat'iy nazar oynalar bilan ishlash uchun yagona interfeysni taminlaydi, shuning, uchun quyidagi ilova tuzilmalari Windows, Linux va boshqa OSlar uchun o'zgarmas bo'lib qoladi.

GLUT funksiyasi o'zining vazifalari bo'yicha bir qancha guruhlarda tavsiflanishi mumkin:

- initsiallash;
- hodisalarni qayta ishlashning boshlanishi;
- oynalarni boshqarish (Upravlenie oknami);
- menyularni boshqarish (Upravlenie menyu);
- teskari chaqiruvlar funksiyasini ro'yxatdan o'tkazish;
- indekslangan ranglar palitrasini boshqarish;
- shriftlarni aks ettirish;
- qo'shimcha geometrik shakllar (tor, konus va hokazolar)ni tasvirlash.

Initsiallash quyidagi funksiyalar yordamida amalga oshiriladi:

glutInit (int *argcp, char **argv)

argcp o'zgaruvchi bu main() funksiyasida tasvirlanadigan argc standart o'zgaruvchisidagi ko'rsatgichidir, argv esa ushbu funsiyadagi tasvirlanayotgan dastur boshlanishida uzatiladigan parametrlarga ko'rsatgich bo'ladi. Bu funksiya ilova oynasi strukturasi uchun zarur bo'lgan boshlang'ich harakatlarni amalga oshiradi va bir nechtagina GLUT funksiyalari uchungina chaqirilishi mumkin. Bularga:

glutInitWindowPosition (int x, int y)

glutInitWindowSize (int width, int height)

glutInitDisplayMode (unsigned int mode)

Birinchi ikkala funksiya mos ravishda oyna holati va o'lchamini ifodalaydi, oxirgi funksiya esa ma'lumotlarni har xil aks ettirishni ifodalaydi, bular «yoki» (« | «) bitlik operatsiyasi bilan birgalikda berilishi mumkin:

GLUT_RGBA RGBA rejimi. **GLUT_RGBA** yoki **GLUT_INDEX** rejimlari aniq ko'rsatilmagan holda ishlatiladi.

GLUT_RGB Bu ham **GLUT_RGBA** kabi.

GLUT_INDEX indeksirlangan ranglar rejimi (palitrlardan foydalanish). **GLUT_RGBA**ni bekor qiladi.

GLUT_SINGLE Alohida buferli oyna. Odatdagi bo'yicha foydalaniladi.

GLUT_DOUBLE Ikkitali buferli oyna. **GLUT_SINGLE** ni bekor qiladi.

GLUT_STENCIL Niqob buferli oyna.

GLUT_ACCUM To'plagich buferli oyna.

GLUT_DEPTH Chuqurlik buferli oyna.

Bu berilgan funksiyaning parametrlari ro'yxati to'liq emas, biroq ko'p holatlar uchun bular yetarli bo'ladi.

GLUT kutubxonasi funksiyasi hodisa-boshqaruv deb nomlangan mehanizmni amalga oshiradi. Bu initsiallashi davomida aniqlangan barcha hodisalarni birma-bir qayta ishlaydi, tegishli initsiallashdan keyin ishga tushadigan qandaydir ichki siklning borligini anglatadi. Bularga: sichqoncha chertkisi, oynaning yopilishi, oynalar hossalarining o'zgarishi, kursor holatining o'zgarishi, klavish bosilishi va hech narsa sodir bo'lmayotgan paytdagi hodisalar. Biror voqeaning bajarilishini davriy tekshirish o'tkazish uchun uni qayta ishlaydigan funksiyasini registratsiyadan o'tkazish kerak. Buning uchun funksiya ko'rinishi quyidagicha:

void **glutDisplayFunc** (void (*func) (void))

void **glutReshapeFunc** (void (*func) (int width, int height))

void **glutMouseFunc** (void (*func) (int button, int state, int x, int y))

void **glutIdleFunc** (void (*func) (void))

void **glutMotionFunc** (void (*func)(int x, int y));

void **glutPassiveMotionFunc** (void (*func)(int x, int y));

Shu parametr tipiga mos nomi beriladi. **glutDisplayFunc()** yordamida oyna ilovasi uchun kerak bo'ladigan tasvirlarni yaratish yoki qayta tiklash mumkin. Aniq ko'rsatilgan oyna uchun void **glutPostRedisplay** (void) funksiyasidan foydalanib yangilash qulay sanaladi.

glutReshapeFunc() orqali yangi o'lchamlar berishda foydalanuvchi oyna o'lchami o'zgarishi uchun qayta ishlash funksiyasi sozlanadi.

glutMouseFunc() funksiyasi aniqlaydi – sichqoncha orqali beriladigan buyruqlarni qayta ishlash uchun, **glutIdleFunc()** funksiyasi esa, foydalanuvchi hodisada qatnashmagan paytda o'zi har gal avtomatik chaqiriladi.

glutMotionFunc funksiyasi foydalanuvchi tomonidan sichqoncha ko'rsatkichi harakatlantirilganda, sichqoncha tugmasini ushlab turilganda bajariladi.

glutPassiveMotionFunc funksiyasi foydalanuvchi tomonidan sichqoncha ko'rsatkichi harakatlantirilganda sichqoncha tugmasiga hech qanday ta'sir ko'rsatilmaganda bajariladi.

void **glutMainLoop** (void) funksiyasi cheksiz ichki siklda barcha hodisalarni nazorat qilib, odatda GLUT kutubxonasi yordamida ixtiyoriy dasturning oxirida chaqiriladi. Quyidagi dastur ilova strukturasida animatsiyadan foydalanib ishlash ko'rsatilgan:

```
#include <GL/glut.h>
 void MyCut(void)
 {
/*Quyidagi kadrlarni belgilovchi, o'zgaruvchilarni
almashtiradigan kod */
...
};
void MyDisplay(void)
{
/* Kadrni tasvirlash OpenGL kodi */
...
/* Chizishdan keyin buferlarni joylashtiramiz */
glutSwapBuffers();
};
void main(int argcp, char **argv)
{
/* GLUT Initsializatsiyasi */
glutInit(&argcp, argv);
glutInitWindowSize(640, 480);
```

```
    glutInitWindowPosition(200, 200);
    /* Oynani ochish */
    glutCreateWindow("My OpenGL Application cutting");
    /* tartibni tanlash:ikkilangan bufer va RGBA ranglari */
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE |
GLUT_DEPTH);
    /* funksiyani chaqirishni registrasiyalash */
    glutDisplayFunc(MyDisplay);
    glutIdleFunc(MyCut);
    /* voqealarni qayta ishlash mexanizmini yoqish*/
    glutMainLoop();
    };
```

GLUT_DOUBLEni **GLUT_SINGLE**ga almashtirish uchun ilovada tasvirning statik ko'rinishini tuzish kerak bo'ladi, chunki bu holatda bitta bufer yetarli bo'ladi. *glutIdleFunc()* chaqiruv funksiyasini olib tashlaydi.

ILOVA B. GLU VA GLUT KUTUBXONALARINING PRIMITIVLARI

GLU va GLUT kutubxonasida amalga oshirilgan primitivlarni qurishning standart buyruqlarini ko'rib chiqamiz.

GLU kutubxonasida primitiv qurish uchun dastlab gluNew Quadric() buyrug'i yordamida quadric-obyekt ko'rsatkichini yaratish kerak, so'ngra, gluSphere(), gluCylinder(), gluDisk(), gluPartialDisk() buyrug'idan bittasini chaqirish kerak. Bu buyruqlarni alohida ko'rib chiqamiz:

void **gluSphere** (GLUquadricObj **qobj*, GLdouble *radius*, GLint *slices*, GLint *stacks*)

Bu funksiya koordinata boshi sfera markazida va radiusi *radius* bo'lgan sfera yaratadi. Bunda sferaning *z* o'qi atrofidagi kesimlari soni *slices* parametrini beradi, *z* o'qi bo'ylab esa *stacks* parametrini beradi.

void **gluCylinder** (GLUquadricObj **qobj*,
GLdouble *baseRadius*,
GLdouble *topRadius*,
GLdouble *height*, GLint *slices*,
GLint *stacks*)

Berilgan funksiya *z* o'qiga parallel o'q bo'ylab, orqa asos radiusi *baseRadius* bo'lgan va *z*=0 tekislikda joylashgan, oldingi asosi radiusi *topRadius* va *z=height* tekislikda joylashgan asossiz silindr (ya'ni xalqa) tuzadi. Agar radiuslardan biri nolga teng bo'lsa konus hosil bo'ladi, *slices* va *stacks* parametrlari analogik ma'noga ega bo'lib, ular haqida yuqorida buyrug'larni ko'rdik.

void **gluDisk** (GLUquadricObj **qobj*,
GLdouble *innerRadius*,
GLdouble *outerRadius*, GLint *slices*,
GLint *loops*)

Funksiya koordinata boshi markazi bo'lib, radiusi *outerRadius* yassi disk (ya'ni doira) quradi. Agar bunda *innerRadius* qiymati noldan farqli bo'lsa, shuningdek, disk markazida radiusi *innerRadius* teshik joylashgan bo'ladi. *Slices* parametri *z* o'qi atrofidagi desk kesimlar sonini, *loops* parametri esa – *ox* va *z* o'qiga perpendikulyar konsentrik xalqalar soni.

void **gluPartialDisk** (GLUquadricObj **qobj*,
GLdouble *innerRadius*,
GLdouble *outerRadius*, GLint *slices*,
GLint *loops*, GLdouble *startAngle*,
GLdouble *sweepAngle*);

Bu buyruqning oldingilardan farqi shundaki, u soat strelkasiga qarama-qarshi *y* o'qi musbat yo'nalishidan hisoblanganda va *startAngle* va *sweepAngle* parametrlar berilganda boshlang'ich va oxirgi burchaklaridan doira sektorini hosil qiladi. Burchaklar graduslarda beriladi.

GLUT kutubxonasidan primitivlar tuzishni olib boradigan buyruqlar, OpenGL va GLU standart primitivlari orqali amalga oshiriladi. Kerakli primitivlarni hosil qilish uchun mos keluvchi buyruqlarni chaqirish kerak bo'ladi.

void **glutSolidSphere** (GLdouble *radius*, GLint *slices*, GLint *stacks*)

void **glutWireSphere** (GLdouble *radius*, GLint *slices*, GLint *stacks*)

glutSolidSphere() buyrug'i sfera yasash uchun, **glutWireSphere()** – buyrug'i esa *radius* radiusli sfera karkasini yasash uchun kerak bo'ladi. Qolgan parametrlar oldingi buyruqlar kabi bo'ladi.

void **glutSolidCube** (GLdouble *size*)

void **glutWireCube** (GLdouble *size*)

buyruqlari qirra uzunligi *size* va koordinata boshida markaz bilan kub yoki kubning karkasini chizadi.

void **glutSolidCone** (GLdouble *base*, GLdouble *height*, GLint *slices*, GLint *stacks*)

void **glutWireCone** (GLdouble base, GLdouble height, GLint *slices*, GLint *stacks*)

Bu buyruq orqali *z* o'qi bo'ylab joylashgan balandligi *height*, asos radiusi *base* bo'lgan konus yoki uning karkas chiziladi. Asoslari z=0 tekislikda joylashgan bo'ladi.

void **glutSolidTorus** (GLdouble *innerRadius*, GLdouble *outerRadius*, GLint *nsides*, GLint *rings*)

void **glutWireTorus** (GLdouble *innerRadius*, GLdouble *outerRadius*, GLint *nsides*, GLint *rings*)

Bu buyruqlar z=0 tekislikda tor yoki uning karkasini quradi. Ichki va tashqi radiuslar *innerRadius*, *outerRadius* parametrlari bilan beriladi. *nsides* parametri torning ortogonal kesimini tashkil qiluvchi tomonlari soni, *rings esa* – tor radial kesimlari sonini amalga oshiradi.

void **glutSolidTetrahedron** (void)

void **glutWireTetrahedron** (void)

Bu buyruqlar radiusi 1 ga teng bo'lgan sferaga tashqi chizilgan tetraedr (muntazam uchburchakli) yoki uning karkasini hosil qiladi.

void **glutSolidOctahedron** (void)

void **glutWireOctahedron** (void)

Bu buyruqlar radiusi 1 ga teng bo'lgan sferaga tashqi chizilgan oktaedr yoki uning karkasini hosil qiladi.

void **glutSolidDodecahedron** (void)

void **glutWireDodecahedron** (void)

Bu buyruqlar sferaga tashqi chizilgan ikosaedr yoki uning karkasini hosil qiladi.

void **glutSolidIcosahedron** (void)

void **glutWireIcosahedron** (void)

Bu buyruqlar radiusi 1 ga teng bo'lgan sferaga tashqi chizilgan ikosaedr yoki uning karkasini hosil qiladi.

Quyidagi keltirilgan primitivlarni to'g'ri tashkil qilish uchun ko'rinmas chiziq va sirtlarni olib tashlash kerak bo'ladi, buning uchun **glEnable**(GL_DEPTH_TEST) buyrug'iga mos keluvchi rejimni chaqirish kerak.

ILOVA C. OPENGL ILOVALARINI SOZLASH

Borland C++ 5.02 muhitida ilova yaratish.

Dastlab BorlandC\Include\Gl, BorlandC\Lib, Windows\System kataloglarida glut.h, glut32.lib, glut32.dll fayllarning mavjudligini ta'minlash zarur. Shuningdek, ushbu kataloglarda gl.h, glu.h, openg l32.lib, glu32.lib, opengl32.dll, glu32.dll, fayllarning mavjudligini tekshirish kerak. Ular, odatda BorlandC++ va Windows tarkibiga kiradi. Shu bilan birgalikda Microsoftning versiyalaridagi opengl32.lib, glu32.lib, glut32.lib fayllar Borland C++ muhiti uchun to'g'ri kelmaydi va faqat mos tushadigan versiyalardan foydalanish kerak. Bunday versiyalar yaratish BorlandC\Bin katalogida 'implib' standart dasturini ishlatish kerak. Buning uchun *.lib faylini mos keluvchi *.dll faylida yaratish quyidagi tartibda bajarilishi kerak:

implib BorlandC\Lib\filename.lib filename.dll

Yana ta'kidlab o'tish kerakki Borland noma'lum sabablarga ko'ra BorlandC++5.02 tarkibiga kiruvchi GLAUX kutubxonasiga kiruvchi ilova kompilyasiyasi glaux.lib faylini ishlata olmaydi, shuning uchun bu kutubxonadan voz kechishga to'g'ri keladi. Ilova yaratish uchun quyidagi amallarni bajarish kerak:

- Loyiha yaratish: buning uchun *Project->New Project* tanlanadi va *Target Expert* oynasidagi maydonni quyidagi tartibda to'ldirish kerak: *Platform* maydonida Win32 tanlab, *Taget Model* maydonida *Sonsole* ni tanlab, *Advanced* ni bosib va '*.rc ' va ' *.def ' tanlash punklarini bekor qilish kerak.
- Loyihaga OpenGL kutubxonalarini qo'shish. Buning uchun loyiha oynasida ijro qilinayotgan (*.exe) faylni tanlab va sichqonchaning o'ng tugmasini bosib kontekst menyudan *Add node* punktini tanlash kerak. Keyin bu opengl32.lib, glu32.lib, glut32.lib fayllarni joylashish holatini aniqlash kerak.
- Kompilyatsiya uchun *Project->Build All* ni tanlash, bajarish uchun – *Debug->Run* ni tanlash kerak bo'ladi.

MS Visual C++ 6.0 muhitida ilova yaratish.

Ishni boshlashdan oldin MSVC\Include\Gl, MSVC\Lib, Windows\System kataloglariga glut.h, glut32.lib, glut32.dll fayllarni mos ravishda ko'chirib olish kerak. Shuningdek, bu kataloglarda Visual C++ va Windows tarkibiga kiradigan gl.h, glu.h,

opengl32.lib, glu32.lib, opengl32.dll, glu32.dll fayllar borligini tekshirish kerak. GLAUX kutubxonasida joylashgan buyruqlardan foydalanish uchun sanab o'tilgan fayllar ro'yxatiga glaux.h, glaux.lib fayllarni ham qo'shish kerak.

Ilova yaratish uchun quyidagi amallarni bajarish kerak:

- Loyiha yaratish: buning uchun *File->New->Projects->Win32 Console Application* ni tanlab, loyihaga nom berib OK ni bosish kerak.
- Paydo bo'lgan oynadan '*An empty project*' ni tanlab, Finish, OK ni bosish kerak.
- Dastur matni hosil qilingan tekstli faylga (*File->New->Files->Text File* ni tanlash mumkin), kengaytmasi *.c yoki *.cpp bo'lgan faylni loyihaga qo'shish kerak (*Project->Add To Project->Files* ni tanlash orqali).
- Loyihaga OpenGL kutubxonalarini qo'shish. Buning uchun *Project->Settings->Link* ni tanlab va *Object/library modules* maydonida kerakli kutubxona nomini terish kerak: opengl32.lib, glu32.lib, glut32.lib va zarur hollarda glaux.lib.
- kompilyatsiya uchun *Build->Build program.exe* ni, ishga tushirish uchun – *Build->Execute program.exe* ni tanlash zarur.
- Ishga tushgan paytda matnli maydon hosil bo'lmasligi uchun *Project->Settings->Link* buyrug'ini tanlab *Project Options* maydonida 'subsystem:console' o'rniga 'subsystem: windows' ni terish kerak va shu qatorning o'zida '/entry:mainCRTStartup' ni terish kerak.
- Dastur tayyor bo'lgandan keyin 'Release' rejimini yana bir bor kompilyasiyadan o'tkazish maqsadga muvofiqdir, bu dasturni tez ishlashi va optimallashtirish uchun foydali hisoblanadi. Buning uchun avval *Build->Set Active Configuration...* ni tanlab '...-Win32 Release'ni belgilab keyin yana bir bor kerakli kutubxonani qo'shish kerak.

Borland C++ Builder 6. muhitida ilova yaratish.

Ishni boshlashdan oldin CBuilder6Mnclude\Gl, CBuilder6\Lib, Windows\System kataloglariga glut.h, glut32.lib glut32.dll fayllarni mos ravishda ko'chirib olish kerak. Shuningdek, bu kataloglarda Borland C++ va Windows tarkibiga kiradigan gl.h, glu.h,

opengl32.lib, glu32.lib, opengl32.dll, glu32.dll fayllar borligini tekshirish kerak.

Shu bilan birgalikda Microsoftning versiyalaridagi opengl32.lib, glu32.lib, glut32.lib fayllar Borland C++ Builder 6 muhiti uchun to'g'ri kelmaydi va faqat mos tushadigan versiyalardan foydalanish kerak. Bunday versiyalarni yaratish uchun SVuilder6\Bin katalogida 'implib' standart dasturini ishlatish kerak. Buning uchun *.lib faylini mos keluvchi *.dll faylida yaratish quyidagi tartibda bajarilishi kerak:

implib glut32.lib glut32.dll

Ilova yaratish uchun quyidagi amallarni bajarish kerak:

- Loyiha yaratish: buning uchun *File->New->Other->Console Wizard* ni tanlab OK ni bosish kerak.
- Paydo bo'lgan oynadan Source Type - S++, Console Application ni tanlab, 'Use VCL', 'Use CLX', 'Multi Threaded' opsiyalarini olib tashlab OK ni bosish kerak.
- Dastur matnini yaratilgan matnli faylga joylashtirish mumkin, yoki uni loyihadan o'chirib tashlash mumkin (*Project->Remove From Project*) va loyihaga *.c yoki *.cpp kengaytmali fayllarni qo'shish mumkin (*Project ->Add To Project* ni tanlash orqali).
- Yaratilgan loyihani kerakli katalogda saqlang (*File ->Save All* ni tanlab).
- Loyihaga GLUT kutubxonasini qo'shish. Buning uchun *Project->Add To Project* ni tanlab glut32.lib faylini qo'shish kerak.
- Kompilyatsiya uchun *Project->Build* ni, ishga tushirish uchun – *Run->Run* ni tanlash zarur.
- Dastur tayyor bo'lganda, tezkorligi va xajmi bo'yicha optimallashtirish uchun uni 'Release' rejimida qaytadan kompilyatsiyaga berish tavsiya etiladi. Buning uchun avvalo *Project->Options ->Compiler* ni tanlab 'Release' tugmasini bosish kifoya.

ILOVA D. AMALIY TOPSHIRIQLARGA MISOLLAR

Masala 1: Sodda GLUT-ilovasi

Ushbu oddiy masala GLUT ilovasi strukturasini va OpenGL ning oddiy asoslarini ko'rsatadi. Dastur natijasi uchburchaklar rangini sichqonchaning chap tugmasini bosganda o'zgaradigan tasodifiy to'plamidir.

Sichqonchaning o'ng tugmasi bilan uchburchaklar rangini o'zgartirish tartibiga yordam beradi.

```
#include <stdlib.h>
#include <gl/glut.h>

#ifdef random
#undef random
#endif

#define random(m) (float)rand()*m/RAND_MAX

/* oynaning kengligi va balandligi*/
GLint Width = 512, Height = 512;
/* oynadagi to'g'ri to'rtburchaklar soni*/
int  Times = 100;
/* to'ldiruvchilar bilan */
int  FillFlag = 1;
long  Seed = 0;
/* uchburchaklarni tasvirlash  funksiyasi*/
void
DrawTriang( float x1, float y1, float x2, float y2, float x3, float y3,
int FillFlag )
{
glBegin(FillFlag ? GL_TRIANGLES : GL_LINE_LOOP);
glVertex2f(x1, y1);
glVertex2f(x2, y2);
glVertex2f(x3, y3);
glEnd();
}
```

```
/* ekranga chiquvchi barcha axborotlarni boshqarish */
void
Display(void)
{
int i;
float x1, y1, x2, y2, x3, y3;
float r, g, b;
srand(Seed);
glClearColor(0, 0, 0, 1);
glClear(GL_COLOR_BUFFER_BIT);

for( i = 0; i < Times; i++ ) {
r = random(1);
g = random(1);
b = random(1);
glColor3f( r, g, b );

x1 = random(1) * Width;
y1 = random(1) * Height;
x2 = random(1) * Width;
y2 = random(1) * Height;
x3 = random(1) * Width;
y3 = random(1) * Height;

DrawRect(x1, y1, x2, y2, x3, y3, FillFlag);
}
glFinish();
}
/* oyna o'lchamlari o'zgaradigan funksiya */
void
Reshape(GLint w, GLint h)
{
Width = w;
Height = h;
glViewport(0, 0, w, h);

glMatrixMode(GL_PROJECTION);
```

```
glLoadIdentity();
glOrtho(0, w, 0, h, -1.0, 1.0);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
}
/*sichqonchadan keladigan ma'lumotni qayta ishlash */
void
Mouse(int button, int state, int x, int y)
{
if( state == GLUT_DOWN ) {
switch( button ) {
case GLUT_LEFT_BUTTON:
Seed = random(RAND_MAX);
break;
case GLUT_RIGHT_BUTTON:
FillFlag = !FillFlag;
break;
}
glutPostRedisplay();
}
}

/* klaviaturadan keladigan ma'lumotni qayta ishlash */
void
Keyboard ( unsigned char key, int x, int y )
{
#define ESCAPE '\033'

if( key == ESCAPE )
exit(0);
}

main(int argc, char *argv[])
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_RGB);
```

```
glutInitWindowSize(Width, Height);
glutCreateWindow("Rect draw example (RGB)");

glutDisplayFunc(Display);
glutReshapeFunc(Reshape);
glutKeyboardFunc(Keyboard);
glutMouseFunc(Mouse);
glutMainLoop();
}
```

Masala 2: OpenGL yoritish modeli

1. Ushbu dastur OpenGL modelini ishlatishda oddiy sahna orqali tor, kub, konus va shardan tuzilgan bo'ladi. Obyekt qilib har xil materiallarni olamiz. Albatta, sahnada yorug'lik manbai bo'ladi.

```
#include <GL/glut.h>
#include <stdlib.h>

/*tor materialining parametrlari*/
float mat1_dif[]={0.8f,0.8f,0.0f};
float mat1_amb[]= {0.2f,0.2f,0.2f};
float mat1_spec[]={0.6f,0.6f,0.6f};
float mat1_shininess=0.5f*128;

/* kub materialining parametrlari */
float mat2_dif[]={0.2f,0.6f,0.0f,0.0f};
float mat2_amb[]= {0.4f,0.4f,0.2f,0.0f};
float mat2_spec[]={0.4f,0.6f,0.4f,0.0f};
float mat2_shininess=0.3f*128;

/* shar materialining parametrlari */
float mat3_dif[]={0.9f,0.2f,0.0f};
float mat3_amb[]= {0.2f,0.2f,0.2f};
float mat3_spec[]={0.6f,0.6f,0.6f};
float mat3_shininess=0.1f*128;

/* konus materialining parametrlari */
float mat4_dif[]={0.0f,0.0f,0.8f};
float mat4_amb[]= {0.2f,0.2f,0.2f};
```

```
float mat4_spec[]={0.6f,0.6f,0.6f};
float mat4_shininess=0.7f*128;

/* yorug'lik manbai va materiallar parametrlarini
initsiallashtirish */
void init (void)
{
GLfloat light_ambient[] = { 0.0, 0.0, 0.0, 1.0 };
GLfloat light_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat light_specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };

/* yorug'lik manbaining parametrlarini o'rnatish */
glLightfv (GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv (GL_LIGHT0, GL_DIFFUSE, light_diffuse);
glLightfv (GL_LIGHT0, GL_SPECULAR, light_specular);
glLightfv (GL_LIGHT0, GL_POSITION, light_position);

/* yorug'lik manbai va yoritgichni qo'shish hamda yoqish */
glEnable (GL_LIGHTING);
glEnable (GL_LIGHT0);

/*  z-buferni yoqamiz*/
glEnable(GL_DEPTH_TEST);
}

/* Tasvirlarni chizish uchun kerak bo'ladigan funksiyani
chaqirish. Unda barcha geometrik shakllarni chiqarish amalga
oshiriladi. */
void display (void)
{
/* kadr buferi va chuqurlik buferini tozalash*/
glClear (GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT);

glPushMatrix ();
glRotatef (45.0, 1.0, 0.0, 0.0);
```

```
/* tor tasviri*/
glMaterialfv (GL_FRONT,GL_AMBIENT,mat1_amb);
glMaterialfv (GL_FRONT,GL_DIFFUSE,mat1_dif);
glMaterialfv (GL_FRONT,GL_SPECULAR,mat1_spec);
glMaterialf  (GL_FRONT,GL_ShININESS,mat1_shininess);

glPushMatrix ();
glTranslatef (-0.75, 0.5, 0.0);
glRotatef (90.0, 1.0, 0.0, 0.0);
glutSolidTorus (0.275, 0.85, 15, 15);
glPopMatrix ();

/*kub tasviri */
glMaterialfv (GL_FRONT,GL_AMBIENT,mat2_amb);
glMaterialfv (GL_FRONT,GL_DIFFUSE,mat2_dif);
glMaterialfv (GL_FRONT,GL_SPECULAR,mat2_spec);
glMaterialf  (GL_FRONT,GL_ShININESS,mat2_shininess);

glPushMatrix();
glTranslatef(1,-0.5,1);
glRotatef(901, 1.0, 0.0, 0.0);
glutSolidCube(1);
glPopMatrix();

/* shar tasviri */
glMaterialfv (GL_FRONT,GL_AMBIENT,mat3_amb);
glMaterialfv (GL_FRONT,GL_DIFFUSE,mat3_dif);
glMaterialfv (GL_FRONT,GL_SPECULAR,mat3_spec);
glMaterialf  (GL_FRONT,GL_ShININESS,mat3_shininess);

glPushMatrix ();
glTranslatef (0.75, 0.0, -1.0);
glutSolidSphere (1.0, 25, 25);
glPopMatrix ();

/*konus tasviri */
glMaterialfv (GL_FRONT,GL_AMBIENT,mat4_amb);
```

```
    glMaterialfv (GL_FRONT,GL_DIFFUSE,mat4_dif);
    glMaterialfv (GL_FRONT,GL_SPECULAR,mat4_spec);
    glMaterialf  (GL_FRONT,GL_ShININESS,mat4_shininess);

    glPushMatrix ();
    glTranslatef (-0.75, -0.5, 0.0);
    glRotatef (270.0, 1.0, 0.0, 0.0);
    glutSolidCone (1.0, 2.0, 15, 15);
    glPopMatrix ();

    glPopMatrix ();
    /*ekranga sahnani chiqarish */
    glFlush ();
    }
    /* foydalanuvchi tomonidan chaqiriladigan oyna o'lchamlarini
o'zgartiruvchi funksiya */
    void reshape(int w, int h)
    {
    /* oyna o'lchamiga teng bo'lgan chiqarish sohasi o'lchamini
o'rnatish */
    glViewport (0, 0, (GLsizei) w, (GLsizei) h);

    /* oyna o'lchamini hisobga olib proeksiyalar matritsasini
beramiz */
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();

    gluPerspective(
    40.0, /* gradusda berilgan ko'rish burchagi */
     (GLfloat)w/h, /*oynaning siqilish koeffitsiyenti */
    1,100.0);  /*tekislik kesimigacha masofa*/
    glMatrixMode (GL_MODELVIEW);

    glLoadIdentity ();
    gluLookAt(
    0.0f,0.0f,8.0f, /* kamera holati */
    0.0f,0.0f,0.0f, /* sahnalar markazi */
```

```
    0.0f,1.0f,0.0f); /* y o'qining musbat yo'nalishi*/
    }

    /* klaviaturadagi klavishlarni bosganda chaqiriladigan funksiya
*/
    void keyboard(unsigned char key, int x, int y)
    {
    switch (key) {
    case 27: /* escape */
    exit(0);
    break;
    }
    }

    /* Ilovaning asosiy sikli.
    * Chuqurlik buferi bilan ekran rejimi o'rnatiladi, oyna
yaratiladi. */
    int main(int argc, char** argv)
    {
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB
                    | GLUT_DEPTH);
    glutInitWindowSize (500, 500);
    glutCreateWindow (argv[0]);
    init ();
    glutReshapeFunc (reshape);
    glutDisplayFunc (display);
    glutKeyboardFunc (keyboard);
    glutMainLoop();
    return 0;
    }
```

2. Ushbu dastur OpenGL muhitida beshta kubdan tashkil topgan va markaziy kubdan bir xil masofada yotuvchi kublardan tuzilgan bo'ladi. Albatta sahnada yorug'lik manbai bo'ladi.

```
#include <GL/glut.h>
```

```
void init()
{
/*  z-buferni yoqamiz*/
glEnable(GL_DEPTH_TEST);
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glEnable(GL_COLOR_MATERIAL);
}

/* Tasvirlarni chizish uchun kerak bo'ladigan funksiyani
chaqirish. Unda barcha kublarni chiqarish amalga oshiriladi.  */
void display()
{
/* kadr buferi va chuqurlik buferini tozalash*/
glClear(GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT);

/* oyna o'lchamini hisobga olib proeksiyalar matritsasini
beramiz */
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
GLint viewport[4];
glGetIntegerv(GL_VIEWPORT, viewport);
double aspect = (double)viewport[2] / (double)viewport[3];
gluPerspective(60, aspect, 1, 100);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

// move back a bit
glTranslatef( 0, 0, -35 );

static float angle = 0;
angle += 1.0f;

/* markaziy kub parametrlarini o'rnatish */
glPushMatrix();
```

```
glTranslatef(0,0,0);
glRotatef(angle, 0.1, 0.2, 0.5);
glColor3ub(255,0,255);
glutSolidCube(5);
glPopMatrix();

/* quyidan ikkinchi kub parametrlarini o'rnatish */
glPushMatrix();
glTranslatef(10,-10,0);
glRotatef(angle, 0.1, 0.2, 0.5);
glColor3ub(255,0,0);
glutSolidCube(5);
glPopMatrix();

/* yuqoridan ikkinchi kub parametrlarini o'rnatish */
glPushMatrix();
glTranslatef(10,10,0);
glRotatef(angle, 0.1, 0.2, 0.5);
glColor3ub(0,255,0);
glutSolidCube(5);
glPopMatrix();

/* yuqoridan birinchi kub parametrlarini o'rnatish */
glPushMatrix();
glTranslatef(-10,10,0);
glRotatef(angle, 0.1, 0.2, 0.5);
glColor3ub(0,0,255);
glutSolidCube(5);
glPopMatrix();

/* quyidan birinchi kub parametrlarini o'rnatish */
glPushMatrix();
glTranslatef(-10,-10,0);
glRotatef(angle, 0.1, 0.2, 0.5);
glColor3ub(255,255,0);
glutSolidCube(5);
glPopMatrix();
```

```
    glutSwapBuffers();
    }
    /* foydalanuvchi tomonidan chaqiriladigan oyna o'lchamlarini
o'zgartiruvchi funksiya */
    void reshape(int w, int h)
    {
    glViewport(0, 0, w, h);
    }

    void timer(int extra)
    {
    glutPostRedisplay();
    glutTimerFunc(16, timer, 0);
    }

    int main(int argc, char **argv)
    {
    glutInit(&argc, argv);
    glutInitWindowSize(640,480);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH |
GLUT_DOUBLE);
    glutCreateWindow("CUBES");

    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutTimerFunc(0, timer, 0);

    init();
    glutMainLoop();
    return 0;
    }
```

3. Ushbu dastur OpenGL muhitida oddiy sahna orqali sfera, konus va tordan tuzilgan materiallarni ifodalaydi. Ushbu obyektlarning to'r shaklida berilgan tasvirini va uning ranglar bilan to'ldirilgan variantlari turli burchak ostidagi harakati keltirilgan va sahnada yorug'lik manbai tushurilgan.

```
#include <windows.h>
#include <gl/glut.h>
#include <stdlib.h>

static int slices = 26;
static int stacks = 16;
/* GLUT qayta aloqalarga ishlov berish */
static void
resize(int width, int height)
{
/* oyna o'lchamini hisobga olib proeksiyalar matritsasini
beramiz */
const float ar = (float) width / (float) height;
glViewport(0, 0, width, height);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(-ar, ar, -1.0, 1.0, 2.0, 180.0);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity() ;
}
static void
display(void)
{
const double t = glutGet(GLUT_ELAPSED_TIME) / 1000.0;
const double a = t*90.0;
/* kadr buferi va chuqurlik buferini tozalash*/
glClear(GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT);
glColor3d(0,1,1);

/* sfera materiali tasviri va uning parametrlari */
glPushMatrix();
glTranslated(-2.4,1.2,-6);
glRotated(60,1,0,0);
glRotated(a,0,0,1);
glutSolidSphere(1,slices,stacks);
```

```
    glPopMatrix();

    /* konus materiali tasviri va uning parametrlari */
    glPushMatrix();
    glTranslated(0,1.2,-6);
    glRotated(45,1,0,0);
    glRotated(a,0,0,1);
    glutSolidCone(1,1,slices,stacks);
    glPopMatrix();

    /* tor materiali tasviri va uning parametrlari */
    glPushMatrix();
    glTranslated(2.4,1.2,-6);
    glRotated(60,1,0,0);
    glRotated(a,0,0,1);
    glutSolidTorus(0.2,0.8,slices,stacks);
    glPopMatrix();

    /* to'r shaklidagi sfera materiali tasviri va uning parametrlari */
    glPushMatrix();
    glTranslated(-2.4,-1.2,-6);
    glRotated(60,1,0,0);
    glRotated(a,0,0,1);
    glutWireSphere(1,slices,stacks);
    glPopMatrix();

    /* to'r shaklidagi konus materiali tasviri va uning parametrlari
*/
    glPushMatrix();
    glTranslated(0,-1.2,-6);
    glRotated(60,1,0,0.8);
    glRotated(a,0,0,1);
    glutWireCone(1,1,slices,stacks);
    glPopMatrix();

    /* to'r shaklidagi tor materiali tasviri va uning parametrlari */
    glPushMatrix();
```

```
glTranslated(2.4,-1.2,-6);
glRotated(120,1,0,0);
glRotated(a,0,0,1);
glutWireTorus(0.2,0.8,slices,stacks);
glPopMatrix();

glutSwapBuffers();
}

static void
key(unsigned char key, int x, int y)
{
switch (key)
{
case 27 :
case 'q':
exit(0);
break;

case '+':
slices++;
stacks++;
break;

case '-':
if (slices>3 && stacks>3)
{
slices--;
stacks--;
}
break;
}

glutPostRedisplay();
}

static void
```

```
idle(void)
{
glutPostRedisplay();
}

/* yorug'lik manbai va materiallar parametrlarini
initsiallashtirish */
const GLfloat light_ambient[]  = { 0.0f, 0.0f, 0.0f, 1.0f };
const GLfloat light_diffuse[]  = { 1.0f, 1.0f, 1.0f, 1.0f };
const GLfloat light_specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };
const GLfloat light_position[] = { 2.0f, 5.0f, 5.0f, 0.0f };

const GLfloat mat_ambient[]    = { 0.7f, 0.7f, 0.7f, 1.0f };
const GLfloat mat_diffuse[]    = { 0.8f, 0.8f, 0.8f, 1.0f };
const GLfloat mat_specular[]   = { 1.0f, 1.0f, 1.0f, 1.0f };
const GLfloat high_shininess[] = { 100.0f };

/* Program entry point */

int
main(int argc, char *argv[])
{
glutInit(&argc, argv);
glutInitWindowSize(640,480);
glutInitWindowPosition(20,20);
glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE |
GLUT_DEPTH);

glutCreateWindow("GLUT ilovasi shakllari");

glutReshapeFunc(resize);
glutDisplayFunc(display);
glutKeyboardFunc(key);
glutIdleFunc(idle);

glClearColor(5,0.9,0.9,0);
glEnable(GL_CULL_FACE);
```

```
glCullFace(GL_BACK);
glEnable(GL_DEPTH_TEST);
glDepthFunc(GL_LESS);

/* yorug'lik manbai va yoritgichni qo'shish hamda yoqish */
glEnable(GL_LIGHT0);
glEnable(GL_NORMALIZE);
glEnable(GL_COLOR_MATERIAL);
glEnable(GL_LIGHTING);

/* yorug'lik manbai va materialning parametrlarini o'rnatish */
glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);

glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT, GL_SHININESS, high_shininess);

glutMainLoop();

return 0;
}
```

Masala 3: Teksturalash

Ushbu dasturni bajarishdan olingan natija halqalar falagi atrofida aylanib turuvchi teksturalar kiritilgan tetraedrni qurish hisoblanadi. MS Visual C++ muhitida dastur nomsiz ham kompilyatsiyalanishi mumkin, Borland C++ da kompilyatsiyalashda esa TextureInit() funksiyasini tasvirlash va chaqirishni tavsiflashga to'g'ri keladi, shundagina teksturalarni kiritish amalga oshirilmaydi. Yuqorida aytilganidek, GLAUX kutubxonasidagi funksiyalardan foydalanishga urinib ko'rish dasturni kompilyatsiyalashda hatoliklar haqidagi ma'lumotlarga olib keladi.

MS Visual C++ da dasturni kompilyatsiyalashda 'texture.bmp' faylini loyiha katalogiga ko'chirish yoki unga to'liq

yo'lni '/' belgisidan foydalanib ko'rsatish kerak. Agar yo'l ko'rsatilmagan bo'lsa, u holda operatsion tizimdan bajariluvchi faylni yuklashda teksturali fayl aynan bajarilayotgan fayl joylashgan katalogda bo'lishi kerak.

```
#include <GL\glut.h>
#include <gl\glaux.h>
#include <math.h>
#define TETR_LIST 1

GLfloat light_col[] = {1,1,1};
float mat_diff1[]={0.8,0.8,0.8};
float mat_diff2[]={0.0,0.0,0.9};
float mat_amb[]= {0.2,0.2,0.2};
float mat_spec[]={0.6,0.6,0.6};
float shininess=0.7*128,
float CurAng=0, RingRad=1, RingHeight=0.1;
GLUquadricObj* QuadrObj;
GLuint TexId;
GLfloat TetrVertex[4][3], TetrNormal[4][3];

/* berilgan a,b,c nuqtaning tekislikdagi normalini hisoblash */
void getnorm (float a[3],float b[3],float c[3],float *n)
{
 float mult=0,sqr;
 int i,j;
 n[0]=(b[1]-a[1])*(c[2]-a[2])-(b[2]-a[2])*(c[1]-a[1]);
 n[1]=(c[0]-a[0])*(b[2]-a[2])-(b[0]-a[0])*(c[2]-a[2]);
 n[2]=(b[0]-a[0])*(c[1]-a[1])-(c[0]-a[0])*(b[1]-a[1]);
 /* normalning kerakli yo'nalishini belgilash: (0,0,0) nuqtadan*/
 for (i=0;i<3;i++)  mult+=a[i]*n[i];
 if (mult<0) for (j=0;j<3;j++) n[j]=-n[j];
}

/* tetraedr uchlari koordinatalarini hisoblash */
void InitVertexTetr()
{
```

```
float alpha=0;
int i;
TetrVertex[0][0]=0;
TetrVertex[0][1]=1.3;
TetrVertex[0][2]=0;
 /* tetraedr asosi koordinatalarini hisoblash */
 for (i=1;i<4;i++)
 {
TetrVertex[i][0]=0.94*cos(alpha);
TetrVertex[i][1]=0;
TetrVertex[i][2]=0.94*sin(alpha);
alpha+=120.0*3.14/180.0;
}
}

/* tetraedr tomonlari normalini hisoblash*/
void InitNormsTetr()
{
 getnorm(TetrVertex[0], TetrVertex[1],
       TetrVertex[2], TetrNormal[0]);
 getnorm(TetrVertex[0], TetrVertex[2],
       TetrVertex[3], TetrNormal[1]);
 getnorm(TetrVertex[0], TetrVertex[3],
       TetrVertex[1], TetrNormal[2]);
 getnorm(TetrVertex[1], TetrVertex[2],
       TetrVertex[3], TetrNormal[3]);
}

/* tetraedr qurish ro'yxatini yaratish */
void MakeTetrList()
{
 int i;
glNewList(TETR_LIST,GL_COMPILE);
/*tetraedr tamonlarini berish */
 glBegin(GL_TRIANGLES);
for (i=1;i<4;i++)
{
```

```
glNormal3fv(TetrNormal[i-1]);
glVertex3fv(TetrVertex[0]);
glVertex3fv(TetrVertex[i]);
if (i!=3) glVertex3fv(TetrVertex[i+1]);
else glVertex3fv(TetrVertex[1]);
}
glNormal3fv(TetrNormal[3]);
glVertex3fv(TetrVertex[1]);
glVertex3fv(TetrVertex[2]);
glVertex3fv(TetrVertex[3]);

glEnd();
glEndList();
}

void DrawRing()
{
 /* z o'qiga parallel joylashgan silindr (halqa) qurish.
* Ikkinchi va uchinchi parametrlar asos radiusini belgilaydi,
* to'rtinchisi balandligini, oxirgi ikkita son
* z o'qi bo'ylab va atrofni bo'lishni belgilaydi.
* Shunday ekan, keyinchalik silindrning asosi
* z=0 tekislikda joylashadi   */
  gluCylinder (QuadrObj,RingRad,RingRad,RingHeight,30,2);
}
void TextureInit()
{
 char strFile[]="texture.bmp";
 AUX_RGBImageRec *pImage;

 /* bayt bo'yicha *.bmp ga to'g'rilash*/
 glPixelStorei(GL_UNPACK_ALIGNMENT,1);
 /* tekstura uchun identifikator yaratish*/
 glGenTextures(1,&TexId);
 /* xotiraga tasvirni yuklash*/
 pImage = auxDIBImageLoad(strFile);
```

```
    /* tekstura xossalarini tavsiflashni boshlash */
    glBindTexture (GL_TEXTURE_2D,TexId);
    /* teksturalarni detallashtirish va initsiallash darajalarini
yaratish */
    gluBuild2DMipmaps(GL_TEXTURE_2D,3,pImage->sizeX,
                pImage->sizeY,GL_RGB,GL_UNSIGNED_BYTE,
                pImage->data);
    /* quadric-obyektida bu teksturani sozlashga ruxsat etish */
    gluQuadricTexture (QuadrObj, GL_TRUE);
    /* tekstura parametrlarini berish*/
    /* s va t parametrik o'qlar bo'yicha tasvirni takrorlash */
    glTexParameteri (GL_TEXTURE_2D,
GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri (GL_TEXTURE_2D,
GL_TEXTURE_WRAP_T, GL_REPEAT);
    /* teksturada nuqta tanlashda qo'shimcha kiritishlardan
foydalanilmaydi */
    glTexParameteri (GL_TEXTURE_2D,
GL_TEXTURE_MIN_FILTER, GL_NEAREST);
    glTexParameteri (GL_TEXTURE_2D.
GL_TEXTURE_MAG_FILTER, GL_NEAREST);
    /* tekstura va material obyektlarini birga qo'shish*/
    glTexEnvi (GL_TEXTURE_ENV,
GL_TEXTURE_ENV_MODE, GL_MODULATE);
    }
    void Init(void)
    {
    InitVertexTetr();
    InitNormsTetr();
    MakeTetrList();
    /* material xossalarini aniqlash */
    glMaterialfv (GL_FRONT_AND_BACK, GL_AMBIENT,
mat_amb);
    glMaterialfv (GL_FRONT_AND_BACK, GL_SPECULAR,
mat_spec);
    glMaterialf (GL_FRONT, GL_ShININESS, shininess);
    /* yoritgich xossalarini aniqlash */
```

```
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_col);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    /* qo'rinmas chiziqlar va sirtlarni olib tashlash */
    glEnable(GL_DEPTH_TEST);
    /* normallarni normalashtirishni olib borish*/
    glEnable(GL_NORMALIZE);
    /* obyektlar materiallari ranglarning diffuzion aks etishi bilan
farqlanadi */
    glEnable(GL_COLOR_MATERIAL);
    glColorMaterial(GL_FRONT_AND_BACK,GL_DIFFUSE);
   /* halqa qurish uchun quadric-obyektiga ko'rsatgich yaratish */
    QuadrObj=gluNewQuadric();
    /* tekstura xossalarini aniqlash */
    TextureInit();
    /* markaziy proeksiyani tayinlash */
    glMatrixMode(GL_PROJECTION);
    gluPerspective(89.0,1.0,0.5,100.0);
   /* keyinchalik faqatgina sahnadagi obyektlarni almashtirish olib
boriladi */
    glMatrixMode(GL_MODELVIEW);
   }
   void DrawFigures(void)
   {
    /* teksturalar kiritish rejimini qo'shish (yoki yoqish) */
    glEnable(GL_TEXTURE_2D);
    /* halqa uchun rangning diffuzion tasvirini tayinlash */
    glColor3fv(mat_diff1);
    /*birlik matritsani yuklashda oldingi matritsa bilan birgalikda
ko'rsatmaslik */
   glLoadIdentity();
   /* tekshirish nuqtalarini aniqlash */
   gluLookAt(0.0, 0.0, 2.5,0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
   /* matritsa ko'rinishini saqlash, chunki keyingi
   * halqa burilishini ko'rsatish uchun kerak */
    glPushMatrix();
   /* yangi burilish burchaklarda bir nechta burilishlarni
```

```
    * amalga oshiradi (bu yozilgan burilish burchagi matritsasi
    * oldingi matritsa ko'rinishidagiga nisbatan tezkor) */
    glRotatef (-CurAng,1,1,0);
    glRotatef (CurAng,1,0.0);
    /* har bir halqani chizish uchun ularni alohida yangilash
    * kerak. Shuning uchun matritsa ko'rinishini saqlashni
    * boshlash, keyin asl holiga qaytarish kerak */
    glPushMatrix();
    glTranslatef (0,0,-RingHeight/2);
    DrawRing();
    glPopMatrix();
    glPushMatrix();
    glTranslatef (0,RingHeight/2,0);
    glRotatef (90,1,0,0);
    DrawRing();
    glPopMatrix();
    glPushMatrix();
    glTranslatef (-RingHeight/2,0,0);
    glRotatef (90,0,1,0);
    DrawRing();
    glPopMatrix();
    /* tetraedrni burish uchun matritsani asl holiga keltirish */
    glPopMatrix();
    /* teksturalar rejimida to'xtatish funksiyasini o'rnatish */
    glDisable(GL_TEXTURE_2D);
    /* bo'rilishlarga o'tkazish*/
    glRotatef (CurAng,1,0,0);
    glRotatef (CurAng/2,1,0,1);
    /* tetraedrni o'z o'qi atrofida aylantirish  uni OZ o'qi bo'ylab
pastga siljitish */
    glTranslatef (0,-0.33,0);
    /* tetraedr uchun diffuzion tasvirlash  ranglarini berish  */
    glColor3fv(mat_diff2);
    /* tetraedr qurishni boshlash */
    glCallList(TETR_LIST);
    }
    void Display(void)
```

```
    {
    /* joriy chuqurlik va kadr buferini  initsializatsiya qilish */
    glClear(GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT);
    /*obyektlarni tuzish*/
    DrawFigures();
    /* kadr buferlari joyini o'zgartirish */
    glutSwapBuffers();
    }
    void Redraw(void)
    {
    /* joriy burilish burchagini kattalashtirish */
    CurAng+=1;
    /* tasvirni yaratish prosedurasini chaqirish signali (yangilash
uchun) */
    glutPostRedisplay();
    }
    int main(int argc, char **argv)
    {
     /*GLUT kutubhonasi funksiyalarini initsializatsiyalash */
     glutInit(&argc, argv);
     /* RGB formatda ranglarni aks ettiruvchi ikkilangan buferli
rejimni tayinlash,
     * chuqurlik buferidan foydalanish  */
     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB |
GLUT_DEPTH);
     /* ilova oynasini hosil qilish*/
     glutCreateWindow("Example of using OpenGL");
     /* tasvirlarni yaratish funksiyasini ro'yxatdan o'tkazish*/
     glutDisplayFunc(Display);
     /*tasvirlarni yangilash funksiyasini ro'yxatdan o'tkazish */
     glutIdleFunc(Redraw);
     /* OpenGL funksiya initsializatsiyasi */
     Init();
     /* hodisalarni qayta ishlash sikli */
     glutMainLoop();
     return 0;
```

}

ILOVA E. MUSTAQIL BAJARISH UCHUN MASALALAR

1. Paravoz. Ekranda harakatlanayotgan poezdni boshqaruvchi mashinist ko'rishi mumkin bo'lgan manzara: relslar, ustunlar, yo'l qurilishi va boshqalar tasvirlansin.

Masalaga oydinlik. Burilishlar, strelkalar, poezd tezligini o'zgarishi, qarama-qarshi yo'nalishdagi poezlar va xokazolar hisobga olinsin.

2. Parda. Ekranda qandaydir teatr tomoshasining final sahnasi yakunida chapdan va o'ngdan yopilayotgan parda tasvirlansin. Pardada «Tamom» degan yozuv yozilgan.

3. Atom. Kimyoviy elementning atom modeli, ya'ni o'z orbitasida aylanuvchi yadro va elektronlar tasvirlansin. Elektronlarning orbitalar bo'yicha taqsimlanishi beriladi.

Masalaga oydinlik. Oldindan tayyorlangan faylda Mendeleyev davriy sistemasidagi barcha elementlar elektronlarning orbitalar bo'yicha taqsimlanishi saqlanadi. Foydalanuvchi faqat kimyoviy element tartib raqamini yoki belgilanishini kiritadi.

4. Mayatnik. Ekranda harakatlanayotgan matematik mayatnik tasvirlansin. Matematik mayatnikning uzunligi boshlang'ich holati beriladi.

Masalaga oydinlik. Xavoning qarshiligi e'tiborga olinsin. Mayatnikning o'rniga arg'imchoq tasvirlansin.

5. Bayroq. Shamolda xilpirayotgan bayroqni tasvirlang (masalan yurtimiz bayrog'i).

6. Reklama. Ixtiyoriy mahsulot yoki xizmat turining dinamik reklamasi ishlab chiqilsin.

7. Xisob-kitoblar. Qo'shish va ayirish operatsiyalarini namoyish qiluvchi buxgalterlik hisob-kitobi ishlab chiqilsin. O'n va arifmetik amallar belgisi klaviaturadan kiritiladi. Razryad bo'yicha olib boriladigan amallar ko'rinib turishi uchun kutish vaqti hisobga olinsin.

8. Ekranda bayram salyuti, ya'ni (bir nechta stvoldan otilgan) raketalarning uchishi, portlashi va oskolkalarning yerga tushishi tasvirlansin. Rangli effektlarni har kim o'z xohishi bo'yicha tovush yordamida berishi mumkin.

9. Oyning orbita bo‘yicha aylanib chiqish sikli 28 sutkani tashkil qiladi. Harakatlanish orbitasini doimiy deb hisoblash mumkin. Oyning harakatlanish jarayonida tug‘ilishi, o‘sishi va «eskirishi» yulduzli osmonda tasvirlansin.

10. Ekranda 10 ta aylanadan va har biriga mos ravishda qo‘yilgan aylanadan iborat (sport yoki armiya) nishonlar tasvirlangan. Berilgan vaqtda yoki kompyuter tugmasi bosilganda o‘q otiladi. Natijada nishonda o‘q tekkan joy bo‘lib qoladi. Bu jarayon zahiradagi o‘q tamom bo‘lguncha yoki tugma bosilguncha (to‘plangan ballarni hisoblash davom etadi). Foydalanuvchi o‘qlarni nishonga tekkizish nuqtalarini o‘zi berishi mumkin.

Masalaga oydinlik. Nishonga urish nuqtasini sichqoncha yoki klaviatura yordamida belgilanadi. Oldin urilgan nuqtalarga bir nechta o‘q tegishi mumkin.

11. Qimor. Kompyuter-olib boruvchi, odam-o‘yin ishtirokchisi rolida. Ekranda uchta idish, ularning bittasini ostida sharik (o‘yin boshlangan paytda u ko‘rinadi) tasvirlansin. Olib boruvchi berilgan tezlikda sharni idishlar ostida almashtiradi. Almashtirish tugaganidan so‘ng foydalanuvchi sharning qaerdaligini topishi kerak.

TEST SAVOLLARI

1. «Kompyuter grafikasi»ni qanday tasvirli ko‘rinishda ifodalash mumkin?

A. belgilar->tasvir
B. tasvir->belgilar
C. belgilar ->belgilash
D. tasvir ->tasvir
E. to‘g‘ri javob yo‘q

2. Qaysi almashtirishlar matritsalari simmetrik?

A. kuchish, masshtablash
B. masshtablash, akslantirish
C. masshtablash, burish
D. burish, akslantirish
E. barcha almashtirishlar

3. Qaysi almashtirishlarda matritsalar simmetrik emas?

A. masshtablash, akslantirish
B. akslantirish, burish
C. masshtablash, burish
D. ko‘chish, burish
E. to‘g‘ri javob yo‘q

4. Qaysi almashtirishning matritsasi determinanti 1 ga teng?

A. burish
B. masshtablash
C. ko‘chish
D. akslantirish
E. to‘g‘ri javob yo‘q

5. Qaysi almashtirishning matritsasi determinanti -1 ga teng?

A. burish
B. masshtablash
C. ko‘chirish

D. akslantirish
E. to‘g‘ri javob yo‘q

6. Qaysi Platon jismining 20 ta yoqi, 30 ta qirrasi, 12 ta uchi mavjud?

A. geksaedr
B. tetraedr
C. oktaedr
D. dodekaedr
E. ikosaedr

7. Qaysi Platon jismining 12 ta yoqi, 30 ta qirrasi, 20 ta uchi mavjud?

A. tetraedr
B. geksaedr
C. oktaedr
D. dodekaedr
E. ikosaedr

8. Qaysi Platon jismining 8 ta yoqi, 12 ta qirrasi, 6 ta uchi mavjud?

A. geksaedr
B. oktaedr
C. tetraedr
D. dodekaedr
E. ikosaedr

9. Platon jismlarini yoqi (Yo), qirrasi (Q) va uchi (U) orasida qanday bog‘liqlik (Eyler tenglik) mavjud?

A. Yo+U=Q+1
B. Yo+U=Q+3
C. Yo+U=Q
D. Yo+U=Q+4
E. Yo+U=Q+2

10. Keltirilgan proeksiyalardan qaysilari parallel hisoblanadi?

A. ortografik proeksiyalash

B. aksonometrik proeksiyalash
C. kavale proeksiyalash
D. kabine proeksiyalash
E. barcha javoblar to‘g‘ri

11. Keltirilgan proeksiyalardan qaysilari markaziy proeksiyalash hisoblanadi?

A. ortografik proeksiyalash
B. aksonometrik proeksiyalash
C. kavale proeksiyalash
D. bir nuqtali proeksiyalash
E. kabine proeksiyalash

12. Proeksiyalash matritsasi qanday bo‘ladi?

A. determinanti 0 ga teng emas
B. simmetrik
C. nosimmetrik
D. determinanti 0 ga teng
E. birlik matritsa

13. Ikki qo‘shni piksellar 8 bog‘lamli hisoblanadi, agarda P1(x1,y1) va P2(x2,y2) nuqtalar orasida quyidagi bog‘liqlik mavjud bo‘lsa.

A. |x1-x2|<=1, |y1-y2|<=1
B. |x1-x2|+|y1-y2|<=1
C. |x1-x2|+|y1-y2|<=2
D. |x1-y1|<=1, |x2-y2|<=1
E. |x1-y1|+|x2-y2|<=1

14. Ikki qo‘shni piksellar 4 bog‘lamli hisoblanadi, agarda P1(x1,y1) va P2(x2,y2) nuqtalar orasida quyidagi bog‘liqlik mavjud bo‘lsa.

A. |x1-x2|<=1, |y1-y2|<=1
B. |x1-x2|+|y1-y2|<=1
C. |x1-x2|+|y1-y2|<=2
D. |x1-y1|<=1, |x2-y2|<=1
E. |x1-y1|+|x2-y2|<=1

15. Keltirilgan algoritmlardan qaysi biri kesmani to‘g‘ri to‘rtburchak bilan kesishga kiradi?

A. Brezenxeym algoritmi
B. Sazerland-Xogdman algoritmi
C. Sazerland-Koxen algoritmi
D. Roberts algoritmi
E. Varnok algoritmi

16. Kesmani rastr algoritmi orqali chizuvchi algoritmni aniqlang.

A. Sazerland-Koxen algoritmi
B. Roberts algoritmi
C. Appel algoritmi
D. Brezenxeym algoritmi
E. Sazerland-Xogdman algoritmi

17. Qaysi splayn egri chizig‘i ikki tayanuvchi nuqta va shu nuqtalardagi urunuvchi vektorlar bilan beriladi?

A. Beze egri chizig‘i
B. Ermit egri chizig‘i
C. B-splayn egri chizig‘i
D. Beta-splayn egri chizig‘i
E. barcha splayn egri chiziqlari

18. Qaysi splayn egri chiziqlari 4 ta tayanuvchi nuqtalar bilan beriladi?

A. Ermit, Beze, B-splayn
B. B-splayn , Ermit, Beta-splayn
C. Beze, B-splayn , Veta-splayn
D. Beze, Ermit, Beta-splayn
E. to‘g‘ri javob yo‘q

19. Keltirilgan olib tashlash algoritmlardan qaysi birida obyekt nuqtasidan chizma tekisligigacha bo‘lgan masofa haqida ma’lumot har bir piksel uchun hisobga olinadi?

A. Roberts algoritmi
B. Appelya algoritmi

C. Varnok algoritmi.
D. z-bufer algoritmi.
E. tartiblash algoritmi.

20. Keltirilgan qaysi olib tashlash algoritmida sonli-ko‘rinmaslik tushunchasi hisobga olinadi?

A. Appel algoritmi.
B. z-bufer algoritmi.
C. Roberts algoritmi.
D. Varnok algoritmi.
E. satrma-satr tekshirish algoritmi.

21. Keltirilgan qaysi uzoqlashtirish algoritmida maydonni to‘g‘ri burchakli 4 qismga bo‘lish ishlatiladi?

A. z-bufer algoritmi.
B. Roberts algoritmi.
C. Appel algoritmi.
D. Varnok algoritmi.
E. chuqurligi bo‘yicha tartiblash algoritmi.

22. Poligonal setkani qaysi bo‘yash metodida tasvir realroq bo‘ladi?

A. Guro metodi.
B. Fong metodi.
C. diffuzion akslantirish.
D. Aks.
E. barcha javoblar to‘g‘ri.

23. Keltirilgan rang modellaridan qaysi biri oddiy foydalanuvchilar uchun mo‘ljallangan?

A. RGB
B. CMY
C. CMYK
D. HSV
E. barcha javoblar to‘g‘ri.

24. Keltirilgan rang modellaridan qaysilari apparatli ta'minotlar uchun mo'ljallangan?

A. RGB, HSV
B. CMY, HLS
C. HSV, HLS
D. RGB, CMY
E. CMYK, HSV

25. Beze va B-splayn tekisligining elementar bo'lagi nechta tayanuvchi nuqtalar bilan beriladi?

A. 16
B. 4
C. 10
D. 20
E. 8

26. OpenGL ga berilgan to'g'ri ta'rifni belgilang?

A. ikki va uch o'lchovli grafika sohasida ilovalar yaratish uchun mo'ljallangan amaliy dasturiy interfeys.

B. ochiq grafik kutubxonaga ega bo'lgan amaliy dasturiy interfeys.

C. kompyuter grafikasi sohasida ilovalar yaratish uchun mo'ljallangan amaliy dasturiy interfeys.

D. kompyuter grafikasi sohasida tasvirlarni vizuallashtirish uchun mo'ljallangan amaliy dasturiy interfeys.

E. barcha javoblar to'g'ri.

27. OpenGL kutubxonasining o'ziga xos xususiyatlarini belgilang?

A. Barqarorlik, ishonchlilik, qo'llashning osonligi.

B. Barqarorlik, ishonchlilik va uzatuvchanlik, qo'llashning osonligi, ma'lumotlardan birgalikda foydalanish.

C. Barqarorlik, ishonchlilik va uzatuvchanlik, qo'llashning osonligi.

D. Barqarorlik, ishonchlilik, qo'llashning osonligi, ma'lumotlardan birgalikda foydalanish.

E. Barqarorlik, uzatuvchanlik, qo'llashning osonligi, ma'lumotlardan birgalikda foydalanish.

28. OpenGL ning asosiy imkoniyatlarini belgilang?

A. Primitivlarni tavsiflash, ranglar manbasini tavsiflash, atributlarni tayinlash, vizuallashtirish va geometrik o'zgartirish funksiyalari.

B. Primitivlarni tavsiflash, ranglar manbasini tavsiflash, atributlarni tayinlash, vizuallashtirish, geometrik o'zgartirish va tasvirlash funksiyalari.

C. Primitivlarni tavsiflash, ranglar manbasini tavsiflash, atributlarni tayinlash, geometrik o'zgartirish va tasvirlash funksiyalari.

D. Primitivlarni tavsiflash, atributlarni tayinlash, vizuallashtirish, geometrik o'zgartirish va tasvirlash funksiyalari.

E. Primitivlarni tavsiflash, atributlarni tayinlash, vizuallashtirish, geometrik o'zgartirish funksiyalari.

29. OpenGL funksiyalari qanday texnologiyasi asosida qurilgan?

A. Mijoz texnologiyasi.
B. Server texnologiyasi.
C. Mijoz-server texnologiyasi.
D. Avtomatlashtirilgan loyihalash texnologiyasi.
E. Dasturlash texnologiyasi.

30. OpenGL kutubxonasi qaysi dasturlash tizimlarida avtomatik o'rnatiladi?

A. Microsoft Visual C++, S##, DevC++
B. Microsoft Visual C++, S##, Borland C++
C. Microsoft Visual C##, Delphi, DevC++, Borland C++
D. Microsoft Visual C##, DevC++, Borland C++
E. Microsoft Visual C++, DevC++, Borland C++

31. Nuqta, chiziq, ko'pburchak va boshqa geometrik obyektlar OpenGL da qanday funksiya sifatida qaraladi?

A. Primitivlarni tavsiflash.
B. Atributlarni tayinlash.
C. Geometrik obyektlarni berish.
D. Obyektni vizuallashtirish.
E. Protsedurali funksiya.

32. OpenGL da virtual fazoda kuzatuvchi holatini belgilash qaysi funksiyaga mansub?

A. Primitivlarni tavsiflash.
B. Atributlarni tayinlash.
C. Vizuallashtirish.
D. Geometrik o‘zgartirish.
E. Prosedurali funksiya.

33. OpenGL dan foydalanganda ekranda nima hosil bo‘lishini qaysi funksiya belgilaydi?

A. Primitiv.
B. Atribut.
C. Ranglar.
D. Geometrik obyektlar.
E. Funksiyalar.

34. OpenGLda tasvirlarni yangilash funksiyasi qanday vazifalarni bajaradi?

A. OpenGL buferlarini tozalash.
B. Kuzatuvchining holatini o‘rnatish.
C. Geometrik obyektlarni chizish.
D. Geometrik obyektlar o‘zgartirish.
E. Barcha javoblar to‘g‘ri.

35. OpenGLda tasvirlanyotgan obyektni ekranda chiqarish usulini nima belgilaydi?

A. Primitiv.
B. Atribut.
C. Ranglar.
D. Funksiyalar.
E. Geometrik obyektlar.

36. GL kutubxonasining barcha o‘zgarmaslari qanday qo‘shimcha bilan boshlanadi?

A. GL_
B. GL

C. gl/gl.h
D. void GL
E. #include <gl/gl.h>

37. GLUT kutubxonasining barcha o'zgarmaslari qanday qo'shimcha bilan boshlanadi?

A. GLUT _
B. GLUT
C. gl/glut.h
D. void GLUT
E. #include <gl/glut.h>

38. GL_DEPTH_BUFFER_BIT buyrug'i qanday vazifani bajaradi?

A. Bufer rangini tozalash
B. Bufer rangini to'ldirish
C. Bufer chuqurligini tozalash
D. Bufer chuqurligini to'ldirish
E. To'g'ri javob berilmagan

39. GLU kutubxonasining barcha buyruqlari qanday qo'shimcha bilan boshlanadi?

A. GLU_
B. GLU
C. gl/glu.h
D. void GLU
E. #include <gl/glu.h>

40. OpenGL da kadr buferini tozalash uchun qanday buyruq ishlatiladi?

A. glClearColor
B. glClear
C. glVertex
D. glTranlsate
E. GL_COLOR_BUFFER_BIT

41. OpenGL da kadr buferini to'ldirish uchun qanday buyruq ishlatiladi?

A. glClearColor
B. glClear
C. glVertex
D. glTranlsate
E. GL_DEPTH_BUFFER_BIT

42. OpenGL da uchlarning joriy rangini tayinlash uchun qanday buyruq ishlatiladi?

A. glColor
B. glClear
C. glVertex
D. glTranlsate
E. glColor3fv

43. OpenGL da uchlarning koordinatasini tayinlash uchun qanday buyruq ishlatiladi?

A. glClearColor
B. glClear
C. glVertex*
D. glTranlsate*
E. glPushMatrix()

44. OpenGL da kadr buferini qora rang bilan to'ldirish qaysi javobda to'g'ri berilgan?

A. glClearColor(1, 0, 0, 0)
B. glClearColor(0, 0, 0, 1)
C. glClearColor(0, 1,1, 1)
D. glClearColor(0, 1, 1, 0)
E. glClearColor(1, 1, 1, 0)

45. OpenGL da obyektni burish uchun qanday buyruq ishlatiladi?

A. glRotate*
B. glScale*
C. glVertex*

D. glTranlsate*
E. glDisable*

46. OpenGL da obyektni ko'chirish uchun qanday buyruq ishlatiladi?
A. glRotate*
B. glScale*
C. glDisable*
D. glVertex*
E. glTranlsate*

47. OpenGL da obyektni masshtablashtirish uchun qanday buyruq ishlatiladi?
A. glRotate*
B. glScale*
C. glVertex*
D. glDisable*
E. glTranlsate*

48. glVertex2* buyrug'i qanday qiymatlarni qabul qiladi?
A. x va y qiymatlarini qabul qiladi, z koordinatasiga 0, w koordinatasiga 1 qiymatini kiritadi.
B. x va y qiymatlarini qabul qiladi, z va w koordinatalariga 0 qiymatini kiritadi.
C. x va y qiymatlarini qabul qiladi, z va w koordinatalariga 1 qiymatini kiritadi.
D. x va y qiymatlarini qabul qiladi, z koordinatasiga 1, w koordinatasiga 0 qiymatini kiritadi.
E. A va B javoblari to'g'ri.

49. OpenGL da uchburchakli primitivlarning qayday turlari mavjud?
A. GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN
B. GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_ LOOP
C. GL_TRIANGLES, GL_TRIANGLE_ FAN, GL_TRIANGLE_ LOOP

D. GL_TRIANGLES, GL_TRIANGLE_ FAN, GL_TRIANGLE_ LOAD

E. GL_TRIANGLES, GL_TRIANGLE_ LOOP, GL_TRIANGLE_ LOAD

50. void glPolygonMode (GLenum *face*, Glenum *mode*) **buyrug'ida** *mode* **parametrining vazifasi?**

A. Ko'pburchakning qanday akslanishini belgilaydi.

B. Ko'pburchak tipini o'rnatishda ishlatiladi.

C. Ko'pburchakning ekranda qanday tasvirlanishini belgilaydi.

D. Ko'pburchakni faqat uchlarini tasvirlaydi.

E. yorug'likni hisobga olib ko'pburchaklar joriy rang bilan bo'yab chiqiladi.

51. Bir qancha uchlarni glVertex*() buyrug'ini chaqirmasdan foydalanish imkonini beruvchi buyruq to'g'ri berilgan javobni belgilang?

A. void glVertexPointer(GLint *size*, GLenum *type*, GLsizei *stride*, void* *ptr*).

B. void glNormalPointer (GLenum type, GLsizei stride, void *pointer).

C. void glColorPointer (GLint size, GLenum type, GLsizei stride, void *pointer).

D. void glEnableClientState (GLenum array)

E. void glDrawArrays (GLenum mode, GLint first, GLsizei count)

52. Joriy tipdagi matritsa elementlarini aniqlash uchun qaysi buyruq chaqiriladi?

A. void glLoadMatrix [f d] (GLtype *m)

B. void glLoadIdentity (void)

C. void glPushMatrix (void)

D. void glPopMatrix (void)

E. void glMultMatrix [f d] (GLtype **m*)

53. Joriy matritsani boshqa matritsaga ko'paytirish uchun qaysi buyruq chaqiriladi?

A. void glLoadMatrix [f d] (GLtype *m)

B. void glLoadIdentity (void)
C. void glPushMatrix (void)
D. void glPopMatrix (void)
E. void glMultMatrix [f d] (GLtype *m)

54. OpenGL da tuman effektini yoqish uchun qanday buyruqni chaqirish zarur?

A. glEnable(GL_FOG)
B. glEnable (GL_LIGHTING)
C. glLightModel
D. GL_SPOT_DIRECTION
E. GL_MAX_LIGHT

55. OpenGL da tekstura bilan ishlash uchun qanday harakatlar ketma-ketligini amalga oshirish zarur?

A. Tasvirni tanlash va uni kerakli formatga keltirish
B. Tasvirni OpenGL ga uzatish
C. Tekstura obyektga qanday qo‘yilishini va u bilan qanday munosabatda bo‘lishini aniqlab olish
D. Teksturani obyekt bilan bog‘lash
E. Barcha javoblar to‘g‘ri

FOYDALANILGAN ADABIYOTLAR

1. Боресков А. Расширения OpenGL. – СПб.: БХВ-Петербург. 2006. – 288 с.

2. Васильев С.А. OpenGL. Компьютерная графика. Учебное пособие. - Тамбов: Изд-во Тамб. гос. техн. ун-та, 2005. – 80 с.

3. Гайдуков С.А. OpenGL. Профессиональное программирование трехмерной графики на C++. – СПб.: БХВ-Петербург. 2004. – 736 с.

4. Компьютерная графика: Полигональные модели / А.В. Боресков, Е.В. Шикин. – М.: Изд-во «Диалог-МИФИ», 2005. – 461 с.

5. Миронов Д. Компьютерная графика в дизайн. Учебник. – СПб.: БХВ-Петербург, 2008. – 560 с.

6. Петров М., Молочков К. Компьютерная графика. Учебник. – Питер, 2002. – 736 с.

7. Петров М.Н. Компьютерная графика. – СПб.: Питер. 2011. – 544 с.

8. Порев В.Н. Компьютерная графика. – СПб.: БХВ-Петербург, 2005. – 432 с.

9. Рейнбоу В. Компьютерная графика. Энциклопедия. Питер, 2003. – 876 с.

10. Рихсибоев Т. Компьютер графикаси. – Тошкент: Ўзбекистон ёзувчилар уюшмаси «Адабиёт» жамғармаси нашриёти, 2006. – 154 б.

11. Херн, Дональд, Бейкер, М.Паулин. Компьютерная графика и стандарт OpenGL, 3-е издание.: Пер. с анг. –М.: Издательский дом «Вильямс», 2005. – 1168 с.

12. Хилл Ф. OpenGL: Программирование компьютерной графики. – СПб.: Питер, 2002. – 1089 с.

13. Шикин А.В., Боресков А.В. Компьютерная графика. Полигональные модели. Москва, ДИАЛОГ-МИФИ, 2001. – 464 с.

14. Шрайнер Дэйв, Ву М., Нейдер Дж., Девис Том. OpenGL. Руководство по программированию. Библиотека программиста. 4-е изд. Питер, 2006.–624с.

15. Эйнджел Эдвард. Интерактивная компьютерная графика. Вводный курс на базе OpenGL, 2 изд.: Пер. с анг. – М.: Издательский дом «Вильямс», 2001. – 592 с.

16. Яцюк О.Г., Романычева Э.Т. Компьютерные технологии в дизайне. Эффективная реклама. – СПб.: БХВ-Петербург, 2002. – 432 с.

17. David Salomon. The Computer Graphics Manual. Volume 1. – Springer, 2012. – 1564 p.

18. Donald Hearn, M. Pauline Baker. Computer graphics. C version. – Prentice Hall, 1997. – 662 p. – 2nd edition. – ISBN: 0135309247.

19. Mamarajabov M., Ashurov M., Umarova U. CorelDRAW dasturi va uning imkoniyatlari. Metodik qo'llanma. – Toshkent: TDPU, 2011. – 82 b.

20. Nazirov Sh.A., Nuraliyev F.M., Aytmuratov B.Sh. Rastr va vector grafika. – T.: G'.G'ulom, 2007. – 192 b.

21. Nazirov Sh.A., Nuraliyev F.M., Tillayeva M.A. Uch o'lchovli modellashtirish. – T.: «Ilm ziyo», 2012. – 144 b.

22. Гребенников К.А. Компьютерная графика как средство профессиональной подготовки специалистов-дизайнеров. (на материалах среднего профессионального образования): Автореф. дис.... канд. пед. наук. – Воронеж: РУДН, 2002. – 28 с.

23. Крайнова О.А. Проектирование методической системы обучения студентов дисциплине «Компьютерная графика» на примере специальности 030100 «Информатика»: Автореф. дис. ... канд. пед. наук. – Москва: МГУ, 2004. – 24 с.

24. Нодельман Л.Я. Технология обучения студентов художественно-графического факультета компьютерной графике: Автореф. дис. ... канд. пед. наук. – Москва: МГУ, 2000. – 275 с.

25. Петрова Н.П. Компьютерная графика и анимация как средство медиа-образования: Автореф. дис. ... канд. пед. наук. – Москва: МПГУ, 1997. – 156 с.

26. Чернякова Т.В. Методика обучения компьютерной графике студентов вуза: Дисс. ... канд. пед. наук. – Екатеринбург: РГППУ, 2010. – 201 с.

27. Eminov A.G'. Bo'lajak o'qituvchilarning kompyuter grafikasi bo'yicha kompetentligini rivojlantirish metodikasi («Informatika va axborot texnologiyalari» o'quv fani misolida): Dis. ... ped. fan. nomz. – Toshkent, 2012. – 139 b.

GLOSSARIY

Affin almashtirishlari – chiziqli almashtirish, masalan, koordinatani almashtirish.

Alfa-kanal – shaffoflik xarakteristikasi.

Amaliy dasturlar interfeysi (Application Program Interface – API) - o‘zlarining dasturlarini tegishli operatsion tizimlar bilan uyg‘unlashuvi uchun dasturiy ta’minot ishlab chiquvchilar amal qilishlari kerak bo‘lgan vazifalar yig‘masining spetsifikatsiyasi.

Animatsiya (animation) – bir necha tasvir yoki kadrlarni ko‘rsatish orqali kino sifatida qabul qilinadigan kadrlar ketma-ketligi.

Antialiasing (antialiasing) – alohida burchakli piksellar rangini belgilanishini silliqlash yo'li orqali rastrli tasvirlarning pog'onali effektlarini bartaraf etish.

Avtomatlashgan loyihalash tizimi, SAPR (Computer Aided Design, CAD) – kompyuter yordamida murakkab ob'ektlarni loyihalash uchun mo'ljallangan tizim.

Bitli massiv – xotira yoki diskda saqlanadigan rastr.

BMP – fayllarning rastrli grafik formati, Windows operasion tizimi muhitidagi dasturlarda keng ishlatiladi. Tasvir bitli massiv shaklida saqlanadi.

CMY (Cyan, Magenta, Yellow) – bu rang modeli subtraktiv, ya`ni biror kerakli bo‘lgan rangni hosil qilish uchun asosiy ranglar, oq rangdan ajraladi.

CMYK (Cyan, Magenta, Yellow, blacK) – to’rt hil rangga asoslangan subtraktiv rang modeli.

Diffuzion qaytish – yorug‘likning barcha yo‘nalishlar bo‘yicha tekis tarqalishi.

GIF (Graphics Interchange Format) - pikselli grafik tasvirni global kompyuter tarmog'iga uzatish uchun maxsus ishlab chiqilgan format.

Grafik qurilmalar interfeysi (Graphic Device Interface, GDI) – Windows operasion tizimining quyi tizimi.

Kompyuter grafikasi – kompyuter yordamida yaratiladigan tasvir.

JPEG (Joint Photographic Experts Group) – rastrli tasvirlar uchun ma'lumotlarni samarali zichlovchi standart fayl formati.

OpenGL (Open Graphics Library) – ikki va uch o'lchovli grafika sohasida ilovalar yaratish uchun ancha keng tarqalgan, apparatga bog'liq bo'lmagan amaliy dasturiy interfeys.

Oyna (window) – grafik natijalar tekisligi qismi. Windows operasion tizimida bu fundamental tushuncha hisoblanadi.

Palitra – aniq bir grafik tasvir uchun foydalanadigan biror bir rang modeli asosida tashkil qilingan ranglar to'plami.

Piksel (pixel) – rastr elementi.

Platon jismlari – barcha yoqlari to'g'ri ko'pburchaklar va barcha uchlariga tegishli burchaklar o'zaro teng bo'lgan qavariq ko'pyoqliklar.

Poligonal setka – bu fazoviy obyektni tasvirlovchi o'zaro bog'liq balandliklar, qirralar va yoqlar (ko'pburchaklar) to'plami.

Proeksiya – n o'lchovli koordinatalar sistemasida berilgan nuqta(lar)ni n dan kam bo'lgan o'lchovli koordinatalar sistemasidagi nuqta(lar)ga geometrik almashtirish.

Rang chuqurligi – bitta pikselning rangi haqida ma'lumot saqlash uchun ajratilgan bitlar soni bilan aniqlanadi.

Rang modeli – poligrafiyada yoki monitorning rangli kanallarida foydalanish mumkin bo'lgan bo'yoqlarning chegaralangan soni yordamida ranglar namoyish qilinadigan tizim.

Rasterizasiyalash (rasterization) – tasvir elementlarini vektorli tavsiflash asosida rastr tasvir yaratish.

Rendering (rendering) – axborotlarni grafik ko'rinishda tasvirlash jarayoni.

RGB (Red, Green, Blue) – bu rang modeli additiv, ya'ni biror bir kerakli rangni hosil qilish uchun uning asosiy ranglari yig'iladi.

Shrift (front) – kompyuter tizimlari, poligrafiyada matnni aks ettirish uchun mo'ljallangan simvolli belgilar to'plami.

Splayn (spline) – murakkab shakllar sirtini yoki chiziqlar qismini aproksimasiyalash uchun ishlatiladigan maxsus tipdagi egri chiziq yoki sirt. Bir qancha bog'liq splaynlar yagona bir butun sifatida shaklni ifodalaydi.

Tekstura (texture) – obyektni bo'yash usuli, ko'proq rastrli obrazlar ko'rinishida ishlatiladi. Ko'proq rastrli obrazlar ko'rinishida ishlatiladi.

TIFF (Target Image File Format) – rangli tasvirlarni skanerlashdan olingan natijalarni saqlash uchun universal format.

Vektor grafika – alohida obyektlarning vektorli tavsiflanishi asosida tasvirning yaratilishi.

Videoadapter – bu grafik qurilma bo'lib, bevosita uning yordamida kompyuter monitori ekranida tasvir shakllanadi.

Virtual reallik (virtual reality) – haqiqatan mavjud bo'lmaydi, biroq, kompyuter tizimlari insonning ko'rish, eshitish va boshqa hissiyot organlariga ta'sir qilib, ushbu dunyodan erkin foydalanish illuziyasini keltirib chiqaradi.

MUNDARIJA

СОДЕРЖАНИЕ

CONTENTS

Chapter I. DESIGN BASICS

Chapter II. THEORETICAL ASPECTS OF COMPUTER GRAPHICS

Chapter III. OPENGL GRAPHICS LIBRARY

QAYDLAR UCHUN

SH.A.NAZIROV, F.M.NURALIYEV,
B.Z.TO‘RAYEV

KOMPYUTER GRAFIKASI VA DIZAYN

Toshkent – «Fan va texnologiya» – 2015

Muharrir:	A.Eshov
Tex. muharrir:	M.Holmuhamedov
Musavvir:	D.Azizov
Musahhih:	N.Hasanova
Kompyuterda sahifalovchi:	Sh.Mirqosimova

E-mail: tipografiyacnt@mail.ru Tel: 245-57-63, 245-61-61.
Nashr.lits. AI№149, 14.08.09. Bosishga ruxsat etildi 30.10.2015.
Bichimi 60x84 $^{1}/_{16}$. «Timez Uz» garniturasi. Ofset bosma usulida bosildi.
Shartli bosma tabog‘i 15,75. Nashriyot bosma tabog‘i 16,0.
Tiraji 300. Buyurtma № 156.

«Fan va texnoiogiyalar Markazining
bosmaxonasi» da chop etildi.
100066, Toshkent sh., Olmazor ko'chasi, 171-uy.

ISBN 978-9943-990-80-7