Міністерство освіти і науки України
Міжнародний економіко-гуманітарний університет імені академіка Степана Дем'янчука

Р.М.Літнарович, І.Ф. Чернецький, М.І. Дєдух

Сучасні технології опрацювання графічної інформації
Курс лекцій

Частина 1

Рівне – 2012 р.

УДК 378.147.31:004.92
Літнарович Р.М., Чернецький І.Ф. , М.І. Дєдух. Сучасні технології опрацювання графічної інформації. Курс лекцій. Частина 1. МЕГУ, Рівне, 2012.- 130 с.

Приведений курс лекцій, який читається магістрантам-інформатикам факультету Кібернетики МЕГУ.
Ключові слова: комп'ютерна графіка, візуалізація, опрацювання, модель.

Приведён курс лекций, который читается магистрантам-информатикам факультета Кибернетики МЕГУ.
Ключевые слова: компьютерная графика, визуализация, обработка, модель.

Adjusted lectures, which reads undergraduate Informatics Faculty of Cybernetics IEGU.
Keywords: computer graphics, visualization, processing, model.
МЕГУ, Рівне, 2012.- 129с.

Рецензенти: В.Г.Бурачек, доктор технічних наук, професор
Є.С. Парняков, доктор технічних наук, професор
В.О.Боровий , доктор технічних наук, професор

ЗМІСТ

Лекція 1. Загальне введення в комп'ютерну графіку
Зміст

Предмет і область застосування комп'ютерної графіки

Комп'ютерна графіка - це область інформатики, яка охоплює всі сторони формування зображень за допомогою комп'ютера. З'явившись в 1950-х роках, вона спочатку давала можливість виводити лише кілька десятків відрізків на екрані. В наші дні засоби комп'ютерної графіки дозволяють створювати реалістичні зображення, не поступаються фотографічним знімкам. Створено різноманітне апаратне і програмне забезпечення для отримання зображень самого різного виду та призначення - від простих креслень до реалістичних образів природних об'єктів. Комп'ютерна графіка використовується практично у всіх наукових і інженерних дисциплінах для наочності сприйняття і передачі інформації. Застосування її для підготовки демонстраційних слайдів уже вважається нормою. Тривимірні зображення використовуються в медицині (комп'ютерна томографія), картографії, поліграфії, геофізиці, ядерної фізики та інших областях. Телебачення і інші галузі індустрії розваг використовують анімаційні засоби комп'ютерної графіки (комп'ютерні ігри, фільми). Загальноприйнятою практикою вважається також використання комп'ютерного моделювання при навчанні пілотів і представників інших професій (тренажери). Знання основ комп'ютерної графіки зараз необхідно і інженеру, і вченому.

Кінцевим результатом застосування засобів комп'ютерної графіки є зображення, яке може використовуватися для різних цілей. Оскільки найбільша кількість інформації людина отримує за допомогою зору, вже в стародавні часи з'явилися схеми і карти, використовувані при будівництві, в географії і в астрономії.

Сучасна комп'ютерна графіка - це досить складна, грунтовно опрацьована і різноманітна науково-технічна дисципліна. Деякі її розділи, такі як геометричні перетворення, способи опису кривих і поверхонь, до теперішнього часу вже досліджені досить повно. Ряд областей продовжує активно розвиватися: методи растрового сканування, видалення невидимих ліній і поверхонь, моделювання кольору й освітленості, текстурування, створення ефекту прозорості та напівпрозорості та ін.

Сфера застосування комп'ютерної графіки включає чотири основні області.

1. Відображення інформації

Проблема подання накопиченої інформації (наприклад, даних про кліматичні зміни за тривалий період, про динаміку популяцій тваринного світу, про екологічний стан різних регіонів і т.п.) краще за все може бути вирішена за допомогою графічного відображення.

Жодна з областей сучасної науки не обходиться без графічного представлення інформації. Крім візуалізації результатів експериментів і аналізу даних натурних спостережень існує велика область математичного моделювання процесів і явищ, яка просто немислима без графічного виводу. Наприклад, описати процеси, що протікають в атмосфері або океані, без відповідних наочних картин течій або полів температури

практично неможливо. В геології в результаті обробки тривимірних натурних даних можна отримати геометрію пластів, що залягають на великій глибині.

В медицині в даний час широко використовуються методи діагностики, які використовують комп'ютерну візуалізацію внутрішніх органів людини. Томографія (зокрема, ультразвукове дослідження) дозволяє отримати тривимірну інформацію, яка потім піддається математичній обробці і виводиться на екран. Крім цього застосовується і двовимірна графіка: енцефалограми, миограмма, що виводяться на екран комп'ютера або Графобудівник.

2. Проектування

У будівництві та техніці креслення давно являють собою основу проектування нових споруд або виробів. Процес проектування з необхідністю є ітеративним, тобто конструктор перебирає безліч варіантів з метою вибору оптимального з яких-небудь параметрами. Не останню роль в цьому відіграють вимоги замовника, який не завжди чітко уявляє собі кінцеву мету і технічні можливості. Побудова попередніх макетів - досить довгий і дорога справа. Сьогодні існують розвинені програмні засоби автоматизації проектно-конструкторських робіт (САПР), що дозволяють швидко створювати креслення об'єктів, виконувати розрахунки на міцність і т.п. Вони дають можливість не тільки зобразити проекції виробу, але і розглянути його в об'ємному вигляді з різних сторін. Такі кошти також надзвичайно корисні для дизайнерів інтер'єру, ландшафту.

3. Моделювання

Під моделюванням в даному випадку розуміється імітація різного роду ситуацій, що виникають, наприклад, при польоті літака або космічного апарату, русі автомобіля і т.п. В англійській мові це найкраще передається терміном моделювання. Але моделювання використовується не тільки при створенні різного роду тренажерів. У телевізійній рекламі, в науково-популярних та інших фільмах тепер синтезуються рухомі об'єкти, візуально мало поступаються тим, які можуть бути отримані за допомогою кінокамери. Крім того, комп'ютерна графіка надала кіноіндустрії можливості створення спецефектів, які в минулі роки були просто неможливі. В останні роки широко поширилася ще одна сфера застосування комп'ютерної графіки - створення віртуальної реальності.

4. Графічний інтерфейс користувача

На ранньому етапі використання дисплеїв як одного з пристроїв комп'ютерного виводу інформації діалог "людина-комп'ютер" в основному здійснювався в алфавітно-цифровому вигляді. Тепер же практично всі системи програмування застосовують графічний інтерфейс. Особливо вражаюче виглядають розробки в області мережі Інтернет. Існує безліч різних програм-браузерів, що реалізують в тому чи іншому вигляді засобу спілкування в мережі, без яких доступ до неї важко собі уявити. Ці програми працюють в різних операційних середовищах, але реалізують, по суті, одні і ті ж функції, що включають вікна, банери, анімацію і т.д.

У сучасній комп'ютерній графіці можна виділити такі основні напрямки: образотворча комп'ютерна графіка, обробка та аналіз зображень, аналіз сцен (перцептивна комп'ютерна графіка), комп'ютерна графіка для наукових абстракцій (когнітивна комп'ютерна графіка, тобто графіка, сприяє пізнанню).

Образотворча комп'ютерна графіка своїм предметом має синтезовані зображення. Основні види завдань, які вона вирішує, зводяться до наступних:

• побудова моделі об'єкта й формування зображення;

• перетворення моделі і зображення;

• ідентифікація об'єкта та отримання необхідної інформації.

Обробка та аналіз зображень стосуються в основному дискретного (цифрового) подання фотографій та інших зображень. Засоби комп'ютерної графіки тут використовуються для:

• підвищення якості зображення;

• оцінки зображення - визначення форми, місця розташування, розмірів та інших параметрів необхідних об'єктів;

• розпізнавання образів - виділення і класифікації властивостей об'єктів (при обробці аерокосмічних знімків, введенні креслень, в системах навігації, виявлення і наведення).

Аналіз сцен пов'язаний з дослідженням абстрактних моделей графічних об'єктів і взаємозв'язків між ними. Об'єкти можуть бути як синтезованими, так і виділеними на фотознімках. До таких завдань належать, наприклад, моделювання "машинного зору" (роботи), аналіз рентгенівських знімків з виділенням і відстеженням об'єкта, що цікавить (внутрішнього органу), розробка систем відеоспостереження.

Когнітивна комп'ютерна графіка, тільки формується новий напрям, поки ще недостатньо чітко окреслене. Це-комп'ютерна графіка для наукових абстракцій, що сприяє народженню нового наукового знання. Технічною основою для неї є потужні ЕОМ і високопродуктивні засоби візуалізації.

Одним з найбільш ранніх прикладів використання когнітивної комп'ютерної графіки є робота Ч.Страуса "Несподіване застосування ЕОМ у чистій математиці" (ТІІЕР, т. 62, № 4, 1974, с.96-99). У ній показано, як для аналізу складних алгебраїчних кривих використовується "n-мірна" дошка на основі графічного термінала.Пользуясь пристроями введення, математик може легко отримувати геометричні зображення результатів спрямованої зміни параметрів досліджуваної залежності. Він може також легко управляти поточними значеннями параметрів ", поглиблюючи тим самим своє розуміння ролі варіацій цих параметрів". В результаті отримано "кілька нових теорем і визначені напрямки подальших досліджень".

У цьому курсі передбачається розглянути наступні питання:

• представлення зображення в комп'ютерній графіці;

• способи підготовки зображення до візуалізації;

• методи виведення зображення на екран;

• методи роботи із зображенням;

• методи обчислювальної геометрії.

Отменить изменения

Коротка історія

У цьому короткому історичному нарисі ми будемо згадувати алгоритми та методи, за якими читач зможе познайомитися в даному курсі. Ці попередні згадки не повинні бентежити при першому прочитанні. По завершенні курсу можна повернутися до цього розділу і пройти його заново.
Комп'ютерна графіка в початковий період свого виникнення була далеко не настільки ефектною, якою вона стала в справжні дні. У ті роки комп'ютери перебували на ранній стадії розвитку і були здатні відтворювати тільки найпростіші контури (лінії). Ідея комп'ютерної графіки не відразу була підхоплена, але її можливості швидко росли, і поступово вона стала займати одну з найважливіших позицій в інформаційних технологіях.

Першою офіційно визнаною спробою використання дисплея для виведення зображення з ЕОМ стало створення в Массачусетському технологічному університеті машини Whirlwind-I в 1950 р. Таким чином, виникнення комп'ютерної графіки можна віднести до 1950-х років. Сам же термін "комп'ютерна графіка" придумав в 1960 г.сотрудник компанії Boeing У. Феттер.

Перше реальне застосування комп'ютерної графіки пов'язують з ім'ям Дж. Уїтні. Він займався кіновиробництвом в 50-60-х роках і вперше використовував комп'ютер для створення титрів до кінофільму.

Наступним кроком у своєму розвитку комп'ютерна графіка зобов'язана Айвен Сазерленду, який в 1961 р., Ще будучи студентом, створив програму малювання, названу

ним Альбом (альбом для малювання). Програма використовувала світлове перо для малювання найпростіших фігур на екрані. Отримані картинки можна було зберігати і відновлювати. У цій програмі було розширено коло основних графічних примітивів, зокрема, крім ліній і точок був введений прямокутник, який задавався своїми розмірами і розташуванням.

Спочатку комп'ютерна графіка була векторної, т.е.ізображеніе формувалося з тонких ліній. Ця особливість була пов'язана з технічною реалізацією комп'ютерних дисплеїв. У подальшому більш широке застосування отримала растрова графіка, заснована на уявленні зображення на екрані у вигляді матриці однорідних елементів (пікселів).

У тому ж 1961 студент Стів Рассел створив першу комп'ютерну відеогру Spacewar ("Зоряна війна"), а науковий співробітник Bell LabsЕдвард Зеджек створив анімацію "Моделювання двох-жиро ваги системи управління".

У зв'язку з успіхами в області комп'ютерної графіки великі корпорації почали виявляти до неї інтерес, що в свою чергу стимулювало прогрес в області її технічної підтримки.

Університет штату Юта стає центром досліджень в області комп'ютерної графіки завдяки Д.Евансу і А.Сазерленду, які в цей час були найбільш помітними фігурами в цій галузі. Пізніше їх коло став швидко розширюватися. Учнем Сазерленда став Е.Кетмул, майбутній творець алгоритму видалення невидимих поверхонь з використанням Z-буфера (1978). Тут же працювали Дж.Варнок, автор алгоритму видалення невидимих граней на основі розбивки області (1969) і засновник Adobe System (1982), Дж.Кларка, майбутній засновник компанії Silicon Graphics (1982). Всі ці дослідники дуже сильно просунули алгоритмічну сторону комп'ютерної графіки.

У тому ж 1971 Гольдштейн і Нагель вперше реалізували метод трасування променів з використанням логічних операцій для формування тривимірних зображень.

У 1970-і роки стався різкий стрибок у розвитку обчислювальної техніки завдяки винаходу мікропроцесора, в результаті чого почалася мініатюризація комп'ютерів і швидке зростання їх продуктивності. І в цей же час починає інтенсивно розвиватися індустрія комп'ютерних ігор. Одночасно комп'ютерна графіка починає широко використовуватися на телебаченні і в кіноіндустрії. Дж.Лукас створює відділення комп'ютерної графіки на Lucasfilm.

У 1977 р. з'являється новий журнал «Computer Graphics World".

В середині 1970-х років графіка продовжує розвиватися в бік все більшої реалістичності зображень. Е.Кетмул в 1974 р. створює перші алгоритми текстурування криволінійних поверхонь. У 1975 г.появляется згаданий раніше метод зафарбовування Фонга. У 1977 г.Дж.Блін пропонує алгоритми реалістичного зображення шорсткуватих поверхонь (мікрорельєфів); Ф.Кроу розробляє методи усунення східчастого ефекту при зображенні контурів (антіелайзінг) Дж.Брезенхем створює ефективні алгоритми побудови растрових образів відрізків, кіл і еліпсів .. Рівень розвитку обчислювальної техніки до цього часу вже дозволив використовувати "жадібні" алгоритми, що вимагають великих обсягів пам'яті, і в 1978 г.Кетмул пропонує метод Z-буфера, в якому використовується область пам'яті для зберігання інформації про "глибині" кожного пікселя екранного зображення. В цьому ж році Сайрус і Бек розвивають алгоритми кліпування (відсікання) ліній. А в 1979 р. Кей і Грінберг вперше реалізують зображення напівпрозорої поверхні.

У 1980 р. Т.Уіттед розробляє загальні принципи трасування променів, що включають відображення, заломлення, затінення та методи антіелайзінга. В1984 р. групою дослідників (Горел, Торренс, Грінберг та ін) була запропонована модель излучательности, одночасно розвиваються методи прямокутного кліпування областей.

У 1980-і роки з'являється цілий ряд компаній, що займаються прикладними розробками в галузі комп'ютерної графіки. У 1982 г.Дж.Кларк створює Silicon Graphics,

тоді ж виникає Ray Tracing Corporation, Adobe System, в 1986 р. компанія Pixar відгалужується отLucasfilm.

У ці роки комп'ютерна графіка вже міцно впроваджується в кіноіндустрію, розвиваються додатки до інженерних дисциплін. До речі, 1990-ті роки у зв'язку з виникненням мережі Інтернет у комп'ютерної графіки з'являється ще одна сфера застосування.

Тут перераховані далеко не всі серйозні кроки на шляху розвитку графіки, але більш детальне знайомство з її історією вимагає досить гарного подання про теорію і алгоритмах цієї дисципліни, тому ми обмежуємося лише коротким оглядом. Неважко помітити, що пріоритет у розвитку даного напрямку в інформаційних технологіях досить міцно утримують американські дослідники. Але і у вітчизняній науці теж були свої розробки, серед яких можна назвати ряд технічних реалізацій дисплеїв, виконаних в різні роки:

1968 року, ВЦ АН СРСР, машина БЕСМ-6, ймовірно, перший вітчизняний растровий дисплей з відеопам'яттю на магнітному барабані;

1972, Інститут автоматики і електрометрії (ІАіЕ), векторний дисплей "Символ";

1973 року, ІАіЕ, векторний дисплей "Дельта";

1977 р., ІАіЕ, векторний дисплей ЕПМ-400;

1982 р., Київ, НДІ периферійного устаткування, векторний дисплей СМ-7316, 4096 символів, дозвіл 2048 2048?;

1979-1984, Інститут прикладної фізики, серія растрових кольорових напівтонових дисплеїв "Гамма". Останні дисплеї даної серії мали таблицю кольоровості, підтримували вікна, плавне масштабування.

Таким чином, в процесі розвитку комп'ютерної графіки можна виділити кілька етапів.

У 1960-1970-і роки вона формувалася як наукова дисципліна. В цей час розроблялися основні методи та алгоритми: відсікання, растрова розгортка графічних примітивів, зафарбування візерунками, реалістичне зображення просторових сцен (видалення невидимих ліній і граней, трасування променів, що випромінюють поверхні), моделювання освітленості.

У 1980-і графіка розвивається більш як прикладна дисциплина.Разрабатываются методи її застосування в самих різних областях людської діяльності.

В 1990-е роки методи комп'ютерної графіки стають основним засобом організації діалогу "людина-комп'ютер" і залишаються такими по теперішній час.13.06.2012

Технічні засоби підтримки комп'ютерної графіки

Розвиток комп'ютерної графіки багато в чому обумовлено розвитком технічних засобів її підтримки. Насамперед це пристрої виведення, якими є дисплеї. В даний час існує декілька типів дисплеїв, що використовують електронно-променеву трубку, а також дисплеї на рідкокристалічних індикаторах і інші їх види. Нас цікавлять головним чином функціональні можливості дисплеїв, тому ми не будемо торкатися їх внутрішнього устрою і електронних схем.

Виникнення комп'ютерної графіки, як уже говорилося раніше, можна віднести до 50-х років. Дисплейне графіка на першому етапі свого розвитку використовувала електронно-променеві трубки (ЕПТ) з **довільним скануванням променя** для виведення у вигляді зображення інформації з ЕОМ. З експерименту в Массачусетському технологічному інституті почався етап розвитку векторних дисплеїв (дисплеїв з довільним скануванням променя).

Найпростішим з пристроїв на ЕЛТ є **дисплей із запам'ятовуючою трубкою** з прямим копіюванням зображення. Запам'ятовувюча трубка має властивість запам’ятовувати тривалий час після світіння: зображення залишається видимим протягом тривалого часу (до однієї години). При виведенні зображення інтенсивність електронного променя збільшують до рівня, при якому відбувається запам'ятовування

сліду променя на люмінофорі. Складність зображення практично не обмежена. Стирання відбувається шляхом подачі на всю трубку спеціальної напруги, при якій світіння зникає, і ця процедура займає приблизно 0,5 с. Тому зображення, отримані на екрані, не можна стерти частково, а стало бути, динамічні зображення або анімація на такому дисплеї неможливі. Дисплей з пам'ятаючою трубкою є векторним, або дисплеєм з довільним скануванням, тобто він дозволяє провести відрізок з однієї адресної точки в будь-яку іншу. Його досить легко програмувати, але рівень інтерактивності в нього нижче, ніж у ряду дисплеїв інших типів через низьку швидкості і погані характеристики стирання.

Наступний тип - це **векторні дисплеї з регенерацією зображення**.При переміщенні променя по екрану в точці, на яку потрапив промінь, збуджується світіння люмінофора по екрану. Це світіння досить швидко припиняється при переміщенні променя в іншу позицію (звичайний час післясвітіння - менше 0,1 с). Тому, для того щоб зображення було постійно видимим, доводиться його "перемальовувати" (регенерувати зображення) 50 або 25 разів на секунду. Необхідність регенерації зображення вимагає збереження його опису в спеціально виділеній пам'яті, званої пам'яттю регенерації. Сам опис зображення називається дисплейним файлом. Зрозуміло, що такий дисплей вимагає досить швидкого процесора для обробки дисплейного файлу та управління переміщенням променя по екрану.

Зазвичай серійні векторні дисплеї встигали 50 разів в секунду будувати тільки близько 3000-4000 відрізків. При більшій кількості відрізків зображення починає мерехтіти, так як відрізки, побудовані на початку чергового циклу, повністю гаснуть до того моменту, коли будуть будуватися останні.

Іншим недоліком векторних дисплеїв є мале число градацій по яскравості (зазвичай від двох до чотирьох). Були розроблені, але не знайшли широкого застосування двох-і триколірні ЕЛТ, також забезпечували кілька градацій яскравості.

У векторних дисплеях легко стерти будь-який елемент зображення, досить при черговому циклі побудови видалити стирається елемент з дисплейного файлу.

Текстовий діалог підтримується за допомогою алфавітно-цифрової клавіатури. Непрямий графічний діалог, як і у всіх інших дисплеях, здійснюється переміщенням перехрестя (курсору) по екрану за допомогою тих чи інших засобів управління перехрестям - координатних коліс, керуючого важеля (джойстика), трекбола (кульовий рукоятки), планшета і т.д. Відмінною рисою векторних дисплеїв є можливість безпосереднього графічного діалогу, яка полягає в простому вказівці за допомогою світлового пера об'єктів на екрані (ліній, символів і т.д.).

Векторні дисплеї звичайно підключаються до ЕОМ високошвидкісними каналами зв'язку. Перші серійні векторні дисплеї за кордоном з'явилися наприкінці 1960-х років.

Прогрес в технології мікроелектроніки привів до того, що з середини 1970-х років переважне поширення одержали дисплеї з растровим скануванням променя. Растрове пристрій можна розглядати як матрицю дискретних точок (пікселів), кожна з яких може бути підсвічена. Таким чином, воно є точечно-який малює пристроєм. Тому будь-який зображається на екрані дисплея відрізок будується за допомогою послідовності точок, апроксимуючих ідеальну траєкторію відрізка, подібно до того, як можна будувати зображення по клітках на картатому листку паперу. При цьому відрізок виходить прямим тільки у випадках, коли він горизонтальний, вертикальний або спрямований під кутом 45 градусів до горизонталі. Всі інші відрізки виглядають як послідовність "сходинок" (ступінчастий ефект).

При побудові зображення в растрових графічних пристроях використовується буфер кадру, що представляє собою великий безперервний ділянку пам'яті комп'ютера. Для кожної точки в растрі приділяється як мінімум один біт пам'яті. Буфер кадру сам по собі не є пристроєм виведення, він лише використовується для зберігання рісунка.Наіболее часто в якості пристрою виводу, що використовується з буфером кадру, виступає відеомонітор.

Щоб зрозуміти принципи роботи растрових дисплеїв, ми розглянемо в загальних рисах пристрій кольорової растрової електронно-променевої трубкі.Ізображеніе на екрані виходить за допомогою сфокусованого електронного променя, який, потрапляючи на екран, покритий люмінофором, дає яскраве колірне пляма. Луч в растровому дисплеї може відхилятися тільки в строго певні позиції на екрані, що утворюють своєрідну мозаїку. Люмінофорне покриття теж не безперервно, а являє собою безліч близько розташованих дрібних точок, куди може позиціонуватися промінь. Дисплей, що формує чорно-білі зображення, має одну електронну гармату, і її промінь висвітлює однотонні колірні плями. У кольоровій ЕПТ знаходяться три електронних гармати, по одній на кожен основний колір: червоний, зелений і сіній.Електронние гармати часто об'єднані в трикутний блок, відповідний трикутним блокам червоного, зеленого і синього люмінофорів на екране.Електронние промені від кожної з гармат, проходячи через спеціальну тіньову маску, потрапляють точно на пляму свого люмінофора. Зміна інтенсивності кожного з трьох променів дозволяє отримати не тільки три основних кольори, але і кольори, одержувані при їх змішенні в різних пропорціях, що дає дуже велику кількість квітів для кожного пікселя екрана.

Дисплеї на рідкокристалічних індикаторах працюють аналогічно індикаторам в електронному годиннику, але, звичайно, зображення складається не з декількох великих сегментів, а з великого числа окремо керованих точок. Ці дисплеї мають найменші габарити і енергоспоживання, тому широко використовуються в портативних комп'ютерах. Вони мають як переваги, так і недоліки в порівнянні з дисплеями на ЕЛТ. Хоча історично такий спосіб виводу зображення з'явився раніше, ніж растровий дисплей з ЕПТ, але швидко розвиватися він почав значно пізніше. Ці дисплеї також є растровими пристроями (їх теж можна представити як матрицю елементів - рідких кристалів).

Існують і інші види дисплеїв, наприклад плазмова панель, але ми не будемо їх торкатися, оскільки вони також є растровими, а технічна реалізація не є предметом нашого курсу. Важливо те, що розглянуті нами алгоритми розроблені для растрових графічних дисплеїв, а загальні принципи роботи цих пристроїв нам зрозумілі.

Крім дисплеїв, як пристрої виводу зображень використовуються плотери (графобудівники), призначені для виведення графічної інформації на папір. Ранні графічні пакети були орієнтовані саме на модель пір'яного плоттера, що формує зображення за допомогою пера. Перо може переміщатися уздовж двох напрямних, що відповідають двом координатним осям, причому воно може перебувати в двох станах - піднятому і опущеному. У піднятому стані воно просто переміщається над поверхнею паперу, а в опущеному залишає на папері лінії, що формують зображення. Таким чином, плоттер стоїть ближче до векторних дисплеїв, але відрізняється від них тим, що прати виводяться зображення неможливо. Тому для них зображення спочатку повністю формується в пам'яті комп'ютера, а потім виводиться.

Крім того, слід згадати принтери, що виводять зображення на папір або плівку. Зображення, одержуване з допомогою сучасних принтерів, також формується як точкове (растрове), але, як правило, з кращим дозволом, ніж екранне. Як і у випадку з графопостроителем, стерти зображення або його частина неможливо.

Тепер зробимо невеликий огляд пристроїв вводу інформації, що дозволяють вирішувати різні завдання комп'ютерної графіки, не вдаючись у деталі фізичних принципів їх роботи. Ці пристрої дозволяють організувати діалог "людина-комп'ютер", а особливості конструкції кожного пристрою дозволяють йому спеціалізуватися на виконанні певного кола завдань. Нас вони цікавлять саме як логічні пристрої, тобто з точки зору виконуваних ними функцій.

Першу групу пристроїв, за допомогою яких користувач може визначити розташування на екрані, назвемо **вказівними пристроями** (**pointing device**): миші, трекбол ((trackball), світловим пером (lightpen), джойстик (joystick), спейсбол (spaceball).

Майже всі пристрої цієї групи оснащені парою або кількома кнопками, які дозволяють сформувати і передавати в комп'ютер будь-якиі сигнали або переривання.

Рис. 1.1.Мишка

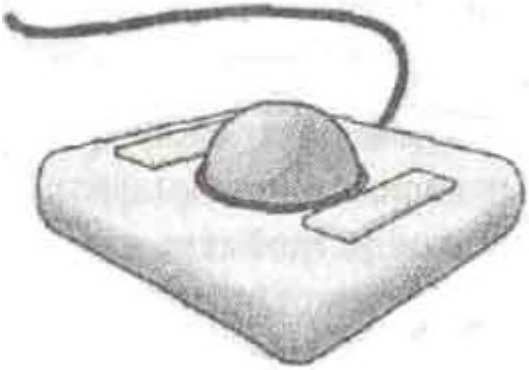

Рис. 1.2.Трекбол

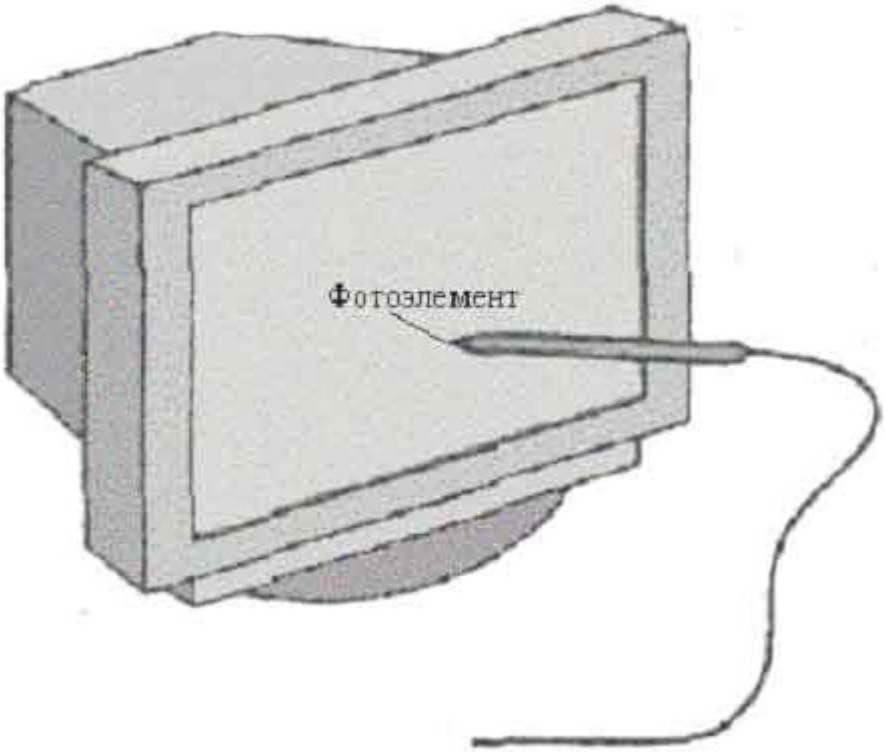

Рис. 1.3.Світлове перо

Рис. 1.4.Джойстик

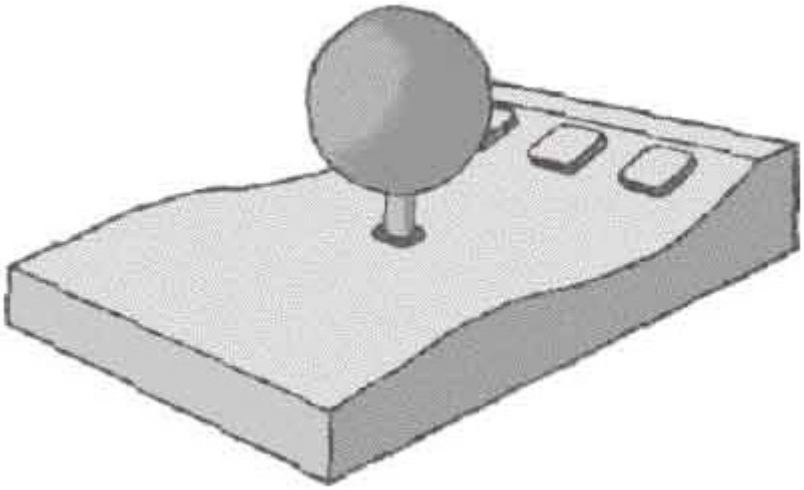

Рис. 1.5.Спейсбол

Рис. 1.6.Рукавичка для вводу даних

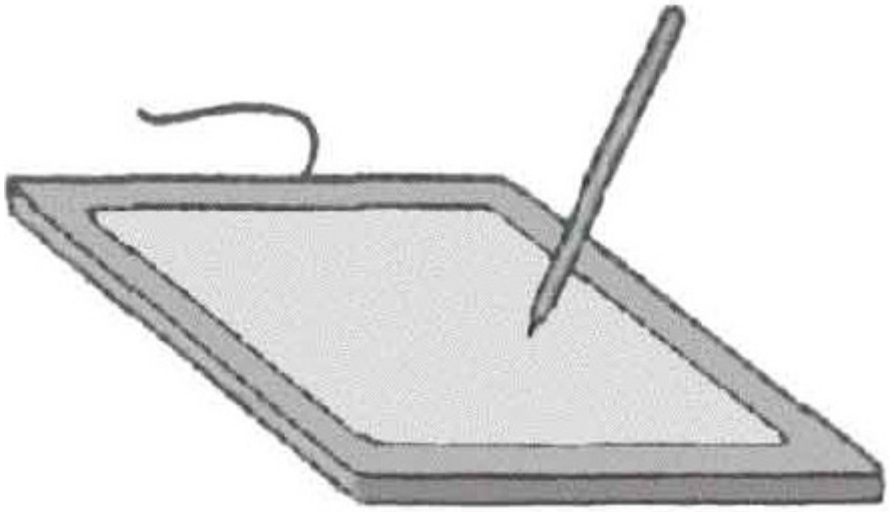

Рис. 1.7.Планшет

Питання та вправи

1. Назвовіть чотири основні області застосування комп'ютерної графіки

2.Які основні напрямки розвитку комп'ютерної графіки? Які задачі вони рішають ?

3. Де і коли був вперше використаний дисплей комп'ютера як пристрій виводу ЕОМ?

4.Коли і ким була розроблена перша інтерактивна програма для малювання?

5. Назвіть основних розробників методів зафарбовування гладких поверхонь.

6. Хто є автором ряду алгоритмів побудови растрових зображень різних геометричних об'єктів?

7. Назвіть ім'я авторів алгоритмів, видалення невидимих ліній.

8. В чому є основна відмінність між дисплеями з довільним скануванням і растровим скануванням?

9. В чому полягає відмінність між дисплеєм на запам’ятовуючій трубці від векторного дисплею з регенерацією зображення ? 10. Які основні принципи роботи кольорової растрової електронно-променевої трубки ?

11. Як працює плоттер з пера?

12. Назвіть основні пристрої введення, що використовуються в комп'ютері графіка.

13. Які пристрої вводу дають можливість працювати в абсолютних координатах?

14. Перечисліть області застосування сканерів..

Лекція 2. Колір в комп'ютерній графіці

Зміст

- Про природу світла і кольору
- Колірний графік МКО
- Кольорові моделі RGB і CMY
- Кольорові моделі HSV і HLS
- Простір CIE Лув
- Питання та вправи

Про природу світла і кольору

Світло як фізичне явище являє собою потік електромагнітних хвиль різної довжини і амплітуди. Око людини, будучи складною оптичною системою, сприймає ці хвилі в діапазоні довжин приблизно від 350 до 780 нм. Світло сприймається або безпосередньо від джерела, наприклад, від освітлювальних приладів, або як відбитий від поверхонь об'єктів або переломлений при проходженні крізь прозорі і напівпрозорі об'єкти. Колір - це характеристика сприйняття оком електромагнітних хвиль різної довжини, оскільки саме довжина хвилі визначає для ока видимий цвет.Амплітуда, що визначає енергію хвилі (пропорційну квадрату амплітуди), відповідає за яскравість кольору. Таким чином, саме поняття кольору є особливістю людського "бачення" навколишнього середовища.

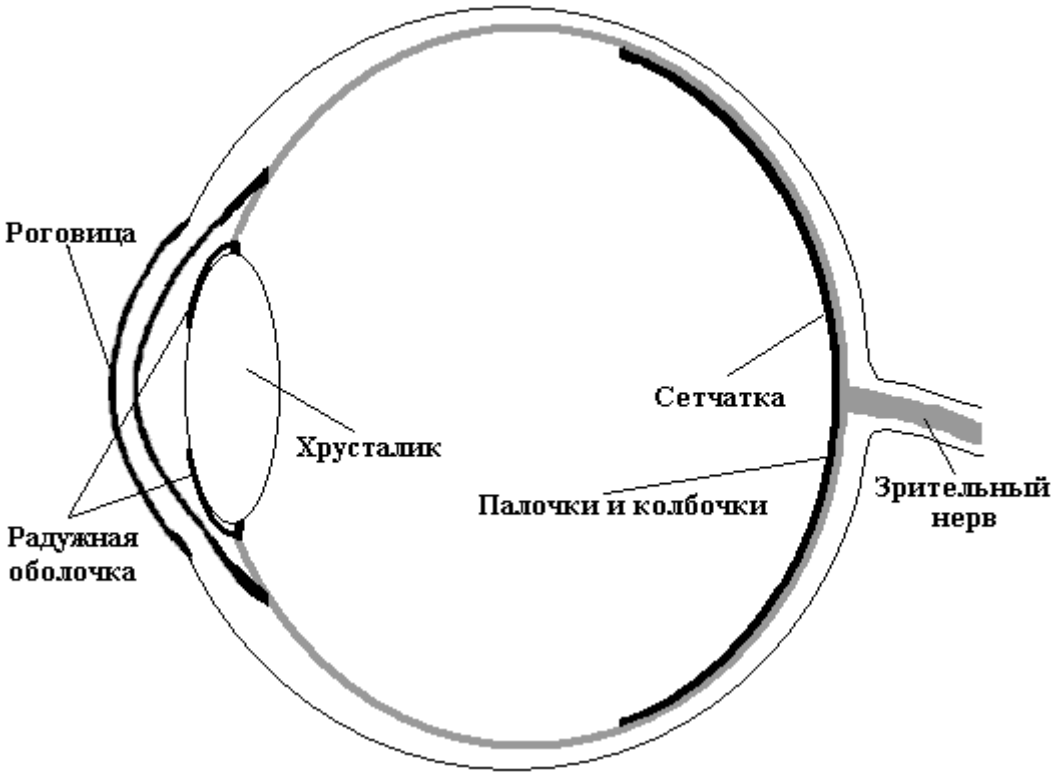

Рис. 2.1. Око людини

На рис. 2,1 схематично зображено око людини. Фоторецептори, розташовані на поверхні сітківки, відіграють роль приймачів света.Хрусталік - це своєрідна лінза, що формує зображення, а райдужна оболонка виконує роль діафрагми, регулюючи кількість світла, що пропускається всередину очі. Чутливі клітини ока неоднаково реагують на хвилі різної довжини. Інтенсивність світла є міра енергії світла, що впливає на око, а яскравість - це міра сприйняття оком цього впливу. Інтегральна крива спектральної чутливості ока наведена на **рис. 2.2**, це **стандартна крива Міжнародної комісії з освітлення (МКО, або CIE - Міжнародна комісія де l'Eclairage).**

Фоторецептори поділяються на два види: палички і колбочкі.Палочкі є високочутливими елементами і працюють в умовах слабкого освітлення. Вони нечутливі до довжини хвилі і тому не "розрізняють" кольору. Колбочки ж, навпаки, мають вузької спектральної кривої і "розрізняють" кольору. Паличок існує тільки один тип, а колбочки поділяються на три види, кожен з яких чутливий до певного діапазону довжин хвиль (довгі, середні або короткі). Чутливість їх також різна.

На рис. 2,3 представлені криві чутливості колбочек для всіх трьох видів. Видно, що найбільшою чутливістю володіють колбочки, що сприймають кольори зеленого спектра, небагато слабкіше - "червоні" колбочки й істотно слабкіше - "сині".

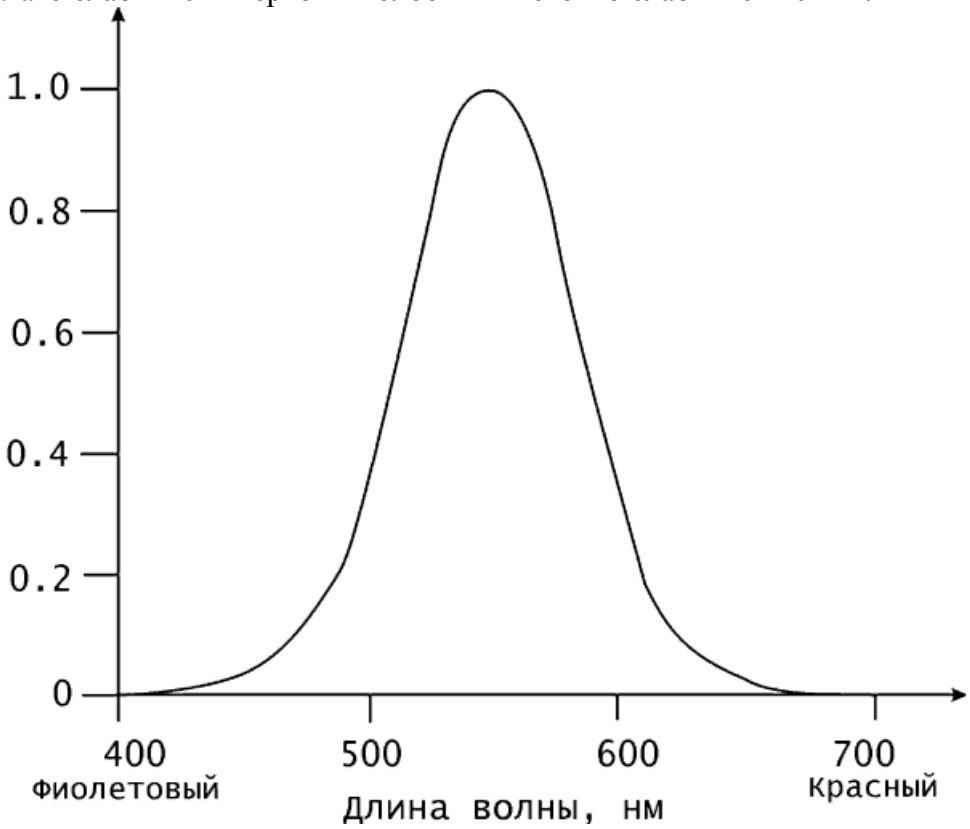

Рис. 2.2. Інтегральна крива спектральної чутливості ока

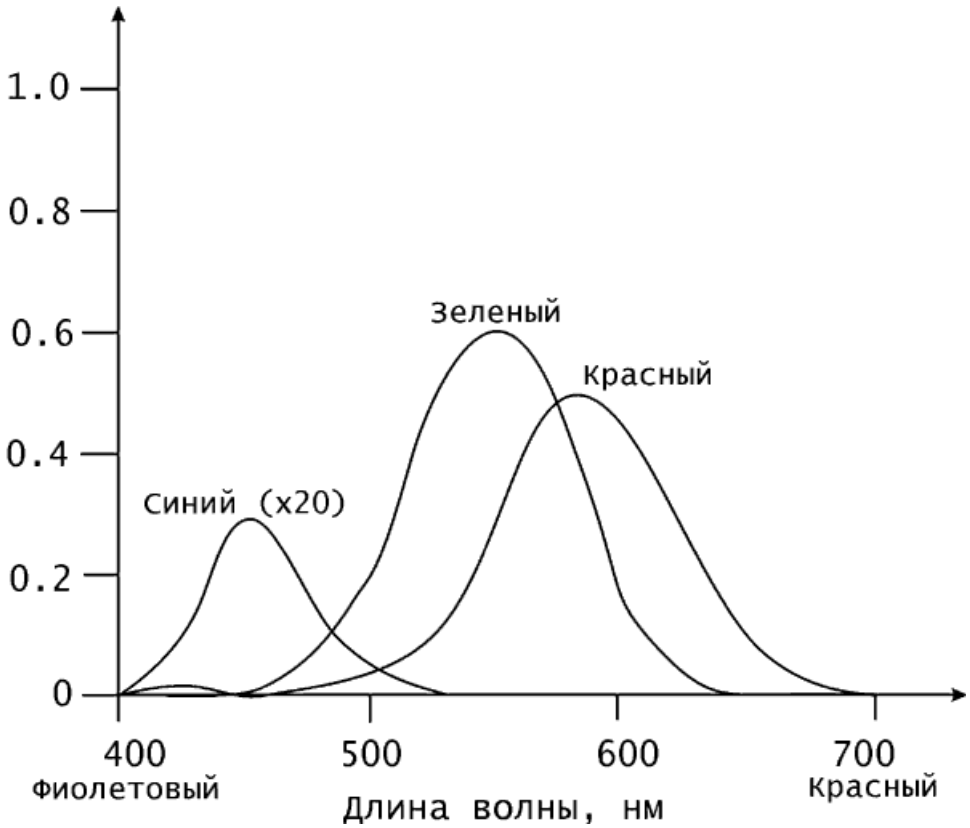

Рис. 2.3. Криві чутливості різних рецепторів

Таким чином, якщо функція $C(\lambda)$ характеризує спектральне розкладання світлового випромінювання від деякого джерела (рис. 2.4), т. е.распределеніе інтенсивності по довжинах хвиль, то три типи колбочок будуть посилати в мозок сигнали (червоний, зелений, синій), потужність яких визначається інтегральними співвідношеннями

$$R = \int C(\lambda) S_R(\lambda) d\lambda,$$

$$G = \int C(\lambda) S_G(\lambda) d\lambda,$$

$$B = \int C(\lambda) S_B(\lambda) d\lambda,$$

де - S_R, S_G, S_B функції чутливості відповідних типів колбочок.

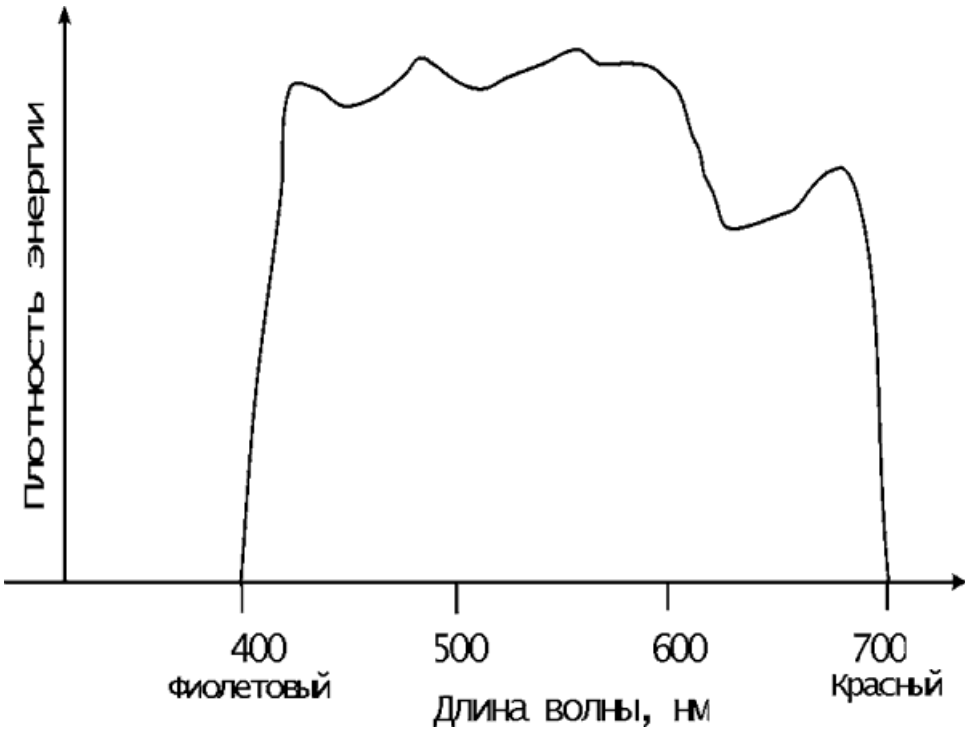

Рис. 2.4. Характерна спектральна крива

Якщо сприймається світло містить всі видимі довжини хвиль у приблизно рівних кількостях, то він називається ахроматичні і при максимальній інтенсивності сприймається як білий, а при більш низьких інтенсивностях - як відтінки сірого кольору. Інтенсивність відбитого світла зручно розглядати в діапазоні від 0 до 1, і тоді нульове значення буде відповідати чорному кольору. Якщо ж світло містить довжини хвиль у нерівних пропорціях, то він являетсяхроматіческім. Об'єкт, що відображає світло, сприймається як кольоровий, якщо він відображає або пропускає світло у вузькому діапазоні довжин волн.Точно так само і джерело світла сприймається як кольоровий, якщо він випускає хвилі у вузькому діапазоні довжин. При висвітленні кольорової поверхні кольоровим джерелом світла можуть виходити досить різноманітні колірні ефекти.

Колірний графік МКО

Тривимірна природа сприйняття кольору дозволяє відображати його в прямокутній системі координат. Будь-який колір можна зобразити у вигляді вектора, компонентами якого є відносні ваги червоного, зеленого і синього кольорів, обчислені за формулами

$$r = \frac{R}{R+G+B}, \quad g = \frac{G}{R+G+B}, \quad b = \frac{B}{R+G+B}.$$

Оскільки ці координати в сумі завжди становлять одиницю, а кожна з координат лежить в діапазоні від 0 до 1, то всі представлені таким чином точки простору будуть лежати в одній площині, причому тільки в трикутнику, відсікає від неї позитивним октантів системи координат (рис. 2,5 а). Ясно, що при такому поданні вся безліч точок цього трикутника можна описати за допомогою двох координат, так як третя виражається через них за допомогою співвідношення

$$b = 1 - r - g.$$

Таким чином, ми переходимо до двовимірному поданням області, тобто до проекції області на площину (рис. 2.5б).

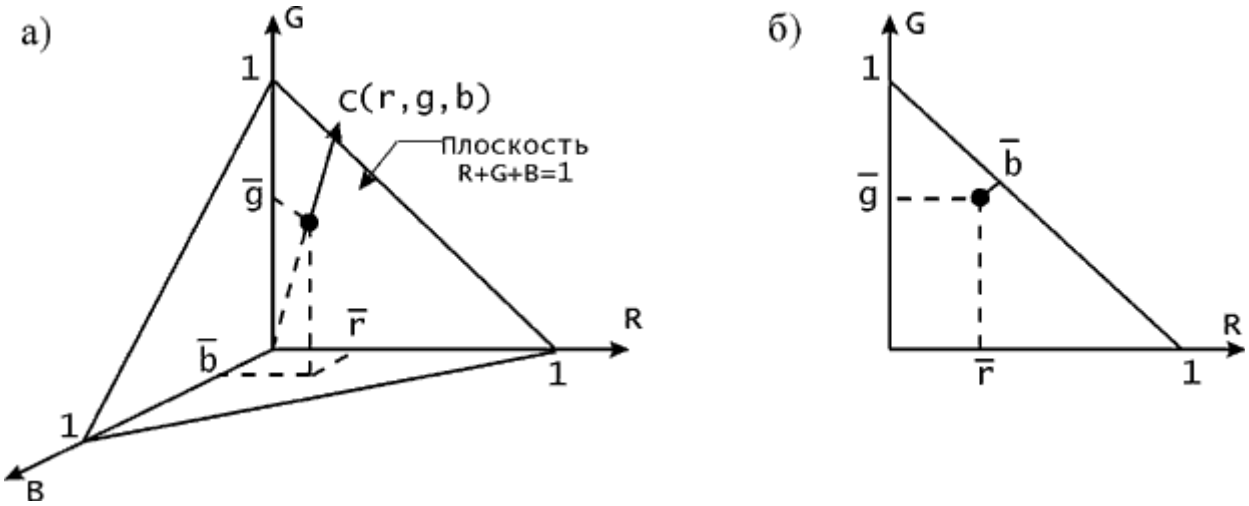

Рис. 2.5. Тривимірний коліровий простір

З використанням такого перетворення в 1931 р. були вироблені міжнародні стандарти визначення та вимірювання квітів. Основою стандарту став так званий двовимірний кольоровий графік МКО.Поскольку, як показали фізичні експерименти, складанням трьох основних кольорів можна одержати не всі можливі колірні відтінки, то в якості базисних були обрані інші параметри, отримані на основі дослідження стандартних реакцій очі на світ. Ці параметри - є чисто теоретичними, оскільки побудовані з використанням негативних значень основних складових кольору. Трикутник основних квітів був побудований так, щоб охоплювати весь спектр видимого світла. Крім того, рівна кількість усіх трьох гіпотетичних квітів у сумі дає білий колір. Координати кольоровості будуються так само, як і в наведеній вище формулі:

$$x = \frac{X}{X+Y+Z}, \quad y = \frac{Y}{X+Y+Z}, \quad z = \frac{Z}{X+Y+Z}, \quad x+y+z=1$$

При проекції цього трикутника на площину виходить коліровий графік МКО. Але координати кольоровості визначають тільки відносні кількості основних квітів, не задаючи яскравості результуючого цвета.Яскравість можна задати координатою Y, а X, Z визначити виходячи з величин (x, y, Y), за формулами

$$X = \frac{Y}{y}x, \quad Z = \frac{Y}{y}(1-x-y).$$

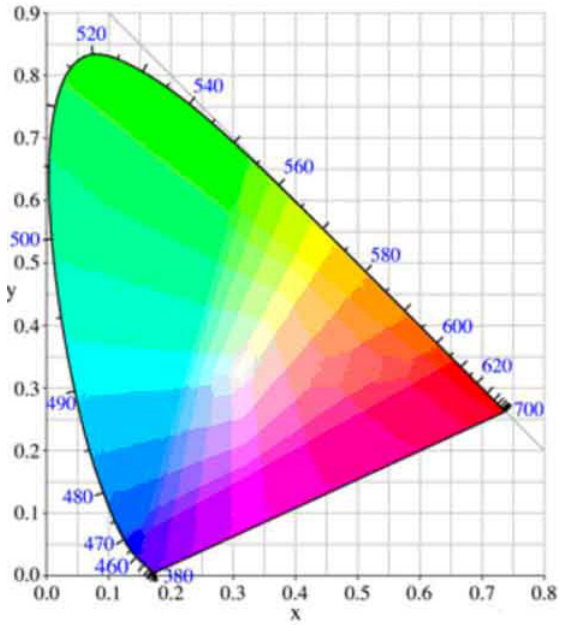

Рис. 2.6. Кольоровий графік МКО. На контурі зазначені довжини хвиль в нанометрах

Кольоровий графік МКО наведено на рис. 2.6. Область, обмежена кривою, охоплює весь видимий спектр, а сама крива називається лінією спектральних цветностей. Числа, проставлені на малюнку, означають довжину хвилі у відповідній точці. Точка C, відповідна полуденного висвітленню при суцільній хмарності, прийнята в якості опорного білого кольору.

Кольоровий графік зручний для цілого ряду завдань. Наприклад, з його допомогою можна отримати додатковий колір: для цього треба провести промінь від даного кольору через опорну точку до перетину з іншою стороною кривої (кольори є додатковими один до одного, якщо при додаванні їх у відповідній пропорції виходить білий колір). Для визначення домінуючої довжини хвилі якого кольору також проводиться промінь з опорної точки до перетину з даними кольором і триває до перетину з найближчою точкою лінії цветностей.

Для змішування двох кольорів використовуються закони Грассмана. Нехай два кольори задані на графіку МКО координатами $D_1 = (x_1, y_1, Y_1)$ і $D_2 = (x_2, y_2, Y_2)$. Тоді змішання їх дає колір $D_{12} = (x_1 + x_2, y_1 + y_2, z_1 + z_2)$. Якщо ввести позначення $t_1 = \frac{Y_1}{y_1}, t_2 = \frac{Y_2}{y_2}$, то отримаємо координати кольоровості суміші

$$x_{12} = \frac{x_1 t_1 + x_2 t_2}{t_1 + t_2}, \quad y_{12} = \frac{y_1 t_1 + y_2 t_2}{t_1 + t_2}, \quad Y_{12} = Y_1 + Y_2.$$

Координати МКО є точним стандартом визначення кольору. Але в різних областях, які мають справу з кольором, є свій підхід до його моделювання. Зокрема, може використовуватися інший набір основних кольорів. Комп'ютерна графіка спирається на систему RGB, тому представляє інтерес перехід між цими двома наборами квітів (іншими словами, перетворення координат кольоровості).

Кольорові моделі RGB і CMY

Колірні моделі, використовувані в комп'ютерній графіці, - це кошти опису квітів у певному діапазоні.

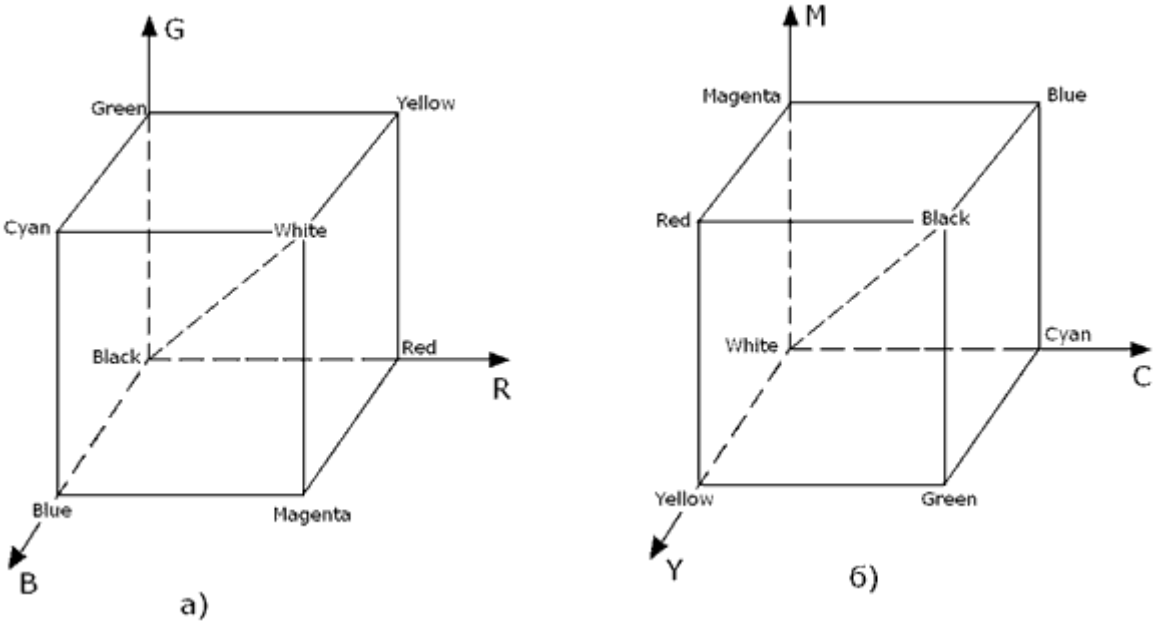

Рис. 2.7. Кольоровий куб для моделей RGB і CMY

На основі описаних вище фізичних уявлень в комп'ютерній графіці була прийнята так звана адитивна кольорова модель, яка використовує три первинних складових кольору. Ця модель припускає, що будь-який колір можна розглядати як зважену суму

трьох основних кольорів. Проілюструвати її можна на прикладі висвітлення сцени за допомогою трьох прожекторів різного кольору. Кожен прожектор управляється незалежно, і шляхом зміни потужності кожного з них можна відтворити практично всі кольори. У моделі RGB колір можна представити у вигляді вектора в тривимірній системі координат з початком відліку в точці (0,0,0). Максимальне значення кожної з компонент вектора приймемо за 1. Тоді вектор (1,1,1) відповідає білому цвету.Все колірні вектори, таким чином, укладені усередині одиничного куба, називаного колірним кубом (рис. 2.7а).

Інша модель змішання квітів - субстрактивной колірна модель, або модель CMY, що використовує в якості первинних складових кольору Cyan, Magenta, Yellow (блакитний, пурпурний, жовтий), які є додатковими до Red, Green, Blue. У цій моделі відтінки кольору виходять шляхом «віднімання» з падаючого світла хвиль певної довжини. Цей підхід потребує пояснення. В цій системі координат вектор (0,0,0) відповідає білому кольору, а вектор (1,1,1) - черному.Соответствующій колірної куб представлений на рис. 2.7б.

Зв'язок між значеннями (R, G, B) і (C, M, Y) для одного і того ж кольору виражається формулою

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} + \begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

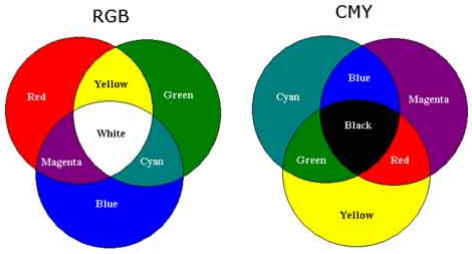

Рис. 2.8. Схема змішування кольорів для моделей RGB і CMY

Кольори однієї моделі є додатковими до квітів іншої (додатковий колір - це колір, результатом змішання якого з даним є білий). Схема змішування кольорів для двох моделей представлена на рис. 2.8. Приклад субстрактивной формування відтінків зображений на рис. 2.9. При висвітленні падаючим білим світлом у шарі блакитний (Cyan) фарби із спектру білого кольору поглинається (віднімається) червона частина як додатковий колір, потім з залишився світла в шарі пурпурної (Magenta) фарби поглинається зелена частина спектра, і, нарешті, від білої поверхні відбивається синій колір, який ми і бачимо. Таким чином, змішання блакитного і пурпурного кольорів дає в підсумку синій колір.

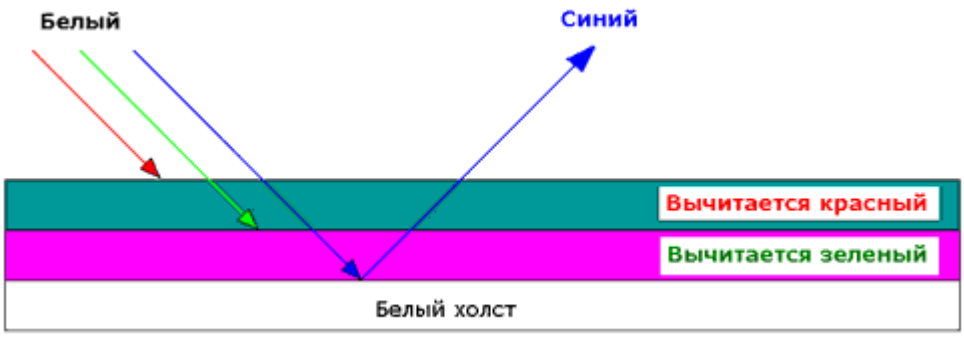

Рис. 2.9. Субстрактивне формування відтінків

Растрові дисплеї, як правило, використовують апаратно-орієнтовану модель квітів RGB. Існують також дисплеї з таблицею кольоровості, що представляє собою матрицю, кожний елемент якої-деякий колір (вектор RGB). В таких дисплеях значення кодів пікселів, що заносяться в відеопам'ять, являють собою індекси матриці цветності.Прі відображенні деякого пікселя на екран за значенням коду вибирається елемент таблиці кольоровості, що містить трійку значень R, G, B. Ця трійка і передається на монітор для завдання кольору пікселя на екрані.

В повнокольорових дисплеях для кожного пікселя в відеопам'ять заноситься трійка значень R, G, B. У цьому випадку для відображення пікселя з відеопам'яті безпосередньо вибираються значення R, G, B, які і передаються на монітор (але можуть і передаватися в таблицю кольоровості).

У моделях RGB і CMY легко задавати яскравості для одного з основних кольорів, але досить важко задати відтінок з необхідним колірним тоном і насиченістю, відповідним якомусь зразку кольору. У різного роду графічних редакторах це завдання найчастіше вирішується за допомогою інтерактивного вибору з палітри кольорів і формуванням квітів у палітрі шляхом підбору значень координат до одержання необхідного візуального результату. Іноді така палітра наочно відображає вибір вектора з колірного куба: спочатку за допомогою одного движка вибирається колірна площину, а потім на цій площині вибирається конкретна точка. Але й таким методом не відразу вдається досягти бажаного ефекту, оскільки не так просто вибрати правильну колірну площину.

Кольорові моделі HSV і HLS

Наведені моделі не охоплюють усього діапазону видимого кольору, оскільки їх колірний обхват - це лише трикутник на графіку МКО, вершинам якого відповідають базові кольори. Вони є апаратно орієнтованими, тобто відповідають технічної реалізації кольору в пристроях графічного виводу. Але психофізіологічне сприйняття світла визначається не інтенсивністю трьох первинних квітів, а колірним тоном, насиченістю і светлотой. Тон дозволяє розрізняти кольори, насиченість задає ступінь "розведення" чистого тону білим кольором, а светлота - це інтенсивність світла в цілому. Тому для адекватного нашому сприйняттю підбору відтінків більш зручними є моделі, в числі параметрів яких присутня тон (Hue). Цей параметр прийнято вимірювати кутом, відлічуваних навколо вертикальної осі. При цьому червоному кольору відповідає кут 0 °, зеленому - 120 °, синьому - 240 °, а доповнюють один одного кольору розташовані один навпроти іншого, тобто кут між ними складає 180 °. Кольори CMYрасположени посередині між складовими їх компонентами RGB.Существует дві моделі, що використовують цей параметр.

Модель HSV (Hue, Saturation, Value, або тон, насиченість, кількість світла) можна представити у вигляді світлової шестигранної піраміди (рис. 2.10), по осі якої відкладається значення V, а відстань від осі до бічної грані в горизонтальному перетині відповідає параметру S (за діапазон зміни цих величин приймається інтервал від нуля до одиниці). Значення S дорівнює одиниці, якщо точка лежить на бічній грані піраміди. Шестикутник, що лежить в основі піраміди, являє собою проекцію колірного куба в напрямку його головної діагоналі (рис. 2.11).

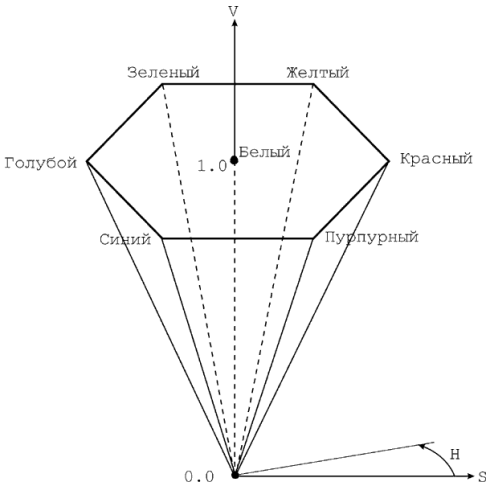

Рис. 2.10. Кольорова палітра HSV

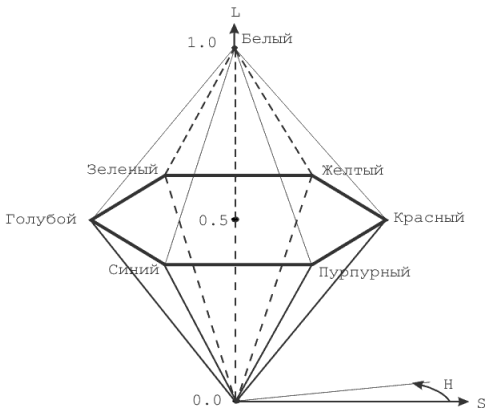

Рис. 2.11. Кольорова палітра HLS

Перетворення колірного простору HSV в RGB здійснюється безпосередньо за допомогою геометричних співвідношень між шестигранною пірамідою і кубом.

Колірна модель HLS (відтінок, яскравість, насиченість, або тон, светлота, насиченість) є розширенням моделі HSV. Тут колірний простір уже представляється у вигляді подвійної піраміди (рис. 2.11), в якій по вертикальній осі відкладається L (светлота), а інші два параметра задаються так само, як і в попередній моделі. У літературі ці піраміди іноді називають шестигранним конусом.

На рис. 2,12 і 2.13пріведени блок-схеми перетворення моделей HSVi HLS в модель RGB. Алгоритми зворотного перетворення пропонуються читачеві в якості вправи.

У першому алгоритмі використовується функція Ent, що означає цілу частину числа. Крім того, використовується операція присвоювання для векторов.Константа NDF (скорочене від вираження "не визначено") використовується при вході в алгоритм для того, щоб з'ясувати, задано чи значення змінної X. Наприклад, за угодою NDF може бути деяким негативним значенням, так як тон - це завжди позитивна величина. У другому алгоритмі застосовується допоміжна функціяValue (H, M1, M2) для обчислення значення компоненти R, G або B в залежності від ситуації.

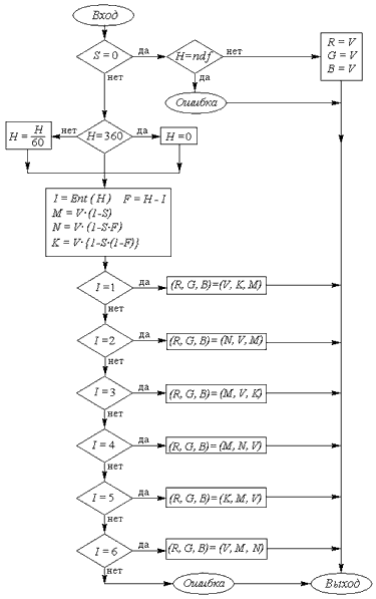

Рис. 2.12. Перетворення моделі HSV в RGB

Алгоритм перетворення:

Приведення H до заданого діапазону:

Поки H <0, H = H +360

Поки H> 360 H = H-360

Визначення координат

Якщо H <60 то Value = M1 + (M2-M1) * H/60

Якщо 60 <= H <180 то Value = M2

Якщо 180 <= H <240 то Value = M1 + (M2-M1) * (240-H) / 60

Якщо 240 <= H то Value = M1

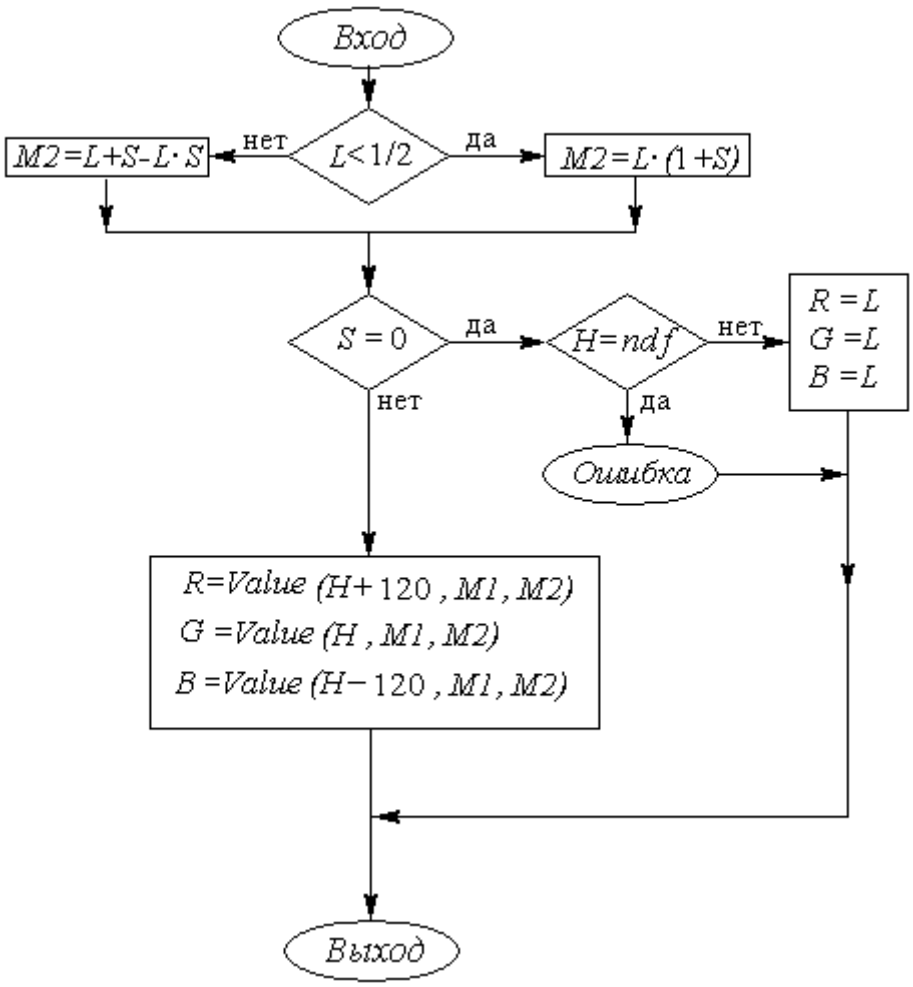

Рис. 2.13. Перетворення моделі HLS в RGB

Простір CIE Luv

Один з істотних мінусів колірного простору XYZ - це те, що воно не є перцептивно (візуально) рівномірним і не може використовуватися для обчислення колірних відстаней. Тому CIE (МКО) продовжила розробку перцептивно рівномірного простору. Метою комітету CIE було створення повторюваної системи стандартів передачі кольору для виробників фарб, чорнила, пігментів та інших барвників. Найважливіша функція цих стандартів - надати універсальну схему, в рамках якої можна було б встановлювати відповідність кольорів.

В результаті було створено колірний простір CIE Luv, що дозволяє визначити розрізнення кольорів для людини з «усередненими» зором, (тобто різні люди неоднаково сприймають різницю між квітами). Свою назву простір отримало завдяки його компонентам L, U і т. Параметр L відповідає яскравості кольору, uотвечает за перехід від зеленого до червоного (при збільшенні), а при збільшенні параметра V відбувається перехід від синього до фіолетовому.Еслі U і V рівні 0, то, змінюючи L, одержуємо кольору, що є градаціями сірого.

Цей колірний простір було розроблено для кількісного виміру відмінності двох кольорів. CIE були проведені дослідження за участю великої кількості людей, результатом чого стало створення простору Luv. Виміри проводилися в "хороших" умовах (достатнє освітлення і неяскравий монотонний фон); перед випробуваним перебували два аркуші паперу, пофарбованих відповідно двома кольорами, і він повинен був дати відповідь, наскільки, на його думку, розрізняються ці кольори. У випадку реального зображення, ми повинні знайти відмінності між кольорами для більш

складного фону, не завжди в умовах гарне освітлення (наприклад, занадто яскраво). Але освітлення залежить приміщень і час доби, і що кут нахилу поверхні, джерело світла, перехід від RGB Luv виглядає наступним чином. Спочатку нормуємо R, G, b:

$$\begin{pmatrix} R^* \\ G^* \\ B^* \end{pmatrix} = \begin{pmatrix} R/255 \\ G/255 \\ B/255 \end{pmatrix}$$

В подальшому виконуємо перетворення простору RGB в XYZ

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.412453 & 0.35758 & 0.180423 \\ 0.212671 & 0.71516 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{pmatrix} \times \begin{pmatrix} R^* \\ G^* \\ B^* \end{pmatrix}$$

Цветовое пространство CIE Luv — непрерывное однородное преобразование пространства CIE XYZ, описываемое следующими формулами:

Кольоровий простір CIE Luv- безперервне однорідне перетворення простору CIE XYZ, який описується слідуючими формулами:

$$L = \begin{cases} 116 \times \sqrt[3]{\frac{Y}{Y_n}} - 16, & \frac{Y}{Y_n} > 0.008856 \\ 903.3 \times \frac{Y}{Y_n}, & \frac{Y}{Y_n} \le 0.008856 \end{cases}$$

$$u' = \frac{4X}{X + 15Y + 3Z} = \frac{4x}{-2x + 12y + 3}, \quad v' = \frac{9Y}{X + 15Y + 3Z} = \frac{9y}{-2x + 12y + 3}$$

$$u = 13L(u' - u_n), \quad v = 13L(v' - v_n)$$

Щоб визначити параметри Y_n, u_n і v_n вводиться поняття білої точки (**white point**). Біла точка – це пара параметрів кольоровості (x, y), що визначає стандарт білого кольору для різних джерел світла. CIE склала таблицю білих точок до джерел світла різної яскравості. Значення компонентів білої точки в XYZ нормалізована до 100 (у вище приведених формулах Y_n просто відповідає нормалізованій Y компоненті). Параметри u_n і v_n оцінюються за тими ж формулами що u' і v' ,які використовують значення x та y для білої точки. Як вже згадувалося вище, компонента L відповідає яскравості кольору,а із формул видно , що L пропорційна кубічному кореню із компоненти Y простору XYZ. Однак, вважається, що сприйняття людини більше відповідає корінь другого степеня із освітленості. Так наприклад, в кольоровому просторі Lab параметр L обчислюється з використанням квадратного кореня.

Дещо про властивостеі L, u, v:

- L змінюється від 0 до 100;
- u,v лежить в межах -200, 200;
- u відповідає за перехід від зеленого до червоного (при збільшені u)
- v відповідає за перехід від синього до фіолетового (при збільшенні v);
- якщо u і v рівні 0, змінюючи L отримують зображення, що містить відтінки сірого кольору(grayscale).

Нарешті, і саме головне, те, до чого ми прагнули, переходячи в цей простір. У нас є два кольори- L_1, u_1, v_1 і L_2, u_2, v_2. Як визначити відстань між кольорами, Тобто, наскільки б людина замітила б різницю між ними? Виявляється, вона задається Евклідовою нормою

$$D = \sqrt{(L_1 - L_2)^2 + (u_1 - u_2)^2 + (v_1 - v_2)^2}$$

При відстані між двома кольорами $D > 5$ більшість людей вже помічають різницю, при $D > 10$ вона очевидна для всіх. В цьому і є головна перевага цього

простору. Вона враховує сприйняття кольорів людиною, а відмінності між кольорами визначаються дуже простою формулою. Слід зазначити, що ця формула застосовується за певних умов: освітлення, фон має не турбувати і відволікати. Паралельно з розвитком CIE Luv був також розроблений проникливо рівномірним кольором простору для CIE Lab. Із цих двох моделей, більш широко використовується модель системи CIE Lab. Структура кольорового простору Lab основана на на теорії, що колір не може бути одночасно як зеленим так і червоним або жовтим і блакитним (рис 2.14). Таким чином, для опису атрибутів "червоний/ зелений/ і жовтий/синій" можна використати однакові значення. Форомули переходу від простору XYZ до простору Lab в виконуються наступним чином:

$$L = \begin{cases} 116 \cdot [(Y/Y_n)^{1/3}] - 16 & \text{если } (Y/Y_n) > 0.008856 \quad a = 500 \cdot [f(X/X_n) - f(Y/Y_n)] \\ 903.3 \cdot Y/Y_n & \text{если } (Y/Y_n) \le 0.008856 \quad b = 200 \cdot [f(Y/Y_n) - f(Z/Z_n)] \end{cases}$$

$$\text{где } f(t) = \begin{cases} t^{1/3} & \text{если } (Y/Y_n) > 0.008856 \\ 7.787 \cdot t + 16/116 & \text{если } (Y/Y_n) \le 0.008856 \end{cases}$$

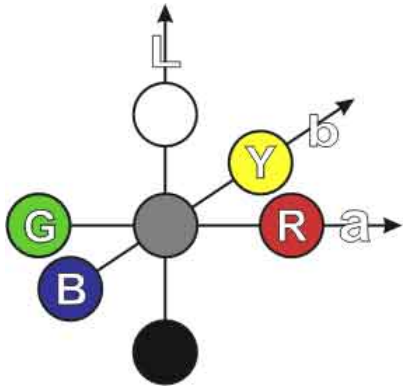

Рис. 2.14. Представлення кольору в просторі CIE Lab

Рис. 2.15. Видимий стандартним спостерігачем простір Lab.

Кожна кольорова модель, крім переваг, також має свої індивідуальні недоліки. Існують і інші моделі, які не розглядаються.

Питання і вправи.

1. Розташуйте за спаданням чутливість рецепторів ока до кольорів: червоний, зелений, синій.

2. Що таке хроматичний спектр?

3.Що таке ахроматичний спектр?

4. Як реалізується , проекція тривимірного кольорового простіру на площину?

5. У чому полягає різниця між кольоровим графіком МКО від трикутної проекційної області кольорового простору?

6. Що таке додатковий кольор?

7.Що таке аддитивна і субстрактивна кольорові моделі? Чим відрізняються їх кольорові куби?

8. Що лежить в основі кольорової моделі HSV і HLS?

9. Чи є кольорові моделі HSV і HLS аддитивними чи субстрактивними?

10.Побудуйте алгоритм перетворення моделі RGB в HSV.

11.Побудуйте алгоритм перетворення моделі RGB в HLS.

12.В чому полягає головна перевага кольорового простору Luv ?

13. В чому полягає головна перевага кольорового простору Lab?

Лекція 3: Геометричні перетворення

Зміст

Системи координат і вектори

Для подальшого викладу нам знадобляться деякі відомості з аналітичної геометрії та лінійної алгебри. Не ставлячи перед собою завдання докладного розгляду всіх цих питань, наведемо (або нагадаємо) ті основні поняття і операції, які використовуються в алгоритмах комп'ютерної графіки.

Дві взаємно перпендикулярні пересічні прямі із заданим масштабом утворюють декартову прямокутну систему координат на площині. Точка перетину O називається початком координат, прямі називаються осями координат. Одну з осей називають віссю OX, або віссю абсцис, іншу - віссю OY, або віссю ординат. Ці осі також називаютькоординатними осями.

Візьмемо довільну точку M на площині із заданою системою координат. Нехай M_x і M_y - проекції цієї точки на осі абсцис і ординат відповідно, причому довжина відрізка OM_x дорівнює x, а довжина OM_y дорівнює y. Тоді пара чисел (x, y) називається декартовими координатами точки M на площині (абсцисою і ординатою точки).

Три взаємно перпендикулярні пересічні прямі із заданим масштабом утворюють декартову прямокутну систему координат у просторі. Так само як і у випадку площини, точка перетину **O** називается початком координат, прямі називаються осями координат. Одну з осей називають віссю OX, або віссю абсцис, іншу - віссю OY, або віссю ординат, третю - віссю OZ, або віссю аплікат.

Нехай, M_x, M_y і M_z - проекції довільної точки M в просторі на осі абсцис, ординат і аплікат відповідно, причому довжина відрізка OM_x дорівнює x, довжина OM_y дорівнює y, а довжина OM_z дорівнює z. Тоді трійка чисел x, y, z називається декартовими координатами точки M в просторі (абсцисою, ординатою і аплікатой точки).

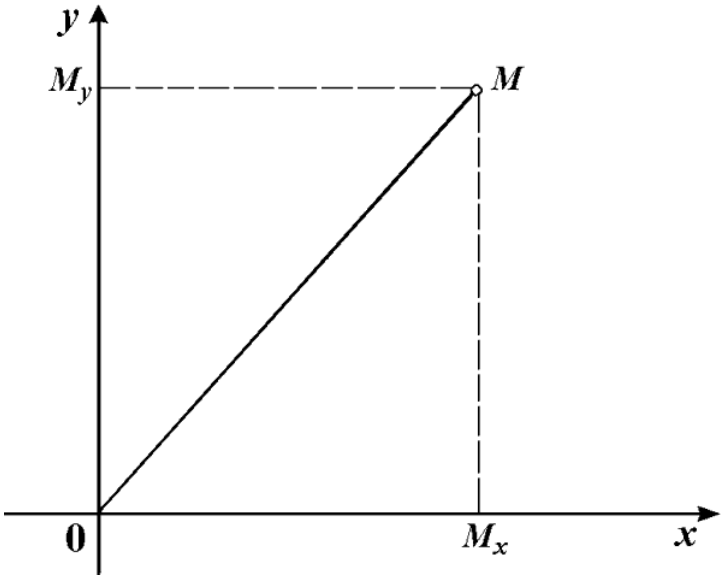

Рис. 3.1. Система координат на площині

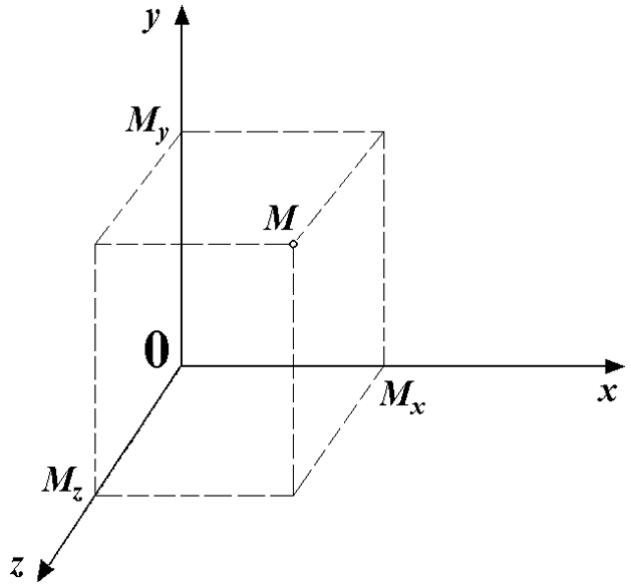

Рис. 3.2. Система координат в просторі

Нехай на площині задана декартова система координат. Візьмемо дві точки з координатами (x_1, y_1) та (x_2, y_2) відповідно. Тоді, використовуючи теорему Піфагора, можна отримати, що відстань між цими двома точками виражається формулою

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

Відстань між двома точками в просторі з координатами (x_1, y_1, z_1) (x_2, y_2, z_2) і виражається аналогічною формулою: $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}.$

Відрізок на площині і в просторі задається за допомогою двох точок, які вказують його межі. Геометричним вектором, або простовектором в просторі, будемо називати відрізок, у якого зазначено, яка з його граничних точок є початком, а яка - кінцем (т.е.указано напрямок вектора). Початок вектора називають точкою його застосування. Вектор називається нульовим, якщо його початок і кінець співпадають. Вектори називаються колінеарними, якщо вони лежать на паралельних прямих. Вектори вважаються рівними, якщо вони колінеарні, мають однакову довжину і однаковий напрямок. Таким чином, всі вектори, що виходять паралельним переносом з одного і

того ж вектора, рівні между собою. Будь-яка точка на площині і в просторі може розглядатися як вектор, початок якого збігається з початком координат (радіус-вектор), а кожен вектор, перенесений на початок координат, задає своїм кінцем єдину точку пространства.Поетому будь-який вектор може бути представлений сукупністю своїх координат в декартовій системі.

Лінійними операціями над векторами прийнято називати операції додавання векторів і операцію множення вектора на число.

Сумою двох векторів $\overrightarrow{a}$ і $\overrightarrow{b}$ називається вектор, що йде з початку вектора $\overrightarrow{a}$ в кінець вектора $\overrightarrow{b}$, за умови, що вектор $\overrightarrow{b}$ прикладений до кінця вектора $\overrightarrow{a}$.

Перелічимо основні властивості операції додавання векторів:

- $\overrightarrow{a} + \overrightarrow{b} = \overrightarrow{b} + \overrightarrow{a}$.
- $(\overrightarrow{a} + \overrightarrow{b}) + \overrightarrow{c} = \overrightarrow{a} + (\overrightarrow{b} + \overrightarrow{c})$.
- Існує нульовий векто $\overrightarrow{0}$ р, такий, що $\overrightarrow{a} + \overrightarrow{0} = \overrightarrow{a}$ для будь-якого вектора $\overrightarrow{a}$.
- Для кожного вектора $\overrightarrow{a}$ існує протилежний йому вектор $\overrightarrow{a}'$, такий, що $\overrightarrow{a} + \overrightarrow{a}' = \overrightarrow{0}$.

Різницею двох векторів $\overrightarrow{a}$ і $\overrightarrow{b}$ називається такий вектор $\overrightarrow{c}$, який в сумі з вектором $\overrightarrow{b}$ дає вектор $\overrightarrow{a}$.

Добутком $\alpha\overrightarrow{a}$ вектора $\overrightarrow{a}$ на число α називається вектор $\overrightarrow{b}$, колінеарний вектору $\overrightarrow{a}$, що має довжину $|\alpha| \cdot |\overrightarrow{a}|$ і напрямок, що збігається з напрямком вектора $\overrightarrow{a}$ при $\alpha > 0$ й протилежне напрямку $\overrightarrow{a}$ при $\alpha < 0$. Геометричний сенс множення вектора на число полягає в тому, що довжина вектора збільшується в $|\alpha|$ раз.

Операція множення вектора на число має такі властивості:

- $\alpha(\overrightarrow{a} + \overrightarrow{b}) = \alpha\overrightarrow{a} + \alpha\overrightarrow{b}$ (розподільна властивість числового співмножника щодо суми векторів);
- $(\alpha + \beta)\overrightarrow{a} = \alpha\overrightarrow{a} + \beta\overrightarrow{a}$ (розподільна властивість векторного співмножника щодо суми чисел);
- $(\alpha\beta)\overrightarrow{a} = \alpha(\beta\overrightarrow{a})$ (сполучна властивість числових співмножників);
- якщо вектор $\overrightarrow{b}$ колінеарний ненульовому вектору $\overrightarrow{a}$, то існує дійсне число β, таке, що $\overrightarrow{b} = \beta\overrightarrow{a}$.

Рівняння прямої і площини

Рівняння прямої на площині в декартовій системі координат можна задати рівнянням виду

$$y = kx + b$$

для випадку, коли пряма не паралельна осі OY, і рівнянням

$$x = c$$

для вертикальної прямої. Але пряма може бути також задана й іншим способом. Досить вказати вектор напрямку цій прямій $\overrightarrow{l} = (l_x, l_y)$ і якусь точку $\overrightarrow{r}_0 = (x_0, y_0)$, що лежить на цій прямій. При цьому точки, що лежать на прямій, можуть бути задані з використанням векторних операцій у вигляді так званого **параметричного рівняння прямої**

$$\overrightarrow{r} = \overrightarrow{r}_0 + t\overrightarrow{l},$$

в якому параметр т пробігає всі значення числової прямой.Координати точки, що відповідає деякому значенню цього параметра, визначаються співвідношеннями

$$x = x_0 + tl_x, \quad y = y_0 + tl_y. \text{(3.4)}$$

Пряму в просторі теж можна задавати параметричним рівнянням, яке дуже легко отримати з попереднього простим переходом від двовимірних векторів до тривимірних. Нехай $\overrightarrow{l} = (l_x, l_y, l_z)$, $\overrightarrow{r}_0 = (x_0, y_0, z_0)$. Тоді це рівняння буде визначати пряму в просторі, а координати точок цієї прямої будуть визначатися формулами

$$x = x_0 + tl_x, \quad y = y_0 + tl_y, \quad z = z_0 + tl_z, \quad -\infty < t < +\infty. \text{(3.5)}$$

Як відомо з елементарної геометрії, через будь-які три точки в просторі проходить площину. З іншого боку, через кожну точку площини можна провести єдину пряму, перпендикулярну даної площини. При цьому всі ці прямі будуть паралельні один одному, а значить, вони мають загальний вектор напряму. Цей вектор будемо називати **нормалью до площини**. Якщо довжина вектора дорівнює одиниці, ми будемо називати його **одиничною нормаллю**. У комп'ютерній графіці часто доводиться вирішувати завдання побудови нормалі до деякої площини, заданої трьома точками, а також завдання перетину прямої з площиною і двох площин.

Площина в просторі можна задати, вказавши вектор нормалі до неї і будь-яку точку, що належить даній площині. Нехай $\overrightarrow{n} = (n_1, n_2, n_3)$- вектор одиничної нормалі, а $\overrightarrow{r}_0 = (x_0, y_0, z_0)$- деяка точка на площині. Тоді для будь-якої точки $\overrightarrow{r} = (x, y, z)$, що лежить на площині, вектор $\overrightarrow{r} - \overrightarrow{r}_0$ буде ортогонален вектору нормалі, а отже, виконується рівність $((\overrightarrow{r} - \overrightarrow{r}_0) \cdot \overrightarrow{n}) = 0.$

Розкриваючи це вираження в координатному вигляді, одержуємо

$$n_1x + n_2y + n_3z - n_1x_0 - n_2y_0 - n_3z_0 = 0.$$

Тепер перепишемо це рівняння у вигляді

$$n_1x + n_2y + n_3z + d = 0, \text{(3.6)}$$

де $d = -n_1x_0 - n_2y_0 - n_3z_0$. Це рівняння називається канонічним рівнянням площини. При цьому абсолютно ясно, що якщо все це рівняння помножити на який-небудь відмінний від нуля множник, то воно буде описувати ту ж саму площину, тобто коефіцієнти n_1, n_2, n_3 для кожної площини задаються з точністю до довільного ненульового множника. Але якщо при цьому вектор $\overrightarrow{n}$ має одиничну довжину, то $|d|$ задає відстань від початку координат до даної площини.

В алгоритмах комп'ютерної графіки досить часто доводиться стикатися із завданням побудови площини, що проходить через три задані точки. Нехай три точки $\overrightarrow{r}_1, \overrightarrow{r}_2$ і $\overrightarrow{r}_3$, не лежать на одній прямій, мають координатами $(x_1, y_1, z_1), (x_2, y_2, z_2)$ і (x_3, y_3, z_3). Для канонічного рівняння необхідно побудувати нормаль до площини, що легко можна здійснити, використовуючи операцію векторного добутку. Оскільки вектори $\overrightarrow{v}_1 = \overrightarrow{r}_2 - \overrightarrow{r}_1$ і $\overrightarrow{v}_2 = \overrightarrow{r}_3 - \overrightarrow{r}_1$ лежать в шуканої площини, то вектор $\overrightarrow{N} = \overrightarrow{v}_1 \times \overrightarrow{v}_2$ буде ортогонален цій площині. Нехай $\overrightarrow{N} = (N_x, N_y, N_z)$, тоді рівняння площини буде мати вигляд

$$N_xx + N_yy + N_zz + D = 0.$$

Залишається визначити значення D. Так як точка $\overrightarrow{r}_1$ належить цій площині, то її координати повинні задовольняти отриманому рівнянню. Підставимо їх у рівняння і отримаємо

$$N_xx_1 + N_yy_1 + N_zz_1 + D = 0,$$

Отже

$$D = -N_x x_1 - N_y y_1 - N_z z_1,$$

і після підстановки остаточно одержимо:

$$N_x(x - x_1) + N_y(y - y_1) + N_z(z - z_1) = 0 \quad (3.7)$$

У більшості алгоритмів, що використовують площині, досить знати нормаль до неї і будь-яку точку, що належить площині. Очевидно, що за аналогією можна вивести канонічне рівняння прямої на площині, якщо задана нормаль до неї і належить прямий точка.

Аналітичне подання кривих і поверхонь

Нехай на площині задана декартова система координат.

Крива на площині - це геометричне місце точок (x, y), що задовольняють рівнянню

$$F(x, y) = 0 \quad (3.10)$$

де F - функція двох змінних. Ясно, що далеко не кожна функція буде задавати лінію. Так, наприклад, рівняння

$$x^2 + y^2 + 1 = 0$$

не задовольняє ні одна точка площині, а рівнянню

$$x^2 + y^2 = 0$$

задовольняє тільки одна точка $(0, 0)$.

Для аналітичного подання кривої в багатьох випадках зручніше задавати криву параметричними рівняннями, використовуючи допоміжну змінну (параметр) t :

$$x = \varphi(t), \quad y = \psi(t), t \in [a, b], \quad (3.11)$$

де φ і ψ - безперервні функції на заданому інтервалі зміни параметра. Якщо функція $\varphi(t)$ така, що можна виразити t через $x(t = \varphi^{-1}(x))$, то від параметричного подання кривої легко перейти до рівняння (3.10):

$$y = \psi(\varphi^{-1}(t)) = 0.$$

Систему рівнянь (3.11) можна записати у векторному вигляді:

$$\vec{r} = \vec{f}(t), \quad \vec{r} = (x, y), \quad \vec{f}(t) = (\varphi(t), \psi(t)).$$

Відрізок прямої являє собою окремий випадок кривої, причому параметричне подання його може мати вигляд

$$x = t, \quad y = at + b, \quad t \in [t_1, t_2]$$

або

$$x = at + b, y = t, t \in [t_1, t_2].$$

Коло радіуса r з центром в точці (x_0, y_0) може бути представлена параметричними рівняннями

$$x = x_0 + r \cdot \cos t, \quad y = y_0 + r \cdot \sin t, t \in [0, 2\pi].$$

Перейдемо до тривимірного простору з заданої декартовой системою координат. Поверхня в просторі - це геометричне місце точок (x, y, z), що задовольняють рівнянню виду

$$F(x, y, z) = 0. \quad (3.12)$$

Так само як і у випадку кривої на площині, не всяка функція F описує будь-яку поверхню. Наприклад, рівнянню

$$x^2 + y^2 + z^2 + 1 = 0$$

не задовольняє ні одна точка простору. Поверхня також може бути задана в параметричному вигляді, але на відміну від кривої для цього потрібні дві допоміжні змінні (параметри):

$$x = \varphi(u,v), \quad y = \psi(u,v), \quad z = \zeta(u,v), \quad u \in [a,b], \quad v \in [c,d]. \text{(3.13)}$$

Наприклад, сфера радіуса r з центром в точці (x_0, y_0, z_0) може бути задана рівнянням

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - r^2 = 0$$

або ж параметричними рівняннями

$$x = x_0 + r \cdot \cos u \cdot \cos v, \quad y = y_0 + r \cdot \sin u, \quad z = z_0 + r \cdot \cos u \cdot \sin v.$$

Криву в просторі можна описати як перетин двох поверхонь, тобто за допомогою системи рівнянь

$$F_1(x,y,z) = 0, \quad F_2(x,y,z) = 0 \text{ (3.14)}$$

або параметричними рівняннями виду

$$x = \varphi(t), \quad y = \psi(t), z = \zeta(t), \quad t \in [a,b]. \text{(3.15)}$$

Перетин променя з площиною і сферою

Пряма на площині і в просторі є нескінченною в обидва боки. **Променем** називається полупрямая, тобто множина всіх точок прямої, що лежать по одну сторону від заданої її точки, званої початком променя. Луч будемо задавати в параметричному вигляді, як це було описано в одному з попередніх розділів. Нехай $\overrightarrow{l} = (l_x, l_y, l_z)$ - напрямний вектор прямої, а $\overrightarrow{r}_0 = (x_0, y_0, z_0)$ - початкова точка. Тоді координати точок променя будуть визначатися формулами

$$x = x_0 + tl_x, \quad y = y_0 + tl_y, \quad z = z_0 + tl_z. \text{ (3.8)}$$

Будемо вважати, що спрямовуючий вектор одиничний, тобто $l_x^2 + l_y^2 + l_z^2 = 1$.

Спочатку розглянемо задачу про знаходження точки перетину променя з площиною, заданої канонічними рівнянням

$$n_1 x + n_2 y + n_3 z + d = 0. \text{ (3.9)}$$

Вектор нормалі $\overrightarrow{n} = (n_1, n_2, n_3)$ теж будемо вважати одиничним. Спочатку треба визначити значення параметра **t**, при якому промінь перетинає плоскость.Для цього підставимо координати з формули (3.8) в рівняння (3.9) і отримаємо

$$n_1(x_0 + tl_x) + n_2(y_0 + tl_y) + n_3(z_0 + tl_z) + d = 0,$$

звідки легко визначити, що промінь перетинає площину в точці із значенням

$$t_0 = -\frac{(\overrightarrow{r}_0 \cdot \overrightarrow{n}) + d}{(\overrightarrow{l} \cdot \overrightarrow{n})}.$$

Очевидно, що така точка існує тільки за умови $(\overrightarrow{l} \cdot \overrightarrow{n}) \neq 0$. У свою чергу, ця величина звертається в нуль тільки у випадку, коли вектори $\overrightarrow{l}$ і $\overrightarrow{n}$ ортогональні один одному.

Нехай тепер нам задана сфера з центром у точці $\overrightarrow{r}_c = (x_c, y_c, z_c)$ і радіусом d . Тоді рівняння сфери буде мати вигляд $(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 = d^2.$

Підставивши сюди координати променя з рівняння (3.9), отримаємо, що параметр, при якому промінь перетинає сферу, повинен задовольняти квадратному рівнянню

$$at_0^2 + bt_0 + c = 0,$$

де $a = |\vec{r}_c|^2,\ b = 2((\vec{r}_0 - \vec{r}_c)\cdot\vec{l}),\ c = |\vec{r}_0 - \vec{r}_c|^2 - d^2$. Визначимо коріння цього рівняння. Якщо дискримінант $D = \frac{b^2}{4} - c \geq 0$, то коріння існують. Їх може бути або два $(D > 0)$, або один $(D = 0)$. У першому випадку маємо дві точки перетину, у другому - одну (промінь стосується сфери). Відповідні значення параметра визначаються співвідношенням $t_{1,2} = -\frac{b}{2} \mp \sqrt{D}$

Інтерполяція функцій однієї та двох змінних

Крім функцій, заданих аналітично (тобто за допомогою елементарних функцій, значення яких легко можуть бути обчислені в будь-якій точці області визначення), на практиці часто доводиться мати справу з таблично заданими функціями. В цьому випадку функція задається своїми значеннями на деякому дискретній множині точок (**вузлів)** з області визначення. Якщо необхідно отримати значення функції в будь-якій точці, не збігається з вузлом, використовують різні методи наближеного обчислення, які грунтуються на деяких апріорних припущеннях щодо цієї функції. При цьому сама процедура обчислення називається **інтерполяцією** у випадку, коли точка належить заданій області, і **екстраполяцією**, якщо вона лежить поза областю.

Як припущень про характер дискретно заданої функції найбільш часто використовуваною і простий є те, що вона кусочно-лінійна, т. е. що в проміжках між вузлами вона веде себе відповідно до лінійним законом. Тоді інтерполяція називается **лінійною**, і цей метод ми будемо досить часто застосовувати в алгоритмах комп'ютерної графіки.

Нехай на площині задана система координат XOY і відрізок $[x_1, x_2]$ на осі OX, на кінцях якого задані значення y_1, y_2 деякої **лінійної** функції (рис. 3.3). Тоді для будь-якої точки x всередині заданого відрізка відповідне значення y, яке обчислюється за формулами

$$y = ty_1 + (1-t)y_2, \qquad t = (x_2 - x)/(x_2 - x_1).$$

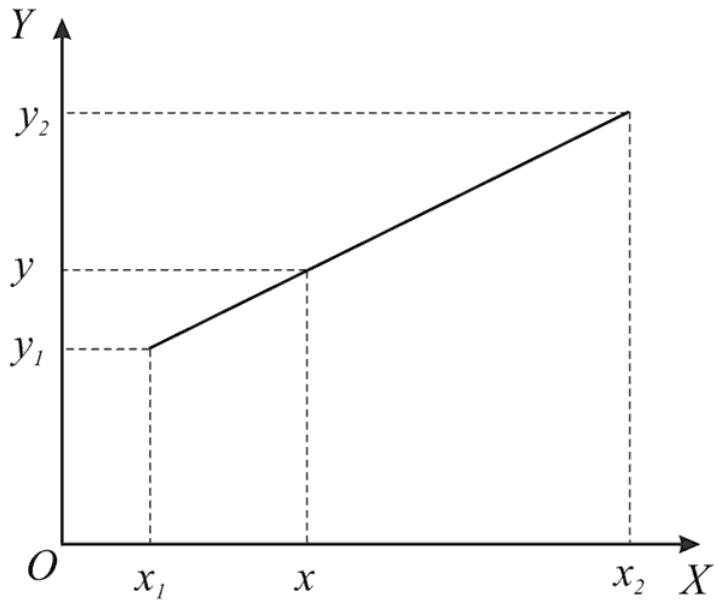

Рис. 3.3. Лінійна інтерполяція функції однієї змінної

Звернемося тепер до задачі інтерполяції функцій двох змінних. У цьому випадку найбільш простої також є інтерполяція за трьома заданим точкам знову ж за допомогою кусочно-лінійної функції. Нехай на площині заданий трикутник з вершинами $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ і задані значення функції в цих точках z_1, z_2, z_3. Тоді три точки (x_i, y_i, z_i) визначають у просторі трикутник, який є плоскою фігурою. Передбачається, що площа трикутника більше нуля, або, як кажуть, **трикутник невироджений**. Для визначення значення функції в довільній точці (x, y), що лежить всередині трикутника, скористаємося так званими **барицентричними координатами** (α, β, γ) цієї точки. Геометричний зміст цих координат полягає в тому, що вони рівні відношенню площ трикутників, зображених на рис. 3.4:

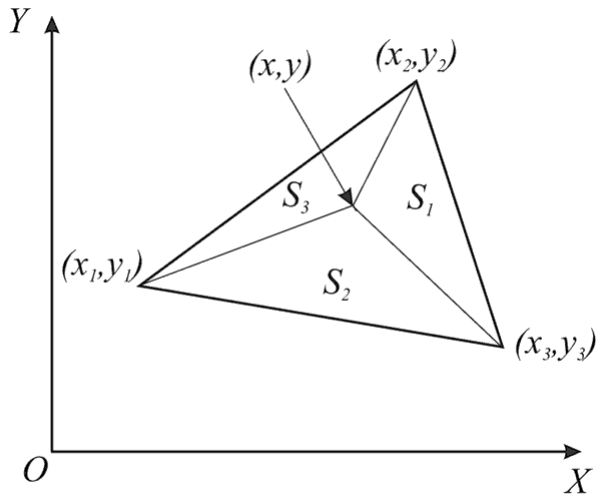

Рис. 3.4. Лінійна інтерполяція функції двох змінних

$$\alpha = \frac{S_1}{S},\ \beta = \frac{S_2}{S},\ \gamma = \frac{S_3}{S},\ S = S_1 + S_2 + S_3.$$

Ці числа ненегативні і задовольняють наступним співвідношенням:

$$\left.\begin{aligned} &\alpha + \beta + \gamma = 1 \\ &\alpha x_1 + \beta x_2 + \gamma x_3 = x \\ &\alpha y_1 + \beta y_2 + \gamma y_3 = y \end{aligned}\right\}.$$

Ці співвідношення будемо розглядати як рівняння для знаходження чисел (α, β, γ).

Визначник цієї системи рівнянь є

$$\Delta = \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{vmatrix} = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1),$$

і він по модулю дорівнює подвоєної площі трикутника, тому $\Delta \neq 0$, отже, система має єдине рішення за будь-якої правої частини. Скористаємося формулами Крамера і випишемо вид цього рішення:

$$\alpha = \frac{\Delta_1}{\Delta}, \quad \beta = \frac{\Delta_2}{\Delta}, \quad \gamma = 1 - \alpha - \beta,$$

Де

$$\Delta_1 = \begin{vmatrix} 1 & 1 & 1 \\ x & x_2 & x_3 \\ y & y_2 & y_3 \end{vmatrix} = (x_2 y_3 - x_3 y_2) + x(y_2 - y_3) + y(x_3 - x_2),$$

$$\Delta_2 = \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x & x_3 \\ y_1 & y & y_3 \end{vmatrix} = (x_3 y_1 - x_1 y_3) + x(y_3 - y_1) + y(x_1 - x_3).$$

Після того як отримані баріцентріческіе координати точки (x, y), значення функції в ній розраховується за формулою

$z = \alpha z_1 + \beta z_2 + \gamma z_3.$

Існують добре розроблені методи гладкої інтерполяції функцій. Особливо часто при інтерполяції кривих і поверхонь використовуються **сплайн-функції**, які гладко "склеюються" з **поліномів**. Серед них слід виділити **кубічні сплайни**, які будуються з поліномів третього ступеня. Вони широко використовуються в інженерній геометрії завдяки простоті їх обчислення та іншим корисним властивостям. Ми їх розглянемо докладніше в наступних розділах.

Матриці

Для виконання перетворень векторів в просторі ми будемо використовувати матричний метод. **Матрицею** розмірності $n \times n$ називається таблиця чисел виду

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}.$$

Надалі будемо використовувати скорочений запис для матриці: $A = (a_{ij})$. Рядки матриці $A_i = (a_{i1}, a_{i2}, \dots, a_i n)$ будемо називати **вектор-рядками** (оскільки кожна з них визначає вектор), а стовпці A^j - **вектор-стовпцями** . Матриці є ефективним інструментом для виконання перетворень на площині і в просторі. У цих випадках застосовуються матриці розмірності 2×2 і 3×3.

Спочатку введемо ряд операцій над матрицями і векторами.

Нехай задані матриці $A = (a_{ij})$ і $B = (b_{ij})$. **Сумою матриць** називається матриця $C = (c_{ij})$, елементами якої є $c_{ij} = a_{ij} + b_{ij}$.

Визначимо також операцію **множення матриці на число**. Результатом множення матриці $A = (a_{ij})$ на число α є матриця $B = (b_{ij})$, елементи якої $b_{ij} = \alpha a_{ij}$.

Добутком двох матриць $A = (a_{ij})$ і $B = (b_{ij})$ називається матриця $C = (c_{ij})$, елементи якої визначаються таким чином:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}.$$

Добуток матриць некомутативний, тобто в загальному випадку $A \cdot B \neq B \cdot A$.

Попередні визначення ми вводили для матриць довільної розмірності. Наступні операції будуть пов'язані з векторами, і ми будемо мати на увазі, що $n = 2$ або $n = 3$. Нехай задана матриця $A = (a_{ij})$ і вектор $\overrightarrow{r} = (x_1, \dots, x_n)$. Результатом **множення матриці на вектор** є вектор $\overrightarrow{r}_0 = (x_1^0, \dots, x_n^0)$, координати якого обчислюються як скалярний добуток рядка матриці на вектор.:

$$x_i^0 = \sum_{k=1}^{n} a_{ik} x_k \equiv (A_i \cdot \overrightarrow{r}).$$

Якщо матриця $B = (b_{ij})$ отримана з матриці $A = (a_{ij})$ шляхом заміни всіх вектор-рядків на вектор-стовпці, тобто $b_{ij} = a_{ji},\ i,\ j = 1, \ldots, n$, то її називають **транспонованою** матрицею A і позначають A^T.

Аналогічним чином визначається **множення вектора на матрицю**, тільки в цьому випадку вектор скалярно множиться на вектор-рядки матриці. Матриця виду

$$E = \begin{pmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 \end{pmatrix}$$

називається **одиничною** і має такі властивості:

• $A \cdot E = E \cdot A = A$ для будь матриці A.

• $\overrightarrow{r} = E \cdot \overrightarrow{r}$ для будь-якого вектора $\overrightarrow{r}$.

• Якщо для матриці A існує матриця B, така, що $A \cdot B = E$, то B називається зворотною матрицею до A і позначається A^{-1}. При цьому $A \cdot A^{-1} = A^{-1} \cdot A = E$, і для будь-якого вектора $\overrightarrow{r}$ одержуємо співвідношення: якщо $\overrightarrow{r}' = A \cdot \overrightarrow{r}$, то $\overrightarrow{r} = A^{-1} \cdot \overrightarrow{r}'$.

• Якщо для матриць A і B існують зворотні матриці, то існує і зворотна матриця для їх добутку $(a \cdot B)^{-1} = B^{-1} \cdot A^{-1}$.

Завдяки операції множення матриці на вектор будь матриця визначає перетворення в просторі, за яким кожному вектору зіставляється деякий інший по цілком певним законом.

Зазначимо, що для геометричних перетворень зручно використовувати матриці розмірністю на одиницю більше, ніж розмірність простору, але про це докладніше йтиметься в наступному розділі.

Геометричні перетворення (перенос, масштабування, обертання)

Геометричні об'єкти на площині і в просторі можна піддавати ряду різних перетворень. Найбільш вживаними в задачах комп'ютерної графіки є:

• переміщення (паралельний перенос);

• зміна розмірів (масштабування);

• повороти навколо деякої точки на площині або деякої осі в просторі (обертання).

Надалі ми часто будемо ототожнювати точки простору з радіус-вектором, визначеним цією точкою.

Спочатку розглянемо перетворення на площині, або **двовимірні перетворення**.

Паралельний перенос об'єкта зводиться до переміщення всіх його точок на одну і ту ж відстань d в одному і тому ж напрямку, заданому певним вектором $\overrightarrow{v}$. Якщо цей вектор має довжину d, то операція переносу може бути реалізована шляхом складання всіх точок об'єкта з вектором $\overrightarrow{v}$. Досить просто довести, що при такій операції зберігаються відстані між точками і, як наслідок, кути між відрізками. Зрозуміло також, що відрізки прямих перейдуть у відрізки прямих. Тому при перенесенні багатокутника немає необхідності піддавати цієї операції нескінченна безліч крапок, досить просто перенести вершини, а потім з'єднати їх відрізками.

Масштабування об'єкта можна реалізувати шляхом множення координат всіх його точок на деяке число. Нехай є точки з координатами (x_1, y_1) та (x_2, y_2), над якими

виконується таке перетворення. Результатом будуть нові точки з координатами $(S_x x_1, S_y y_1)$ і $(S_x x_2, S_y y_2)$. Якщо $S_x = S_y = S_0$, то нескладно довести, що обидві точки перемістяться уздовж прямих, що проходять через саму точку і початок координат, тобто в напрямку свого ж радіус-вектора (рис. 3.5). При цьому відстань між новими точками буде в S_0 раз відрізнятися від колишнього, але кути між відрізками збережуться (це можна показати, якщо виразити косинус кута через скалярний добуток векторів). Ясно, що якщо коефіцієнт масштабування S_0 більше одиниці, відповідний відрізок розтягується, а якщо менше, то стискається. Крім того, при такому перетворенні об'єкт зміщується.

У випадку, коли $S_x \neq S_y$, відстані між точками зміняться нерівномірно, оскільки розтягування в горизонтальному і вертикальному напрямках будуть різними. Кути між відрізками також не збережуться (рис. 3.6).

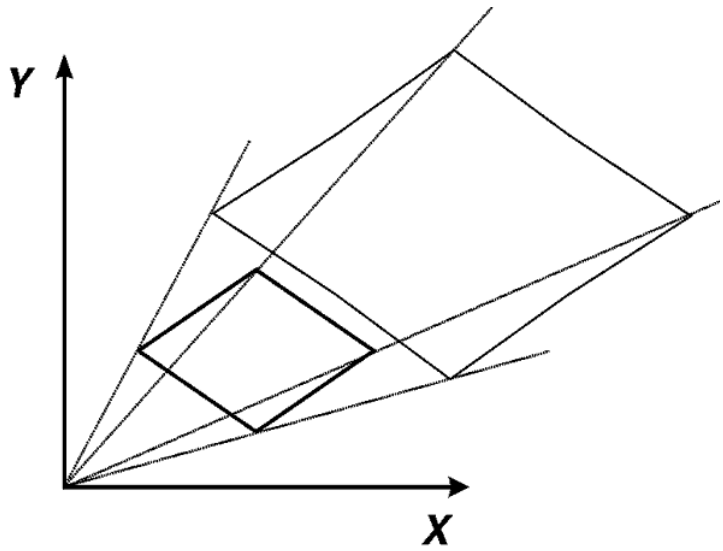

Рис. 3.5. Масштабування зі збереженням кутів

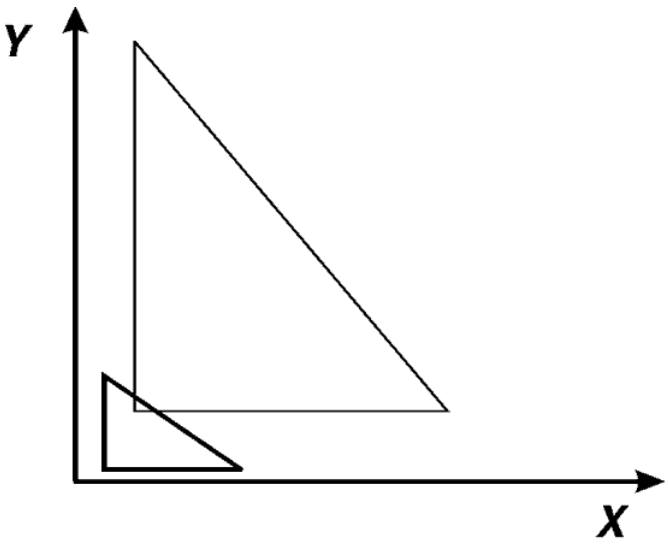

Рис. 3.6. Масштабування з перекручуванням кутів

Обертання в площині переміщають точки по дузі кола, центр якої знаходиться в початку координат. Розглянемо спочатку рух однієї точки при повороті на кут α (позитивним є напрямок проти годинникової стрілки), т. е. поворот радіус-вектора на кут

(рис. 3.7). Нехай точка розташовувалася на відстані r від початку координат, а її радіус-вектор становив кут β з віссю абсцис. Тоді координати точки визначаються формулами

$$x = r\cos\beta, \quad y = r\sin\beta.$$

Після повороту вектор буде становити кут $\beta + \alpha$, а нові координати точки будуть визначатися співвідношеннями

$$x' = r\cos(\beta+\alpha) = r\cos\beta\cos\alpha - r\sin\beta\sin\alpha = x\cos\alpha - y\sin\alpha \quad (3.7)$$

$$y' = r\sin(\beta+\alpha) = r\sin\beta\cos\alpha + r\cos\beta\sin\alpha = x\sin\alpha + y\cos\alpha$$

Можна показати, що при такому перетворенні зберігаються відстані між точками, а отже, і кути між відрізками.

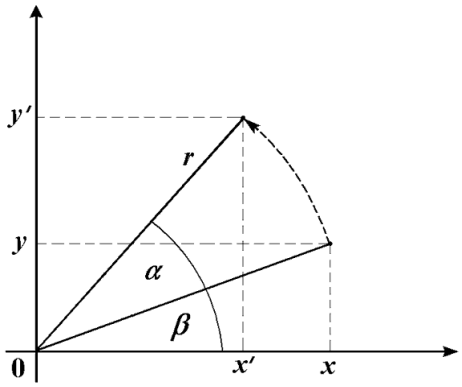

Рис. 3.7. Поворот на площині

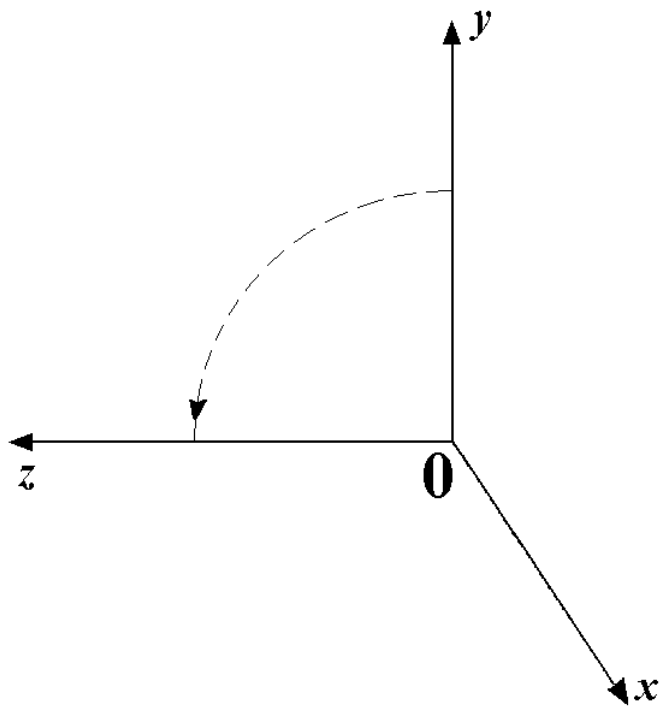

Рис. 3.8. Поворот в просторі

У разі тривимірного простору міркування, що стосуються перенесення і масштабування, повністю аналогічні, тільки вони поширюються на третю координату точок. З обертанням же справа йде інакше, оскільки тут обертальний рух є переміщення уздовж поверхні сфери і поворот на якийсь кут відносно точки не можна визначити однозначно. Але переміщення з однієї точки сфери в іншу завжди можна здійснити послідовністю поворотів щодо осей координат, тому виведемо формули для цих трьох обертань.

При повороті щодо осі OX на кут α у всіх точках координата x залишається незмінною. Якщо дивитися на площину YOZ з боку кінця осі OX, то осі будуть розташовані так, як показано на рис. 3.8.Додатнім вважається поворот від осі OY до осі OZ. Якщо скористатися формулами для плоских поворотів, то координати y' і z' нової точки визначаються виразами

$$y' = y\cos\alpha - z\sin\alpha$$
$$z' = y\sin\alpha + z\cos\alpha.$$

Формули повороту щодо осі OZ повністю збігаються з тими, які були виведені для плоского випадку, а поворот щодо осі OY виглядає так:

$$x' = x\cos\alpha + z\sin\alpha$$
$$z' = -x\sin\alpha + z\cos\alpha.$$

У всіх цих формулах слід звернути увагу на знаки, так як вони залежать від того, який поворот вважається позитивним (в даному випадку ми маємо справу з правою трійкою базисних векторів).

Перетворення масштабування і повороту на площині і в просторі можна виразити за допомогою матриць. Якщо задані коефіцієнти масштабування S_x, S_y, S_z, то перетворення точки здійснюється за допомогою множення матриці на її радіус-вектор,

$$\vec{r}' = S \cdot \vec{r} = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & s_z \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} S_x x \\ S_y y \\ S_z z \end{pmatrix}. \quad (3.8)$$

Двовимірний випадок виглядає подібним же чином.

Поворот на площині можна здійснити за допомогою матриці

$$R = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}. \quad (3.9)$$

І нарешті, повороти в просторі щодо осей координат можна виконати за допомогою трьох матриць обертання

$$R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}, \quad R_y(\alpha) = \begin{pmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{pmatrix}, \quad (3.10)$$

$$R_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Неважко перевірити, що для матриць обертання справедливе співвідношення

$$R(-\alpha) = R^{(}-1)(\alpha)$$

Для виконання послідовних поворотів навколо осей на кути α, β, γ можна створити матрицю перетворення шляхом перемноження трьох матриць:

$$R = R_z(\gamma) \cdot R_y(\beta) \cdot R_x(\alpha).$$

Використання цієї матриці дасть помітну економію в обчисленнях у порівнянні з послідовними множеннями на кожну з трьох матриць обертання.

Перехід в іншу систему координат

Ми розглянули перетворення геометричних об'єктів, заданих у певній декартовій системі координат. Але в багатьох випадках зручно розглядати ті ж об'єкти в іншій системі координат, оскільки їх опис може стати більш простим. Найпростіший приклад - задання координат паралелепіпеда: найпростіше це зробити в системі координат, поєднаної з однією з його вершин з осями, направленими вздовж ребер. У зв'язку з цим

зупинимося на питанні, як зміняться координати точки при переході від однієї декартової системи координат до іншої.

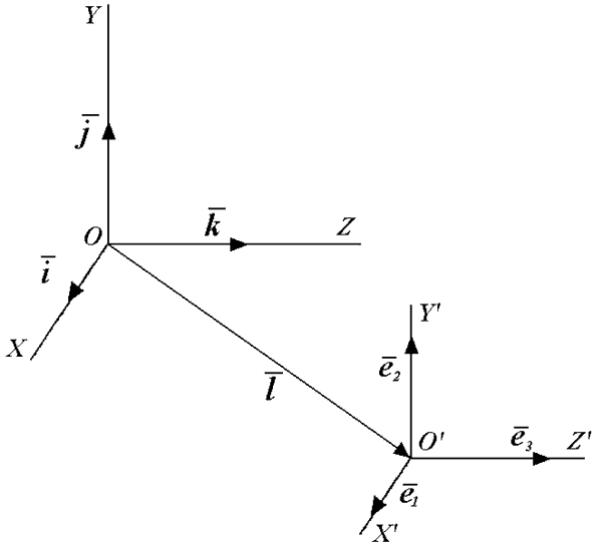

Рис. 3.9. Дві системи координат у просторі

Нехай одиничні орти першої системи координат позначаються $\vec{i}, \vec{j}, \vec{k}$, а осі координат OX, OY, OZ -. Введемо ще одну систему координат, одиничні орти якої позначимо $\vec{e}_1, \vec{e}_2, \vec{e}_3$, а осі координат- $O'X', O'Y', O'Z'$. Ця система має свій початок координат і свої напрямки осей. Вважаємо, що в обох системах координат орти утворюють ліву трійку (рис. 3.9).

Спочатку розглянемо ситуацію, коли точка O' співпадає з точкою O. Вектори $\vec{e}_1, \vec{e}_2, \vec{e}_3$ можна задати в першій системі координат, розклавши їх по векторах $\vec{i}, \vec{j}, \vec{k}$:

$$\left.\begin{aligned} \vec{e}_1 &= e_x^1\vec{i} + e_y^1\vec{j} + e_z^1\vec{k} \\ \vec{e}_2 &= e_x^2\vec{i} + e_y^2\vec{j} + e_z^2\vec{k} \\ \vec{e}_3 &= e_x^3\vec{i} + e_y^3\vec{j} + e_z^3\vec{k} \end{aligned}\right\}.$$

Якщо в першій системі точка $\vec{M}$ має координати (x, y, z), а в другій системі - (x', y', z'), то, очевидно,

$$x'\vec{e}_1 + y'\vec{e}_2 + z'\vec{e}_2 = x\vec{i} + y\vec{j} + z\vec{k}.$$

Множачи скалярно це співвідношення на вектори $\vec{e}_1, \vec{e}_2, \vec{e}_3$, одержимо зв'язок між значеннями координат у різних системах:

$$\left.\begin{aligned} x' &= e_x^1 x + e_y^1 y + e_z^1 z \\ y' &= e_x^2 x + e_y^2 y + e_z^2 z \\ z' &= e_x^3 x + e_y^3 y + e_z^3 z \end{aligned}\right\}.$$

Ці співвідношення можна записати в матричному вигляді

$$(x'//y'//z') = \begin{pmatrix} e_x^1 & e_y^1 & e_z^1 \\ // & // & // \\ e_x^2 & e_y^2 & e_z^2 \\ // & // & // \\ e_x^3 & e_y^3 & e_z^3 \end{pmatrix} \cdot (x//y//z), \quad (3.11)$$

або у векторному запису

$$\overrightarrow{M'} = A \cdot \overrightarrow{M}.$$

Припустимо, що друга система координат отримана з першої шляхом повороту на кут φ щодо осі OY. Тоді

$$\overrightarrow{e}_1 = R_y(\varphi) \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \cos\varphi \\ 0 \\ -\sin\varphi \end{pmatrix}, \quad \overrightarrow{e}_2 = R_y(\varphi) \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix},$$

$$\overrightarrow{e}_3 = R_y(\varphi) \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \sin\varphi \\ 0 \\ \cos\varphi \end{pmatrix},$$

отже

$$A = \begin{pmatrix} \cos\varphi & 0 & -\sin\varphi \\ 0 & 1 & 0 \\ \sin\varphi & 0 & \cos\varphi \end{pmatrix} = R_y(-\varphi) = R_y^{-1}(\varphi).$$

Таким чином, при поворотах системи координат нові координати точок виходять шляхом множення матриці повороту на протилежний кут на вектор вихідних координат.

Якщо нова система координат отримана зі старої шляхом зсуву на вектор $\overrightarrow{l} = (l_x, l_y, l_z)$, то очевидно, що нові координати точки задаються формулами $x' = x - l_x, \quad y' = y - l_y, \quad z' = z - l_z.$

Тепер можна розглянути композицію двох перетворень системи координат - перенесення і обертання. Тоді координати точок перетворюються за формулою

$$\overrightarrow{M'} = A \cdot (\overrightarrow{M'} - \overrightarrow{l}). \quad (3.12)$$

Задача обертання щодо довільної осі

Обертання щодо довільної осі також можна реалізувати за допомогою множення матриці на вектор, але попередньо цю матрицю треба побудувати. Припустимо, що пряма проходить через початок координат і задана одиничним вектором $\overrightarrow{l} = (l_x, l_y, l_z)$, і потрібно виконати поворот точки на кут α щодо неї. Для цього скористаємося наступним алгоритмом:

1. Сумістимо пряму з віссю OZ за допомогою повороту системи координат щодо осі OX на кут φ, а потім повороту щодо осі OY на кут ψ.

2. Виконаємо поворот щодо осі OZ на кут α.

3. Виконаємо повороти системи спочатку щодо осі OY на кут $-\psi$, а потім щодо осі OX на кут $-\varphi$ (в зворотному порядку по відношенню до перших поворотів), тим самим повертаючи її у вихідне положення.

Підсумкова матриця перетворення, таким чином, є твором декількох матриць, а саме

$$R = R_x(\varphi) \cdot R_y(\psi) \cdot R_z(\alpha) \cdot R_y(-\psi) \cdot R_x(-\varphi).$$

Матриці $R_y(-\psi), R_x(-\varphi)$ є матрицями перетворення координат при поворотах системи координат, як було показано в попередньому розділі. Визначимо спочатку кут φ, який є кутом між віссю OZ і його проекцією вектора на площину YOZ $\overrightarrow{l}_1 = (0, l_y, l_z)$. Нехай $d = \sqrt{l_y^2 + l_z^2}$ - довжина цієї проекції. Тоді, $\cos\varphi = \frac{l_z}{d}$, $\sin\varphi = -\frac{l_y}{d}$ (синус негативний, оскільки поворот іде від осі OZ до осі OY, тобто в негативному напрямку). Після повороту системи координат новими

координатами вектора $\overrightarrow{l}$ будуть $l_x, 0, d$. Кут ψ - це кут між векторами $\overrightarrow{l}_1$ і $\overrightarrow{l}$, тому $\cos\psi = d, \sin\psi = l_x$. Тепер ми можемо виписати вид матриць перетворення координат для кожного кроку алгоритму, враховуючи те, що матриці перетворення координат при повороті системи координат назад по відношенню до відповідних матриць обертання:

$$R_x(\varphi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{l_z}{d} & -\frac{l_y}{d} \\ 0 & \frac{l_y}{d} & \frac{l_z}{d} \end{pmatrix}, \quad R_y(\psi =) \begin{pmatrix} d & 0 & -l_x \\ 0 & 1 & 0 \\ l_x & 0 & d \end{pmatrix},$$

$$R_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Неважко переконатися, що послідовне множення матриць $R_x(-\varphi)$ і $R_y(-psi)$ на вектор $\overrightarrow{l}$ дадуть у результаті вектор $(0,0,1)$, тобто цей вектор дійсно стане віссю аплікат.

Залишається тільки виписати остаточний вид матриці R (для скорочення запису введемо наступні позначення :):

$c = \cos\alpha,\ s = \sin\alpha,\ c_1 = 1 - \cos\alpha$:

$$R = \begin{pmatrix} l_x^2 + c & l_x l_y c_1 - l_z s & l_x l_z c_1 + l_y s \\ l_x l_y c_1 + l_z s & l_y^2 c_1 + c & l_y l_z c_1 - l_x s \\ l_x l_z c_1 - l_y s & l_x l_z c_1 + l_x s & l_z^2 c_1 + c \end{pmatrix}. \quad (3.13)$$

Нагадаємо, що l_x, l_y, l_z є направляючими косинусами прямої, щодо якої виконується поворот. Неважко переконатися, що якщо як осей обертання взяти осі координат, то ми в точності одержимо формули (3.10).

Питання та вправи

1. Дайте визначення декартової системи координат.
2. Що таке вектор?
3. Які вектори вважаються рівними?
4. Які вектори називаються лінійно незалежними?
5. Як виразити довжину вектора, використовуючи операцію скалярного добутку ?
6. Як визначити косинус кута між векторами, використовуючи операцію скалярного добутку ?
7. Доведіть, що векторний добуток задовольняє співвідношенню

$\alpha[\overrightarrow{r}_1 \times \overrightarrow{r}_2] = [\overrightarrow{r}_1 \times (\alpha\overrightarrow{r}_2)]$.

8. Як з довільного вектора отримати одиничний вектор, що співпадає з ним по напрямку? (Ця операція називається **нормуванням вектора**).
9. Яке максимальне число лінійно незалежних векторів у просторі?
10. Що таке орти?
11. Як побудувати параметричне рівняння прямої, що проходить через дві задані точки площини або простору?
12. Довести, що якщо у формулі (3,7), замінити координати (x_1, y_1, z_1) координатами будь-якої точки площини, то рівняння буде описати ту ж саму площину. **Примітка**: візьміть довільну точку, що задовольняє рівняння (3,7),і напишіть нове рівняння площини та покажіть, що будь-яка точка другої площини , належить першій площині і навпаки.

13. В яких випадках промінь з площиною перетинаються?

14. В яких випадках промінь перетинає сферу тільки в одній точці?

15. Виходячи із визначення множення матриці на вектор, довести, що для будь-яких двох векторів $\overrightarrow{r}_1, \overrightarrow{r}_2$, і будь-якої матриці A справедливе співвідношення

$$A \cdot (\alpha \overrightarrow{r}) = \alpha(A \cdot \overrightarrow{r}) = (\alpha A) \cdot \overrightarrow{r}.$$

16. Довести, що для будь-якого векторна $\overrightarrow{r}$, числа α і матриці A справедливе співвідношення $A \cdot (\alpha \overrightarrow{r}) = \alpha(A \cdot \overrightarrow{r}) = (\alpha A) \cdot \overrightarrow{r}.$.

17. При якій умові масштабування зберігає кути між відрізками ?

18. Яку траєкторію описують точки об'єкта при повороті.

19. Навколо чого здійснюється поворот на на площині?

20. Навколо чого здійснюється поворот в просторі?

21. Які кроки виконуються в алгоритмі повороту відносно довільної осі у просторі?

22. Докажіть, що якщо матриця A є матрицею обертання, то $A \cdot A^T = E$.

Лекція 4. Представлення геометричної інформації

Зміст

Геометричні примітиви

Під геометричними примітивами розуміють той базовий набір основних геометричних фігур, який лежить в основі всіх графічних побудов, причому ці фігури повинні створювати "базис" в тому сенсі, що жоден з цих об'єктів не може існувати через інший. Однак, питання про те, що включати в набір геометричних примітивів, не можна вважати остаточно вирішеним у комп'ютерній графіці. Наприклад, кількість примітивів можна звести до мінімуму, без якого не можна обійтись і цей мінімум зводиться до апаратно реалізованих графічних об'єктів. У цьому випадку базовий набір обмежений відрізком, багатокутником та набором літер (символів).

Друга точка зору заключається в тому, що в набір примітивів необхідно включити згладжувані криві різних видів (кола, еліпси, криві Безьє), деякі класи поверхонь і навіть суцільні геометричні тіла. В якості трьохмірних геометричних примітивів в даному випадку, пропонуються просторові криві , паралелепіпеди, еліпсоїди, піраміди. Але якщо такий розширений набір примітивів, пов'язаний з апаратною реалізацією, то виникає проблема перенесення програмних додатків з одного комп'ютера на інший, тому що така апаратна підтримка існує далеко не в усіх робочих станціях. Крім того, при створенні трьохмірних геометричних примітивів , програмісти стикаються з проблемою їх математичного опису, а також розробкою методів маніпулювання такими об’єктами , тому що ті типи об'єктів, які не входять до списку базових, необхідно вміти наближувати за допомогою цих примітивів.

У багатьох випадках для апроксимації складних поверхонь використовуються багатогранники, але форма граней може бути різною. Просторовий багатокутник з числом вершин більше трьох не завжди є плоский, а в цьому випадку алгоритми зображення багатогранників можуть призвести до некоректного результату. Таким чином, програміст повинен сам потурбуватися про те, щоб багатогранник був описаний правильно. У цьому випадку, найкращим рішенням є використання трикутників, тому що трикутник завжди плоский. У сучасній графіці це, мабуть, найбільш поширених підхід.

Але є і альтернативний напрямок, який називається **конструктивною геометрією тіл**. В системах, які використовують цей підхід, об'єкти будуються із об’ємних примітивів, з використанням теоретико-множинних операцій (об'єднання, перетин).

Будь-яка графічна бібліотека визначає свій набір примітивів. Так, наприклад, широко поширена інтерактивна система трьохмірної графіки OpenGL включає в свій список примітивів точки (вершини), відрізки, ломані багатокутники, (серед яких особливо виділяються трикутники і чотирикутники), полоси (групи трикутники або

чотирикутників із спільними вершинами) і шрифти. Крім того, вона включає в себе деякі геометричних тіла: сфера, циліндр, конус, тощо.

Ясно, що для зображення таких примітивів повинні бути розроблені ефективні і надійні алгоритми, тому що вони являються конструктивними елементами. Історично склалось так, що перші дисплеї були векторними, тому базовим примітивом був відрізок. Але, як вже відзначили в першому розділі нашого курсу, сама перша інтерактивна програма Sketchpad А. Сазерленда в якості одного із примітивів мала прямокутник, після чого цей об'єкт вже традиційно входив до різних графічних бібліотек.

В подальшому ми проглянемо такі примітиви як **вершина, відрізок, воксель(**voksel′i) **і моделі**, засновані на їх основі, а також **функціональні моделі**.

Полігональні моделі

Для цих просторових моделей використовуються в якості примітивів вершини (точки в просторі), відрізки прямых (вектори),из яких будуються **полілінії**, **полігони** и **полігональні поверхні**.Головним елементом опису являється вершина, всі останні являються похідними. В трьохмірній декартовій системі координат вершины визначаються своїми координатами (x,y,z), линия задається двома вершинами,полілінія представляє собою незамкнуту ломану лінію, полігон -замкнуту ломану лінію. Полігон моделює плоский об’єкт і може описувати плоску грань об’ємного объекта. Декілька граней складають цей об’єкт у вигляді полігональної поверхні - багатокутник або незамкнуту поверхню ("полігональна сітка").

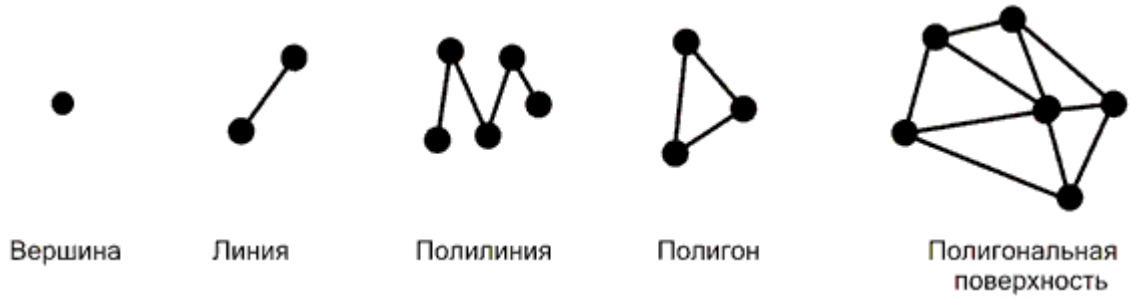

Рис. 4.1. Полігональні моделі

В сучасній комп’ютерній графіці векторно-полігональна модель являється найбільше поширеною.

Вона використовується у системах автоматизованого проектування, комп'ютерних іграх, тренажерах, Г І С, САПР і т. і. Переваги цієї моделі є наступні:

• Зручності масштабування об'єктів.
• Невеликий об’єм даних для опису простий поверхонь.
• Апаратна підтримка багатьох операцій.

До числа недоліків полігональних моделей, можна віднести те, що алгоритми візуалізації виконання топологічних операцій (наприклад, побудова перерізів) є досить складною. Крім того, апроксимація плоскими гранями призводить до значних похибок, особливо при моделюванні поверхонь складної форми.

Воксельні моделі

Воксельна модель – це представлення об'єктів у вигляді трьохмірного масиву об’ємних (кубічних) елементів. Назва воксель "voksel′"-складається з двох слів: volume element. Так як і піксель , воксель має свої атрибути (колір, прозорість, тощо). Повна

прозорість вокселя означає пустоту у відповідній точці об’єму. Чим більше вокселів у визначеному об’ємі і менший їх розмір, тим точніше моделюються трьохмірні об’єкти.

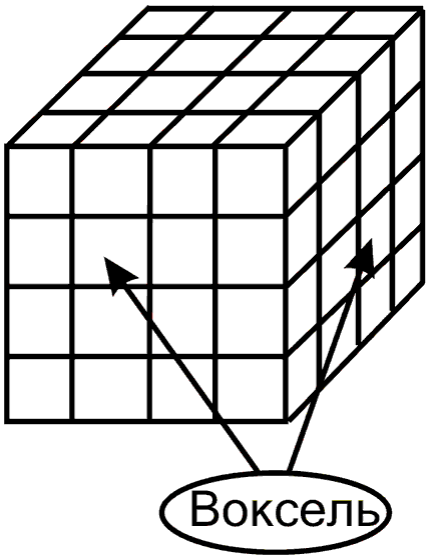

Рис. 4.2. Воксельная модель

Позитивними рисами воксельної моделі являються:

- Можливість представляти внутрішність объекта, а не только зовнішній шар; проста процедура відображення об’ємних сцен.
- Простое виконання топологичних операцій; наприклад, для того, щоб показати переріз просторового тіла, достатньо вокселі зробити прозорими.

До її недоліків відносяться :

- Великий об’єм інформації, необхідний для представлення об’ємних даних.
- Значні затрати пам’яті, що обмежує дозволяючу властивість , точність моделювання.
- Проблеми при збільшенні або зменшенні зображення; наприклад, із збільшенням погіршується дозволяючи властивість зображення.

Поверхні вільних форм (функціональні моделі)

Характерною особливістю пропонуємого способу задання поверхонь являється те, що основним примітивом при цьому являється поверхня другого порядку - **квадрик**. Він визначається за допомогою реальної неперервної функції трьох змінних (x, y, z) у вигляді нерівеності

$$F(x, y, z) \geq 0.$$

Таким чином, **квадрик** є замкнута підмножин евклідового простору, всі точки якого задовольняють вказаній нерівності. Рівняння

$$F(x, y, z) = 0$$

описує **межу** цієї множини. Множина точок, що задовольняє нерівність

$$F(x, y, z) < 0,$$

утворює **зовнішню область** квадрика.

Вільна форма - це довільна поверхня, має властивості гладкості, безперервності і нерозривності. На базі квадриків будуються вільні форми, які описують функціональні моделі.Вільна форма, побудована на цих принципах, має ряд достоїнств, до яких, в першу чергу, необхідно віднести слідуючее:

- Легка процедура розрахунку координат кожної точки.
- Невеликий об’єм інформації для опису достатньо складних форм.
- Можливість будувати поверхні на основі скалярних даних **без попередньої тріангуляції**.

Цей підхід буде більш детально поданий в наступних розділах.

В нашому курсі пердбачається розглянути растрові алгоритми для зображення таких геометричних приміітивів, як відрізки, багатокутники, кола і еліпси. Але спочатку ми займемось тим геометричним апаратом, який дасть можливість адекватно описувати об'єкти в просторі, працювати з ними і формувати зображення.

Системи координат: світова, об'єктна, спостерігача і екранна

Однією із поширених задач комп'ютерної графіки являється зображення двохмірних графікі в деякій системі координат. Ці графіки призначені для відображення залежності між змінними, заданими за допомогою функції. Наприклад, у другому розділі даного курса приведений ряд графіків, характеризуючих сприйняття світла оком людини. Щоб отримати такий графік, прикладная программа повинна описати різні вихідні примітиви (точки, лінії, ланки символів), вказуючи їх месцеположення і розміри в прямокутній системі координат. Одиниці вимірів, в яких задаються ці об'єкти, залежать від їх природи: зміну температури, наприклад, можна відображати в градусах за час, перемещення тіла в просторі - в кілометрах за секунду, и т.і. Ці прикладні (або оріентовані на користувача) координати дають можливість задавати об'єкти в двохмірному або трьохмірному світі користувача, і їх принято називати **світовими координатами**.

Зображення трьохвимірних об'єктів поєднано з цілим рядом задач. Перш за все необхідно пам'ятати, що зображення являється плоским, тому необхідно добитися адекватної передачі візуальних властивостей предметів, дати досить точне наглядне представлення про глибину. В подальшому групы трьохмірних об'єктів, призначеных для зображення, будемо называти **просторовою сценою**, а її двохмірне зображення - **образом**.

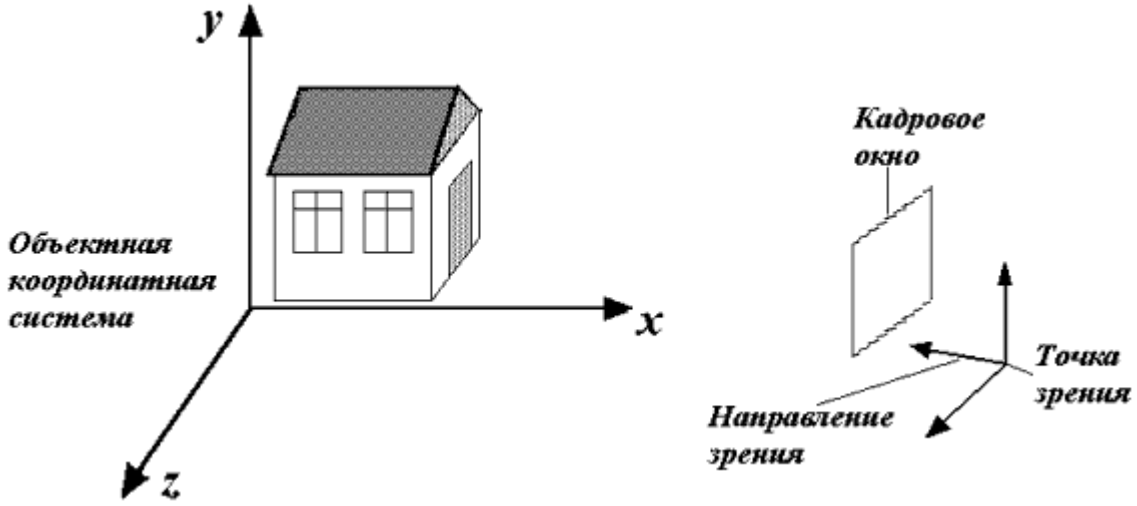

Рис. 4.3. Об'єктна система координат і система координат спостерігача

Як і у випадку із двохмірними об'єктами, першим кроком побудови являється введення інформації про ці об'єкти. Сцена займає деяке визначене місце в просторі, а її опис прив'язується до трьохмірної декартової системи координат, зв'язаної з нею, - **об'єктної координатної системи**. Координати об'єктів, створюють сцену, визначаються на основі їх реальних розмірів і взємного розташування. В залежності від точки, з якої розглядається сцена, можна отримати множину різних її образів. Якщо побудовано достатньо багато таких образів, то по ним можна відновлювати об'ємну структуру предмета. Вибір точки і напрямку зору також можна описати математично, ввівши декартову **систему координат спостерігача** , початок якої знаходиться в точці огляду , а одна із осей співпадає з напрямком зору (рис. 4.3). Перехід від об'єктних координат до координат користувача математично реалізується так, як це було описано в третій лекції . На цьому этапі перетворень зберігаються реальні розміри об'єктів.

Видимий образ формується на деякій площині, яку в подальшому будемо називати **картинною площиною**. Способи перетворення трьохмірного об'єкта в двохмірний образ (**проекції**) можуть бути різними. Так чи інакше, але отриманий образ також повинен бути описаний в деякій двохмірній системі координат. В залежності від способу його отримання реальныі розмір образа також можуть бути різні. Різні види проектування будуть детально розглянуті в наступних лекціях.

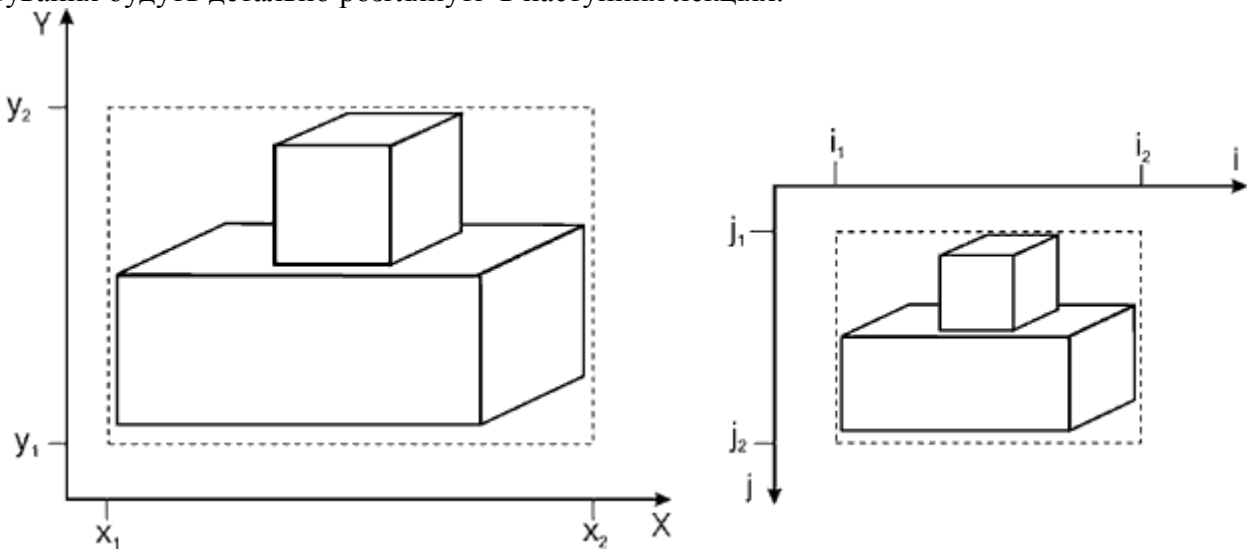

Рис. 4.4. Картинна площина і екран

Так як нашою кінцевою метою являється отримання зображення на екрані, то перенесення образу супроводиться зміною масштабу у відповідності із розмірами екрана. Звичайно, початком координат в системі координат образу вважається лівий нижній кут листа із зображенням. На екрані дисплею початок координат традиційно знаходиться в лівому верхньому куті. Відображення рисунка з картинні площини на экран повинно виконуватися з мінімальним спотворенням пропорцій, що само по собі вносить обмеження на область екрана, заняту рисунком. Зміна масштабу повинна здійснюватися із збереженням пропорцій області (рис. 4.4).

Об'єкти в системі координат картинної площини задаються в деяких одиницях вимірювань, причому масштаб однаковий по обом осям координат. На екрані одиницею виміру являється **піксель**, який слід розглядати як прямокутний, тому масштаби по горизонтальній і вертикальній осям можуть бути різними, що необхідно враховувати при задаванні коефіцієнтів масштабування.

Розглянемо ситуацію, коли зображення займає на картинній площині прямокутну область $x_1 \le x_2,\ y_1 \le y \le y_2$. При відображенні рисунка на экран кожна точка вихідного прямокутника з координатами (x, y) перейде в деяку точку із цілочисленими координатами (i, j). Введемо позначення:

$$\Delta x = x_2 - x_1, \quad \Delta y = y_2 - y_1, \quad \Delta i = i_2 - i_1, \quad \Delta j = j_2 - j_1,$$

$$S_x = \frac{\Delta i}{\Delta x}, \quad S_y = \frac{\Delta j}{\Delta y}$$

(припускається, що зображення займе на екрані прямокутник $i_1 \le i \le i_2,\ j_1 \le j_2$). Визначимо перетворення координат образу (x, y) в экранні координати (i, j) формулами

$$i = i_1 + S_x(x - x_1), \quad j = j_2 - S_y(y - y_1).$$

Ясно, що при такому відображенні прямокутна область образу в точності перейде у відповідний екранний прямокутник, як показано на рисунку. Теперь необхідно

визначити сам екранний прямокутник так, щоб його пропорції відповідали прямокутнику образу, тобто

$$\frac{\Delta x}{\Delta y} = \frac{\Delta i \cdot l_x}{\Delta j \cdot l_y} \equiv \kappa \frac{\Delta i}{\Delta j},$$

де l_x, l_y - горизонтальний і вертикальний розмір одного пікселя. Ці параметри легко встановити, знаючи розміри екрану і розрішення. Звідси отримуємо

$$\Delta j = \kappa \cdot S_x \cdot \Delta y, \quad S_y = \kappa \cdot S_x$$

Тепер достатньо задати на экрані початок відліку і горизонтальний розмір вікна, а остальні параметри легко вычисляються.

Однородні координати. Задання геометричних перетворень в однорідних координатах за допомогою иатриць

В попередній лекції описувались геометричні перетворення на площині і в просторі , а також було показано, як можна використати апарат матриць для таких задач. Для перетворень на площині застосовувались двохмірні вектори і матриці розмірністю 2×2. В просторі, відповідно , з цією ж метою застосовувались трьохмірні вектори і матриці 3×3. Але такий підхід не дає можливості задавати за допомогою матриць петворення переносу і проекції. Тому в проективній геометрії був розроблений апарат, дозволяючий уніфікувати всі геометричені перетворення шляхом введення так званих **однорідних координат**.

Для пояснення такого підходу спочатку розглянемо випадок двохмірного простору. Кожна точка площини з координатами (x, y) може одночасно розглядатися як точка трьохмірного простору з координатами $(x, y, 1)$, тобто як точка, що лежить на площині $z = 1$. З другої сторони, кожній точці трьохмірного простору (x, y, z) при умові $z \neq 0$ відповідає єдина точка цієї ж площини $\left(\frac{x}{z}, \frac{y}{z}, 1\right)$. При цьому виходить , що якщо кожній точці площини $(x, y, 1)$ відповідає пряма, яка проходить через початок координат, тобто встанавлюється взаємно однозначна відповідність між точками площини і множини $(tx, ty, t),\ -\infty < t < \infty,\ t \neq 0$.

Якщо тепер розглядати точку площини як належну **трьохмірному** простору, то її **двохмірні** перетворення можна буде описувати за допомогою матриць 3×3, причому можна буде задавати таким способом не тільки повороти і маштабування, але і зсуви і проекції (як ортографічні, так і центральні).

Поворот на кут α відносно початку координат можна здійснити за допомогою нової матриці повороту:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}.$$

Операція масштабування може бути записана у вигляді

$$\begin{pmatrix} ax \\ by \\ 1 \end{pmatrix} = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}.$$

Перенесення на вектор (x_0, y_0) також можна задати за допомогою матриці:

$$\begin{pmatrix} x + x_0 \\ y + y_0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}.$$

Проекції точки на осі координат визначаються за допомогою матриці проекції:

$$P_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad P_y = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Перейдемо тепер до трьохмірного простору. Кожній точці (x, y, z)будемо ставити у відповідність точку четирьохмірного простору $(x, y, z, 1)$, а для виконання основних перетворень будемо використовувати матриці розмірністю 4×4. Будуються вони цілком аналогічно тому, як це робилося у двохмірному випадку. Матриця зсуву на вектор (x_0, y_0, z_0)має вигляд

$$P = \begin{pmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

матриця масштабування також очевидним чином утворюється із трьохмірної матриці:

$$S = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Проекції точок на координатні площини здійснюються за допомогою матриць (більш детально проекції і їх види будуть розглянуті пізніше):

$$P_{xy} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad P_{yz} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad P_{zx} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Множення цих матриць на вектор приводить до того, що обнуляється одна із координат, і в результаті отримуємо проекцію точки на відповідну площину.

Матриця повороту відносно осі OX на кут α выглядає слідуючим чином:

$$R_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Звідси легко зрозуміти, як будуються матриці повороту відносно других координатних осей, а також матриця повороту відносно довільної осі. Просто беремо матриці, поудовані в третій лекції, і поширюємо їх шляхом додавання вже відомих одиничних вектора-строчки і вектора-стовпчика:

$$\begin{pmatrix} * & * & * & 0 \\ * & * & * & 0 \\ * & * & * & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Шляхом поєднання приведенних элементарних перетворень можна побудувати і більш складні. В третій лекції ми використали добуток простих матриць обертанням для побудови матриці повороту відносно довільної осі. Приведемо один приклад.

Нехай в просторі задані два відрізка - $\overline{AB}$ і $\overline{CD}$. Будемо будувати матрицю перетворення, яка переводить перший відрізок у другий. Це перетворення розкладемо на слідуючі елементарні дії.

1. Зсув, перемещуючий точку $A = (A_x, A_y, A_z)$ в точку $C = (C_x, C_y, C_z)$.
2. Зсув початку координат в ту ж точку.
3. Якщо відрізки неколінеарні:
   - будується вектор нормалі до площини, в якій лежать відрізки (для цього можна використати векторний добуток вихідних векторів);
   - поворот відносно вектора нормалі, що суміщає два відрізка по напрямку (кут повороту можна визначити за допомогою скалярного добутку вихідних векторів).
4. Масштабування з метою вирівнювання довжин відрізків.
5. Повернення початку координат у вихідну точку.

Кожне із цих перетворень реализується за допомогою матриці, а повне перетворення можна виконати, використовуючи добуток матриць.

Використання матриць дуже зручно для виконання перетворень в просторі, хоча в деяких випадках це приводить до надлишкового числа виконуємих операцій. Наприклад, поворот однієї точки в просторі відносно координатної осі OZ за допомогою матриць в однорідних координатах потребує 16 операцій множення і 12 операцій додавання. В той же час він легко може бути виконаний за допомогою формул перетворення

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

Тобто за допомогою всього лише чотирьох множнь і одного додавання і одного віднімання. Операції зсуву також набагато економічніше виконувати без використання матриць. Але коли мова йде про суперпозиції багатьох перетворень (як, наприклад, у випадку повороту відносно довільної осі), то доцільно застосувати відповідну матрицю повороту. Ефективність матричного підходу дуже сильно зростає, якщо матричні операції реализовані аппаратно. Питання про те, в яких випадках використовувати матриці, а в яких ні, в багатьох випадках залежить від можливостей обчислювальної техники, рівня складнсті задачі і вимог до часових характеристик процесу візуалізації.

Питання та вправи

1. Які геометричні об'єкти вважаються примітивами?
2. Які вимоги пред'являються до набору геометричних примітивів?
3. В якій програмі вперше як геометричного примітиву використовувався прямокутник?
4. Що таке об'єктна система координат?
5. Що таке система координат спостерігача?
6. Чи відповідають розміри об'єктів в системі координат спостерігача їхнім реальним розмірам?
7. Що таке картинна площину?
8. Як називається операція переходу від тривимірної системи координат до двовимірної?
9. Що відбувається при перенесенні зображення з картинної площини на екран?
10. Чим відрізняються однорідні координати точки від звичайних декартових координат?
11. З якою метою вводяться однорідні координати?
12. Скільки елементарних дій потрібно для поєднання двох відрізків у просторі?
13. Чи завжди використання матриць для виконання перетворень в просторі ефективніше, ніж інші способи їх реалізації?

Лекція 5: Відсікання (кліпування) геометричних примітивів

Зміст

- Алгоритм Сазерленда-Коена відсікання прямокутною областю
- Відсікання опуклим багатокутником
- Кліпування багатокутників
- Питання та вправи

У комп'ютерній графіці часто доводиться вирішувати завдання виділення деякої області зображуваної сцени, причому завдання це може вирішуватися як у застосуванні до плоскої області (якщо сцена вже спроектована на картинну площину), так і до тривимірної. Алгоритми відсікання застосовуються для видалення невидимих поверхонь і ліній, для побудови тіней, при формуванні текстур. Відсікається область може бути як правильної форми (прямокутник або паралелепіпед зі сторонами, паралельними осям координат або координатним площинам), так і неправильної (довільний багатокутник або багатогранник). Для того щоб ці алгоритми можна було використовувати в задачах зображення динамічних сцен, вони повинні бути ефективними у відношенні часу обчислень. Ми розглянемо декілька найбільш часто застосовуваних алгоритмів.

Алгоритм Сазерленда-Коена відсікання прямокутною областю

Розглянемо плоску сцену, що складається з відрізків різної довжини і напрямків, в якій треба виділити частину, що знаходиться усередині прямокутника. Прямокутник заданий списком ребер: **<top>, <bottom> <left > <right>,** відрізки також задаються координатами кінцевих точек.Область, що відсікає вікном (з урахуванням його межі), складається з точок (x, y), що задовольняють співвідношенням

$$x_l \le x \le x_r, \quad y_b \le y \le y_t. \quad (5.1)$$

Нехай кінці відрізка задані точками (x_1, y_1) і (x_2, y_2). Перший крок алгоритму націлений на те, щоб виявити повністю видимі і повністю невидимі отрезкі.Отрезок цілком належить виділюваної (кліппіруемой) області, якщо обидва його кінця задовольняють умовам (5.1).

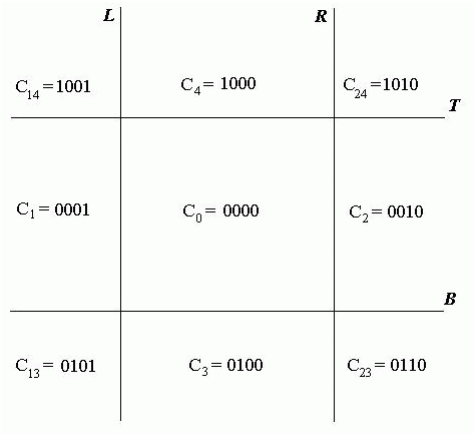

Рис. 5.1. Коди Сазерленда-Коена для областей

Відрізок повністю невидимий, якщо обидва його кінця лежать

1. праворуч від ребра $r(x_1 > x_r, \quad x_2 > x_r)$;
2. ліворуч від ребра $l(x_1 < x_l, \quad x_2 < x_l)$;
3. знизу від ребра $b(y_1 < y_b, \quad y_2 < y_b)$;
4. зверху від ребра $t(y_1 > y_t, \quad y_2 > y_t)$.

У всіх інших випадках відрізок може (але не зобов'язаний) перетинати прямокутне вікно.

Для виконання аналізу повної видимості або невидимості відрізка А.Сазерленд і Д.Коен запропонували наступний алгоритм. Прямі, яким належать ребра прямокутника, розбивають площину на дев'ять областей, кожній з яких присвоюється чотирирозрядний код. Кожен біт цього коду "відповідає" за одну з прямих: 1-й (старший) біт - за пряму t, 2-й - за пряму b, 3-й - за r, 4-й - за l. Якщо в коді області небудь біт встановлений в 1, то це означає, що вона відокремлена від вікна відповідної прямої. Схема ідентифікації областей наведена на рис. 5.1.

Кінцевим точкам відрізків сцени тепер можна привласнити коди в залежності від розташування точок. Ясно, що якщо коди обох кінців відрізка рівні нулю, то відрізок повністю лежить всередині вікна. Для подальшого аналізу скористаємося операцією логічного множення кодів (порозрядне логічне "І"). Тоді таблиця істинності для кодів, згідно зі схемою на рис. 5.1, буде виглядати наступним чином:

Таблиця 5.1. Значення істинності для логичного множення кодів областей

	C_1	C_2	C_3	C_4	C_{13}	C_{14}	C_{23}	C_{24}
C_1	T	F	F	F	T	T	F	F
C_2	F	T	F	F	F	F	T	T
C_3	F	F	T	F	T	F	T	F
C_4	F	F	F	T	F	T	F	T
C_{13}	T	F	T	F	T	T	T	F
C_{14}	T	F	F	T	T	T	F	T
C_{23}	F	T	T	F	T	F	T	T
C_{24}	F	T	F	T	F	T	T	T

Із зіставлення таблиці з малюнком видно, що якщо твір кодів кінців відрізка приймає значення <**True**>, то відрізок цілком лежить по одну сторону якоїсь із прямих, причому зовнішню сторону по відношенню до вікна, отже, він повністю невидимий. У всіх інших випадках відрізок може частково лежати усередині вікна, тому для визначення їх видимої частини треба вирішувати задачу про перетин відрізків з ребрами вікна. При цьому бажано по можливості скоротити число перебираються пар "відрізок-ребро».

У самому загальному випадку існують дві точки перетину відрізка з ребрами, і ці дві точки приймаються за нові кінцеві точки зображуваного відрізка. Але спочатку можна виділити деякі більш прості окремі випадки, пошук перетинань для яких є більш ефективним. Перш за все це горизонтальні і вертикальні відрізки, для яких пошук точки перетину тривіальний. Далі, якщо код одного з кінців відрізка дорівнює нулю, то існує тільки одне перетинання цього відрізка з ребром (або з двома ребрами, якщо відрізок проходить через кутову точку вікна). На рис. 5,2 наведена загальна блок-схема алгоритму відсікання для одного довільно спрямованого відрізка.

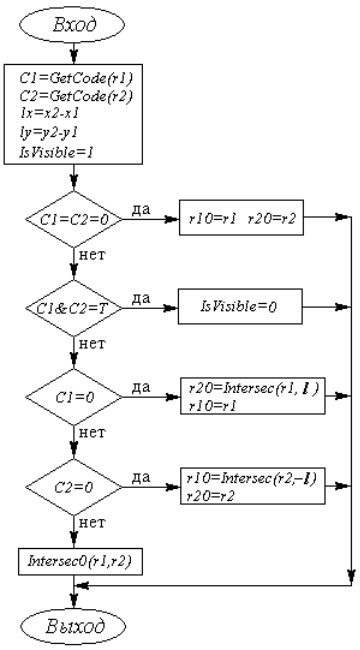

Рис. 5.2. Блок-схема алгоритму Сазерленда-Коена

У блок-схемі використовуються наступні угоди і позначення:

• вхідними даними є точки $\overrightarrow{r}_1 = (x_1, y_1),\ \overrightarrow{r}_2 = (x_2, y_2)$, масив координат вікна $w = \{L, R, B, T\};\ \overrightarrow{l} = (x_2 - x_1, y_2 - y_1) \equiv (l_x, l_y)$;

• на виході отримуємо нові кінцеві точки $\overrightarrow{r}_{10} = (x_{10}, y_{10}),\ \overrightarrow{r}_{20} = (x_{20}, y_{20})$, а також значення змінної IsVisible (0 - відрізок видимий);

• використовуються наступні допоміжні функції:

GetCode(r) - визначення коду точки;

Intersec0(r1,l)- пошук перетину відрізка зі сторонами вікна за умови, що обидві точки лежать поза вікном, якщо перетину немає, встановлює змінну IsVisiblев 0;

Intersec(r1,l)- пошук перетину відрізка зі сторонами вікна за умови, що точка r1лежить у вікні;

C1, C2- коди точок *r1, r2*..

В наведеному алгоритмі тепер залишається тільки деталізовано функції Intersec0і Intersec, ефективність роботи яких є ключовим моментом. Розглянемо один з методів пошуку перетинань, який використовує параметричне рівняння прямої, що проходить через точки $\overrightarrow{r}_1 = (x_1, y_1),\ \overrightarrow{r}_2 = (x_2, y_2)$:

$$\overrightarrow{r} = \overrightarrow{r}_1 + s(\overrightarrow{r}_2 - \overrightarrow{r}_1), \quad s \in [-\infty, +\infty],$$

або в координатному вигляді

$$x = x_1 + sl_x, \quad y = y_1 + sl_y, \quad l_x = x_2 - x_1, \quad l_y = y_2 - y_1.$$

Спробуємо визначити точку перетину відрізка з верхньою межею вікна. Оскільки ця межа описується співвідношеннями

$$L \le x \le R, \quad y = T,$$

то умова перетину з нею кліппіруемого відрізка виглядає наступним чином:

$$s_T = \frac{T - y_1}{l_y}, \quad L \le x_1 + s_T l_x \le R.$$

Аналогічно виглядають формули і для інших кордонів вікна. Якщо точка $\overrightarrow{r}_1$ розташована всередині вікна, то, в залежності від знака l_y, слід шукати перетин або з верхньої, або з нижньою межею вікна. При відсутності таких відшукуються перетину з

лівою або правою стороною вікна. Але перш ніж перебирати ці варіанти, необхідно виключити випадки горизонтальних $(l_y = 0)$ і вертикальних $(l_x = 0)$ напрямів відрізка.

У першому випадку точками перетину з правої або лівої кордоном (в залежності від знака l_x) можуть бути (L, y_1) або (R, y_1), а в другому (x_1, B) - або (x_1, T).

Цей алгоритм реалізований у вигляді функції Intersec, блок-схема якої наведена на рис. 5.3.

Тепер розглянемо випадок, коли жоден з кінців відрізка не лежить усередині вікна. Тут ми можемо знайти першу точку перетину, замінити нею один з кінців відрізка і використовувати попередній алгоритм знаходження другої точки. Зауважимо, що в даному випадку перший крок не завжди закінчується успішно, оскільки попередній аналіз не дозволяє повністю виключити всі невидимі відрізки. Як і в попередньому алгоритмі, спочатку виконується перевірка на горизонтальне і вертикальне напрямки відрізка. Оскільки частина відрізків ми вже виключили з розгляду завдяки попереднім аналізом, то для цих простих ситуацій залишаються тільки дві можливості, наведені нарис. 5.4. Тут точки перетину визначаються абсолютно очевидним чином, причому відразу обидві.

Потім послідовно розглядаються чотири випадки розташування точки $\overrightarrow{r}_1$ щодо прямих, що обмежують вікно. Якщо в якомусь з варіантів буде знайдена точка перетину, то аналіз припиняється, точка $\overrightarrow{r}_1$ замінюється новою точкою і викликається функція **Intersec**. У разі ж, коли всі чотири варіанти не дали позитивного результату, значення змінної **IsVisible** присвоюється 0. Алгоритм реалізується функцією **Intersec0** (блок-схема на рис. 5.5).

Запропонована реалізація алгоритму відсікання не є єдиною в своєму роді. Існують і інші підходи, зокрема використання методу розподілу відрізка навпіл для пошуку точок перетину і виявлення видимої частини відрізка. Такий варіант алгоритму був запропонований Сазерлендом і Спрулом для апаратної реалізації, але він може бути реалізований і на програмному рівні, хоча при цьому ефективність його буде нижче, ніж у попереднього. У ньому немає прямого обчислення координат нової точки по явним алгебраїчним соотношеніям.Поіск здійснюється ітераційним методом, в якому на кожному кроці для відрізка, "підозрюваного" у частковій видимості, знаходиться його середня точка, визначається її код, потім з двох відрізків залишаються або обидва (якщо вони обидва не виявляться повністю невидимими), або тільки один, після чого операції тривають з новими відрізками. Ситуація, коли подальшому дробленню піддаються відразу два знову отриманих відрізка, може виникнути тільки на першому ітераційному кроці в тих випадках, коли початковий відрізок має дві точки перетину з межами вікна. На наступних ітераційних кроках кількість аналізованих відрізків уже не буде збільшуватися. Процес триває до тих пір, поки довжина чергового відрізку не стане менше наперед заданої точності. Після цього знайдена точка перевіряється на предмет перетинання зі стороною вікна. На рис. 5.6пріведени три варіанти відрізків, попередній аналіз яких не класифікує їх як повністю невидимі, і показаний перший ітераційний крок. Відрізок після першого поділу дає два частково видимих відрізка, після чого шукаються дві точки перетину. В інших випадках залишається лише один з двох відрізків, причому в разі відрізка сточка перетину із стороною вікна відсутній, тобто виявляється повна невидимість відрізка. По суті справи загальний алгоритм, показаний нарис. 5.3, зберігається, змінюється лише метод пошуку точки перетину.

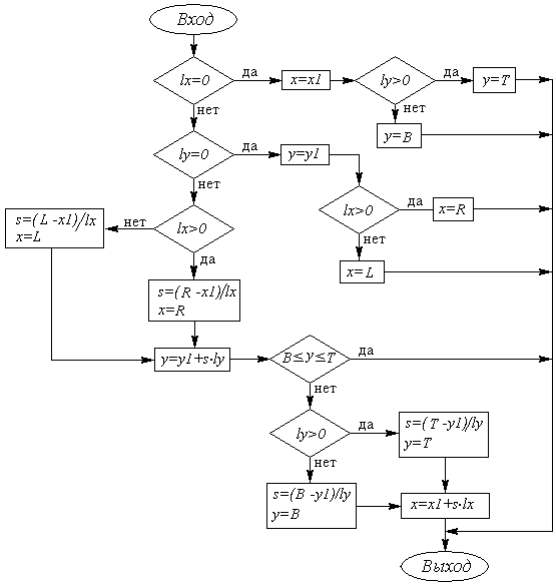

Рис. 5.3. Блок-схема функції Intersec

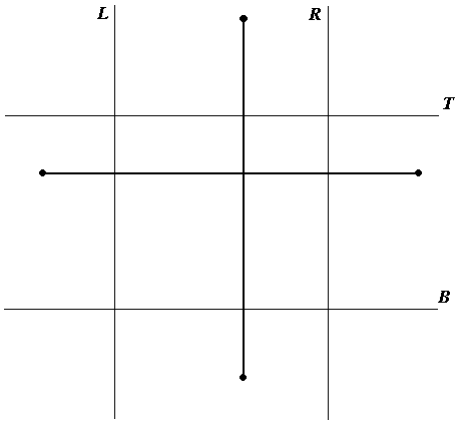

Рис. 5.4. Відрізки паралельні сторонам вікна

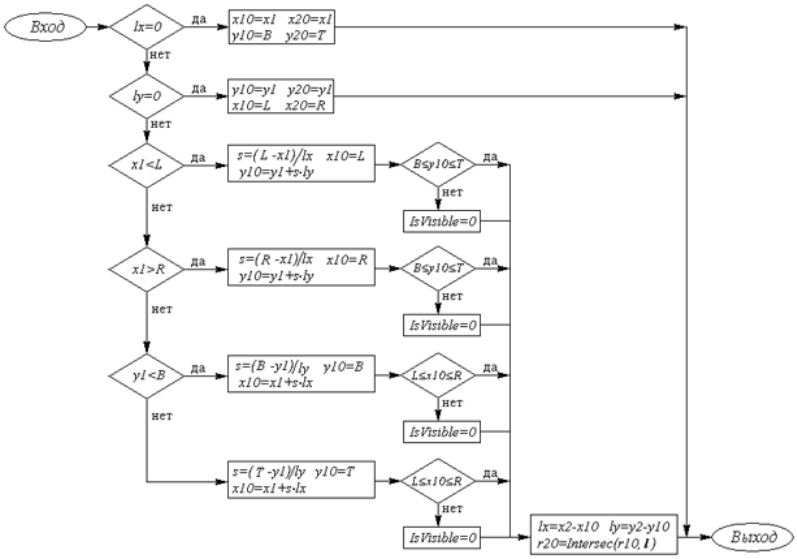

Рис. 5.5. Блок-схема функції Intrsec0

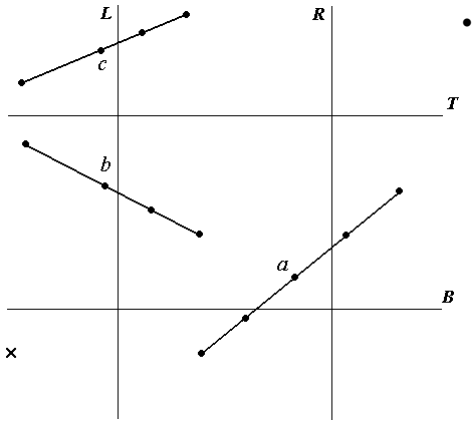

Рис. 5.6. Довільне розташування відрізків

Відсікання опуклим багатокутником

У багатьох задачах комп'ютерної графіки часто доводиться мати справу з відсіканням не тільки простим прямокутним вікном, але і вікном достатньо довільної геометрії. Зокрема, такі завдання можуть виникнути при використанні перспективних проекцій тривимірних сцен, але не тільки в цих випадках. Тому актуальною є задача відсікання опуклим багатокутником. Ясно, що простий аналіз за допомогою кодів Сазерленда-Коена в такій ситуації непридатний. Тут потрібен надійний і досить ефективний алгоритм знаходження точки перетину двох довільно орієнтованих відрізків, а також алгоритм визначення місця розташування точки щодо багатокутника (усередині, зовні або на кордоні).

Розглянемо задачу про перетин відрізка з кінцями $\overrightarrow{r}_1 = (x_1, y_1),\ \overrightarrow{r}_2 = (x_2, y_2)$ з опуклим багатокутником, заданим списком ребер. Ребро може бути задано у вигляді пари точок з безлічі вершин багатокутника $R = (\overrightarrow{p}, \overrightarrow{q}),\ \overrightarrow{p} = (p_x, p_y),\ \overrightarrow{q} = (q_x, q_y)$ (рис. 5.7) Та обставина, що багатокутник опуклий, є дуже суттєвим. Це дозволяє використовувати досить простий алгоритм, який використовує внутрішні нормалі до його сторонам. Під внутрішньою нормаллю розуміється вектор, перпендикулярний стороні і спрямований усередину багатокутника. Як і в попередньому алгоритмі, скористаємося параметричним рівнянням прямої, що проходить через кінці відрізка $\overrightarrow{r} = \overrightarrow{r}_1 + s(\overrightarrow{r}_2 - \overrightarrow{r}_1)$. Якщо при деякому значенні параметра s_0 ця пряма перетинається з прямою, що проходить через точки $\overrightarrow{p}, \overrightarrow{q}$, то вектор, що з'єднує довільну точку ребра з точкою $\overrightarrow{r}_0 = \overrightarrow{r}_1 + s_0(\overrightarrow{r}_2 - \overrightarrow{r}_1)$, буде перпендикулярний вектору нормалі. Отже, скалярний добуток векторів $\overrightarrow{n}$ і $\overrightarrow{p} - \overrightarrow{r}_0$ буде дорівнює нулю. Звідси шляхом нескладних викладок отримуємо $s_0 = \frac{((\overrightarrow{p} - \overrightarrow{r}_1)\cdot\overrightarrow{n})}{((\overrightarrow{r}_2 - \overrightarrow{r}_1)\cdot\overrightarrow{n})}$.

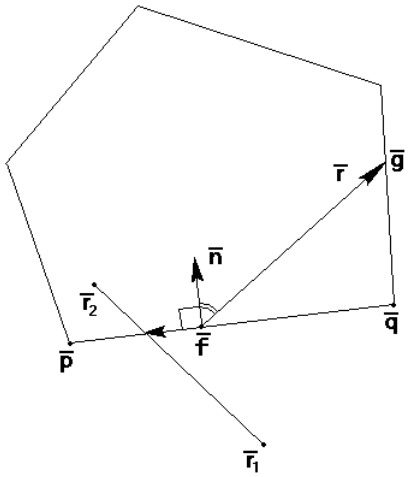

Рис. 5.7. Перетин відрізка багатокутником

Звичайно, використання цієї формули припускає, що, тобто що $d \equiv ((\overrightarrow{r}_2 - \overrightarrow{r}_1) \cdot \overrightarrow{n}) \neq 0$відрізок не паралельний стороні багатокутника, але цей випадок розглядається окремо. Знайдена точка належить відрізку за умови $0 \leq s_0 \leq 1$. Умова приналежності цієї точки ребру багатокутника також можна виразити через скалярний добуток, так як вектори $\overrightarrow{p} - \overrightarrow{r}_0$ і $\overrightarrow{r}_0 - \overrightarrow{q}$ в цьому випадку повинні бути однаково спрямованими, тобто $((\overrightarrow{p} - \overrightarrow{r}_0) \cdot (\overrightarrow{r}_0 - \overrightarrow{q})) \geq 0$.

Для кожного відрізка можливі три випадки взаємного розташування з багатокутником:

- точок перетину немає;
- існує одна точка перетину;
- існують дві точки перетину.

У кожному з цих варіантів для знаходження перетинання відрізка з вікном необхідно вміти визначати приналежність точки опуклому багатокутнику. З рис. 5,7 видно, що якщо для будь-якої точки $\overrightarrow{g}$, що належить багатокутнику (або його кордоні), і довільної точки ребра $\overrightarrow{f}$ побудувати вектор $\overrightarrow{m} = \overrightarrow{g} - \overrightarrow{f}$, то виконується умова $(\overrightarrow{m} \cdot \overrightarrow{n} \geq 0)$, оскільки кут між векторами не може перевищувати 90 градусів. Таким чином, якщо дана умова виконується для всіх ребер багатокутника, то точка є внутрішньою.

Таким чином, алгоритм відсікання відрізка починається з аналізу розташування кінців відрізка по відношенню до вікна. Якщо обидві точки лежать всередині вікна, то відрізок повністю видимий, і подальший пошук припиняється. Якщо тільки одна з точок лежить всередині вікна, то має місце випадок II, і належить знайти одну точку перетину. І, нарешті, якщо обидві точки лежать поза вікном, то існують або дві точки перетину (відрізок перетинає два кордони вікна), або ні однієї (відрізок повністю невидимий). Втім, дві точки перетину можуть збігатися (якщо відрізок проходить через вершину багатокутника), але цей випадок у додатковому аналізі не потребує.

Далі виконується цикл по всіх ребрах багатокутника з метою знаходження точок перетину. Для кожного ребра перед початком пошуку перетинання необхідно перевірити, чи не паралельно воно з отрезком.Еслі це так, то можна обчислити відстань від одного з кінців відрізка до прямої, що проходить через ребро $d = ((\overrightarrow{r}_1 - \overrightarrow{p}) \cdot \overrightarrow{n})$. При $d = 0$ відрізок лежить на прямій, і залишається визначити взаємне розташування кінців відрізка і кінців ребра, що можна зробити простим покоординатного порівнянням. При $d \neq 0$ відрізок не має спільних точок з даними ребром.

Кліпування багатокутників

Від завдання відсікання відрізків можна перейти до більш складною: кліпування довільних багатокутників. Якщо багатокутник неопуклі, то в результаті перетину навіть з прямокутним вікном може вийти трохи не пов'язаних між собою фігур, як це показано на рис. 5.8.

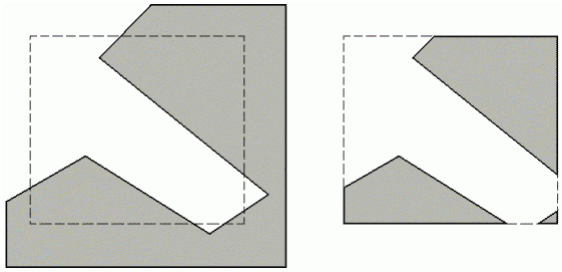

Рис. 5.8. Відсікання неопуклого багатокутника

Коли стоїть завдання штрихування замкнутої області одночасно із завданням відсікання, то важливо правильно визначити приналежність знову отриманих фігур внутрішньої або зовнішньої частини вихідного багатокутника. Нехай вихідний багатокутник заданий упорядкованим списком вершин $\{\overrightarrow{p}_1, \overrightarrow{p}_2, \ldots, \overrightarrow{p}_n$, відповідним ребрам $(\overrightarrow{p}_1, \overrightarrow{p}_2), (\overrightarrow{p}_2, \overrightarrow{p}_3), \ldots, (\overrightarrow{p}_{n-1}, \overrightarrow{p}_n), (\overrightarrow{p}_n, \overrightarrow{p}_1)$. Для відсікання такого багатокутника прямокутним вікном можна застосувати алгоритм, запропонований Сазерлендом і Ходжменом. Ідея його полягає в послідовному відсіканні частини багатокутника прямими, відповідними сторонам вікна. Результатом його роботи є упорядкований список вершин, що лежать у видимій частині вікна. На кожному кроці алгоритму утворюється деяка проміжна фігура, також представлена впорядкованим списком вершин і ребер. Приклад таких послідовних відсікань показаний на рис. 5.9. В процесі відсікання послідовно обходиться список вершин, причому кожна чергова крапка за винятком першої розглядається як кінцева точка ребра, початковою точкою якого є попередня точка з списка.Порядок, в якому розглядаються сторони вікна, не має значення. В процесі обходу формується список нових вершин багатокутника.

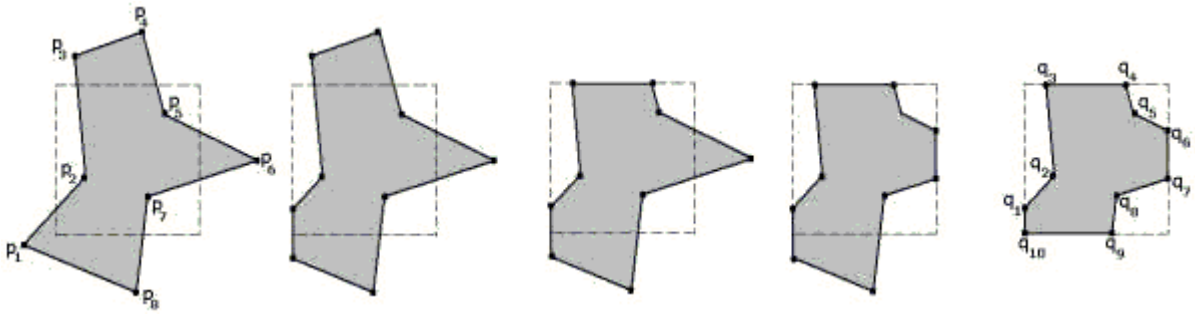

Рис. 5.9. Послідовні кроки кліпування довільного багатокутника

На першому кроці для першої вершини в списку визначається її приналежність видимої області. Якщо вона видима, то вона стає першою точкою першого оброблюваного ребра і заноситься в список нових вершин. Якщо ж вона невидима, то в список нових вершин не заноситься, але все одно стає першою точкою ребра.

Для аналізованого ребра можливі чотири випадки розташування щодо вікна.

1. Ребро повністю мабуть. Чергова точка заноситься в список нових вершин (попередня вже повинна знаходитися в цьому списку, оскільки ребро повністю мабуть).

2. Ребро повністю невидимо. Ніяких дій не проводиться.

3. Ребро виходить з області. Знаходиться точка перетину ребра зі стороною вікна і заноситься в список нових ребер.

4. Ребро входить в область. Також відшукується точка перетину із стороною вікна і заноситься в список нових вершин. Кінцева точка теж заноситься в список нових вершин.

У цьому алгоритмі постійно доводиться визначати видимість точки по відношенню до конкретного ребру відсікаючого вікна. Вікно також можна задати у вигляді впорядкованого списку вершин. Якщо обхід вершин вікна здійснюється за годинниковою стрілкою, то його внутрішня область буде розташована по праву сторону від кордону. При цьому розташування точки $\overrightarrow{p}$ відносно прямої, якій належить ребро, можна встановлювати різними способами:

1. Вибирається початкова точка $\overrightarrow{s}$ цього ребра і будується вектор $\overrightarrow{r} = \overrightarrow{p} - \overrightarrow{s}$ і вектор внутрішньої нормалі n до кордону (ребру) вікна. Обчислюється скалярний добуток $d = (\overrightarrow{r} \cdot \overrightarrow{n})$. Якщо $d \geq 0$, то точка $\overrightarrow{p}$ є видимою.

2. Будується просторовий вектор $\overrightarrow{r}$ (третю координату можна покласти рівною нулю). Вектор $\overrightarrow{l}$ (також просторовий, що і $\rightarrow$ лежить в тій же площині, що й) спрямований уздовж ребра (з урахуванням напрямку обходу). Обчислюється векторний добуток $\overrightarrow{v} = [\overrightarrow{r} \times \overrightarrow{l}]$. Якщо координата z у вектора $\overrightarrow{v}$ позитивна, то точка $\overrightarrow{p}$ лежить праворуч від ребра (є видимою).

3. Виписується канонічне рівняння прямої, що проходить через ребро: $f(x, y) + by + c = 0$

4. Для довільної внутрішньої точки (x_0, y_0) вікна обчислюється значення $d = f(x_0, y_0)$, а також для точки $\overrightarrow{p} = (x_1, y_1)$ обчислюється $d_1 = f(x_1, y_1)$. Якщо числа d і d_1 мають однаковий знак, то точка $\overrightarrow{p}$ є видимою.

Найбільш просто ці алгоритми реалізуються в разі відсікання прямокутним вікном зі сторонами, паралельними осям координат.

Питання та вправи

1. Що таке кліпування?

2. Якщо кінці відрізків мають коди 1000 і 0100, скільки сторін вікна він може перетинати?

3. При якому значенні коду одного з кінців відрізка він обов'язково буде частково видимим?

4. Якщо обидва кінці відрізка лежать поза вікном, то за яких кодах кінців він може проходити уздовж діагоналі вікна?

5. Який з алгоритмів відсікання відрізків ефективніше: наведений у блок-схемі 5,3 або заснований на поділі відрізка навпіл?

6. За допомогою якого умови можна визначити приналежність точки опуклому багатокутнику?

7. Чи буде це умова застосовується у разі довільного багатокутника? (Підтвердіть свою відповідь прикладами).

8. Які випадки розташування ребра щодо вікна розглядаються в алгоритмі кліпування довільного багатокутника

Лекція 6. Видалення невидимих поверхонь і ліній

Зміст

- Видалення нелицьових граней багатогранника
- Алгоритм Робертса
- Алгоритм Варнока
- Алгоритм Вейлера-Азертон
- Метод Z-буфера
- Методи пріоритетів (художника, що плаває обрію)
- Алгоритми рядковому сканування для криволінійних поверхонь
- Метод двійкового розбиття простору
- Метод трасування променів
- Питання та вправи

Завдання видалення невидимих ліній і поверхонь є однією з найбільш цікавих і складних в комп'ютерній графіці. Алгоритми видалення полягають у визначенні ліній ребер, поверхонь чи обсягів, які видимі чи невидимі для спостерігача, що знаходиться в заданій точці простору.
Необхідність видалення невидимих ліній, ребер, поверхонь чи обсягів проілюстрована на рис. 6.1. Малюнок наочно демонструє, що зображення без видалення невидимих ліній сприймається неоднозначно.

Рис. 6.1. Неоднозначність сприйняття зображення куба

Складність завдання видалення невидимих ліній і поверхонь призвела до появи великої кількості різних способів її рішення. Багато хто з них орієнтовані на спеціалізовані додатки. Єдиного (загального) рішення цього завдання, придатного для різних випадків, природно, не існує: для кожного випадку вибирається найбільш підходящий метод.Напрімер, для моделювання процесів у реальному часі потрібні швидкі алгоритми, в той час як для формування складного реалістичного зображення, в якому представлені тіні, прозорість і фактура, враховують ефекти віддзеркалення і заломлення кольору в дрібних відтінках, фактор часу виконання вже не так существенен.Подобние алгоритми працюють повільно, і часто на обчислення потрібно

кілька хвилин або навіть годин. Існує тісний взаємозв'язок між швидкістю роботи алгоритму і детальністю його результату. Жоден з алгоритмів не може досягти хороших оцінок для цих двох показників одночасно. У міру створення все більш швидких алгоритмів можна будувати все більш детальні ізображенія.Реальние завдання, однак, завжди вимагатимуть урахування ще більшої кількості деталей.

Всі алгоритми такого роду так чи інакше включають в себе сортування, причому головна сортування ведеться по геометричному відстані від тіла, поверхні, ребра або точки до точки спостереження або картинній площині. Основна ідея, покладена в основу сортування по відстані, полягає в тому, що чим далі розташований об'єкт від точки спостереження, тим більша ймовірність, що він буде повністю або частково закритий одним з об'єктів, більш близьких до точки наблюденія.После визначення відстаней або пріоритетів по глибині залишається провести сортування по горизонталі і по вертикалі, щоб з'ясувати, чи буде розглянутий об'єкт дійсно закритий об'єктом, розташованим ближче до точки спостереження. Ефективність будь-якого алгоритму видалення значною мірою залежить від ефективності процесу сортування.
Алгоритми видалення невидимих ліній або поверхонь можна класифікувати за способом вибору системи координат або простору, в якому вони працюють. Алгоритми, працюють у об'єктному просторі, мають справу зі світовою системою координат, у якій описані ці об'єкти. При цьому виходять дуже точні результати, обмежені, власне кажучи, лише погрішністю обчислень. Отримані зображення можна вільно масштабувати. Алгоритми, працюють у об'єктному просторі, особливо корисні в тих додатках, де необхідна висока точність. Алгоритми ж, що працюють у просторі зображення, мають справу з системою координат того екрана, на якому об'єкти візуалізуються. При цьому точність обчислень обмежена роздільну здатність екрана.
Ми наведемо деякі з алгоритмів, що працюють як в об'єктному просторі, так і в просторі зображення, кожен з яких ілюструє одну або кілька основних ідей теорії алгоритмів видалення невидимих ліній і поверхонь.
Видалення нелицьових граней багатогранника

Алгоритм Робертса

Цей алгоритм, запропонований в 1963 р., є першою розробкою такого роду й призначений для видалення невидимих ліній при штриховому зображенні об'єктів, складених з опуклих багатогранників. Він відноситься до алгоритмів, що працюють в об'єктному просторі, і дуже елегантний з математичної точки зору. У ньому дуже вдало поєднуються геометричні методи і методи лінійного програмування.
Опуклий багатогранник однозначно визначається набором площин, що утворюють його грані, тому вихідними даними для алгоритму є багатогранники, задані списком своїх граней. Грані задаються у вигляді площин, заданих в канонічній формі (див. лекцію 3) в об'єктній системі координат: $ax + by + cz + d = 0$

.

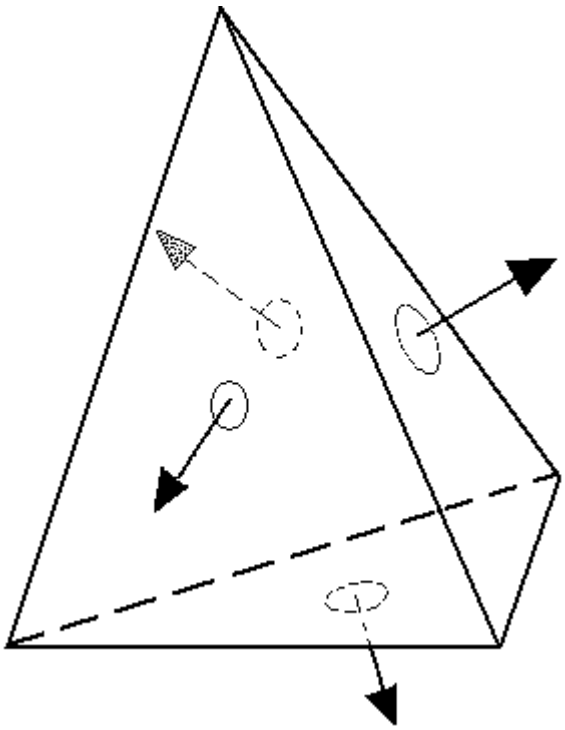

Рис. 6.2. Зовнішні нормалі тетраедра

Таким чином, кожна площина визначається чотиривимірним вектором $\overline{P}$, а кожна точка $\overrightarrow{r}$, задана в однорідних координатах, також являє собою чотиривимірний вектор:

$$\overrightarrow{P} = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}, \overrightarrow{r} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}.$$

Належність точки площини можна встановити за допомогою скалярного добутку, тобто якщо , то точка належить площині, якщо ж ні, то знак добутку показує, по яку сторону від площини ця точка знаходиться. В алгоритмі Робертса площини будуються таким чином, що внутрішні точки багатогранника лежать в позитивній напівплощині. Це означає, що вектор (A, B, C) є зовнішньою нормаллю до багатогранника (мал. 6.2). З векторів площин будується прямокутна матриця порядку $4 \times n$, яка називається узагальненою матрицею опису багатогранника:

$$M = \begin{pmatrix} a_1 & a_2 & a_3 & \dots & a_n \\ b_1 & b_2 & b_3 & \dots & b_n \\ c_1 & c_2 & c_3 & \dots & c_n \\ d_1 & d_2 & d_3 & \dots & d_n \end{pmatrix}.$$

Множачи стовпці матриці на вектор $\overrightarrow{r}$, одержимо n-мірний вектор, і якщо всі його компоненти невід'ємні, то точка належить багатогранника. Ця умова будемо записувати у вигляді $(\overrightarrow{r} \cdot M) \geq 0$ (мається на увазі множення вектор-рядка на матрицю).

У своєму алгоритмі Робертс розглядає тільки відрізки, що є перетином граней багатогранника.

З узагальненої матриці можна отримати інформацію про те, які грані багатогранника перетинаються у вершинах. Дійсно, якщо вершина $\overrightarrow{v} = (x, y, z, 1)$ належить граням

$\vec{P}_1,\ \vec{P}_2,\ \vec{P}_3$, то вона задовольняє рівнянням

$$\left.\begin{aligned}(\vec{v}\cdot\vec{P}_1)&=0\\(\vec{v}\cdot\vec{P}_2)&=0\\(\vec{v}\cdot\vec{P}_3)&=0\\(\vec{v}\cdot\vec{e}_4)&=1\end{aligned}\right\},\quad \text{где}\quad \vec{e}_4=\begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

Цю систему можна записати в матричному вигляді: $\vec{v}\cdot Q=\vec{e}_4$,

де Q - матриця, складена з вектор-стовпців $\vec{P}_1, \vec{P}_2, \vec{P}_3, \vec{e}_4$. Значить, координати вершини визначаються співвідношенням $\vec{v}=\vec{e}_4\cdot Q^{(-1)}$,

тобто вони складають останній рядок зворотної матриці. А це означає, що якщо для будь-яких трьох площин зворотна матриця існує, то площини мають спільну вершину.

Алгоритм насамперед видаляє з кожного багатогранника ті ребра чи грані, які екрануються самим тілом. Робертс використовував для цього простий тест: якщо одна або обидві суміжні грані звернені своєю зовнішньою поверхнею до спостерігача, то ребро є видимим. Тест цей виконується обчисленням скалярного добутку координат спостерігача на вектор зовнішньої нормалі грані: якщо результат негативний, то грань видима.

Потім кожне з видимих ребер кожного багатогранника порівнюється з кожним з решти многогранників для визначення того, яка його частина або частини, якщо такі є, екрануються цими тілами. Для цього в кожну точку ребра проводиться відрізок променя, що виходить з точки розташування спостерігача. Якщо відрізок не перетинає жодного з багатогранників, то точка видимою. Для вирішення цього завдання використовуються параметричні рівняння прямої, що містить ребро, і променя.

Якщо задані кінці відрізка $\vec{r}$ і $\vec{s}$, а спостерігач розташований в точці $\vec{u}$, то відрізок задається рівнянням

$$\vec{v}=\vec{r}+t\cdot(\vec{s}-\vec{r})\equiv\vec{r}+t\cdot\vec{d},\quad 0\le t\le 1,$$

а пряма, що йде в точку, відповідну параметру t, - рівнянням

$$\vec{w}=\vec{v}+\tau\cdot\vec{g}=\vec{r}+t\cdot\vec{d}+\tau\cdot\vec{g}.$$

Для визначення тієї частини відрізка, яка закривається яким-небудь тілом, досить знайти значення t і τ, при яких добуток вектора $\vec{w}$ на узагальнену матрицю позитивно. Для кожної площини $\vec{P}_i$ записується нерівність

$$q_i=(\vec{v}\cdot\vec{P}_i)+t(\vec{d}\cdot\vec{P}_i)+\tau(\vec{g}\cdot\vec{P}_i)>0.$$

Ці умови повинні виконуватися для всіх площин. Вважаючи $q_i=0$, отримуємо систему рівнянь, рішення якої дають нам крапки "зміни видимості" відрізка. Результат можна отримати шляхом спільного вирішення всіляких пар рівнянь з цієї системи. Число всіляких рішень при N площинах рівно $N\cdot(N-1)/2$.

Так як обсяг обчислень зростає із збільшенням числа багатокутників, то бажано у міру можливості скорочувати їх число, тобто якщо ми апроксимуємо деяку поверхню багатогранником, то як грані можна використовувати не трикутники, а більш складні багатокутники. При цьому, зрозуміло, постає проблема, як побудувати такий багатокутник, щоб він мало відхилявся від плоскої фігури.

Алгоритм Варнока

На відміну від алгоритму Робертса, Варнок в 1968 р. запропонував алгоритм, що працює не в об'єктному просторі, а в просторі образу. Він також націлений на зображення багатогранників, а головна ідея його заснована на гіпотезі про спосіб обробки інформації, що міститься в сцені, оком і мозком людини. Ця гіпотеза полягає в тому, що витрачається дуже мало часу і зусиль на обробку тих областей, які містять мало інформації. Більша частина часу і праці витрачається на області з високим інформаційним вмістом. Так, наприклад, розглядаючи приміщення, в якому є тільки картина на стіні, ми швидко оглядаємо стіни, підлогу і стелю, а потім вся увага зосереджуємо на картині. У свою чергу, на цій картині, якщо це портрет, ми побіжно відзначаємо фон, а потім більш уважно розглядаємо особа зображеного персонажа, особливо очі, губи.Как правило, досить детально розглядаються ще й руки і з трохи меншою увагою - одяг.

В алгоритмі Варнока і його варіантах робиться спроба скористатися тим, що великі області зображення однорідні. Така властивість називають когерентністю, маючи на увазі, що суміжні області (пиксели) уздовж обох осей x і y мають тенденцію до однорідності.

У просторі зображення розглядається вікно й вирішується питання про те, порожньо воно, чи його вміст досить просто для візуалізації. Якщо це не так, то вікно розбивається на фрагменти до тих пір, поки вміст фрагмента не стане досить простим для візуалізації або його розмір не досягне необхідної межі дозволу. В останньому випадку інформація, що міститься у вікні, усереднюється, і результат зображується з однаковою інтенсивністю або кольором.

Конкретна реалізація алгоритму Варнока залежить від методу розбиття вікна і від деталей критерію, використовуваного для того, щоб вирішити, чи є вміст вікна досить простим. В оригінальній версії алгоритму кожне вікно розбивалося на чотири однакових подокна.Многоугольнік, що входить в змальовувану сцену, по відношенню до вікна будемо називати (рис. 6.3)

- зовнішнім, якщо він цілком перебуває поза вікном;
- внутрішнім, якщо він цілком розташований усередині вікна;
- пересекающим, якщо він перетинає межу вікна;
- охоплює, якщо вікно цілком розташоване всередині нього.

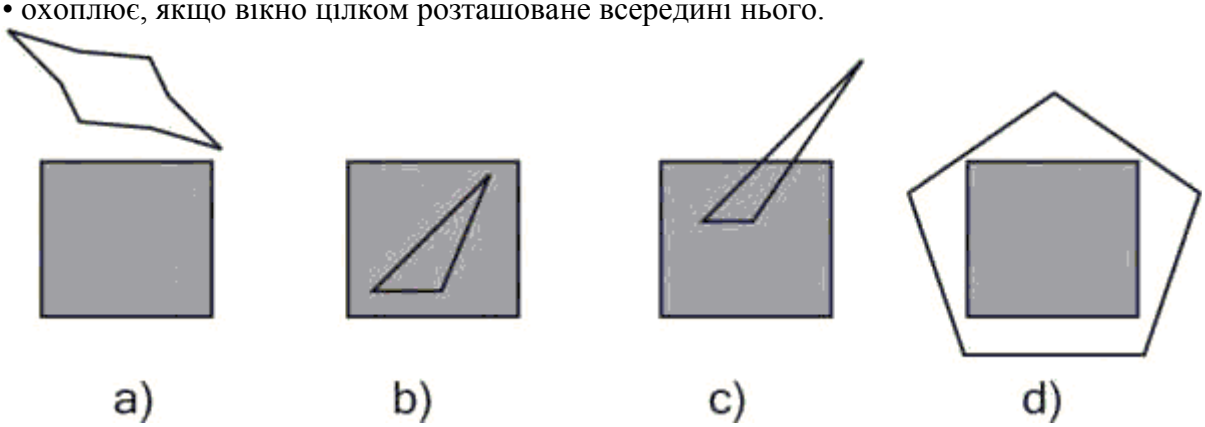

Рис. 6.3. Варіанти розташування багатокутника по відношенню до вікна

Тепер можна в самому загальному вигляді описати алгоритм.

Для кожного вікна:

1. Якщо все багатокутники сцени є зовнішніми по відношенню до вікна, то воно порожньо; зображується фоновим кольором і подальшого розбиттю не підлягає.

2. Якщо тільки один багатокутник сцени має загальні точки з вікном і є по відношенню до нього внутрішнім, то вікно заповнюється фоновим кольором, а сам багатокутник заповнюється своїм кольором.

3. Якщо тільки один багатокутник сцени має загальні точки з вікном і є по відношенню до нього пересекающим, то вікно заповнюється фоновим кольором, а частина багатокутника, що належить вікну, заповнюється кольором багатокутника.

4. Якщо тільки один багатокутник охоплює вікно й немає інших багатокутників, що мають загальні точки з вікном, то вікно заповнюється кольором цього багатокутника.

5. Якщо існує хоча б один багатокутник, що охоплює вікно, то серед всіх таких багатокутників вибирається той, який розташований ближче всіх багатокутників до точки спостереження, і вікно заповнюється кольором цього багатокутника.

6. В іншому випадку проводиться нове розбиття вікна.

Кроки 1-4 розглядають ситуацію перетинання вікна тільки з одним багатокутником. Вони використовуються для скорочення числа подразбіеній.Шаг 5 вирішує завдання видалення невидимих поверхонь. Багатокутник, що знаходиться ближче всіх до точки спостереження, екранує всі інші.

Для реалізації алгоритму необхідні функції, що визначають взаємне розташування вікна і багатокутника, які досить легко реалізуються в разі прямокутних вікон і опуклих многоугольніков.Для визначення, чи є багатокутник охоплює, зовнішнім або внутрішнім, можна скористатися, наприклад, зануренням багатокутника в прямокутну оболонку. Для визначення наявності пересічень можна використовувати опорні прямі (так само, як використовувалися площини в алгоритмі Робертса). Якщо ж багатокутник неопуклого, то завдання ускладнюється. Методи вирішення такого роду завдань будуть розглянуті в розділі, що відноситься до геометричного пошуку.

Слід зауважити, що існують різні реалізації алгоритму Варнока. Були запропоновані варіанти оптимізації, що використовують попереднє сортування багатокутників по глибині, тобто по відстані від точки спостереження, та інші.

Алгоритм Вейлера-Азертон

Вейлер і Азертон спробували оптимізувати алгоритм Варнока у відношенні числа виконуваних розбиттів, перейшовши від прямокутних розбиттів до розбиття вздовж кордонів багатокутників (1977). Для цього вони використовували ними же розроблений алгоритм відсікання багатокутників. Алгоритм працює в об'єктному просторі, і результатом його роботи є багатокутники. У найзагальнішому вигляді він складається з чотирьох кроків.

1. Попереднє сортування по глибині.

2. Відсікання по границі найближчого до точки спостереження багатокутника, називане сортуванням багатокутників на площині.

3. Видалення багатокутників, екраніруемого більш близькими до точки спостереження багатокутниками.

4. Якщо потрібно, то рекурсивне розбиття і нова сортування.

У процесі попереднього сортування створюється список приблизних пріоритетів, причому близькість багатокутника до точки спостереження визначається відстанню до найближчої до неї вершини. Потім виконується відсікання по найпершому з багатокутників. Відсіканню піддаються всі багатокутники зі списку, причому ця операція виконується над проекціями багатокутників на картинну площину. При цьому створюються списки зовнішніх і внутрішніх фігур. Всі потрапили в список зовнішніх не екрануються відтинає багатокутником. Потім розглядається список внутрішніх багатокутників і виконується сортування по відстані до відсікаючого багатокутника. Якщо всі вершини деякого багатокутника виявляються далі від спостерігача, ніж сама

віддалена з вершин екрануючого, то вони невидимі, і тоді вони віддаляються. Після цього робота алгоритму триває із зовнішнім списком.

Якщо якась із вершин внутрішнього багатокутника виявляється ближче до спостерігача, ніж найближча з вершин екрануючого багатокутника, то такий багатокутник є частково видимим. У цьому випадку попередній список пріоритетів некоректний, і тоді в якості нового відсікаючого багатокутника вибирається саме цей "порушник порядку". При цьому використовується саме вихідний багатокутник, а не той, що вийшов в результаті першого отсеченія.Такой підхід дозволяє мінімізувати число розбивок.

Цей алгоритм надалі був узагальнений Кетмул (1974) для зображення гладких Бікубіческая поверхонь. Його підхід полягав у тому, що розбиттю піддавалася поверхня. Коротко цей алгоритм можна описати так:

1. Рекурсивно розбивається поверхню до тих пір, поки проекція елемента на площину зображення не буде покривати не більше одного пікселя.

2. Визначити атрибути поверхні в цьому пікселі і зобразити його.

Ефективність такого методу, як і алгоритм Варнока, залежить від ефективності розбиттів. Надалі цей алгоритм був поширений на сплайнів поверхні.

Метод Z-буфера

Це один з найпростіших алгоритмів видалення невидимих поверхностей.Впервие він був запропонований Кетмул в 1975 р. Працює цей алгоритм у просторі зображення. Ідея Z-буфера є простим узагальненням ідеї про буфері кадру. Буфер кадру використовується для запам'ятовування атрибутів кожного пікселя в просторі зображення, а Z-буфер призначений для запам'ятовування глибини (відстані від картинної площини) кожного видимого пікселя в просторі ізображенія. Оскільки досить поширеним є використання координатної площини XOY у якості картинній площині, то глибина дорівнює координаті z точки , звідси й назва буфера. У процесі роботи значення глибини кожного нового пікселя, який потрібно занести в буфер кадру, порівнюється з глибиною того пікселя, який вже занесений в Z-буфер. Якщо це порівняння показує, що новий піксель розташований попереду пікселя, що знаходиться в буфері кадру, то новий піксель заноситься в цей буфер і, крім того, здійснюється коректування Z-буфера новим значенням глибини. Якщо ж порівняння дає протилежний результат, то ніяких дій не проводиться. По суті, алгоритм є пошуком по x й y найбільшого значення функції $z(x, y)$.

Головна перевага алгоритму - його простота. Крім того, цей алгоритм вирішує задачу про видалення невидимих поверхонь і робить тривіальної візуалізацію перетинань складних поверхонь. Сцени можуть бути будь-якої складності. Оскільки габарити простору зображення фіксовані, оцінка обчислювальної трудомісткості алгоритму не більше ніж лінійна. Оскільки елементи сцени чи картинки можна заносити в буфер кадру або в Z-буфер в довільному порядку, їх не потрібно попередньо сортувати по пріоритету глибини. Тому економиться обчислювальне час, що витрачається на сортування по глибині.

Основний недолік алгоритму - великий обсяг необхідної пам'яті. Останнім часом у зв'язку з швидким зростанням можливостей обчислювальної техніки цей недолік стає менш лімітуючим. Але в той час, коли алгоритм ще тільки з'явився, доводилося винаходити способи створення буфера якомога більшого обсягу при наявному ресурсі пам'яті.

Наприклад, можна розбивати простір зображення на 4, 16 або більше прямокутників або смуг. У граничному варіанті можна використовувати буфер розміром в один рядок розгортки. Для останнього випадку був розроблений **алгоритм рядковому сканування.** Оскільки кожен елемент сцени обробляється багато разів, то сегментування

Z-буфера, взагалі кажучи, призводить до збільшення часу, необхідного для обробки сцени.

Інший недолік алгоритму полягає в трудомісткості реалізації ефектів, пов'язаних з полупрозрачностью, і ряду інших спеціальних завдань, що підвищують реалістичність зображення. Оскільки алгоритм заносить пікселі в буфер кадру в довільному порядку, то досить складно отримати інформацію, яка необхідна для методів, заснованих на попередньому аналізі сцени.

У цілому алгоритм виглядає так:

1. Заповнити буфер кадру фоновим значенням кольору.
2. Заповнити Z-буфер мінімальним значенням z (глибини).
3. Перетворити зображувані об'єкти в растрову форму в довільному порядку.
4. Для кожного об'єкта:

4.1. Для кожного пікселя (x, y) образу обчислити його глибину $z(x, y)$.

4.2. Порівняти глибину $z(x, y)$ зі значенням глибини, що зберігається в Z-буфері в цій же позиції.

4.3. Якщо $z(x, y) > Z - \text{буфер}(x, y)$, то занести атрибути пікселя в буфер кадру і замінити $Z - \text{буфер}(x, y) \text{на} z(x, y)$. В іншому випадку ніяких дій не робити.

Алгоритм, що використовує Z-буфер, можна також застосовувати для побудови перетинів поверхонь. Зміниться тільки оператор порівняння:

$$z(x, y) > Z - \text{буфер}(x, y) \text{ и } z(x, y) = z \text{ сечения}$$

де z сечения - глибина шуканого перерізу.

Методи пріоритетів (художника, що плаває обрію)

Тут ми розглянемо групу методів, що враховують специфіку зображуваної сцени для видалення невидимих ліній і поверхонь.

При зображенні сцен із суцільним зафарбуванням поверхонь можна скористатися методом художника: елементи сцени зображуються в послідовності від найбільш віддалених від спостерігача до більш близьких. При екранування одних ділянок сцени іншими невидимі ділянки просто зафарбовуються. Якщо обчислювальна трудомісткість одержання зображення для окремих елементів досить висока, то такий алгоритм буде не найкращим по ефективності, але зате ми уникнемо аналізу (і цілком можливо, теж дорогого), що дозволяє встановити, які ж з елементів зображувати не треба в силу їх невидимості. Наприклад, при зображенні правильного багатогранника ми досить легко можемо впорядкувати його грані по глибині, але таке сортування для довільного багатогранника можлива далеко не всегда.Ми розглянемо застосування цього методу на прикладі зображення поверхні, заданої у вигляді однозначної функції двох змінних.

Нехай поверхня задана рівнянням

$$z = f(x, y), \quad a \le x \le b, \quad c \le y \le d.$$

В якості картинній площині виберемо площину XOY. В області завдання функції на осях координат побудуємо сітку вузлів:

$$a = x_0 < x_1 < \ldots < x_{n-1} = b, \quad c = y_0 < y_1 < \ldots < y_{m-1} < y_m = d.$$

Тоді $z_{ij} = f(x_i, y_i)$ являють собою набір "висот" для даної поверхні по відношенню до площини XOY. Поверхня будемо апроксимувати трикутниками з вершинами в точках $\overrightarrow{r}_{ij} = (x_i, y_i, z_{ij})$ так, що кожному прямокутнику сітки вузлів будуть відповідати два трикутники: $tr^1_{i\,j} = \{\overrightarrow{r}_{ij}, \overrightarrow{r}_{i+1\,j}, \overrightarrow{r}_{i+1\,j+1}\}$ і $tr^2_{i\,j} = \{\overrightarrow{r}_{ij}, \overrightarrow{r}_{i\,j+1}, \overrightarrow{r}_{i+1\,j+1}\}$. Для побудови наочного зображення поверхні повернемо її на деякий кут спочатку щодо осі OX, а потім щодо осі OY, причому

напрямок обертання виберемо таким чином, що точки, відповідні кутам координатної сітки, розташуються в наступному порядку по далекості від картинної площини: $\overrightarrow{r}_{nm}$, $\overrightarrow{r}_{n0}$, $\overrightarrow{r}_{0m}$, $\overrightarrow{r}_{00}$, тобто точка $\overrightarrow{r}_{nm}$ виявиться найбільш близької до картинної площини (і найбільш віддаленої від спостерігача). Передбачається, що спосіб зафарбовування трикутників вже определен.Тогда процес зображення поверхні можна коротко записати так:

Для $i = n, \ldots, 1$

Для $j = m, \ldots, 1$

Нарисовать $tr^1_{i\,j}$; нарисовать $tr^2_{i\,j}$.

При такій послідовності виводу зображення ми просуваємося від самого віддаленого трикутника до усе більш близьким, частково зафарбовуючи вже зображені ділянки поверхні.

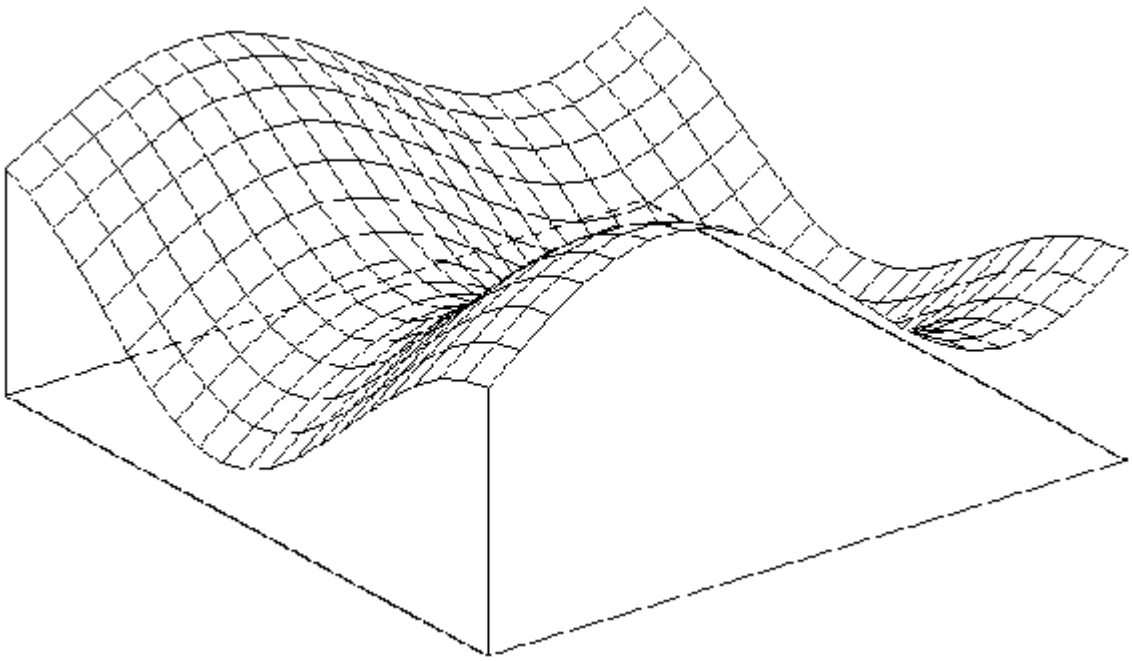

Рис. 6.4. Просте каркасне зображення з поверхні

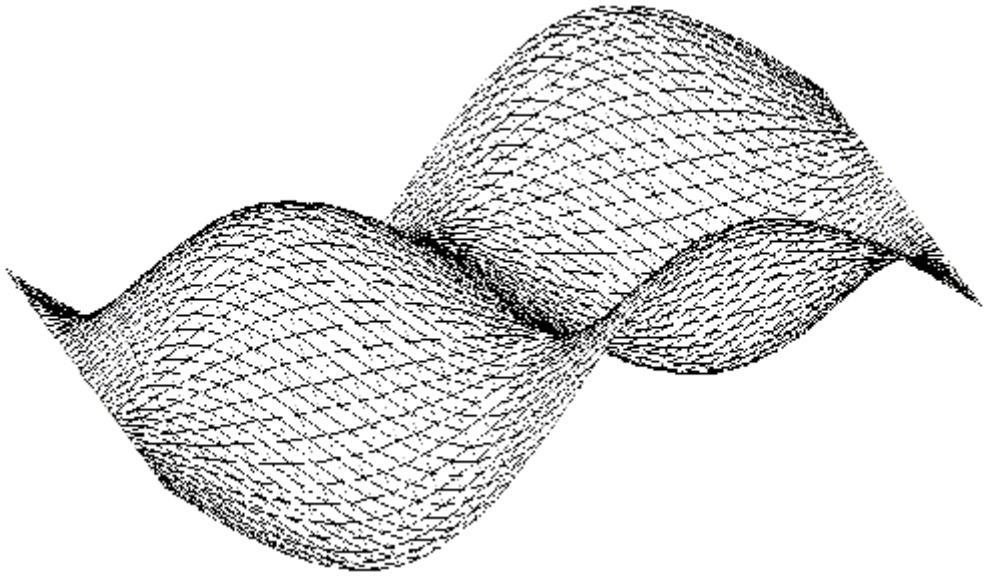

Рис. 6.5. Каркасне зображення діагональними ребрами

Алгоритм художника можна застосовувати для повністю зафарбованої сцени, а для каркасного зображення, коли об'єкт представляється у вигляді набору кривих або ламаних ліній, він непридатний. Для цього випадку запропонований ще один метод, досить ефективний - **метод плаваючого горизонту**. Повернемося до попереднього прикладу зображення поверхності.Каркасное зображення виходить шляхом зображення кривих, одержуваних при перетині цієї поверхні площинами $x = x_i$ і $y = y_i$ (рис. 6.4).

Насправді ми будемо малювати чотирикутник і одну діагональ. У процесі малювання нам знадобляться два цілочислових масиву: $LHor$ (нижній горизонт) і $HHor$ (верхній горизонт) розмірністю, відповідної горизонтальному розміру екрана в пікселях. Вони потрібні для аналізу видимості ділянок зображуваних відрізків. Спочатку ми ініціалізували верхній горизонт нулем, а нижній - максимальним значенням вертикальної координати на екрані. Кожна виводиться на екран крапка може закривати інші точки, які "ховаються за обрієм". У міру малювання нижній горизонт "опускається", а верхній "піднімається", поступово залишаючи дедалі менше незакритого простору. На відміну від методу художника, тут ми просуваємося від ближнього кута до дальнього. Тепер опишемо алгоритм докладніше.

Функція $segment$ в цьому фрагменті призначена для виводу на екран відрізка прямої, причому в момент ініціалізації чергового пикселя (i, j) вона виконує наступні дії:

$$\text{Если } (HHor[i] < j), \text{ то } HHor[i] = j; \text{ вывести пиксель;}$$
$$\text{Иначе если } (LHor[i] > j), \text{ то } LHor[i] = j; \text{ вывести пиксель.}$$

Таким чином, піксель виводиться тільки в тому випадку, якщо він вище верхнього або нижче нижнього горизонту, після чого його координати вже самі стають одним з обріїв. А в цілому алгоритм буде виглядати так:

$$\text{Для } i = 0, \ldots, n-1$$
$$\text{Для } j = 0, \ldots, m-1$$
$$segment(x_i, y_i, x_{i+1}, y_j); segment(x_i, y_i, x_{i+1}, y_{j+1});$$
$$segment(x_i, y_i, x_i, y_{j+1}).$$

На рис. 6.5 наведено приклад зображення поверхні з використанням цього алгоритму.

Алгоритми рядковому сканування для криволінійних поверхонь

Ідея рядковому сканування, запропонована в 1967 р. Уайлі, Ромни, Евансом і Ердалом, полягала в тому, що сцена обробляється в порядку проходження скануючої прямий. В об'єктному просторі це відповідає проведенню січної площини, перпендикулярної простору зображення. Строго кажучи, алгоритм працює саме в просторі зображення, відшукуючи точки перетину скануючої прямий з ребрами багатокутників, складових картину (для випадку зображення багатогранників). Але при перетині чергового елемента малюнка виконується аналіз глибини отриманої крапки й порівняння її з глибиною інших крапок на скануючої площини. У деяких випадках можна побудувати список ребер, упорядкований по глибині, але найчастіше це неможливо виконати коректно. Тому сканування поєднують з іншими методами.

Один з таких методів ми вже розглядали - метод Z-буфера, в якому буфер инициализируется заново для кожної скануючої строкі.Другой метод називають **інтервальним алгоритмом рядковому скануванні**. У ньому скануюча рядок розбивається проекціями точок перетину ребер багатокутників на інтервали, потім у кожному з інтервалів вибираються видимі відрізки. У цій ситуації їх уже можна

відсортувати по глибині. Ми зупинимося трохи докладніше на методі рядковому сканування для криволінійних поверхонь.

В описі методу пріоритетів поверхня задавалася у вигляді функції двох змінних, тут ми будемо задавати поверхню параметричними рівняннями:

$$x = x(u,v), \quad y = y(u,v), \quad z = z(u,v).$$

Перетин скануючої площини $y = y_1$ з поверхнею дає нам так звану **лінію рівня**, або **ізоліній**. Ця крива може бути неодносвязной, тобто складатися з декількох окремих кривих. Щоб отримати цю криву, ми повинні вирішити рівняння

$$y(u,v) = y_i$$

і, визначивши значення параметрів u, v, знайти точки кривої. Для отримання рішення можна скористатися чисельними ітераційними методами, але це вносить додаткові проблеми (наприклад, при поганому виборі початкового наближення ітераційний процес може не зійтися). Вибір відповідного методу рішення лежить поза завданнями нашого курсу, тому перейдемо до опису алгоритму, вважаючи, що він уже обраний і надійно працює.

Для каждой сканирующей строки со значением ординаты y_i :

Для каждого значения абсциссы x_j :

Для всех решений уравнений $u = u(x_j, y_j),\ v = v(x_j, y_j)$

вычислить глубину z=z(u,v).

Определить точку с наименьшим значением глубины

и изобразить.

Другий крок цього алгоритму припускає, що рішення відшукується тільки для тих елементів поверхні (або групи поверхонь, якщо мова йде про більш складної сцені, що містить складені об'єкти), які при даному значенні ординати перетинають скануючу плоскость.Предварітельний аналіз в деяких випадках дозволяє оптимізувати алгоритм.

Метод двійкового розбиття простору

Тепер розберемо один спосіб використання методу художника при зображенні просторових сцен, що містять кілька об'єктів або складові об'єкти. Це так званий **метод двійкового розбиття простору площинами**. Площині, як звичайно, будуть задаватися за допомогою вектора нормалі $\vec{n}$ і відстані до початку координат d (з точністю до знака). Нехай зображувана сцена складається з набору непересічних гране $F_1, F_2, \ldots, F_n$ й (вони можуть мати спільні прямолінійні ділянки кордону). Проведемо площину P_1, розбиває весь простір на два півпростору, в одному з яких знаходиться наблюдатель.Предположім, що площина при цьому не перетинає ні одну з граней (але може містити ділянку її кордону). Тоді грані, що знаходяться в одному півпросторі зі спостерігачем, можуть заслоняти від нього частина граней із другого півпростору, але не навпаки. Це означає, що вони повинні зображуватися пізніше. Розіб'ємо площиною P_2 другий півпростір і знову визначимо, яка група граней з нього повинна зображуватися раніше. Продовжуючи цей процес до того рівня, коли весь простір буде розбито площинами на секції, в кожній з яких буде знаходитися тільки одна грань, ми отримаємо впорядкований набір граней. Цей порядок можна зобразити у вигляді двійкового дерева. У контексті розглянутого алгоритму це дерево являє собою структуру даних T, елементами якої є покажчик на грань зображуваної сцени, площина, що відокремлює цю грань, покажчики на ліве і праве піддерево TL і TR. Такий елемент називається вузлом дерева.

У кожному вузлі дерева ліве піддерево буде містити грані, відділені площиною, а праве - не відокремлені. Малювання сцени здійснюється за допомогою рекурсивного алгоритму наступного виду:

Рисуем дерево (T) :

Если наблюдатель находится в положительной полуплоскости, то:

Если правое поддерево TR не пусто, рисуем дерево (TR).

Рисуем корневую грань.

Если левое поддерево TL не пусто, рисуем дерево (TL).

Иначе

Если левое поддерево TL не пусто, рисуем дерево (TL).

Рисуем корневую грань.

Если правое поддерево TR не пусто, рисуем дерево (TR).

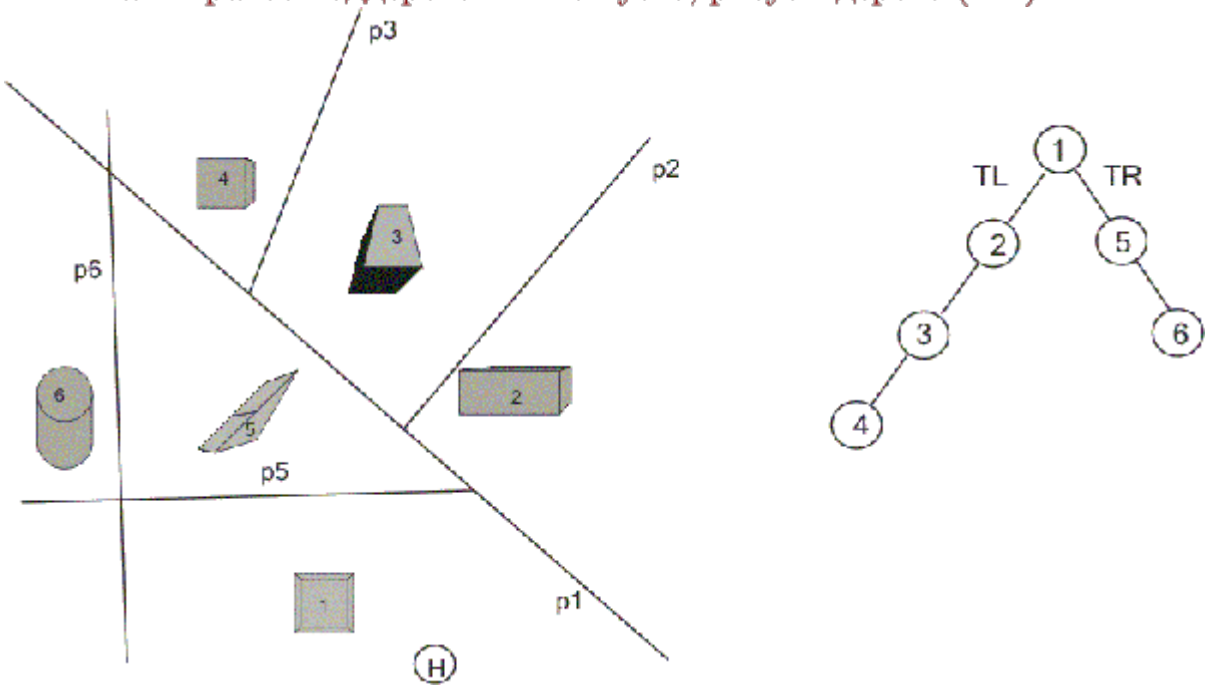

Рис. 6.6. Розбиття простору і відповідне йому дерево

Побудова площин і дерева в даному випадку здійснюється "вручну". Для ефективності роботи алгоритму треба прагнути до того, щоб дерево було збалансованим. Якщо якісь грані не вдається відділити, то їх перетинають площинами і малюють як два об'єкта.Способ визначення, по яку сторону площини перебуває спостерігач, а по яку - грань, дуже простий. Параметр площини d для кожної грані будемо задавати так, щоб грань перебувала в позитивній напівплощині. Тоді якщо при підстановці координат спостерігача в це рівняння одержуємо позитивне значення, то він знаходиться в одній півплощині з гранню, якщо ні, то в різних.

Алгоритм може застосовуватися не тільки до багатогранників, але і взагалі до будь сцені за умови, що є алгоритм зображення складових її об'єктів. На рис. 6.6 зображена проекція сцени, розбитої вертикальними площинами, і відповідне їй дерево.Положеніе спостерігача відзначено гуртком з буквою **Н**. При цій точці зору об'єкти будуть зображуватися в послідовності 5, 6, 1, 2, 3, 4.

Метод трасування променів

Головна ідея цього алгоритму була запропонована в 1968 р. А.Аппелем, а перша реалізація була виконана в 1971 р.

Спостерігач бачить будь-який об'єкт за допомогою випускається якимсь джерелом світла, який падає на цей об'єкт, відбивається або переломлюється згідно законам оптики і потім якимось шляхом доходить до ока спостерігача. З величезної безлічі променів світла, випущених джерелом, лише невелика частина дійде до спостерігача. Отже, відстежувати шляхи променів у такому порядку неефективно з точки зору обчислень. Аппель запропонував відслідковувати (трасувати) промені в зворотному напрямку, тобто від спостерігача до об'єкта. У першій реалізації цього методу трасування припинялася, як тільки промінь перетинав поверхню видимого непрозорого об'єкта; тобто промінь використовувався тільки для обробки прихованих або видимих поверхностей.Впоследствіі були реалізовані алгоритми трасування променів з використанням більш повних моделей висвітлення з урахуванням ефектів відбиття одного об'єкта від поверхні іншого, заломлення, прозорості й затінення.

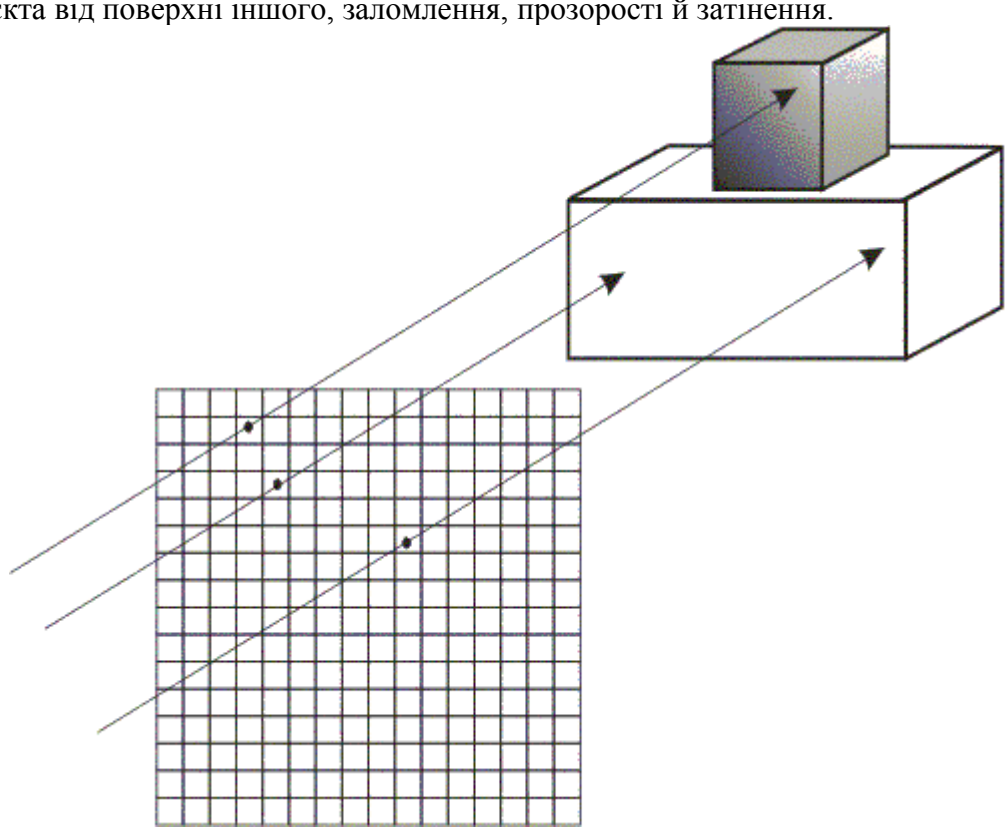

Рис. 6.7. Трасування паралельними променями

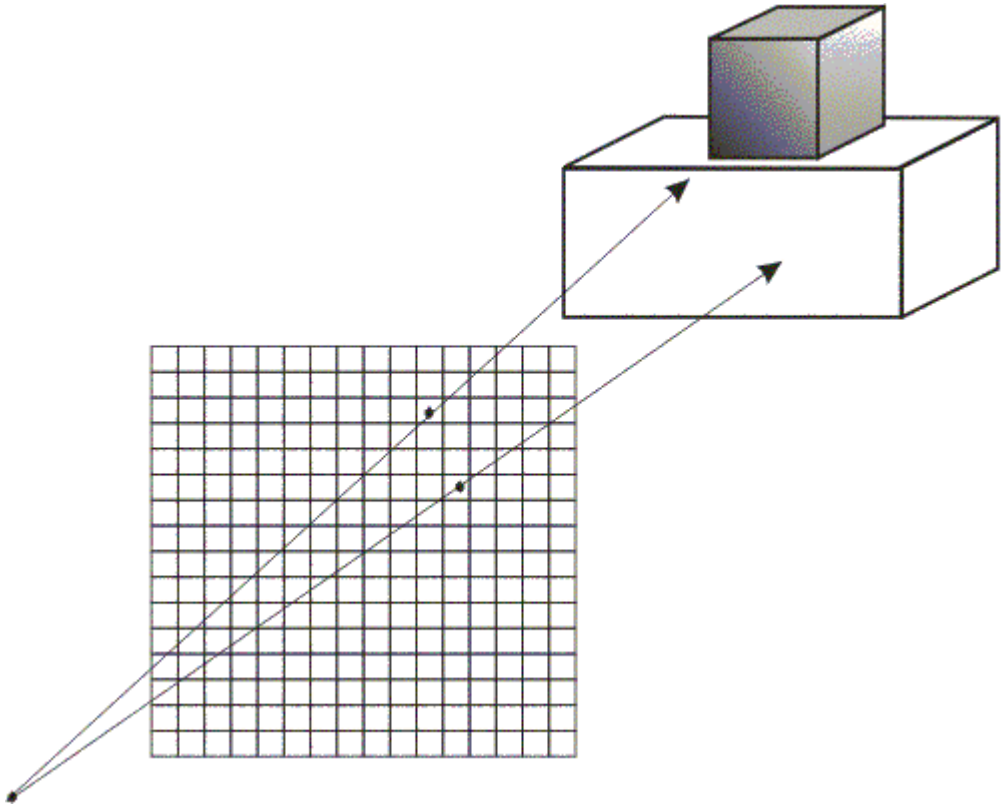

Рис. 6.8. Трасування із центральною точкою

У цьому алгоритмі передбачається, що сцена вже перетворена в простір зображення. Якщо використовується ортографической проекції, то точка зору або спостерігач перебуває в нескінченності на позитивній півосі OZ. У цьому випадку всі світлові промені, що йдуть від спостерігача, паралельні осі (рис. 6.7). Кожен промінь проходить через піксель растра до сцени. Траєкторія кожного променя відстежується, щоб визначити, які саме об'єкти сцени, якщо такі існують, перетинаються з даним променем. Необхідно перевірити те кожного об'єкта сцени з кожним променем. Якщо промінь перетинає об'єкт, то визначаються всі можливі точки перетину променя і об'єкта. Можна отримати велику кількість перетинань, якщо розглядати багато об'єктів. Ці перетину упорядковуються по глибині. Перетин з максимальним значенням z представляє видиму поверхню для даного пикселя. Атрибути цього об'єкта використовуються для визначення характеристик пикселя.

Якщо точка зору знаходиться не в нескінченності (перспективна проекція), алгоритм трасування променів лише незначно усложняется.Здесь передбачається, що спостерігач як і раніше перебуває на позитивній півосі OZ. Картинна площину, тобто растр, перпендикулярна осі OZ, як показано на рис. 6.8.

Найбільш важливим і трудомістким елементом цього алгоритму є процедура визначення перетинань, оскільки це завдання віднімає найбільшу частину часу всієї роботи алгоритму. Тому ефективність методів пошуку особливо важлива. Об'єкти сцени можуть складатися з набору плоских багатокутників, багатогранників або тіл, обмежених замкнутими параметричними поверхнями. Для прискорення пошуку важливо мати ефективні критерії, що дозволяють виключити з процесу свідомо зайві об'єкти.

Одним із способів скорочення числа перетинаються об'єктів є занурення об'єктів в опуклу оболонку - сферичну або у формі паралелепіпеда. Пошук перетину з такою оболонкою дуже простий, і якщо промінь не перетинає оболонки, то не потрібно більше шукати перетинань цього об'єкта з променем.

Особливо просто виконується тест на перетинання зі сферичною оболонкою (в **лекції 3** були розглянуті завдання про перетин променя зі сферою і площиною). Кілька

більшого обсягу обчислень вимагає завдання про перетинання із прямокутним параллелепипедом, оскільки необхідно перевірити те променя щонайменше з трьома нескінченними площинами, обмежують прямокутну оболочку.Поскольку точки перетину можуть опинитися поза граней цього паралелепіпеда, для кожної з них слід, крім того, зробити перевірку на потрапляння всередину. Отже, для трьох вимірів дослідження з прямокутної оболонкою виявляється повільнішим, ніж тест зі сферичної оболонкою.

Після виконання цих первинних тестів починається процес пошуку перетинань із об'єктами, що потрапили в список потенційно видимих. При цьому завдання формування зображення не вичерпується знаходженням самої точки перетину: якщо ми враховуємо ефекти відбиття і заломлення, необхідно відслідковувати подальший шлях променя, для чого, як правило, потрібно відновити нормаль до поверхні, а також визначити напрямок відбитого або переломленого променя. У зв'язку з усіма цими завданнями важливо вибрати досить зручні апроксимації поверхонь, складових сцену. Визначення атрибутів пікселя, виведеного в остаточному підсумку на екран, залежить від вибору моделі висвітлення, про що більш докладно буде розказано в наступних лекціях.

Алгоритм трасування променів для простих непрозорих поверхонь можна представити таким чином.

Створити список об'єктів, що містить:

• повний опис об'єкта: тип, поверхня, характеристики, тип оболонки і т.п.;

• опис оболонки: центр і радіус для сфери або шість значень для паралелепіпеда $(x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max})$.

Для кожного трассируемого променя:

Виконати для кожного об'єкта тривимірний тест на перетинання з оболонкою. Якщо промінь перетинає цю оболонку, то занести об'єкт у списку активних об'єктів.

Якщо список активних об'єктів порожній, то зобразити даний піксел з фоновим значенням кольору і продовжувати роботу. В іншому випадку для кожного об'єкта зі списку активних об'єктів:

• Знайти перетину з усіма активними об'єктами.

• Якщо список перетинань порожній, то зобразити даний піксел з фоновим значенням кольору.

• В іншому випадку в списку припинень знайти найближче до спостерігача (з максимальним значенням z) і визначити атрибути крапки.

• Зобразити даний пиксель, використовуючи знайдені атрибути пересеченного об'єкта і відповідну модель освітленості.

В даний час алгоритм трасування, незважаючи на обчислювальну складність, став дуже популярний, особливо в тих випадках, коли час формування зображення не дуже істотно, але хочеться домогтися як можна більшої реалістичності зображення.

Питання та вправи

• У чому полягає суть видалення невидимих ліній і поверхонь?

• У якому просторі працює алгоритм Робертса?

• Для яких об'єктів приміряється алгоритм Робертса?

• Що являє собою вектор-стовпець узагальненої матриці опису багатогранника?

• Як інтерпретується вираження $(\vec{r} \cdot M) \geq 0$ (M - узагальнена матриця) в алгоритмі Робертса?

• У якому просторі працює алгоритм Варнок?

• Які типи розташування багатокутника щодо вікна розглядаються в алгоритмі Варнок?

• Який із шести кроків алгоритму вирішує завдання про видалення невидимих поверхонь?

• У якому просторі працює алгоритм Вейлера-Азертон?

• У чому принципова відмінність алгоритму Вейлера-Азертон від алгоритму Варнок?

• Яке узагальнення алгоритму Вейлера-Азертон запропонував Кетмул?

• Ким запропонований алгоритм Z-буфера?

• У чому недоліки алгоритму Z-буфера?

• На чому засновані методи пріоритетів?

• Для якого виду зображення розроблений метод художника?

• Для якого виду зображення розроблений метод плаваючого горизонту?

• Що спільного між алгоритмом порядкового сканування і методом Z-буфера?

• В чому полягає ідея методу трасування?

• Які бувають види трасування?

• Які прийоми використовуються для підвищення ефективності алгоритму трасування?

Лекція7. Проектування просторових сцен

Зміст

Основні типи проекцій

- Паралельні проекції
- Центральні проекції
- Математичний апарат
- Ортогональні проекції
- Косокутні проекції
- Центральні проекції
- Спеціальні картографічні проекції. Екзотичні проекції земної сфери
- стереографічній проекції
- Гномоніческая проекція
- ортографической проекції
- Проекції на циліндр
- Проекція Меркатора
- Проекції на багатогранник
- Незвичайні проекції
- Питання та вправи

Основні типи проекцій

У математичному сенсі проекції - це перетворення крапок простору розмірності n в точки простору розмірності меншої, ніж n, або, як ще кажуть, на підпростір вихідного простору. У комп'ютерній графіці розглядаються переважно проекції тривимірного простору образа на двовимірну картинну площину. Проекція тривимірного об'єкта, представленого у вигляді сукупності точок, будується за допомогою прямих проектують променів, які називаються проекторами і які виходять з центру проекції, проходять через кожну точку об'єкта і, перетинаючи картинну площину, утворюють проекцію.

Визначений таким чином клас проекцій називають плоскими геометричними проекціями, оскільки проектування в цьому випадку проводиться на проекційну площину і в якості проекторів використовуються прямі. Існують і інші проекції, в яких проектування здійснюється на криволінійні поверхні або ж проектування здійснюється не за допомогою прямих (такі проекції використовуються, наприклад, у картографії).

Слід зазначити, що, приводячи ілюстрації до даної глави, ми змушені використовувати ті ж самі проекції, методи побудови яких збираємося описати. Хочеться сподіватися, що матеріал через це не буде виглядати більш неясним, ніж при відсутності малюнків.

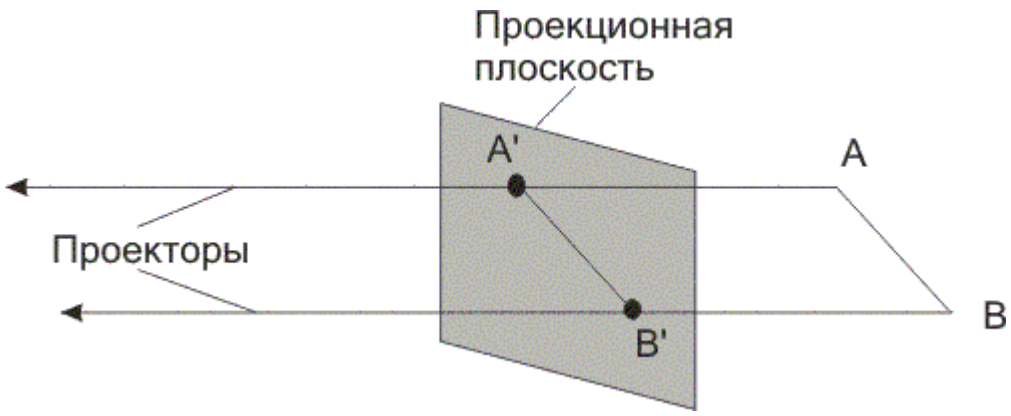

Рис. 7.1. Проекція паралельним пучком променів

Рис. 7.2. Центральна проекція

Плоскі геометричні проекції підрозділяються на два основних класи: центральні та паралельні. Різниця між ними визначається співвідношенням між центром проекції і проекційною площиною. Якщо відстань між ними звичайно, то проекція буде центральною, якщо ж воно нескінченно, то проекція буде паралельною. Паралельні проекції названі так тому, що центр проекції нескінченно віддалений і все проектори паралельні. При описі центральної проекції ми явно задаємо її центр проекції в той час як, визначаючи паралельну проекцію, ми вказуємо напрям проектування. На рис. 7.1 і 7.2показани дві різні проекції одного і того ж відрізка, а також проектори, що проходять через його кінцеві точки. Оскільки проекція відрізка сама є відрізком, то досить спроектувати одні лише кінцеві точки і з'єднати їх.

Центральна проекція породжує візуальний ефект, аналогічний тому, до якого приводять фотографічні системи або зорова система людини, і тому використовується у випадках, коли бажано досягти певного ступеня реалістичності. Цей ефект називаетсяперспективним укорочуванням: у міру збільшення відстані від центру до об'єкта розмір одержуваної проекції зменшується. Це, з іншого боку, означає, що хоча центральна проекція об'єктів є реалістичною, вона виявляється непридатною для подання точної форми та розмірів об'єктів: з проекції можна отримати інформацію про відносних відстанях; кути зберігаються тільки на тих гранях об'єкта, які паралельні проекційної площини; проекції паралельних ліній в загальному випадку не паралельні. Так, при центральній проекції куба в більшості випадків ми одержуємо картину, взагалі не має паралельних відрізків.

Паралельна проекція породжує менш реалістичне зображення, оскільки відсутнє перспективне укорочування, хоча при цьому можуть мати місце різні постійні укорочування уздовж кожної з осей.Проекція фіксує справжні розміри (з точністю до скалярного множника), і паралельні прямі залишаються паралельними. Як і у випадку

центральної проекції, кути зберігаються тільки на тих гранях об'єкта, які паралельні проекційної площини.

Паралельні проекції

Паралельні проекції розділяються на два типи в залежності від співвідношення між напрямком проектування та нормаллю до проекційної площини. Якщо ці напрямки збігаються, тобто напрям проектування є нормаллю до проекційної площини, то проекція називається ортографической. Якщо ж проектори не ортогональні до проекційної площини, то проекція називається косокутних.

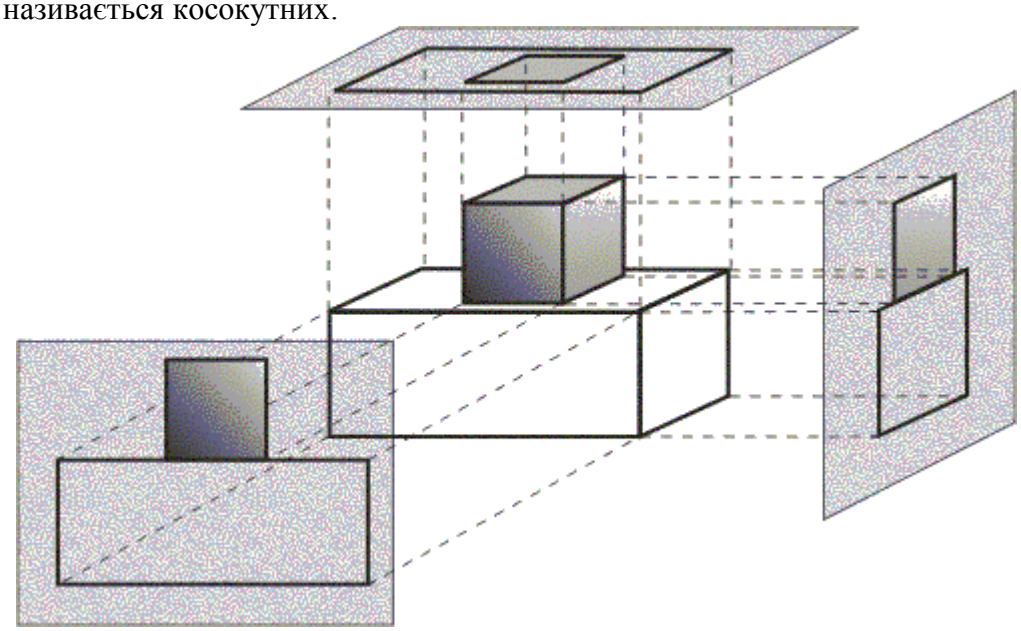

Рис. 7.3. Ортографічні проекції

В інженерній графіці найбільш широко використовуваними видами ортографической проекції є вид спереду, вид зверху (план) і вид збоку, в яких проекційна площина перпендикулярна головним координатним осям, співпадаючим внаслідок цього з напрямком проектування (рис. 7.3). Оскільки кожна проекція відображає лише одну сторону об'єкта, часто зовсім непросто уявити собі просторову структуру проектованого об'єкта, навіть якщо розглядати відразу кілька проекцій одного і того ж об'єкта. Але тим не менш такі креслення дозволяють визначати реальні розміри об'єкта.

У разі аксонометричних ортографической проекції використовуються проекційні площини, не перпендикулярні головним координатним осям, тому на них зображується відразу кілька сторін об'єкта, так само як і при центральному проектуванні, однак в аксонометрії укорочування постійно, тоді як у випадку центральної проекції воно пов'язане з відстанню від центру проекції. При аксонометричному проектуванні зберігається паралельність прямих, а кути змінюються; відстані же можна виміряти уздовж кожної з головних координатних осей (в загальному випадку з різними масштабними коефіцієнтами).

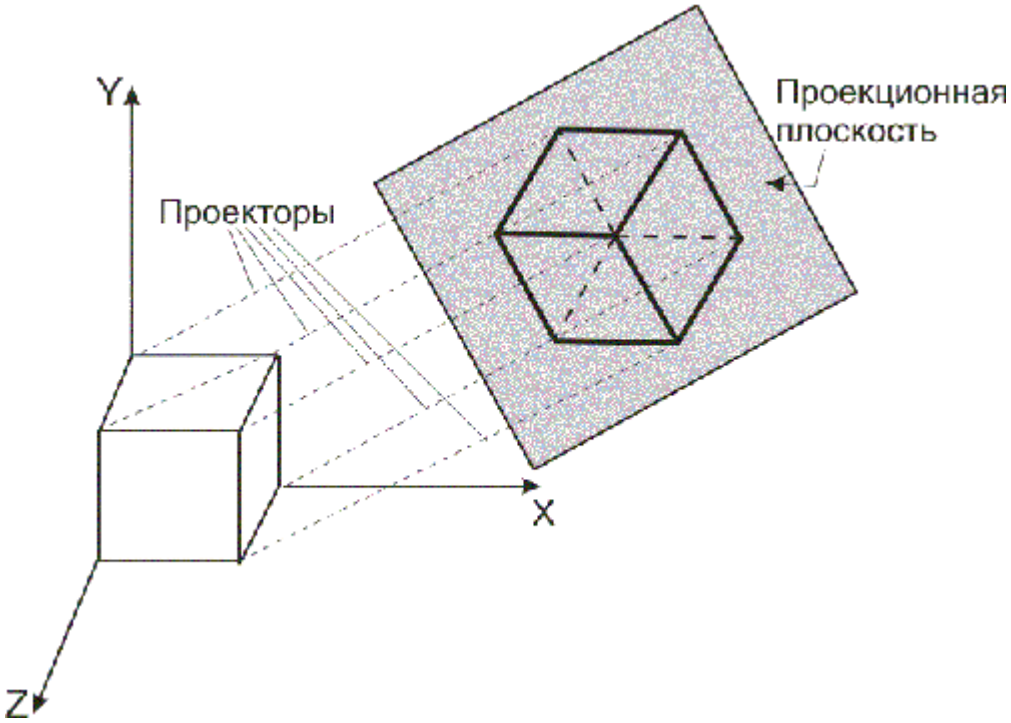

Рис. 7.4. Ізометрична проекція

Аксонометричні проекції підрозділяються на три групи відповідно до розташування проекційної площини по відношенню до осей координат. Якщо нормаль до проекційної площини утворить три різних кута з осями, то проекція називається тріметріческой (триметр). Якщо два з цих кутів однакові, то получаемдіметріческую проекцію (диметр). І, нарешті, якщо всі три кути рівні між собою, то проекція називається ізометричної (ізометрією). Ізометрична проекція володіє тим властивістю, що всі три головні координатні осі однаково коротшають. Тому можна проводити вимірювання вздовж напрямку осей з одним і тим же масштабом (звідси назва: "ізо", що означає "одно", і "метрія" - "вимір"). Крім того, головні координатні осі проектуються так, що їх проекції складають рівні кути один з одним (рис. 7.4).

Косокутні проекції також є паралельними, причому проекційна площина перпендикулярна головної координатної осі.Сторона об'єкта, паралельна цій площині, проектується так, що можна вимірювати кути і відстані. Проекція інших сторін об'єкта також допускає проведення лінійних вимірювань (але не кутових) уздовж головних осей. Ми торкнемося тільки двох найбільш часто використовуваних косокутних проекцій: проекції Кавальє (cavalier) і кабіні (cabinet). У вітчизняній практиці ці проекції називають горизонтальною косокутній ізометрією і кабінетної проекцією.

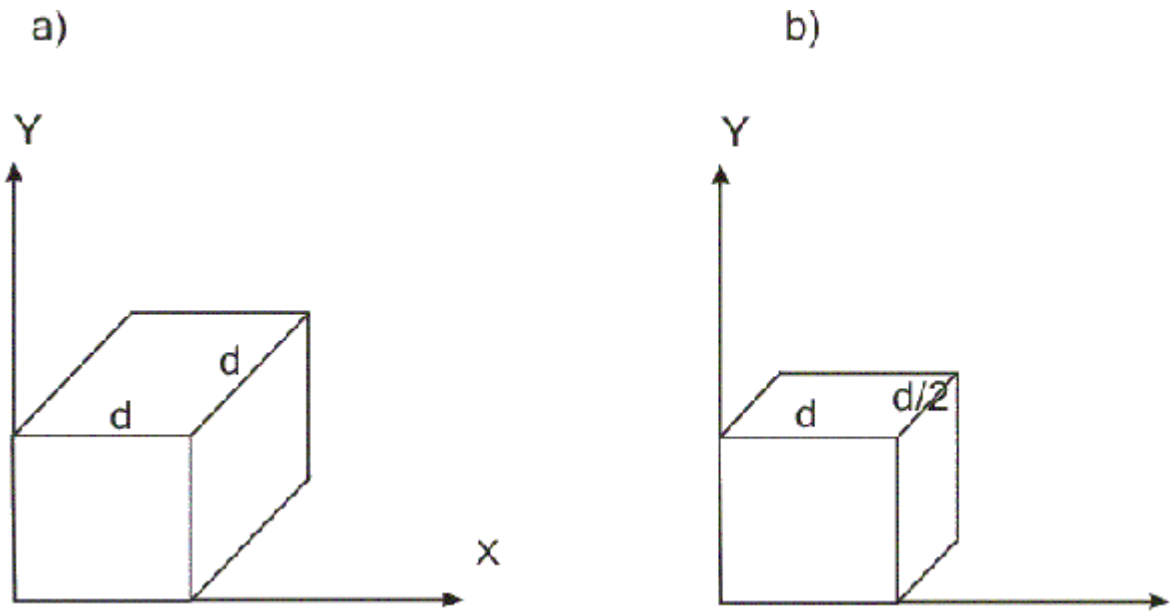

Рис. 7.5. Горизонтальна Косокутна ізометрія (а) і кабінетна проекція (б)

У проекції горизонтальної косокутній ізометрії напрям проектування складає з площиною кут 45 °. В результаті проекція відрізка, перпендикулярного проекційної площини, має ту ж довжину, що й сам відрізок, тобто укорочення відсутня (рис. 7.5а). Кабінетна проекція має напрямок проеціювання, яке складає з

проекційною площиною кут , при цьому відрізки, перпендикулярні проекційної площини, після проектування становлять 1/2 їх дійсної довжини, що більш відповідає нашому візуальному досвіду, тому зображення виглядає більш реалістично (мал. 7.5b).

Центральні проекції

Коли пучок проекторів виходить із заданого центра проекції, то паралельні відрізки на площині проекції вже не будуть паралельними, за винятком випадку, коли вони лежать у площині, паралельній проекційної. При проектуванні декількох паралельних прямих їх проекції перетинаються в так званій точці сходу. Якщо сукупність прямих паралельна однієї з координатних осей, то їх точка сходу називається головної. Таких точок може бути не більше трьох. Наприклад, якщо проекційна площина перпендикулярна осі OZ, то лише на цій осі буде лежати головна точка сходу, оскільки прямі, паралельні як осі OX, так і OY, паралельні також і проекційної площини і тому не мають крапки сходу.
Центральні проекції класифікуються залежно від числа головних точок сходу, якими вони володіють, а отже, і від числа координатних осей, які перетинає проекційна площина. На рис. 7.6 наведені три різні одноточкові проекції куба, причому дві з них мають одну точку сходу, а третя - дві точки.

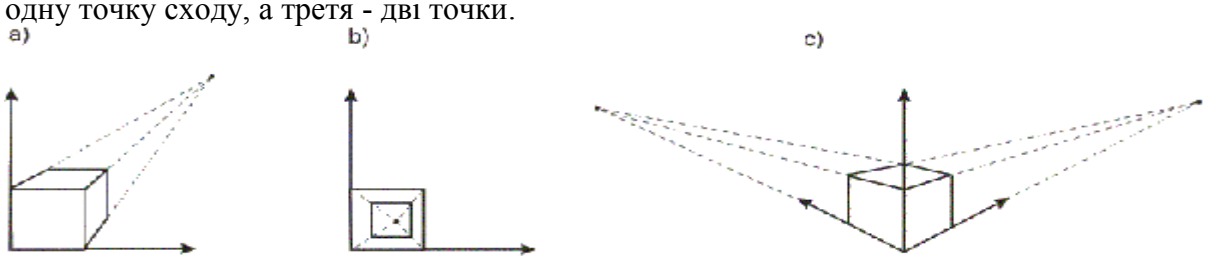

Рис. 7.6. Одноточкові проекції куба.

Двоточкова центральна проекція широко застосовується в архітектурному, інженерному та промисловому проектуванні та в рекламних зображеннях, в яких

вертикальні прямі проектуються як паралельні і, отже, не сходяться. Триточкові центральні проекції майже зовсім не використовуються, по-перше, тому, що їх важко конструювати, а по-друге, через те, що вони додають мало нового з точки зору реалістичності в порівнянні з двоточковою проекцією.

Математичний апарат

Для виконання проектних перетворень будемо використовувати однорідні координати і матриці перетворень, розглянуті раніше влекції 4. Проекція виконується в системі координат спостерігача.

Ортогональні проекції

Спочатку розглянемо математичний опис паралельних проекцій як більш простих. Випадок, коли картинна площина перпендикулярна осі OZ і задається рівнянням $z = 0$ (Тобто ортографической проекції), фактично вже розглядався в лекції 4, де був приведений вид матриць проекції на координатні площини.

Випадок аксонометричній проекції зводиться до послідовності перетворень, подібно до того як здійснювався поворот у просторі щодо довільної осі. Нехай площина задається одиничним вектором нормалі $\vec{n} = (n_x, n_y, n_z)$ і відстанню від початку координат $d \geq 0$

Канонічне рівняння площини, таким чином, має вигляд

$$n_x \cdot x + n_y \cdot y + n_z \cdot z - d = 0.$$

Вектор, спрямований по нормалі від початку координат до перетину з площиною, є $\vec{N} = d \cdot \vec{n} = (n_x d, n_y d, n_z d) = (N_x, N_y, N_z).$

Координати вектора одиничної нормалі є її напрямними косинусами. Проектування в просторі однорідних координат здійснюється наступною послідовністю кроків.

• Зміщення на вектор $-\vec{N}$ за допомогою матриці

$$S_1 = \begin{pmatrix} 1 & 0 & 0 & -N_x \\ 0 & 1 & 0 & -N_y \\ 0 & 0 & 1 & -N_z \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Поворот, що суміщає напрямок нормалі з напрямком осі OZ. Як було показано в лекції 4, цей поворот можна реалізувати у вигляді двох поворотів: перший (щодо осі OZ) переводить нормаль у площину YOZ, а потім - поворот щодо осі OY до суміщення нормалі з віссю OZ. Відповідну матрицю обертання, що є добутком двох матриць, позначимо R.

• Проекція на площину XOY за допомогою матриці

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

• Поворот за допомогою матриці R^{-1}.

• Зміщення на вектор за допомогою матриці S_1^{-1}

Повне перетворення, таким чином, визначається матрицею

$$\Pr = S_1^{-1} \cdot R^{-1} \cdot P_{xy} \cdot R \cdot S_1.$$

Косокутні проекції

Розглянемо косокутну проекцію на площину XOY, при якій орт $\overrightarrow{e}_3 = (0,0,1)$ переходить у вектор $\overrightarrow{r}_0 = (a,b,0)$, тобто напрямок проекції задається вектором $\overrightarrow{p} = \overrightarrow{r}_0 - \overrightarrow{e}_3 = (a,b,-1)$.

Таке перетворення в просторі однорідних координат можна задати за допомогою матриці

$$P = \begin{pmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

У проекції Кавальє вектор $\overrightarrow{e}_3$ переходить у вектор $(\cos(\pi/4), \cos(\pi/4), 0)$, а в кабінетній проекції - у вектор $(0.5 \cdot \cos(\pi/4), 0.5\cos(\pi/4), 0)$, причому в обох проекціях $a = b$.

Центральні проекції

Припустимо, що центр проекції перебуває в точці $\overrightarrow{c} = (c_x, c_y, c_z)$, а картинна площина збігається з площиною XOY. Візьмемо довільну точку зображуваного об'єкта $\overrightarrow{M} = (x,y,z)$ і визначимо її проекцію на обрану площину (рис. 7.7).

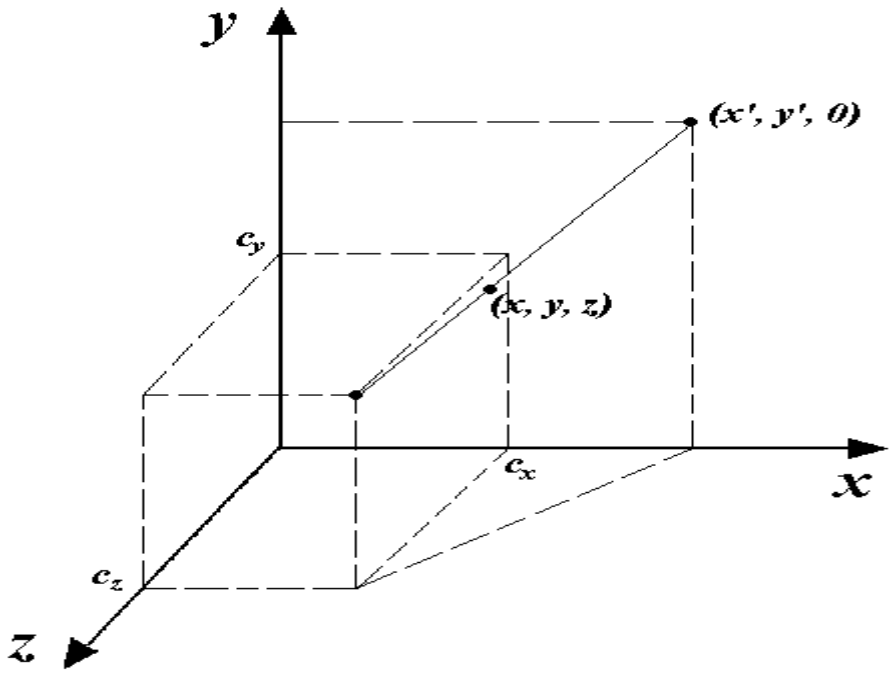

Рис. 7.7. Центральна проекція на площину XOY

Пряму, що проходить через точки $\overrightarrow{c}$ і $\overrightarrow{M}$, задамо в параметричному вигляді:

$$\overrightarrow{r}(t) = \overrightarrow{c} + t \cdot (\overrightarrow{M} - \overrightarrow{c}) = (c_x + t \cdot (x - c_x), c_y + t \cdot (y - c_y) + t \cdot (z - c_z)). \quad (7.1)$$

Тепер знайдемо точку перетину цієї прямої з картинної плоскостью.Она визначається з умови рівності нулю третьої координати:

$$c_z + t \cdot (z - c_z) = 0,$$

звідки визначаємо значення параметра t, при якому точка прямої належить координатній площині:

$$t^* = \frac{c_z}{c_z - z} = \frac{1}{1 - \frac{z}{c_z}}.$$

Підставляючи це значення в формулу (7.1), ми отримаємо координати проекції точки $\overrightarrow{M}$:

$$x^* = c_x + \frac{x - c_x}{1 - \frac{z}{c_z}}, \quad y^* = c_y + \frac{y - c_y}{1 - \frac{z}{c_z}}. \quad (7.2)$$

Фактором, що впливає на перспективне зміна розмірів, є наявність координати z в знаменнику. Чим ближче виявляється крапка до центра проекції, тим більше знаменник, а відповідно і координати точки.
Ми будемо розглядати ситуацію, коли центр проекції лежить на осі OZ, а сама вісь спрямована від спостерігача до проекційної площини, тобто $c_x = x_y = 0, \quad c_z = -d, \; d > 0$. Тоді формули (7.2) набувають вигляду

$$x^* = \frac{x}{1+\frac{z}{d}}, \quad y^* = \frac{y}{1+\frac{z}{d}}. \qquad (7.3)$$

У однорідних координатах таке перетворення можна записати за допомогою двох операцій. Спочатку множимо матрицю проективного перетворення P_z на вихідну точку і отримуємо точку в чотиривимірному просторі:

$$P_z \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/d & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 0 \\ 1+z \end{pmatrix} \qquad (7.4)$$

Потім проектуємо цю крапку в простір однорідних координат шляхом ділення на четверту компоненту:

$$(x^*, y^*, 0, 1) = (x/p, y/p, 0, 1), \quad p = 1 + z/d.$$

Подивимося тепер, що відбувається з пучком паралельних прямих під дією матриці проекції. Нехай заданий пучок прямих, паралельних вектору $\vec{v} = (a, b, c)$. Тоді параметричне рівняння прямої, яка належить цьому пучку, має вигляд

$$\vec{r} = (x + at, y + bt, z + ct).$$

З формули (7.4) випливає, що в результаті проектування отримаємо безліч точок

$$\left(x + at, y + bt, 0, 1 + \frac{z + ct}{d} \right).$$

Переходячи до однорідних координатах і помноживши чисельник і знаменник кожного дробу на d, отримаємо точки $\vec{r}_0$ виду

$$\vec{r}_0 = \left(d\frac{x + at}{d + z + ct}, d\frac{y + bt}{d + z + ct}, 0, 1 \right).$$

Тепер в кожній компоненті вектора чисельник і знаменник поділимо на t:

$$\vec{r}_0 = \left(d\frac{x/t + a}{(d + z)/t + c}, d\frac{y/t + b}{(d + z)/t + c}, 0, 1 \right).$$

Переходячи до межі при $t \to \infty$, отримаємо точку

$$\vec{r}_\infty = \left(\frac{da}{c}, \frac{db}{c}, 0, 1 \right).$$

Таким чином, отримуємо, що після проектування пучок паралельних прямих перетинається в точці сходу $\vec{r}_\infty$. Зрозуміло, що у кожного пучка своя точка сходу. Якщо пучок прямих паралельний площині XOY, тобто $c = 0$, то точка сходу виявляється на нескінченності, а значить, прямі залишаються паралельними.

Для побудови перспективної проекції з декількома точками сходу використовується матриця перспективного перетворення без проектування:

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1/d_1 & 1/d_2 & 1/d_3 & 1 \end{pmatrix}.$$

Тепер точки простору спочатку піддаються перспективному перетворенню, а потім здійснюється проекція.

Визначимо точки сходу для прямих, паралельних осям координат. для прямих $\overrightarrow{r} = (x, y, z + ct)$ результатом проективного перетворення буде безліч точок $(x/f, y/f, (z + ct)/f, 1)$, де $f = x/d_1 + y/d_2 + (z + ct)/d_3 + 1$. При $t \to \infty$ отримаємо точку з координатами $(0, 0, d_3, 1)$. При проекції на площину XOY отримаємо точку $(0, 0)$. пучок прямих $\overrightarrow{r} = (x + ct, y, z)$ перейде в $((x + ct)/f_1, y/f_1, z/f_1, 1),\ f_1 = (x + ct)/d_1 + y/d_2 + z/d_3 + 1$, а точкою сходу для нього буде $(d_1, 0, 0, 1)$, яка при проектуванні перейде в точку, що лежить на осі OX $(d_1, 0)$. Аналогічно для пучка прямих, паралельних осі OY, отримаємо точку сходу на осі OY $(0, d_2)$. Ці три точки на площині є головними точками сходу.

Спеціальні картографічні проекції. Екзотичні проекції земної сфери

З розвитком торгівлі і подорожей придбала більшу важливість досить непросте геометрична задача: як перенести на площину частина земної поверхні, щоб відстані між будь-якими двома точками на ній залишилися неспотвореними? Різні вчені протягом багатьох століть намагалися розв'язати цю проблему по замовленнях своїх урядів, великих комерсантів або просто мандрівників, для яких така карта була б справжньою знахідкою, так як суттєво полегшила б навігацію, як морську, так і повітряну.

В цілому ця задача виявилася нерозв'язною. Поверхня циліндра або конуса можна без спотворень перенести на площину (такі поверхні називаються що розгортаються), відобразити ж на площину поверхня сфери, зберігши відстань між будь-якими двома точками, невозможно.Дело в тому, що навіть малу область сферичної поверхні (на відміну від циліндра або конуса) неможливо розгорнути на площині без тріщин, складок або перекручувань. Будь-яка плоска карта Землі або якоїсь її частини неодмінно буде спотворювати будь властивості. Тому в даний час так необхідні карти з мінімальними (ще краще нульовими) перекручуваннями тих властивостей, для передачі яких призначається карта.Желательно, щоб і інші властивості деформувалися як можна меньше.Всего існує чотири основні типи спотворень:

• спотворення довжин (лінії, однакові на поверхні Землі, зображуються на карті відрізками різної довжини);

• спотворення кутів (кути на карті між узятими напрямками не дорівнюють горизонтальним кутам між тими ж напрямками на поверхні земного еліпсоїда);

• спотворення форм (форма ділянки або зайнятої об'єктом території на карті відмінна від їхньої форми на поверхні Землі);

• спотворення площ (пов'язане з масштабом площі: при сталості величини масштабу площі по всій поверхні карти перекручування площ на ній немає).

Крім класичних карт, велика частина яких була розроблена в середні століття, фантазія вчених надавала навігаторам досить незвичайні способи проекції земної поверхні, але перш ніж описати способи складання незвичайних карт, розглянемо деякі класичні методи картографії.
Центр проекції може бути довільним стосовно проектованої сфері; таким чином, існує нескінченна безліч всіляких різних проекцій. Якщо проводити промені з деякої точки, узятої на прямій, що проходить через центр кулі перпендикулярно деякої площини, то отримаємо на цій площині перспективну проекцію.Рассмотрім деякі з цих проекцій, найбільш корисні з точки зору картографії.

Стереографічна проекція

Важлива властивість будь-якої карти - збереження кутів (кут між будь-якими двома лініями на карті повинен бути таким же, як кут між прообразами цих ліній на земній поверхні). Збереження кутів особливо важливо для мореплавання і аеронавтики, так як воно означає, що спостережуваний кут між будь-якими двома орієнтирами дорівнює куту, вимірюваному на карті за допомогою транспортира. Крім того, на такій карті залишаються незмінними і площі малих областей. Карти, що зберігають кути, називаються конформними. Найпростіше побудувати конформних карту за допомогою стереографічній проекції.

На рис. 7.8 показано, як поверхня сфери в крапці X проектується з точки A (належить сфері) на площину, дотичну до сфери в діаметрально протилежній точці (антипод точки A). Проекція називається екваторіальній, полярної або косою залежно від того, де знаходяться антиподи: на екваторі, полюсах або в якій-небудь іншій точці земної поверхні відповідно. На жаль, конформність викликає спотворення масштабу, зростаюче зі збільшенням відстані від центру карти.

Рис. 7.8. три проекції

Позначимо за довготу і доповнення до широти крапки на сфері буквами λ і відповідно,
$0 \le \lambda \le 360^\circ,\ 0 \le \theta \le 180^\circ$, а через x і - Координати проекцій цієї точки в деякій декартовій системі координат, заданої на площині проекції. Відповідні формули проектування для Північної півкулі мають такий вигляд:

$$x = \operatorname{tg}\theta/2\cos(\lambda - 135°);$$
$$y = \operatorname{tg}\theta/2\sin(\lambda - 135°)$$

Тут і надалі радіус уважається рівним одиниці. Вісь X спрямована уздовж меридіана 135 °.

Гномонічна проекція

Відображення точки X з центру земної кулі B на площину карти в крапку B породжує гномоніческую проекцію (рис. 7.8). Проекція одержала таку назву, оскільки вона нагадує конструкцію сонячних годин з гномоном. Будь дуга великого кола на поверхні земної кулі переходить у пряму на гномоніческой карті. Великим кругомназивается окружність на сфері, площина якої проходить через її центр. Така картка не володіє конформність, але навігатори цінують її за одну важливу властивість, відсутнє у всіх інших проекцій сфери на площину: пряма між будь-якими двома точками на гномоніческой карті є геодезичної, або найкоротшою дугою між цими двома точками, і відповідає дузі великого кола на поверхні землі.

Ортографічна проекції

Якщо центр проекції перебуває в нескінченності (всі проектують промені паралельні), то проекція буде ортографической (рис. 7.8). Наприклад, дивлячись на Місяць із Землі, спостерігач бачить Місяць практично в ортографической проекції. У краю ортографической карти відстані сильно спотворені. Ортографической карта не зберігає ні площ, ні кутів, але, виконана досить мистецьки, створює сильну ілюзію кулястої Землі. Карти, накреслені з точки зору спостерігача, що знаходиться над земною поверхнею, не точні в передачі багатьох її властивостей, але найбільш вірно відповідають нашому зоровому сприйняттю сфери.
Ця проекція виходить при проектуванні на площину, дотичну до сфери в центрі зображуваного явища (λ_0, θ_0), за допомогою променів, перпендикулярних цієї площини. Формули цієї проекції наступні:

$$x = \sin\theta\sin(\lambda - \lambda_0)$$
$$y = \sin\theta_0\cos\theta - \cos\theta_0\sin\theta\cos(\lambda - lambda_0)$$

Проекції на циліндр

Поверхня сфери також можна проектувати на циліндри і конуси, "надягнуті" на сферу. Після побудови циліндричної або конічної проекції поверхня розрізається й розгортається на площину.
Промені, що проектують земну кулю на циліндр, вибираються так, щоб вони були паралельні площини, висікали коло, по якому сфера і циліндр стикаються (мал. 7.9). Якщо циліндр стосується Землі уздовж екватора, то всі меридіани і паралелі на карті переходять у прямі, що перетинаються під прямими кутами.

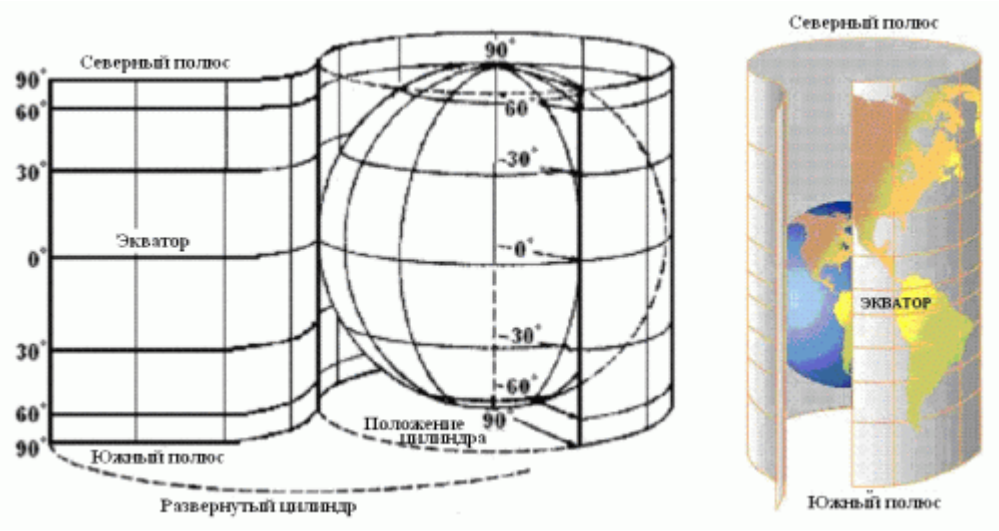

Рис. 7.9. Метод циліндричної проекції із збереженням площ

Циліндрична карта не завжди володіє конформностью і може сильно спотворювати відстані й форму областей. Відзначимо, що жодна карта не може одночасно бути конформної й зберігати площі. Було запропоновано величезне число інших проекцій, що зберігають площу; в сучасних атласах найчастіше зустрічаються зберігають площі карти, побудовані за допомогою циліндричної проекції, яка була запропонована Карлом Б.Мольвейде в 1805 р.

Проекція Меркатора

У XVI столітті фламандський картограф Герхард Меркатор створив знамениту циліндричну проекцію, що володіє властивістю конформності.Конформность в проекції Меркатора досягається за рахунок розтягування циліндра за полюси, при цьому у верхній і нижній частині цього циліндра масштаб стає дуже перекрученим. Незважаючи на це дана проекція володіє однією чудовою властивістю, дуже потрібним для навігаторів: пряма, проведена через будь-які дві точки на карті, є локсодромії, або лінією постійного румба. Локсодромії на сфері або який-небудь іншій поверхні обертання перетинає всі меридіани під постійним кутом (рис. 7.10). Проекція Меркатора задається наступними формулами:

$$x = \begin{cases} \lambda/180^\circ \text{ при } 0^\circ \le \lambda \le 180^\circ, \\ \lambda/180^\circ - 2 \text{ при } 180^\circ \le \lambda \le 360^\circ \end{cases}$$

$$y = \ln(\mathrm{tg}(90^\circ - 0.5\theta))/\pi.$$

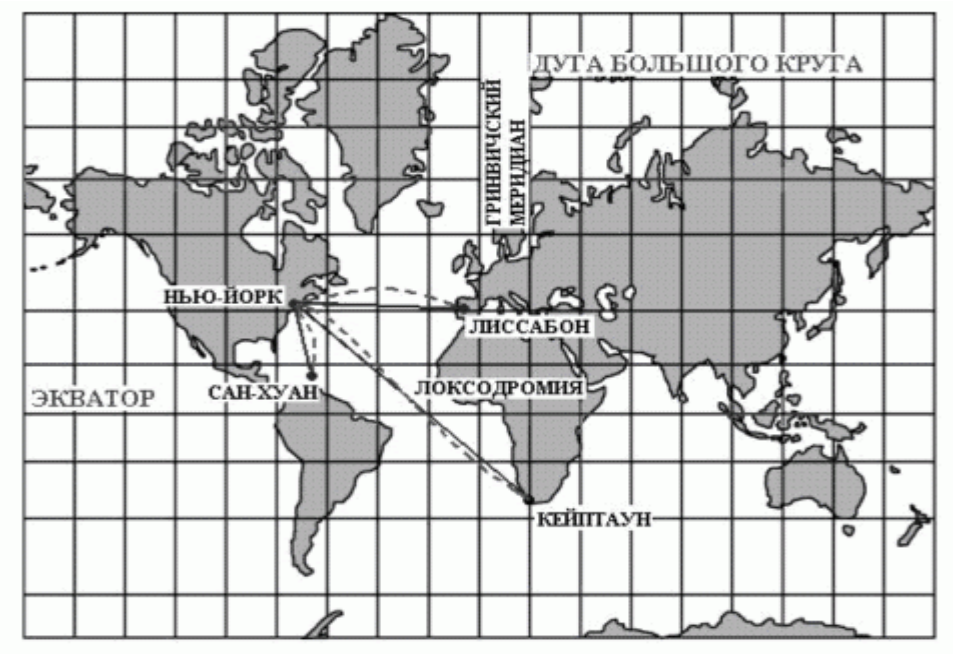

Рис. 7.10. Конформна проекція Меркатора. На карту нанесені локсодроми з Нью-Йорка

Проекції на багатогранник

Будемо називати розрізаної карту світу, спроектовану на той чи інший візерунок з яких-небудь багатокутників. Після складання цих фрагментів утвориться карта з розривами будь-якій частині земної кулі. Одну таку конформних карту склав філософ і математик Ч.Пірс. Земна поверхня спроектована на цій карті на вісім рівнобедрених трикутників, які можна розглядати як грані октаедра, сплющуються до тих пір, поки довжина його просторової діагоналі не звернеться в нуль. Вершин нульовою діагоналі на карті Пірса відповідають північний і південний полюси.

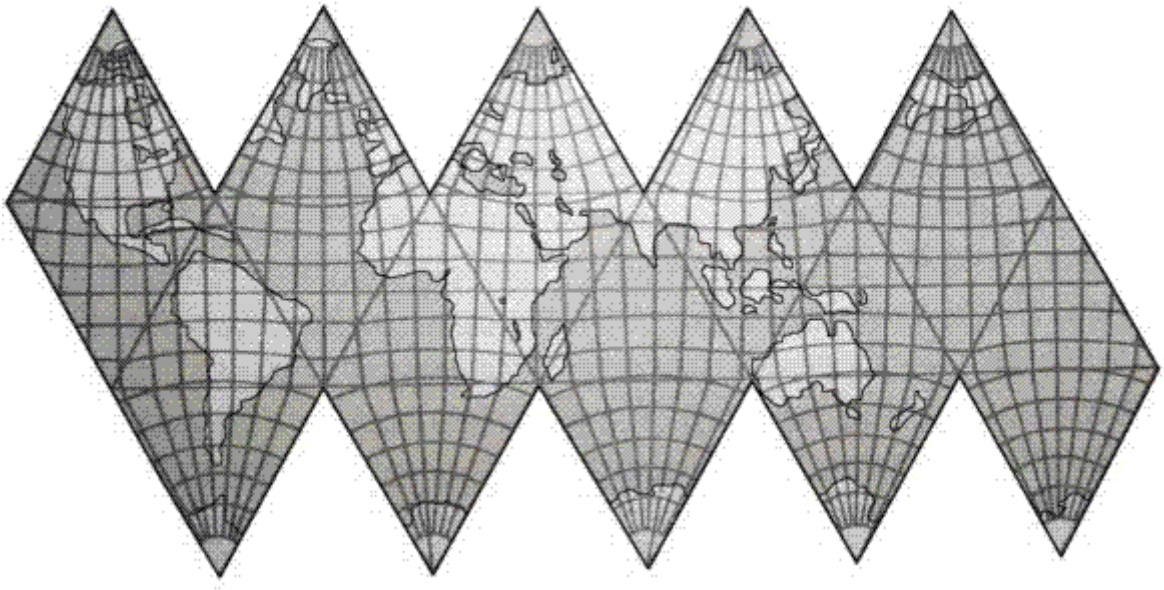

Рис. 7.11. "Лінкаглобус" І. Фішера, складаний у ікосаедр

Приблизно в той же час аналогічна ідея прийшла в голову видатному економісту з Єльського університету Ірвінгу Фішеру: він задумав здійснити гномоніческую проекцію

поверхні Землі на 20 трикутних гранях ікосаедра (рис. 7.11). Ікосаедр є найбільш близьким до ідеалу багатогранником для розрізання "на карти".

Незвичайні проекції

Картографи придумували найрізноманітніші види проекцій, часом дивно виглядають і при цьому володіють досить непоганими властивостями. Одну з таких карт у формі кардіоїда (серця) придумав Йоганн Вернер. Ця карта зберігає площі. Вона користувалася широкою популярністю в XVI в., Але зараз нею вже давно не користуються. У звіті про картографічних курйози, написаному для внутрішнього користування фірми "Лабораторії Белла", математик Едгар Н. Гільберт пише: "... незаслужено забута. Сильно перекручені частини карти лежать далеко від основних мас суши. Скривлені паралелі надають карті приємну ілюзію округлості .. . Паралелі являють собою дуги окружностей, розташовані на однаковій відстані один від одного, з центром у північному полюсі. Меридіани, проведені для указу ¬ ня відстаней уздовж паралелей, такі ж як на сфері ".

Скупчення материків на карті Вернера відбиває нерівномірний розподіл суши по поверхні Землі. Тихий океан настільки великий, що якщо дивитися на Землю з точки, розташованої над протокою Ла-Манш, то погляду відкриється близько 80% усієї суші, а противопо ¬ помилкове півкуля буде майже суцільно покрито водою

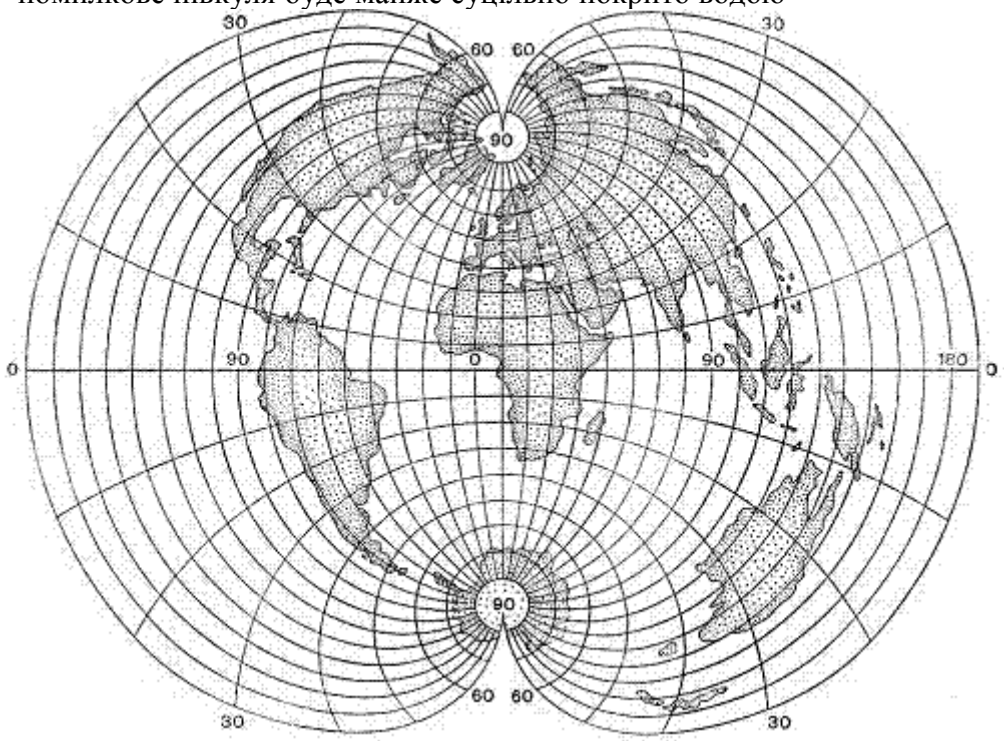

Рис. 7.12. Поліконічній картографічна проекція

Досить незвичайна поліконічній картографічна проекція-картографічна проекція, що будується за допомогою ряду конусів, дотичних до земного еліпсоїда (кулі). У поліконічній картографічній проекції паралелі нормальної сітки - дуги ексцентричних кіл, осьової меридіан - пряма, на якій розташовані центри паралелей, а інші меридіани - криві (рис. 7.12).

Питання та вправи

1. Назвіть два основних види проекцій, обумовлених типом пучка променів.

2. Назвіть чотири види паралельних проекцій.

3. Скільки кроків в алгоритмі ортогональної проекції на довільну площину?

4. Який вигляд має матриця косокутній проекції на площину XOY, переводить вектор $(0,0,1)$ у вектор $(x,y,0)$?

5. Напишіть формули перетворення координат при центральній проекції на площину з центром в точці $(0,0,c)$. Як виглядає матриця такої проекції в однорідній системі координат?

6. Що таке перспективне укорочування?

7. Що таке точка сходу?

8. Як реалізується проекція із трьома точками сходу?

9. Яким властивістю володіє конформна проекція?

10. Яким властивістю володіє циліндрична проекція?

11. У чому цінність проекції Меркатора?

12. Який багатогранник найбільш зручний для побудови розрізаних карт?

Лекція 8. Растрове перетворення графічних примітивів

Зміст

• Алгоритм Брезенхема растрової дискретизації відрізка
• Алгоритми Брезенхема растрової дискретизації окружності й еліпса
• Алгоритми заповнення областей
• Питання та вправи

Алгоритм Брезенхема растрової дискретизації відрізка

При побудові растрового образа відрізка необхідно, перш за все, встановити критерії "хорошою" апроксимації. Перша вимога полягає в тому, що відрізок повинен починатися і закінчуватися в заданих точках і при цьому виглядати суцільним і прямим (при досить високому дозволі дисплея цього можна добитися). Крім того, яскравість уздовж відрізка повинна бути однаковою і не залежати від нахилу відрізка і його довжини. Ця вимога виконати складніше, оскільки горизонтальні і вертикальні відрізки завжди будуть яскравіше похилих, а постійна яскравість уздовж відрізка знову ж досягається на вертикальних, горизонтальних і нахилених під кутом в 45 ° лініях. І, нарешті, алгоритм повинен працювати швидко. Для цього необхідно по можливості виключити операції з речовими числами. З метою прискорення роботи алгоритму можна також реалізувати його на апаратному рівні.

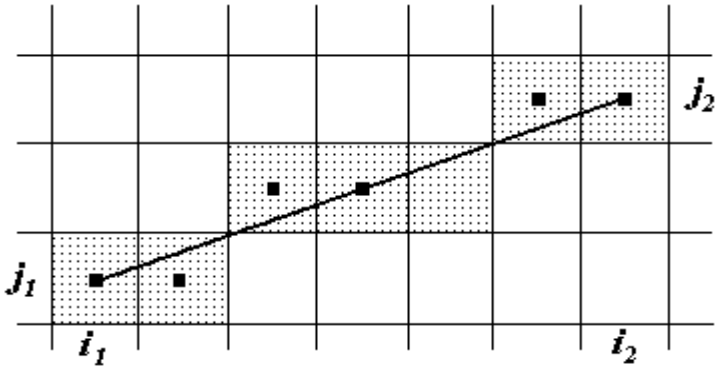

Рис. 8.1. Растровий образ відрізка

У більшості алгоритмів використовується покроковий метод зображення, тобто для знаходження координат чергової точки растрового образу нарощується значення однієї з координат на одиницю растра і обчислюється приріст інший координати.

Завдання полягає в побудові відрізка, що з'єднує на екрані крапки з координатами $(i_1, j_1), (i_2, j_2)$ (будемо вважати, що $i_1 \neq i_2$). Для побудови відрізка прямої на площині з речовими координатами можна скористатися рівнянням прямої, що проходить через дві задані точки, яке має вигляд $j = j_1 + k(i - i_1), \quad k = (j_2 - j_1)/(i_2 - i_1).$
Тепер, вважаючи, що $0 \leq k \leq 1$, (i, j) - координати поточної точки растрового образу, а y - точне значення координати точки відрізка, можна побудувати наступну точку: $i' = i + 1, \quad j' = y + k$
Слід зауважити, що целочисленная координата j зміниться тільки в тому випадку, якщо у перевищить величину $j + 0.5$ (j' є найближче до y ціле число, отримане в результаті операції округлення). Наведений приклад включає операції з числами, які виконуються

істотно повільніше, ніж відповідні цілочисельні операції, а при побудові растрового образа відрізка бажаний алгоритм, по можливості звертається тільки до целочисленной арифметиці. Крім того, алгоритм повинен працювати при будь-якому взаємному розташуванні кінців відрізка.

Алгоритм Брезенхема побудови растрового образа відрізка був спочатку розроблений для графопостроителей, але він повністю підходить і для растрових дисплеїв. В процесі роботи залежно від кутового коефіцієнта відрізка нарощується на одиницю або i , або j , а зміну іншої координати залежить від відстані між дійсним станом точки і найближчою точкою растра (зсуву). Алгоритм побудований так, що аналізується лише знак цього зсуву.

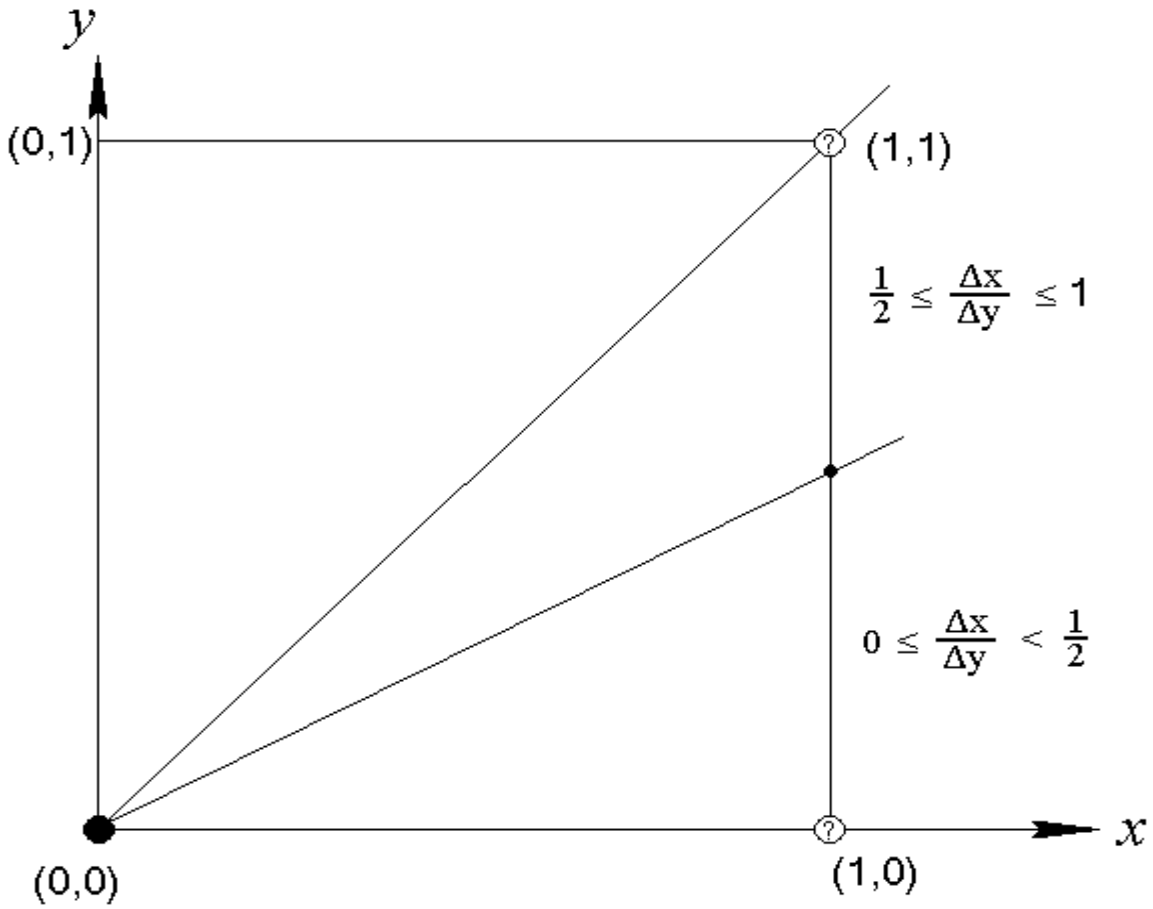

Рис. 8.2. Зв'язок кутового коефіцієнта з вибором пикселя

На рис. 8,2 це ілюструється для відрізка з кутовим коефіцієнтом, що лежить в діапазоні від нуля до одиниці. З малюнка можна помітити, що якщо кутовий коефіцієнт $k \geq \frac{1}{2}$, то при виході з точки $(0,0)$ перетинання з прямою $x = 1$ буде ближче до прямої $y = 1$, ніж до прямої $y = 0$. Отже, точка растра $(1,1)$ краще апроксимує проходження відрізка, ніж точка $(1,0)$. При $k < \frac{1}{2}$ вірно зворотне.

На рис. 8,3 показано, яким чином будуються точки растра для відрізка з тангенсом кута нахилу $3/8$, а на **рис. 8.4** - графік зсуву. На початку побудови зсув покладається рівним $k - 1/2$, а потім на кожному кроці воно нарощується на величину k , і якщо при цьому вертикальна координата точки растра збільшується на одиницю, то зсув в свою чергу зменшується на одиницю. На **рис. 8,5** наведена блок-схема алгоритму для випадку $i_2 > i_1, \quad j_2 > j_1, \quad k > 0$.

Неважко зрозуміти, що від цього алгоритму перейти до цілочисельного: досить замість величини зсуву перейти до величини $\tilde{e} = 2\Delta i \cdot e$.

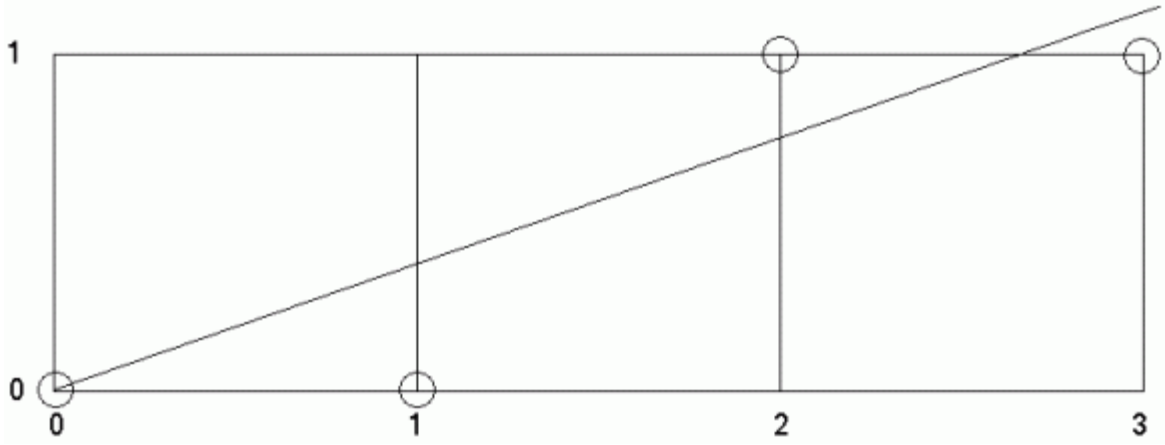

Рис. 8.3. Пікселі, що належать розгортці відрізка

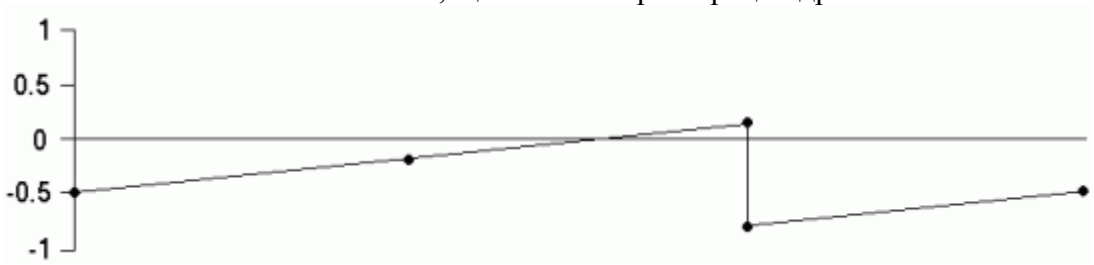

Рис. 8.4. Графік зміни відхилення

Наведемо загальний алгоритм Брезенхема, який враховує всі можливі випадки напрямку відрізка, розглянутого як вектор на координатній площині (на **рис. 8,6** виділені чотири області і зазначені особливості алгоритму в кожній з них). В описі алгоритму використовуються наступні функції:
swap (a, b): обмен значений переменных a, b;

abs (a): абсолютное значение a;

sign (a): 0, если a= 0, 1, если a>0, –1, если a<0;

point (i, j) - инициализация точки (i, j).

Передбачається, що кінці відрізка $(i_1, j_1), (i_2, j_2)$ не збігаються і що всі використовувані змінні є цілими.

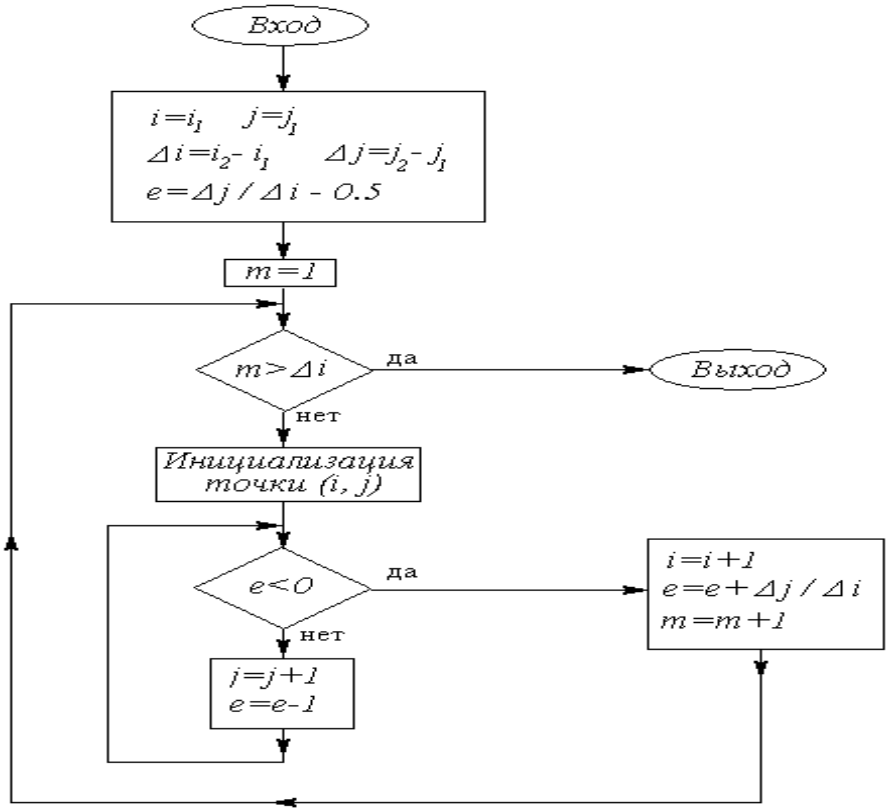

Рис. 8.5. Блок-схема однієї гілки алгоритму Брезенхема

```
i=i1;
j=j1;
di=i2-i1;
dj=j2-j1;
s1=sign(i2-i1);
s2=sign(j2-j1);
di=abs(di);
dj=abs(dj);
if (dj>di)
{
swap(di,dj); c=1;
}
else c=0;
e=2*dj-di;  // Инициализация смещения
// Основной цикл
for (l=0; l<di; l++)
{
point(i,j);
while (e>=0)
{
if (c==1) i=i+s1;
else j=j+s2;
e=e-2*di;
}
if (c==1) j=j+s2; else i=i+s1;
e=e+2*dj;
}
```

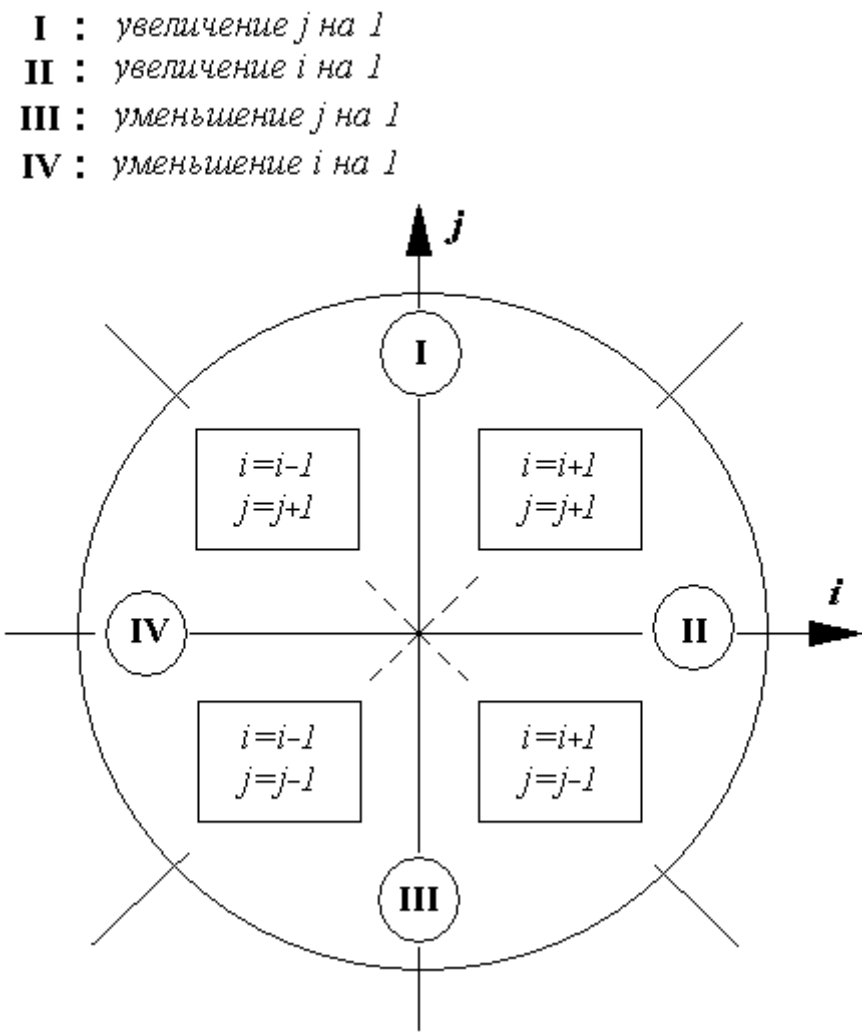

Рис. 8.6. Чотири можливих напрямки відрізка

Алгоритми Брезенхема растрової дискретизації окружності й еліпса

Алгоритм зображення окружності трохи складніше, ніж побудова відрізка. Ми розглянемо його для випадку окружності радіусу r з центром на початку координат. Перенесення його на випадок довільного центру не складає труднощів. При побудові растрової розгортки окружності можна скористатися її симетрією щодо координатних осей і прямих $y = \pm x$. Необхідно згенерувати лише одну восьму частину кола, а інші її частини можна отримати шляхом відображень симетрії. За основу можна взяти частину окружності від 0 до 45 ° в напрямку за годинниковою стрілкою з вихідною точкою побудови $(r, 0)$. В цьому випадку координата окружності x є монотонно спадною функцією координати y .

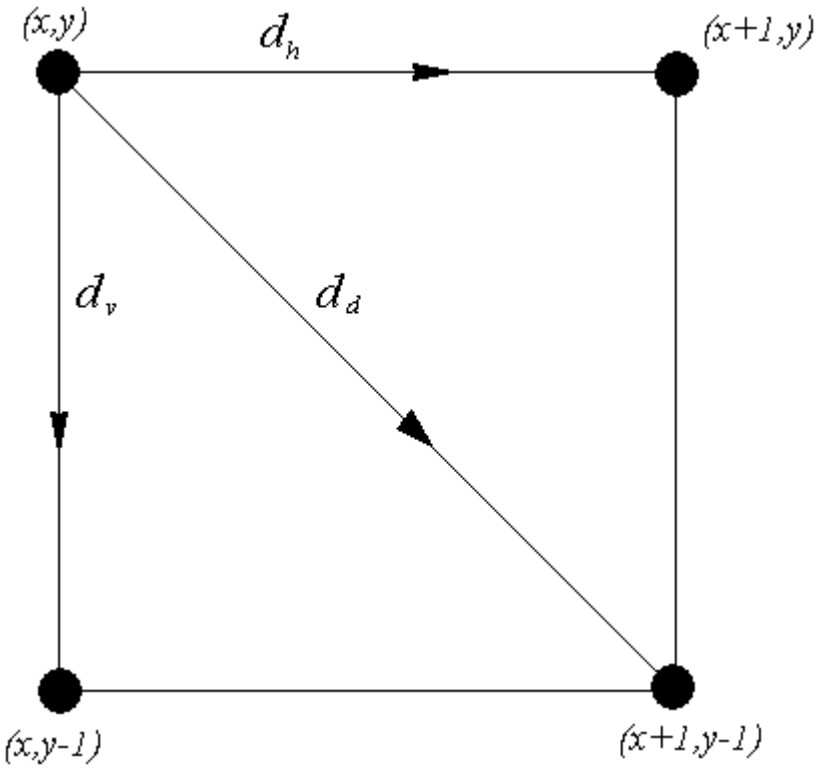

Рис. 8.7. Найближчий піксель при русі по колу

При вибраному напрямку руху по колу є тільки три можливості для розташування найближчого пікселя: на одиницю вправо, на одиницю вниз і по діагоналі вниз (рис. 8.7). Вибір варіанту можна здійснити, обчисливши відстані до цих точок і вибравши мінімальне з них:

$$d_h = |s_h|, \quad d_v = |s_v|, \quad d_d = |s_d|,$$

$$s_h = (x+1)^2 + y^2 - r^2, \quad s_v = x^2 + (y-1)^2 - r^2,$$

$$s_d = (x+1)^2 + (y-1)^2 - r^2.$$

Алгоритм можна спростити, перейшовши до аналізу знаків величин s_h, s_v, s_d. При $s_d < 0$ діагональна точка лежить усередині кола, тому найближчими точками можуть бути тільки діагональна і права. Тепер досить проаналізувати знак вираження $\Delta = d_v - d_d$. Якщо $\Delta \le 0$, вибираємо горизонтальний крок, у противному випадку - діагональний. Якщо ж $s_d > 0$, то визначаємо знак $\Delta^1 = d_d - d_v$, і якщо $\Delta^1 \le 0$, вибираємо діагональний крок, у противному випадку, вертикальний. Потім обчислюється нове значення s_d, причому бажано мінімізувати обчислення не тільки цієї величини, а й величин Δ, Δ^1 на кожному кроці алгоритму. Шляхом нескладних перетворень можна одержати для першого кроку алгоритму, що.

$$\Delta = 2(s_d + y) - 1, \quad \Delta^1 = 2(s_d + x) - 1$$

Після переходу в точку $(x', y'),\ x' = x+1,\ y' = y+1$ по діагоналі нове значення s_d обчислюється за формулою $s'_d = s_d + 2x' - 2y' + 2$, при горизонтальному переході $(x' = x+1, y' = y)$ $s'_d = s_d + 2x' + 1$, при вертикальному

$(x' = x, y' = y-1)$ - $s'_d = s_d - 2y' + 1$.

Таким чином, алгоритм малювання цієї частини окружності можна вважати повністю описаним (блок-схема його наведена **на рис 8.8.**) Все решта її частини будуються паралельно. Після отримання чергової точки (x', y') можна ініціалізувати ще

сім точок з координатами.
$(-x', y'),\ (-x', -y'),\ (x', -y'),\ (y', x'),\ (y', -x'),\ (-y', -x'),\ (-y', x')$

Для побудови растрової розгортки еліпса з осями, паралельними осям координат, і радіусами a, b скористаємося канонічним рівнянням

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1,$$

яке перепишемо у вигляді

$$f(x, y) \equiv b^2x^2 + a^2y^2 - a^2b^2 = 0.$$

На відміну від кола, для якої було досить побудувати одну восьму її частину, а потім скористатися властивостями симетрії, еліпс має тільки дві осі симетрії, тому доведеться будувати одну чверть всієї фігури. За основу візьмемо дугу, що лежить між точками $(0, b)$ і $(a, 0)$в першому квадранті координатної площини.

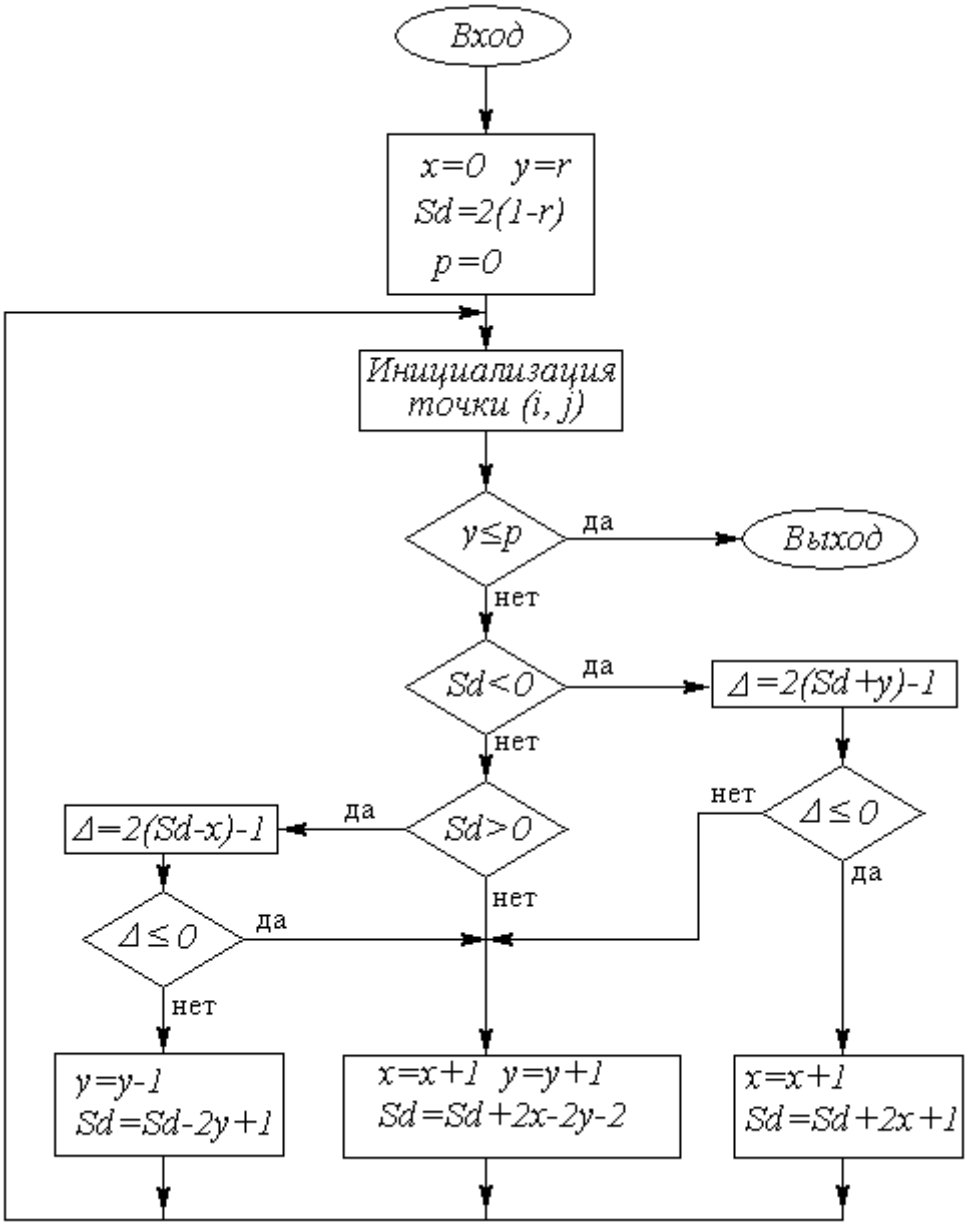

Рис. 8.8. Блок-схема побудови восьмий частини окружності

У кожній точці (x, y)еліпса існує вектор нормалі, що задається градієнтом функції f. Дугу розіб'ємо на дві частини: перша - з кутом між нормаллю і горизонтальною віссю більше 45 ° (тангенс більше 1) і друга - з кутом, меншим 45 °

(**рис. 8.9**). Рух уздовж дуги будемо здійснювати в напрямку за годинниковою стрілкою, починаючи з точки $(0, b)$. Вздовж всієї дуги координата є монотонно спадною функцією від x, але в першій частині вона убуває повільніше, ніж зростає аргумент, а в другій -. Швидше. Тому при побудові растрового образа в першій частині будемо збільшувати x на одиницю і шукати відповідне значення y, а в другій - спочатку зменшувати значення y на одиницю і визначати відповідне значення x.

Напрямок нормалі відповідає вектору

$$grad(x, y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) = (2b^2x, 2a^2y).$$

Звідси знаходимо тангенс кута нахилу вектора нормалі $t = a^2y/b^2x$. Прирівнюючи його одиниці, отримуємо, що координати точки поділу дуги на вищевказані частини задовольняють рівності $b^2x = a^2y$. Тому критерієм того, що ми переходимо до другої області в цілочисельних координатах, буде співвідношення $a^2(y - 1/2) \le b^2(x + 1)$, або, переходячи до цілочисловим операціями $a^2(2y - 1) \le 2b^2(x + 1)$.

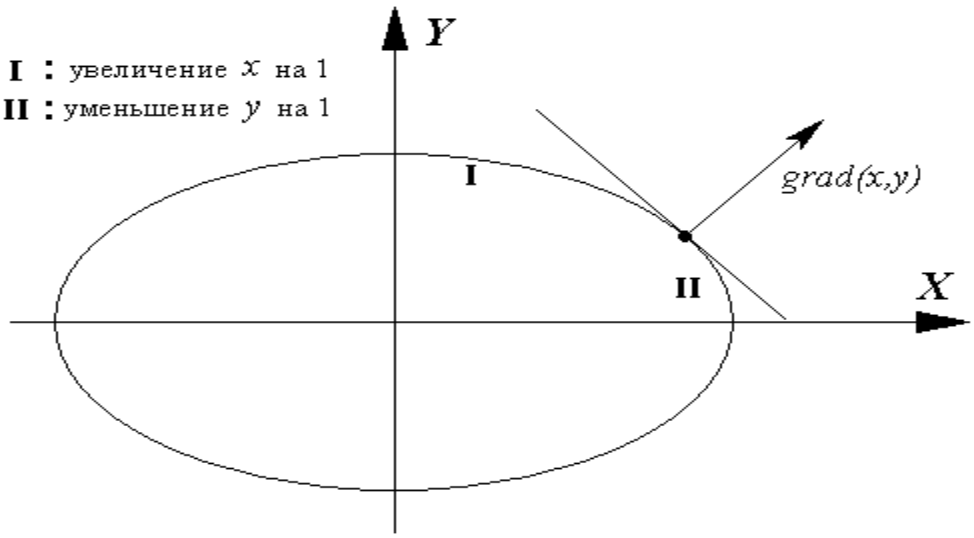

Рис. 8.9. Дві області на ділянці еліпса

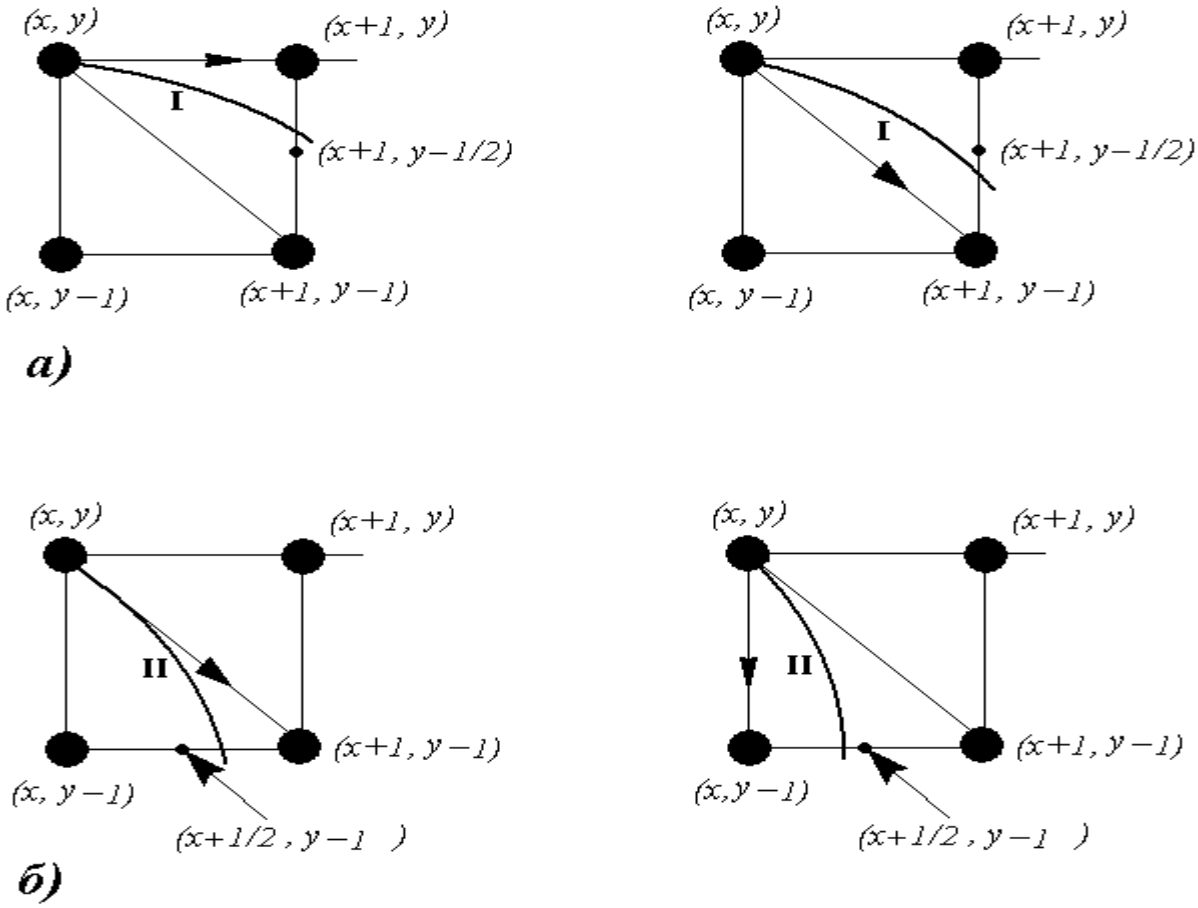

Рис. 8.10. Схема переходу в першій і другій областях дуги еліпса

При переміщенні уздовж першої ділянки дуги ми з кожної точки переходимо або по горизонталі, або по діагоналі, і критерій такого переходу нагадує той, який використовувався при побудові растрового образа окружності. Перебуваючи в точці (x, y), ми будемо обчислювати значення $\Delta = f(x+1, y-\frac{1}{2})$. Якщо це значення менше нуля, то додаткова точка $(x+1, y-\frac{1}{2})$ лежить усередині еліпса, отже, найближча точка растра є $(x+1, y)$, у противному випадку це точка $(x+1, y-1)$ (**рис. 8.10а**).

На другій ділянці дуги можливий перехід або по діагоналі, або по вертикалі, тому тут спочатку значення координати y зменшується на одиницю, потім обчислюється $\Delta = f(x+\frac{1}{2}, y-1)$ і напрямок переходу вибирається аналогічно попередньому випадку (рис. 8.10б).

Залишається оптимізувати обчислення параметра Δ, помноживши його на 4 і представивши у вигляді функції координат точки. Тоді для першої половини дуги маємо

$$\widetilde{\Delta}(x,y) \equiv 4\Delta(x,y) = 4b^2(x+1)^2 + a^2(2y-1)^2 - 4a^2b^2,$$

$$\widetilde{\Delta}(x+1,y) = \widetilde{\Delta}(x,y) + 4b^2(2x+3),$$

$$\widetilde{\Delta}(x+1,y-1) = \widetilde{\Delta}(x,y) + 4b^2(2x+3) - 8a^2(y-1).$$

Для другої половини дуги отримаємо

$$\widetilde{\Delta}(x,y) \equiv 4\Delta(x,y) = b^2(2x+1)^2 + 4a^2(y-1)^2 - 4a^2b^2,$$

$$\widetilde{\Delta}(x+1,y) = \widetilde{\Delta}(x,y) + 8b^2(x+1),$$

$$\widetilde{\Delta}(x+1,y-1) = \widetilde{\Delta}(x,y) + 8b^2(x+1) - 4a^2(2y+3).$$

Всі залишилися дуги еліпса будуються паралельно: після одержання чергової точки (x', y'), можна ініціалізувати ще три точки з координатами $(-x', y'),\ (-x', -y'),\ (x', -y')$. Блок-схему не наводимо з огляду прозорості алгоритму.

Алгоритми заповнення областей

Для заповнення областей, обмежених замкнутою лінією, застосовуються два основних підходи: затравочное заповнення і растрова розгортка.

Методи першого типу виходять з того, що задана деяка точка (запал) всередині контуру і заданий критерій приналежності точки кордоні області (наприклад, заданий колір кордону). В алгоритмах шукають точки, сусідні з затравочной і розташовані всередині контура. Якщо виявлена сусідня точка, що належить внутрішній області контуру, то вона стає затравочной і пошук триває рекурсивно.

Методи растрової розгортки засновані на скануванні рядків растра й визначенні, чи лежить точка всередині заданого контуру області.Сканіроване здійснюється найчастіше "зверху вниз", а алгоритм визначення приналежності точки заданої області залежить від виду її межі.

Спочатку розглянемо простий алгоритм заповнення з запалом з використанням стека. Під стеком в даному випадку ми будемо розуміти масив, в який можна послідовно поміщати значення й послідовно витягати, причому витягають елементи не в порядку надходження, а навпаки: за принципом "першим прийшов - останнім пішов" ("перший в - останній з"). Алгоритм заповнення виглядає наступним чином:

Помістити затравочний піксель в стек
Поки стік не порожній:
Витягти пиксель зі стека
Ініціалізувати піксель
Для кожного з чотирьох сусідніх пікселів:
Перевірити, чи є він граничним і чи був він инициализирован
Якщо ні, то помістити пиксель у стек

Алгоритм можна модифікувати таким чином, що сусідніми будуть вважатися вісім пікселів (додаються елементи, розташовані в діагональному напрямку).

Методи растрової розгортки розглянемо спочатку в застосуванні до заповнення багатокутників. Найпростіший метод побудови полягає в тому, щоб для кожного пікселя растра перевірити його приналежність нутрощі багатокутника. Але такий перебір занадто неекономічний, оскільки фігура може займати лише незначну частину екрана, а геометричний пошук - завдання трудомістка, сполучена з довгими обчисленнями. Алгоритм стане більш ефективним, якщо попередньо виявити мінімальний прямокутник, в який занурений контур багатокутника, але і цього може виявитися недостатньо.

У випадку, коли багатокутник, що обмежує область, заданий списком вершин і ребер (ребро визначається як пара вершин), можна запропонувати ще більш економний метод. Для кожної сканирующей рядки визначаються точки перетину з ребрами багатогранника, які потім упорядковуються по координаті x. Визначення того, який інтервал між парами перетинань є внутрішній для багатогранника, а який ні, є досить простий логічної завданням. При цьому якщо скануюча рядок проходить через вершину багатогранника, то це перетин має бути враховано двічі у випадку, коли вершина є точкою локального мінімуму або максимуму. Для пошуку перетинань сканирующей рядка з ребрами можна використовувати алгоритм Брезенхема побудови растрового образа відрізка.

На закінчення як приклад наведемо алгоритм зафарбовування внутрішньої області трикутника, заснований на складанні повного впорядкованого списку всіх відрізків, що становлять цей треугольнік.Для записи горизонтальних координат кінців цих відрізків

будемо використовувати два масиви X_{min} і X_{max} розмірністю, рівної числу пікселів растра по вертикалі (**рис. 8.11**) .

Побудова починається з ініціалізації масивів X_{min} і X_{max}: масив заповнюється X_{max} нулями, а масив X_{min} - числом N, що дорівнює кількості пікселів растра по горизонталі. Потім визначаємо значення j_{min}, J_{max}, що обмежують трикутник у вертикальному напрямку. Тепер, використовуючи модифікований алгоритм Брезенхема, занесемо кордону відрізків в масиви X_{min} і X_{max}. Для цього всякий раз при переході до чергового пікселю при формуванні відрізка замість його ініціалізації будемо порівнювати його координату i з вмістом j -й осередку масивів. Якщо $X_{min}[j] > i$, то записуємо координату i в масив X_{min}. Аналогічно за умови $X_{max}[j] < i$ координату i записуємо в масив X_{max}.

Якщо тепер послідовно застосувати алгоритм Брезенхема до всіх трьох сторонах трикутника, то ми отримаємо потрібним чином заповнені масиви кордонів. Залишається тільки проініціалізувати пікселі всередині відрізків $\{(X_{min[j],j}),\ (X_{max}[j], j)\}$.

Цей алгоритм можна легко поширити на випадок довільного опуклого багатокутника.

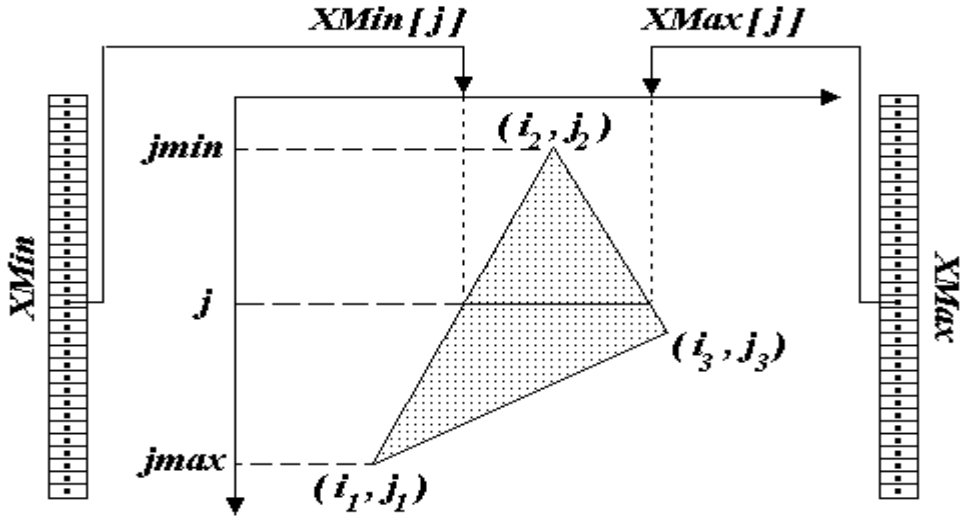

Рис. 8.11. Схема побудови растрової розгортки трикутника

Питання та вправи

1. Що таке розкладання в растр?
2. Яка математична основа растрового розкладання в алгоритмі Брезенхема?
3. За яким критерієм ініціалізується піксель в цьому алгоритмі?
4. Чим відрізняються гілки алгоритму при кутах нахилу <45 ° і> 45 °?
5. Яку частину окружності досить побудувати, щоб потім шляхом відбиттів одержати окружність цілком?
6. Яку частину еліпса досить побудувати, щоб потім шляхом відбиттів одержати еліпс цілком?
7. Назвіть два типи алгоритмів заповнення областей.
8. Яка структура даних використовується в алгоритмах з запалом?
9. Які дані використовуються при побудові растрової розгортки трикутника? 10.

Лекція 9. Замальовування. Рендеринг полігональних моделей

Моделі освітлення. Зафарбовування границь: пласке зафарбовування, метод Гуро, метод Фонга. Видалення ступінчастості (антиалайсінг)

Зміст

Візуальне сприйняття об'єктів оточуючого середовища являє собою складний процес, маючий як фізичні так і психологічні аспекти. В другому розділі ми вже обговорювали деякі складності сприйняття зображень і кольору оком людини. До того що вже було сказано про спектральну чутливість ока, потрібно добавити ще декілька моментів.

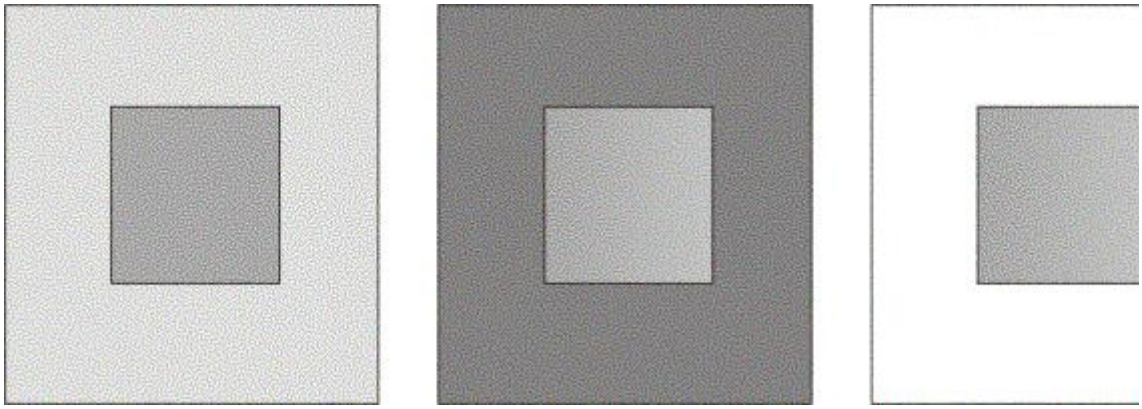

Рис. 9.1. Ефекти сприйняття зображення

Око адаптується до середнього освітлення даної сцени, тому при зміні фона, змінюється сприйняття сцени. Наприклад, однорідно замальована область на більш темному фоні буде здаватися більш яскравою ніж на темному. Крім того, вона буде сприйматись як більш об'ємна (мал. 9.1).

Іще одна особливість сприйняття виражається в тому, що границі рівномірно освітленої області здається більш яскравою в порівнянні з внутрішньою областю. Це явище було вперш винайдено Эрнстом Махом, тому воно отримало назву **ефекту смуг Маха**. Такі особливості необхідно враховувати, якщо ми прямуємо до реалістичного зображення сцен.

При формуванні зображень, маючих дзеркальні і напівпрозорі поверхні, потрібно використовувати закони геометричної оптики, заломлюючих якості матеріалів, ефекти змішування кольорів і т.д.

Проста модель освітлення

Об`єкти оточуючого простору стають видимими для ока дякуючи енергії світла, яка може випромінюватися поверхнею предмета, віддзеркалюватися і проходити крізь неї. В свою чергу віддзеркалення світла від поверхні залежить від фізичних якостей матеріалу, із якого вона виготовлена, а також від характеру та розповсюдження джерела світла. Яскравість (та інтенсивність) освітлення залежить від енергії світлового потоку, яка обумовлюється, по-перше, потужністю джерела освітлення, а по-друге, відбиваючими і пропускними якостями об'єкту.

Спочатку ми розглянемо модель освітлення, враховуючи тільки віддзеркалення. Якості відображення світла залежать в основному від направлення смуг и характеристик відбиття поверхні.

Відбиття може бути двох видів: **Дифузне і Дзеркальне**. Перше виникає у випадку, коли світло проходить під поверхню об'єкту поглинається, а потім рівномірно випромінюється у всіх напрямках. Поверхня в цьому випадку розглядається як ідеальний розсіювач. При цьому виникає ефект матового світла, а видима освітленість тієї або іншої ділянки поверхні не залежить від положення спостерігача. Дзеркальне відображення, навпаки, походить від зовнішньої поверхні, інтенсивність його неоднорідна, тому видимий максимум освітленості залежить від положення ока спостерігача.

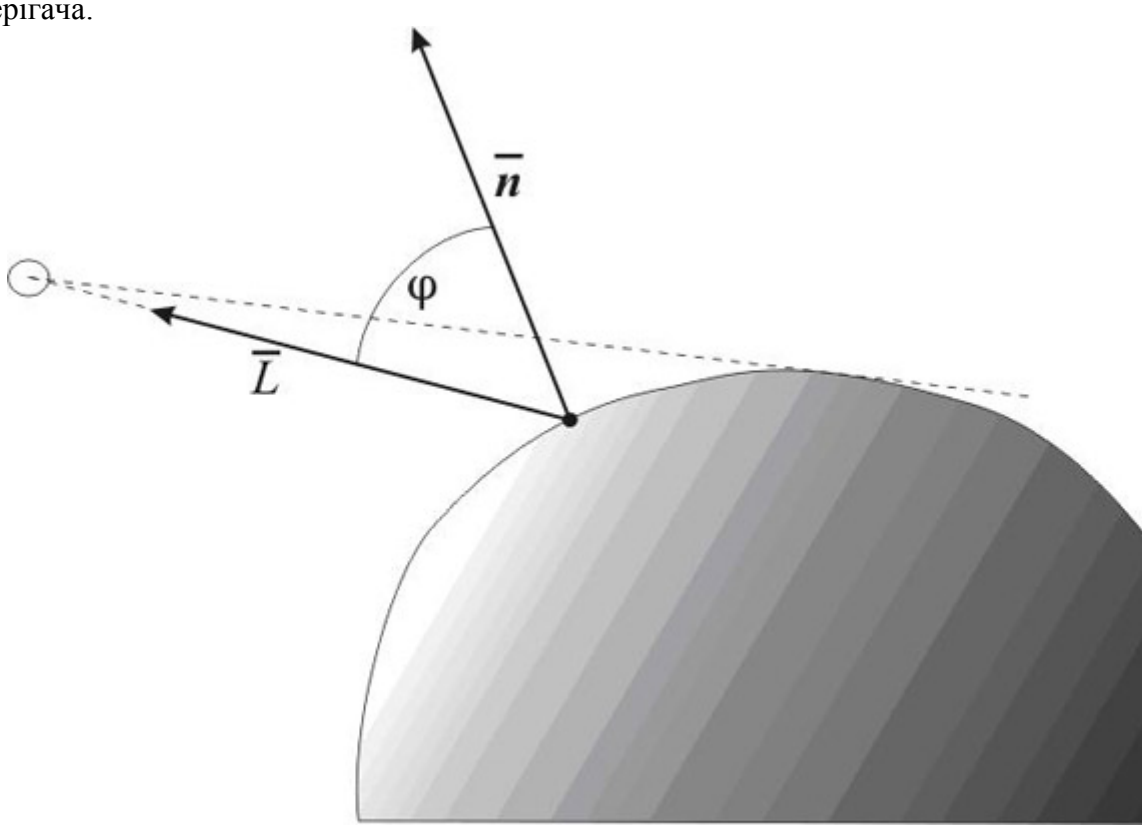

Рис. 9.2. Освітлення точковим джерелом

Світло крапкового джерела відбивається від поверхні розсіювача за законом Ламберта: інтенсивність відбиття пропорційна косинусу кута між зовнішньою нормаллю до поверхні і напрямком до джерела світла (мал. 9.2). Якщо I_S - інтенсивність джерела світла. φ - кут між вектором зовнішньої нормалі до поверхні і напрямком до джерела світла, то інтенсивність відбитого світла визначається формулою

$$I = \begin{cases} I_S \cos\varphi \text{ при } 0 \le \varphi \le \pi/2 \\ 0 \text{ в противном случае} \end{cases}$$

При такому розрахунку інтенсивності вийде дуже контрастна картина, тому ділянки поверхні, на які промені від джерела не потрапляють безпосередньо, залишаться абсолютно чорними. Для підвищення реалістичності необхідно враховувати розсіювання світла в навколишньому просторі. Тому вводиться фонова освітленість, що залежить від інтенсивності розсіяного світла I_F, і інтенсивність відбитого світла визначається вираженням.

$$I = \begin{cases} I_F k_F + k_s I_S \cos\varphi \text{ при } 0 \le \varphi \le \pi/2 \\ \\ I_F k_F \text{ в противном случае} \end{cases}$$

де k_F- коефіцієнт дифузного відбиття розсіяного світла, k_S -__коефіцієнт дифузного відбиття падаючого світла, $0 \le k_S \le 1$, $0 \le k_F \le 1$.

$$I = I_F k_F + \frac{k_S I_S \cos\varphi}{d + C}$$

В описаній моделі поки ніяк не враховувалася далекість джерела світла від освітленості двох об'єктів не можна судити про їх взаємне розташування в просторі отримати перспективне зображення, то необхідно включити загасання інтенсивності з відстанню інтенсивність світла обернено пропорційна квадрату відстані від джерела. В якості відстані до джере перспективного перетворення можна взяти відстань до центру проекції, і якщо він достатньо відда зображення буде досить адекватним. Але якщо цей центр розташований близько до об'єкта відстані міняється дуже швидко, і в цьому випадку краще використовувати лінійне випадку інтенсивність відбитого світла від безпосередньо освітлених ділянок поверхні формулою де:

d - відстань до центра проекції, а C - довільна постійна. Якщо центр проекції перебуває на нескінченності, тобто при паралельному проектуванні, то в якості

d можна взяти відстань до об'єкта, найбільш близького до спостерігача.

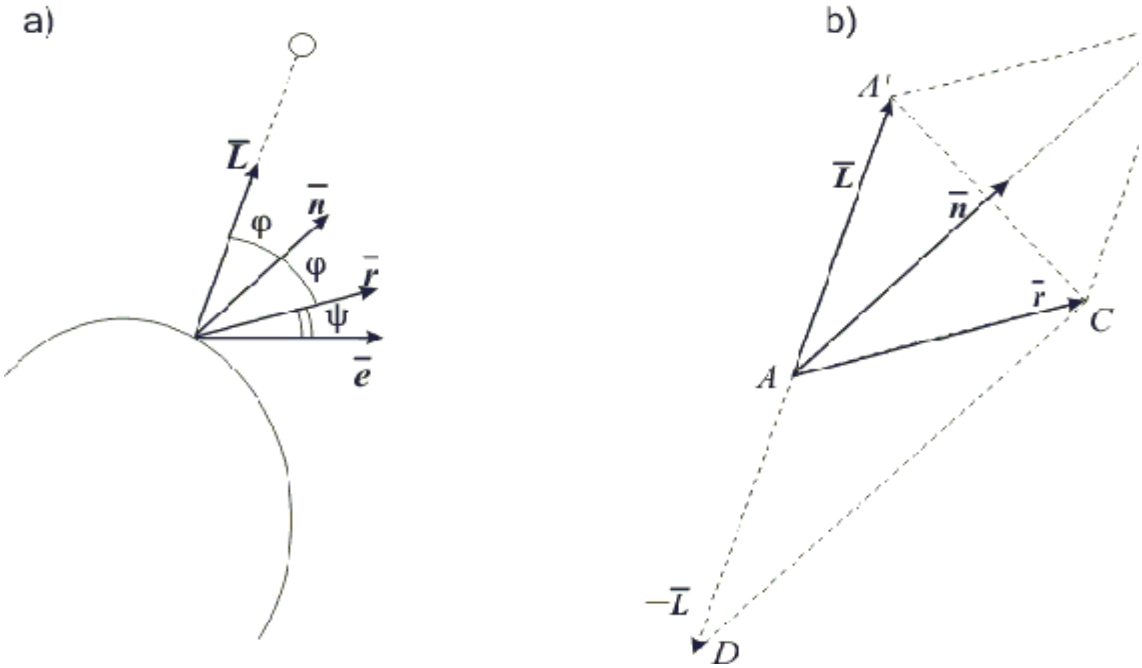

Рис. 9.3. дзеркальне відображення

На відміну від дифузного, дзеркальне відображення є спрямованим. Ідеальне дзеркало відбиває промені за принципом "відбитий і падаючий промені лежать в одній площині, причому кут падіння дорівнює куту відбиття" (мається на увазі кут між напрямом променя і нормаллю до поверхні). Якщо поверхня не ідеально дзеркальна, то промені відбиваються в різних напрямках, але з різною інтенсивністю, а функція зміни інтенсивності має чітко виражений максимум. Оскільки фізичні властивості дзеркального відображення досить складні, то в комп'ютерній графіці використовується емпірична модель Фонга. Суть її полягає в тому, що для ока спостерігача інтенсивність дзеркально відбитого променя залежить від кута між ідеально відбитим променем і напрямком до спостерігача (Мал. 9.3а). Крім того, оскільки дзеркальне відображення залежить ще й від довжини хвилі, це також будемо враховувати у формулі для

обчислення інтенсивності. Модель Фонга описується співвідношенням

$I_Z = \omega(\varphi, \lambda) \cdot I_S \cos^n \psi$ де:

$\omega(\varphi, \lambda)$ - функція відбиття, λ - довжина хвилі. Ступінь, в яку зводиться косинус кута, впливає на розміри світлового відблиску, спостережуваного глядачем. Графіки цієї функції наведено на (мал. 9.4), і вони як раз є характерними кривими поводження функції зміни інтенсивності залежно від властивостей поверхні.

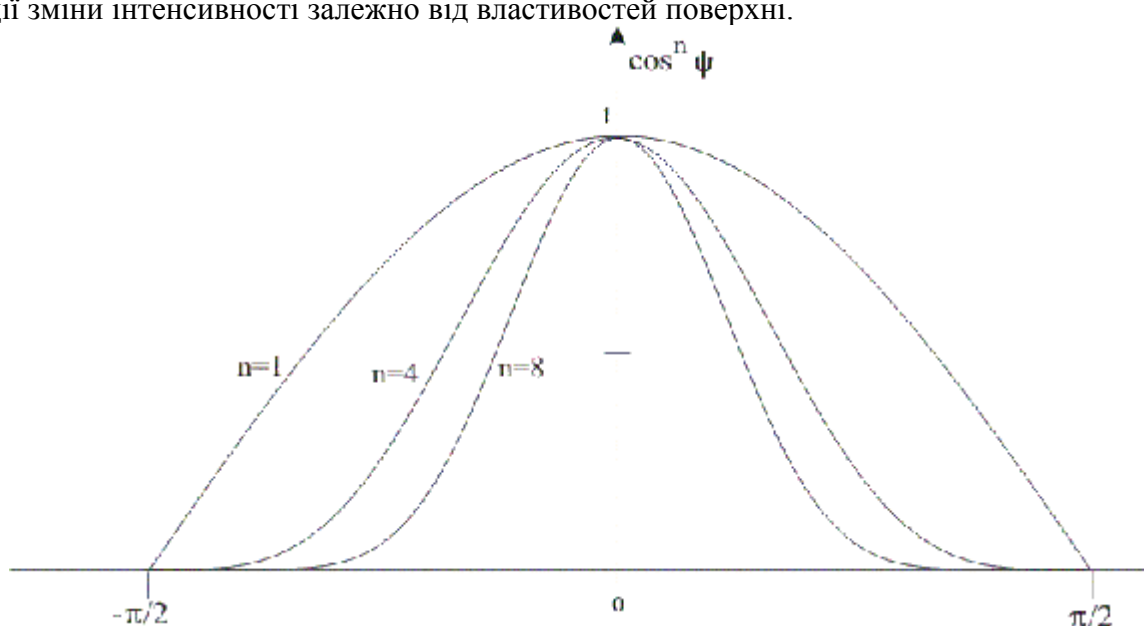

Рис. 9.4. дзеркальне відображення

Тепер модель освітленості, що враховує дзеркальне і дифузне відбиття, можна описати формулою

$$I = I_F k_F + \frac{I_S(k_s \cos\varphi + \omega(\varphi, \lambda) \cdot \cos^n \psi)}{d + C}.$$

Використовуючи одиничні вектори $\overrightarrow{L}$ (_напрямок до джерела) і $\overrightarrow{n}$ (_зовнішня нормаль), косинус кута φ ___можна обчислити через скалярний добуток: $\cos\varphi = (\overrightarrow{L} \cdot \overrightarrow{n})$. Для розрахунку інтенсивності дзеркального відображення спочатку треба визначити відбитий вектор $\overrightarrow{r}$. Із мал. 9.3b видно, що $\overrightarrow{r} = |AB| \cdot \overrightarrow{n} - \overrightarrow{L}$. З іншого боку AB __є діагоналлю ромба $AA'BC$, тому $|AB| = 2 \cdot (\overrightarrow{L} \cdot \overrightarrow{n})$.

Враховуючи всі ці співвідношення, одержуємо формулу

$$I = I_F k_F + \frac{I_S\{k_s(\overrightarrow{L} \cdot \overrightarrow{n}) + \omega(\varphi, \lambda) \cdot [2 \cdot (\overrightarrow{L} \cdot \overrightarrow{n}) \cdot (\overline{e}$$
$$d + C$$

В алгоритмах зафарбовування з використанням колірних моделей інтенсивність розраховується для кожного з базових квітів, оскільки зміна інтенсивності при дзеркальному відображенні залежить від довжини хвилі.

Зафарбування граней

Плоске зафарбовування

Якщо припустити, що джерело світла знаходиться на нескінченності, то промені світла, що падають на поверхню, паралельні між собою. Якщо до цього додати умову, що спостерігач перебуває в нескінченно віддаленій точці, то ефектом ослаблення світла зі збільшенням відстані від джерела також можна знехтувати. Крім того, таке положення спостерігача означає ще й те, що вектори, спрямовані від різних точок поверхні до спостерігача, також будуть паралельні. При виконанні всіх цих умов, як випливає з формули (9.6), плоска грань у всіх точках має однакову інтенсивність освітлення, тому вона зафарбовується одним кольором. Таке зафарбовування називається плоским.

Якщо ми апроксимуємо деяку гладку поверхню багатогранником, то при плоскому зафарбовуванні неминуче виявляться ребра, оскільки сусідні грані з різними напрямками нормалей мають різний колір. Ефект смуг Маху додатково підсилює цей недолік. Для його усунення при використанні цього способу зафарбовування можна лише збільшити число граней багатогранника, що призводить до збільшення обчислювальної складності алгоритму.

Зафарбування методом Гуро

Один із способів усунення дискретності інтенсивності зафарбовування був запропонований Гуро. Його метод полягає в тому, що використовуються не нормалі до плоских гранях, а нормалі до апроксимованої поверхні, побудовані у вершинах багатогранника. Після цього обчислюються інтенсивності у вершинах, а потім у всіх внутрішніх точках багатокутника виконується білінійна інтерполяція інтенсивності. Метод поєднується з алгоритмом рядковому сканування. Після того як грань відображена на площину зображення, для кожної що сканує рядки визначаються її точки перетину з ребрами. У цих точках інтенсивність обчислюється за допомогою лінійної інтерполяції інтенсивності в вершинах ребра. Потім для всіх внутрішніх точок багатокутника, що лежать на скануючого рядку, також обчислюється інтенсивність методом лінійної інтерполяції двох отриманих значень. На мал. 9.5 зображений плоский багатокутник з обчисленими значеннями інтенсивності в вершинах.

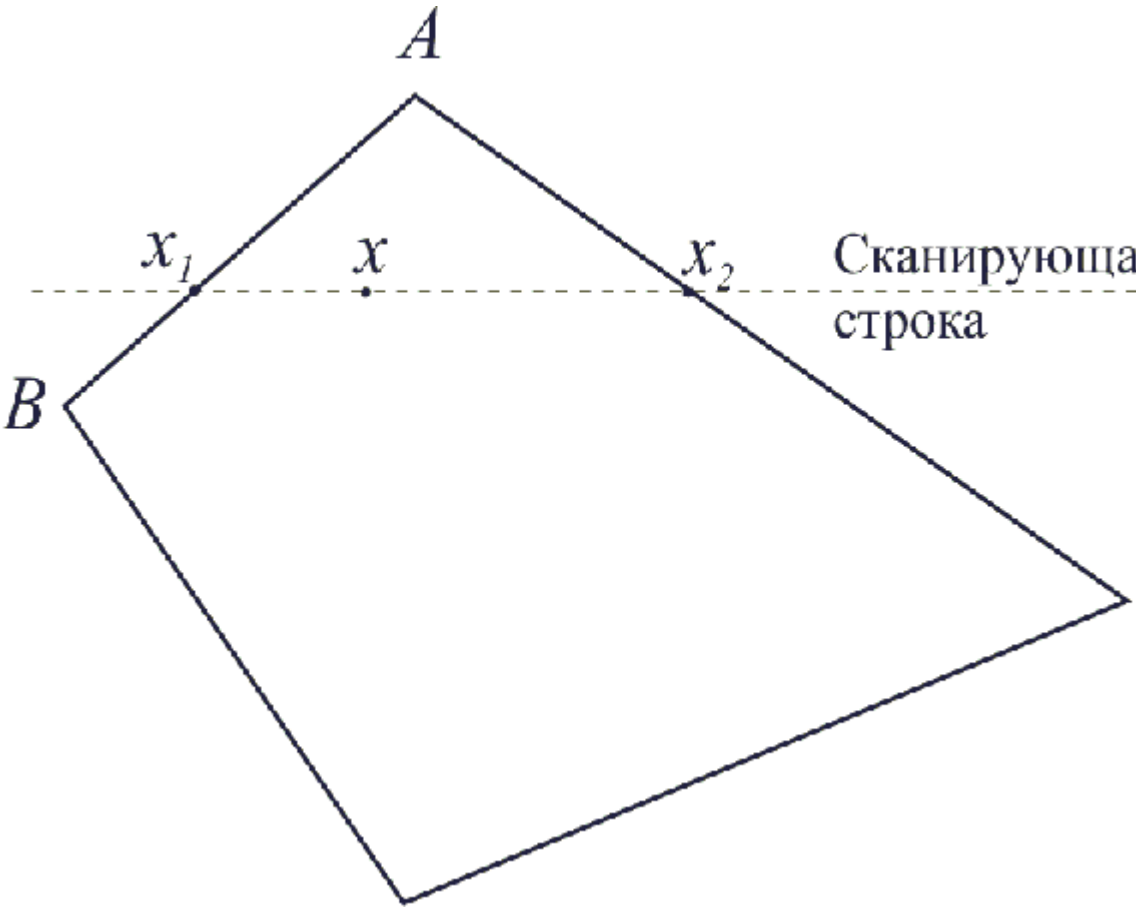

Рис. 9.5. інтерполяція інтенсивності

Нехай I_A, I_B, I_C - інтенсивність в вершинах A, B, C, x_A, x_B, x_C - горизонтальні точок. Тоді в точках перетину сканує рядки з ребрами багатокутника інтенсивності можна обчис формулами інтерполяції:

$$I_1 = t_1 I_A + (1 - t_1) I_B, \quad t_1 = \frac{x_1 - x_B}{x_A - x_B},$$
$$I_2 = t_2 I_A + (1 - t_2) I_C, \quad t_2 = \frac{x_2 - x_C}{x_A - x_C}.$$

Після цього інтенсивність в точці x отримаємо шляхом інтерполяції значень на кінцях відрізка:

$$I = t I_1 + (1 - t) I_2, \quad t = \frac{x_2 - x}{x_2 - x_1}$$

До недоліків методу Гуро варто віднести те, що він добре працює тільки з дифузною моделлю відбиття. Форма відблисків на поверхні і їх розташування не можуть бути адекватно відтворені при інтерполяції на багатокутника. Крім того, є проблема побудови нормалей до поверхні. В алгоритмі Гуро нормаль у вершині багатогранника обчислюється шляхом усереднення нормалей до граней, що примикають до цієї вершини. Така побудова сильно залежить від характеру розбивки.

Зафарбування методом Фонга

Фонг запропонував замість інтерполяції інтенсивності справити інтерполяцію вектора нормалі до поверхні на скануючого рядку. Цей метод вимагає великих обчислювальних витрат, оскільки формули інтерполяції (9.6) - (9.7) застосовуються до трьох компонентів вектора нормалі, але зате дає кращу апроксимацію кривизни поверхні. Тому дзеркальні властивості поверхні відтворюються набагато краще.

Нормалі до поверхні у вершинах багатогранника обчислюються так само, як і в методі Гуро. А потім виконується білінійна інтерполяція в сполученні з порядковим скануванням. Після побудови вектора нормалі в черговій крапці обчислюється інтенсивність.

Рис. 9.6. Три метода замальовування

Цей метод дозволяє усунути ряд недоліків методу Гуро, але не всі. Зокрема, ефект смуг Маху в окремих випадках у методі Фонга буває навіть сильніше, хоча в переважній більшості випадків апроксимація Фонга дає кращі результати. На малий. 9.6 наведені результати зафарбовування поверхні обертання, апроксимувати багатогранником, який складений із трикутних граней: a) - плоске зафарбовування, b) - зафарбовування по методу Гуро, c) - зафарбовування по методу Фонга. Перший з варіантів дає зображення ребристої поверхні з дуже контрастними переходами від однієї грані до другої. Друга модель дає більш гладке зображення, але в районі відблисків чітко спостерігаються лінії ребер, хоча й згладжені. Третій варіант вийшов найбільш гладким, дзеркальні відблиски мають досить реалістичну форму.

Більш складні моделі висвітлення

Коли ми розглядали алгоритми видалення невидимих ліній, передбачалося, що сцена включає тільки непрозорі об'єкти. У простій моделі висвітлення теж мова йшла про непрозорі поверхні. Тепер можна ускладнити завдання, включивши в модель не тільки відбиття світла, але і заломлення.

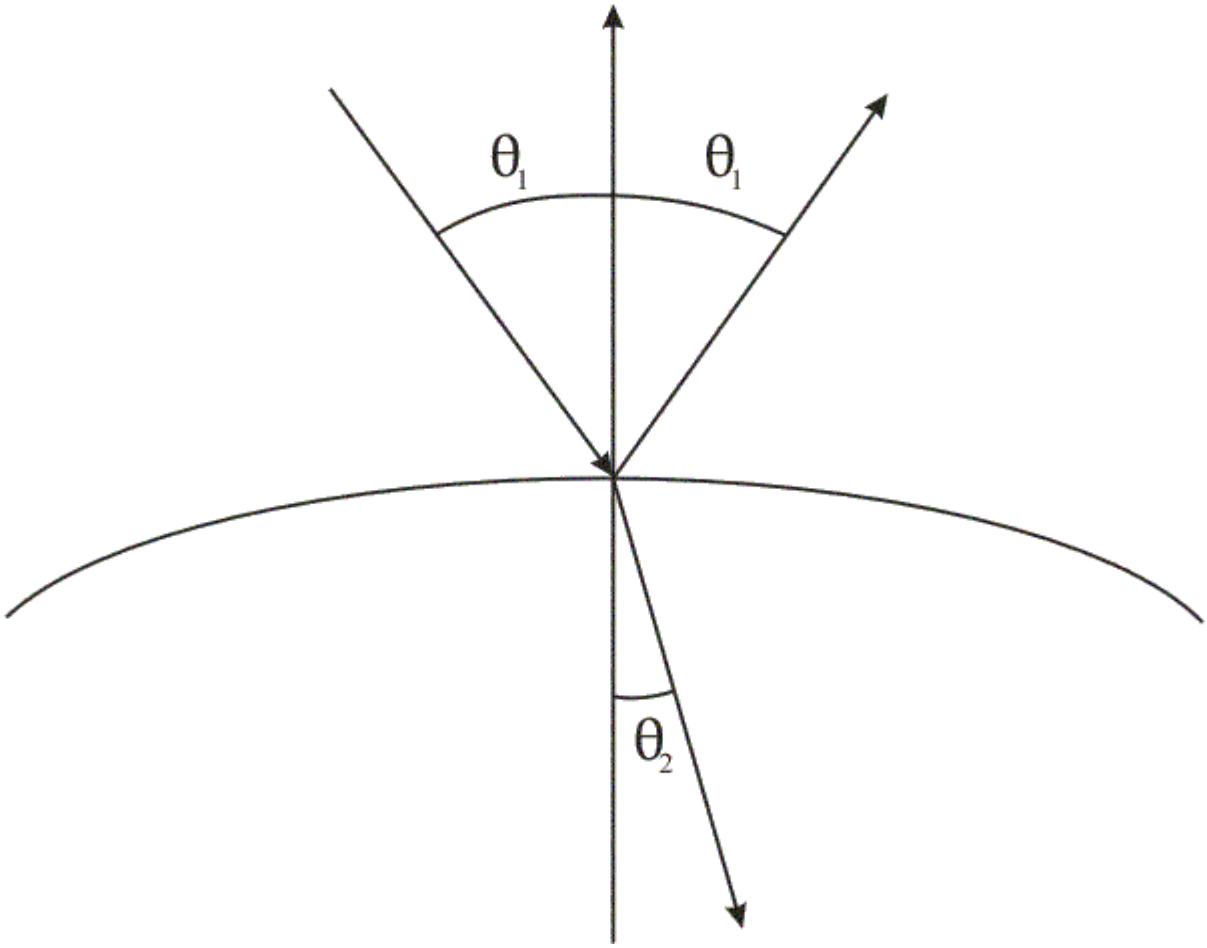

Рис. 9.7. Заломлений і відбитий промені

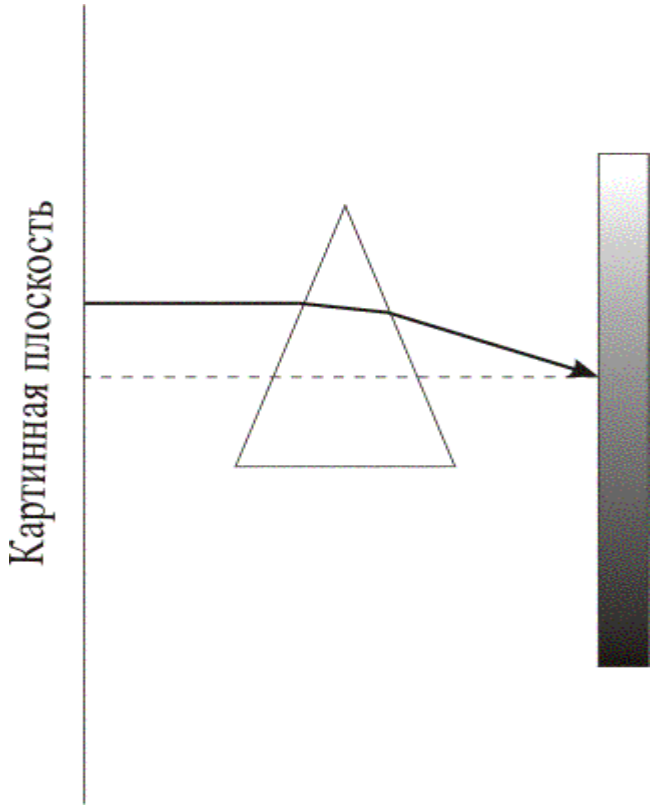

Рис. 9.8. Переломлення в призмі

При переході променя з одного середовища в іншу його напрямок змінюється відповідно до закону Сінелліуса: заломлений промінь лежить в площині, утвореною

нормаллю до площини і падаючим променем, а кути, утворені променями з нормаллю, зв'язані формулою

$\eta_1 \sin\theta_1 = \eta_2 \sin\theta_2$, де η_1, η_2 - показники заломлення двох середовищ (мал. 9.7).

Пропущення світла також може бути дифузним (якщо частина енергії світла розсіюється середовищем) або спрямованим. У першому випадку ми маємо справу з напівпрозорими тілами, які змінюють забарвлення видимих крізь них об'єктів. У другому випадку тіло є прозорим, і воно візуально виявляється тільки завдяки перекручуванням об'єктів за рахунок переломлення променів.

При наявності в просторовій сцені прозорих або напівпрозорих об'єктів треба враховувати, що зображення інших об'єктів буде відрізнятися від звичайної проекції на картинну площину (Мал. 9.8). Ці ефекти добре знайомі всім, хто стикався з різними лінзами. Для побудови зображення таких сцен доцільно використовувати алгоритми зі зворотним трасуванням променів.

Для зображення напівпрозорих поверхонь без урахування заломлення можна ввести так званий коефіцієнт прозорості κ, який дозволяє змішувати інтенсивності для видимої поверхні й тієї, що розташована за нею: $I = \kappa I_1 + (1-\kappa) I_2, \quad 0 \le \kappa \le 1$

При $\kappa = 1$ поверхня непрозора, при $\kappa = 0$ - повністю прозора. Для напівпрозорих тіл необхідно враховувати їх об'ємну структуру.

Методи побудови зображень сцен з прозорими і напівпрозорими об'єктами будуть більш детально розглянуті в наступній лекції.

Усунення ступінчастості (антиалайсінг)

При побудові растрового образа ліній (див. лекцію 8) ми стикаємося з ефектом ступінчастості, пов'язаним з дискретизацією безперервного об'єкта. Спотворення ідеального образу відбувається тому, що з усього безлічі крапок ми вибираємо тільки ті, які

опиняються ближче всього до центру елемента растра, і ініціалізували цей елемент.

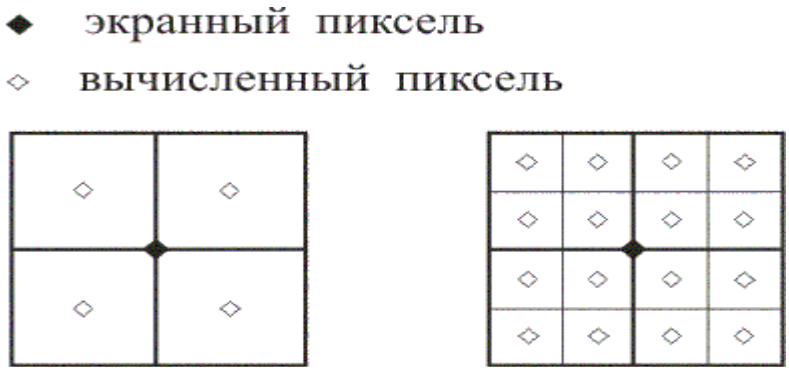

Рис. 9.9. Розподіл при збільшенні зображення в 4 рази

1	2	3	4	3	2	1
2	4	6	8	6	4	2
3	6	9	12	9	6	3
4	8	12	16	12	8	4
3	6	9	12	9	6	3
2	4	6	8	6	4	2
1	2	3	4	3	2	1

Рис. 9.10. Розподіл при збільшенні зображення в 16 разів

Для запобігання сильних перекручувань у цьому випадку можна, по-перше, підвищувати дозвіл растра, що дозволяє відображати все більш дрібні деталі об'єктів. Але у цього підходу є свої чисто фізичні обмеження. Другий підхід полягає в тому, що растр розраховується з більш високою роздільною здатністю, а зображується з більш низьким - шляхом усереднення атрибутів пікселів першого більш детального растра з певними вагами. Якщо ваги однакові, то ми отримуємо рівномірне усереднення, як показано на мал. 9.9. Кращих результатів можна досягти, якщо використовувати різні ваги у пікселів перший растра. На рис. 9.10 показано розподіл ваг при деталізації пікселя екранного растра.

Інший метод усунення ступінчастості полягає в тому, щоб розглядати піксель не як точку, а як деяку кінцеву область. В алгоритмах побудови растрової розгортки піксель вважається приналежним області зафарбовування, якщо його центр знаходився всередині ідеального образу області. Якщо рисюнок чорно-білий, то усунути ефект ступінчастості растра практично неможливо. Але при наявності відтінків півтонів можна задати інтенсивність кольору пікселя в залежності від площі його перетину з областю.

Розглянемо застосування цього методу на прикладі розрисьовки багатокутника. Ребро багатокутника будується з використанням алгоритму Брезенхема, описаного в лекції 8. Тут в цей алгоритм будуть внесені зміни, що включають параметр максирисьного числа рівнів інтенсивності. Визначаючи приналежність пікселя багатокутника, ми будемо використовувати в якості помилки е частку площі, що належить ідеальної фігури (рис. 9.11).

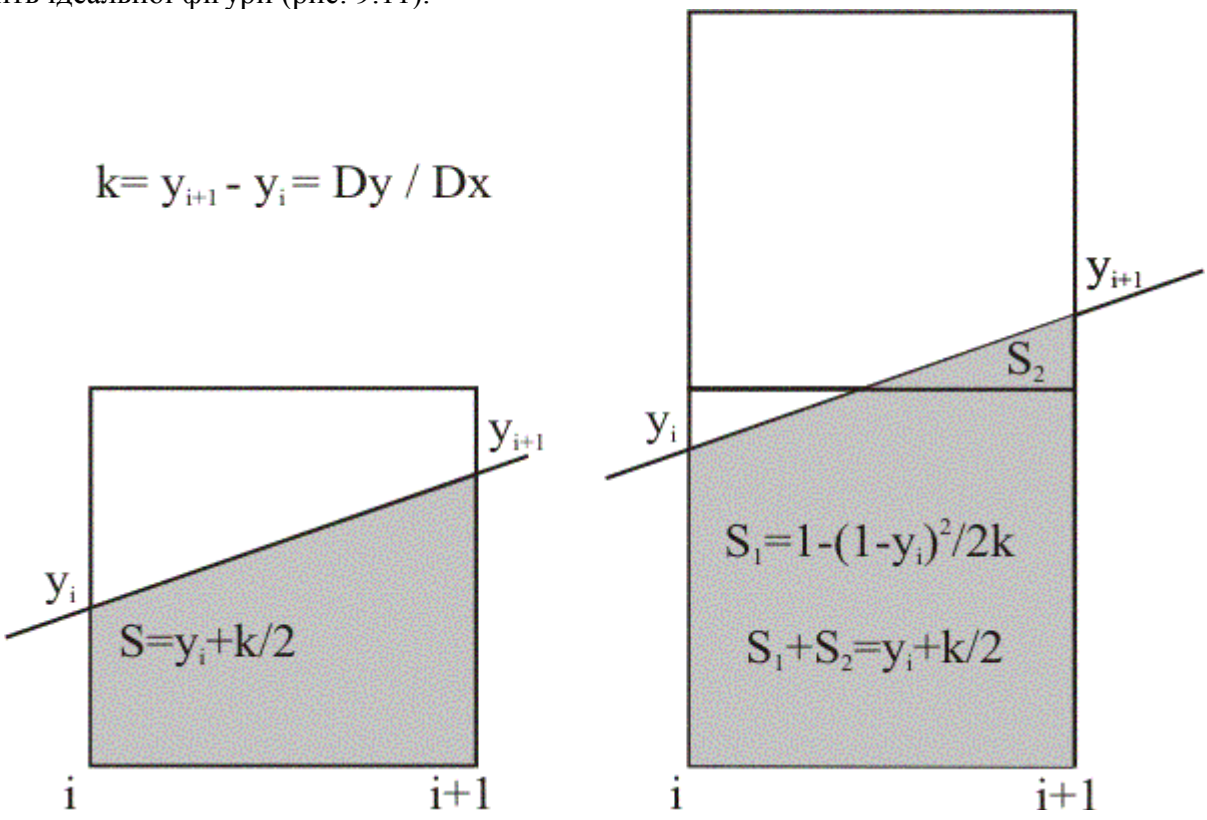

Рис. 9.11. Відрізка, що відсікається площа пікселя

Розглянемо знову випадок, коли відрізок спрямований у позитивний квадрант координатної площини під кутом, меншим. Ідеальний відрізок при заданому значенні цілочисельний координати може перетинати один або два пикселя. У попередній версії алгоритму вибирався піксель, центр якого розташовувався ближче до відрізка. Тепер інтенсивність для обох пікселів буде задаватися залежно від степеню близькості кожного з них. Ініціалізація пікселя буде використовувати інтенсивність як параметр.

Передбачається, що відрізок починається з кута першого пікселя, виходячи з чого і задається початкова інтенсивність. Блок-схема алгоритму приведена на рис. 9.12.

Усунення ефекту ступінчастості з математичної точки зору є завданням згладжування. Наведений тут алгоритм, що використовує площі перетинання растра й ідеального образу, можна описати за допомогою операції згортки функції. Спочатку дамо необхідні визначення. згорткою функції

$f(x)$ називається інтеграл виду

$$C(\xi) = \int_{-\infty}^{+\infty} K(\xi - t) f(t) dt. \qquad (9.9)$$

Функція $K(x)$ називається ядром згортки. В якості ядра згортки зазвичай використовується або функція з кінцевим носієм (тобто відмінна від нуля лише на деякому кінцевому інтервалі), або швидко спадна на нескінченності функція (це може бути необхідною умовою існування інтеграла). Розглянемо як згортальної функції та ядра наступні функції:

$$f(x) = \begin{cases} x \text{ при } 0 \le x \le 1 \\ 0 \text{ при } x < 0 \text{ и } x > 1 \end{cases}, \qquad K(x) = \begin{cases} 1 \text{ при } 0 \le x \le 1 \\ 0 \text{ при } x < 0 \text{ и } x > 1 \end{cases}.$$

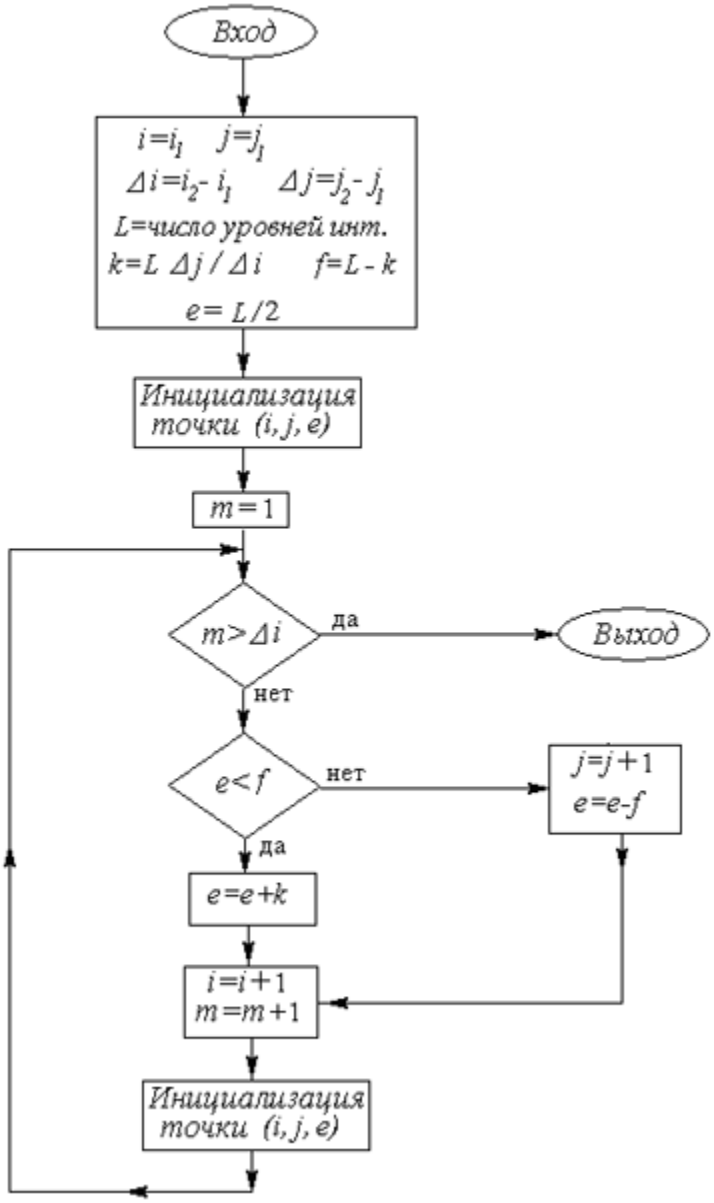

Рис. 9.12. Блок-схема модифікованого алгоритму Брезенхема

Тоді, в силу того, що підінтегральний вираз звертається в нуль при $\xi - t < 0$ и при , отримаємо

$$C(\xi) = \int_{\xi-1}^{\xi} f(t)dt.$$

$\xi - t > 0$

Враховуючи вид функції $f(x)$, отримуємо, що згортка буде відмінна від нуля тільки на інтервалі $0 < \xi < 2$. Значення згортки в деяких точках наведені в таблиці 9.1

.

Таблиця 9.1 Значення згортки у вузлах

ξ	0	1/2	1	3/2	2
$C(\xi)$	0	1/8	1/2	3/8	0

Очевидно, що наша згортка дає площа перетинання трикутника, утвореного згортальної функцією з квадратом, підстава якого є відрізок $[\xi, xi+1]$ на осі OX.

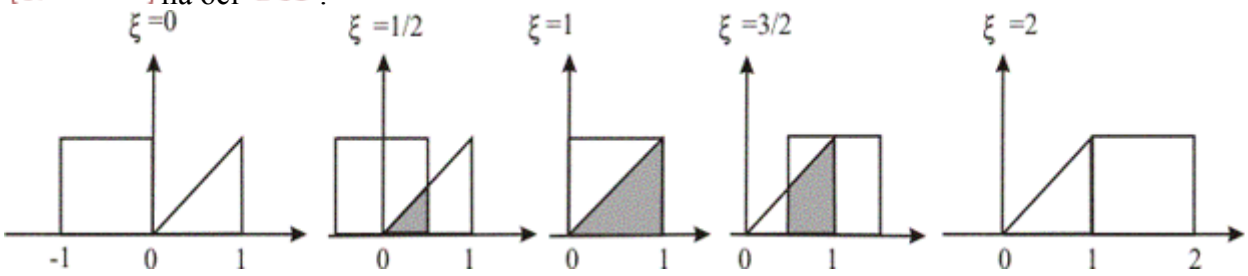

Рис. 9.13. Фігури, відповідні значенням згортки з таблиці

На рис. 9.13 приведений вид для всіх п'яти випадків з таблиці 9.1. Якщо порівняти ці результати з рис . 9.11, то видно, що значення згортки при $k \leq 1$ дають площа тієї частини пікселя, що знаходиться всередині багатокутника (якщо вважати $y_i = 0$), а при $k > 1$ - суму площ двох пересічних пікселя.

На закінчення проілюструємо результат застосування алгоритму усунення ступінчастості на прикладі зображення, отриманого за допомогою програми Corel Draw. Ця програма являє собою розвинений графічний редактор, дозволяє будувати об'єкти векторної графіку.На рис. 9.14 показано зображення простих графічних примітивів, попередньо переведене в растрову форму, на якому при великому збільшенні помітно згладжування із застосуванням відтінків сірого кольору.

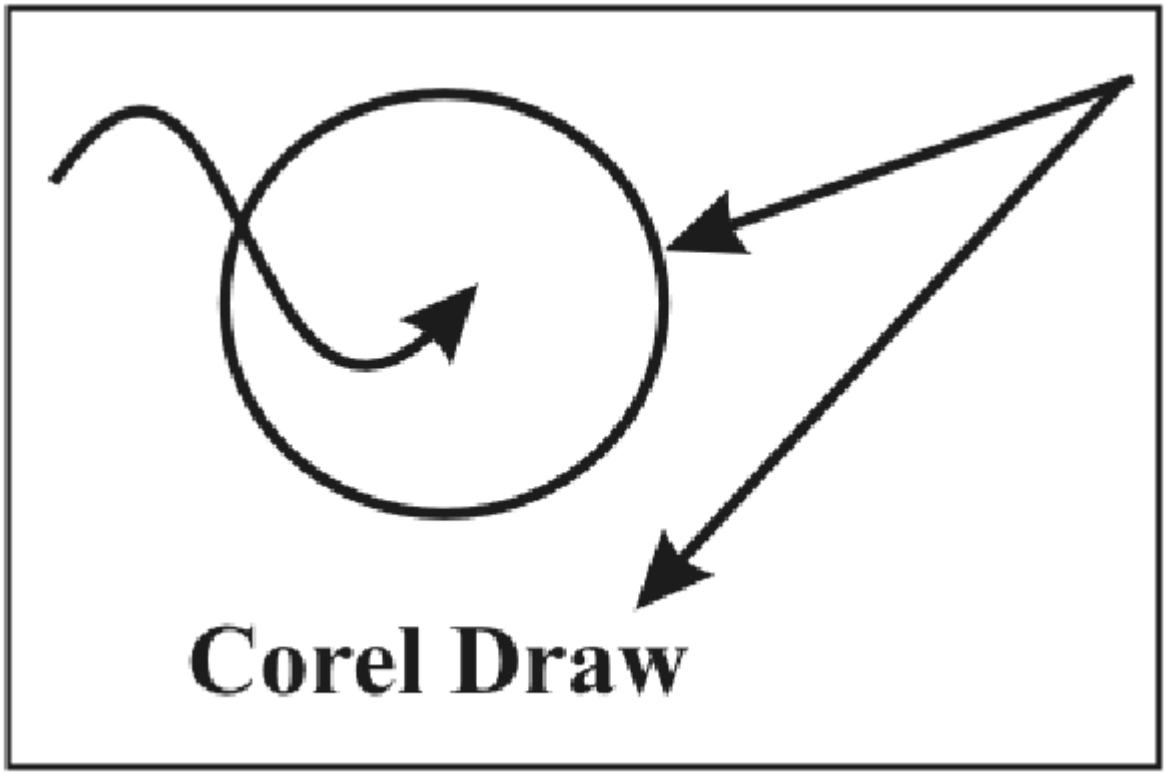

Рис. 9.14. згладжені зображення

Питання та вправи

1. Що таке ефект смуг Маху?
2. Чим відрізняється дифузне віддзеркалення від дзеркальної?

3. Від чого залежить інтенсивність освітлення точки поверхні при дифузному відображенні?

4. Від чого залежить інтенсивність освітлення точки поверхні при дзеркальному відображенні?

5. Які параметри враховує модель дзеркального відображення, запропонована Фонг?

6. Чи змінюється інтенсивність освітлення при плоскому зафарбовуванні грані?

7. Який параметр інтерполюється при зафарбовуванні методом Гуро?

8. Який параметр інтерполюється при зафарбовуванні методом Фонга?

9. У чому полягає ідея алгоритмів антиалайсінга, заснованих на рівні деталізації растра?

10. Який параметр використовується в алгоритмі антиалайсінга, що враховує розміри пікселя?

Лекція 10. Візуалізація просторових реалістичних сцен

Зміст

- Світлозвукові тіньової аналіз
- Метод излучательной
- Глобальна модель висвітлення із трасуванням променів
- Текстури
- Питання та вправи

Світло-тіньової аналіз

Тіні виникають як результат екранування джерела світла предметами, складовими сцену. При цьому існують повні тіні і півтіні, що пов'язано з фізичною природою світла: повна тінь утвориться в центральній частині затінення, а півтінь - поблизу її кордонів. У нашому курсі ми будемо стосуватися тільки повної тіні, хоча й існують технології зображення півтіней, що супроводжуються великим об'ємом обчислень. Якщо вважати, що спостерігач перебуває в одній точці з джерелом світла, то тіні в спостережуваній сцені не виникають: затінюється область є невидимою. У всіх інших випадках тіні видні. Джерело світла може перебувати на нескінченності або на кінцевому відстані, причому в другому випадку він може опинитися в полі зору спостерігача.

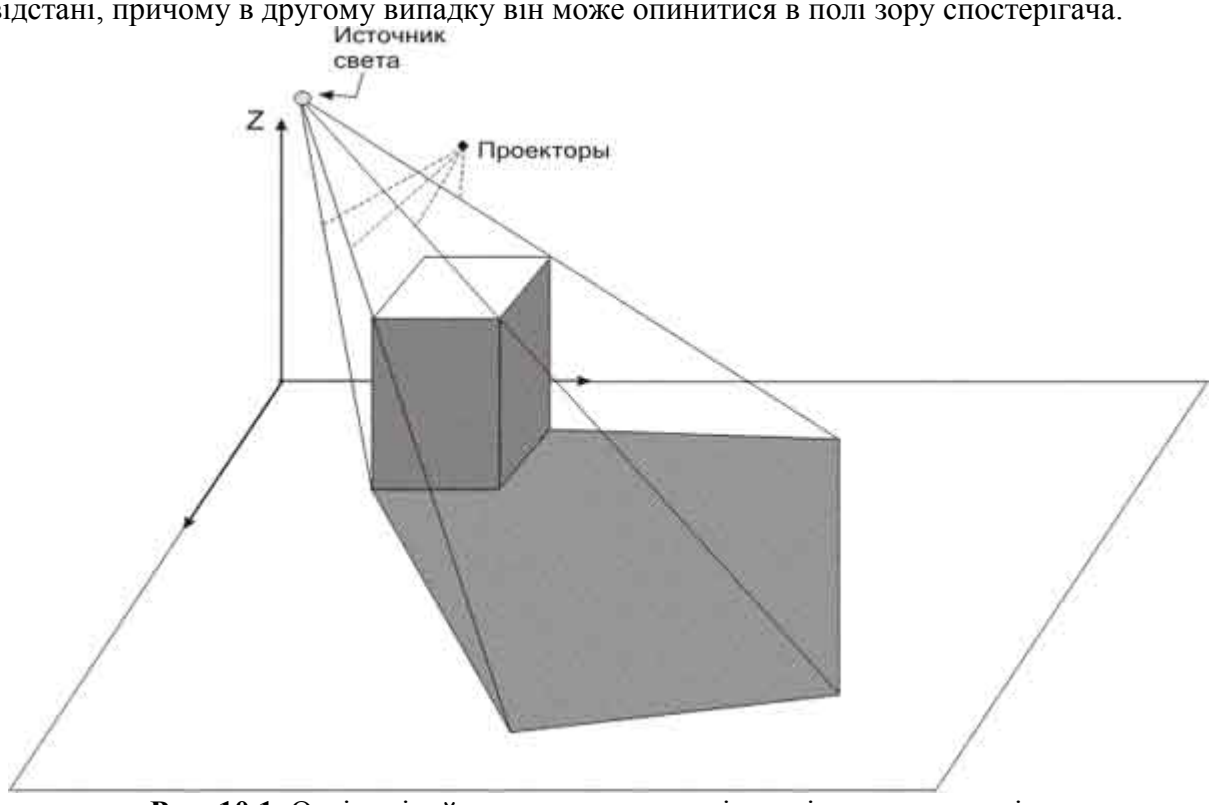

Рис. 10.1. Освіта тіней при остаточному відстані до джерела світла

Для нескінченно віддаленого джерела світла тіні на картинній площині виходять у результаті паралельної проекції об'єктів, а для близького джерела - центральної проекції. Об'єкти та їх частини стають невидимими, якщо вони потрапляють в область тіні, тому при побудові зображення завдання про видалення невидимих областей вирішується двічі: щодо спостерігача в процесі проектування і щодо джерела світла. При визначенні розташування тіней будуються проекції невидимих з позиції джерела світла граней

(неосвітлених) на картинну площину, в результаті чого виходять тіньові багатокутники. Ці багатокутники будуються для всіх об'єктів сцени і заносяться в список. Відзначимо, що тіньові багатокутники не залежать від положення спостерігача, тому при огляді сцен з різних точок зору вони будуються тільки один раз. Неосвітлені області визначаються тими ж методами, які були описані раніше в **лекції 7**, а для побудови проекцій використовуються методи, деякі з яких описані в **лекції 8**. Зокрема, можна застосовувати матриці проекцій в однорідній системі координат. Вперше ідея сполученого аналізу видимості і затіненості була запропонована в 1968 р. Аппель. В якості прикладу розглянемо один алгоритм на основі рядковому сканування, що складається з двох основних етапів.

I. Аналіз сцени по відношенню до джерела світла.

Для всіх багатокутників, отриманих в результаті проектування сцени, визначаються неосвітлені (затінені) ділянки і тіньові багатокутники (проекційні тіні), причому багатокутники утворять пронумерований список. Для цих багатокутників формується матриця $A = (a_{ij})$, що дозволяє визначити, відкидає чи багатокутник тінь і які з багатокутників він може закривати. Якщо для деяких значень $i, j \quad a_{ij} = 1$, то це означає, що багатокутник з номером i може відкидати тінь на багатокутник з номером j.

Таким чином, на цьому етапі основним є питання про ефективне алгоритмі побудови такої матриці. Якщо проекція включає N багатокутників, то необхідно розглянути "взаємини" $N \cdot (N-1)$ пар багатокутників. Скоротити перебір можна за рахунок занурення об'єктів в прямокутні або сферичні оболонки або шляхом використання сортування по глибині.

II. Аналіз сцени по відношенню до спостерігача

Виконуються два процеси сканування. Перший - для визначення відрізків, видимих з позиції спостерігача (про нього розповідалося раніше в **лекції 7**). Другий-для визначення перетинань відрізків з тіньовими багатокутниками зі списку. Рекурсивний алгоритм другого сканування складається з чотирьох основних кроків.

Для кожного видимого відрізка

1. Якщо немає жодного тіньового багатокутника, то відрізок зображується з основною інтенсивністю.
2. Якщо багатокутник, що містить видимий відрізок, не перетинається з тіньовими багатокутниками, то відрізок зображується з основною інтенсивністю.
3. Якщо відрізок повністю закривається деякими тіньовими багатокутниками, то інтенсивність його зображення визначається з урахуванням затінення всіма цими багатокутниками.
4. Якщо кілька тіньових багатокутників частково закривають відрізок, то він розбивається на ряд відрізків крапками перетинання з тіньовими багатокутниками.

Цей алгоритм припускає, що затінення не абсолютне, т.е.затененние ділянки таки є видимими, тільки їх освітленість падає залежно від кількості і освітленості затінюючих багатокутників. При повному затіненні в третьому пункті алгоритму відрізок стає повністю невидимим, а в четвертому подальшому аналізу піддаються тільки незатененние відрізки.

Способи розрахунку інтенсивності при неповному затіненні можуть бути різні. У цьому випадку всі затінені багатокутники мають свою інтенсивність залежно від обраної моделі освітленості. При цьому можна враховувати відстань затіненої ділянки від поверхні, що відкидає тінь.

Ще один алгоритм, часто застосовуваний при побудові тіней, носить назву методу тіньового буфера. Він будується на основі методу Z-буфера, описаного в лекції 7. Тіньовий буфер - це той же Z-буфер, тільки з точки зору джерела світла. Таким чином, використовуються два буфери: один - для відстані від картинної площини до точок

зображуваної сцени, а інший - для відстаней від цих же точок до джерела світла. Алгоритм дозволяє зображувати сцени з повним затіненням і зводиться до двох основних етапів:

1. Сцена розглядається з точки розташування джерела світла у відповідній системі координат. Підсумком побудови є повністю заповнений тіньовий буфер.

2. Сцена розглядається з точки зору спостерігача, застосовується звичайний метод Z-буфера з невеликим доповненням. Якщо точка (x, y, z) є видимою в цій системі координат, то обчислюються її координати в системі, пов'язаної з джерелом світла - (x', y', z'), потім перевіряється, чи є точка видимої із цієї позиції. Для цього значення z' порівнюється зі значенням, що містяться в тіньовому буфері для цієї крапки, і в разі видимості значення інтенсивності заноситься в буфер кадру в крапці (x, y).

Обидва наведених алгоритму працюють у просторі зображення, тобто мають справу з проекціями на площину і деяку додаткову інформацію про точках сцени, які відповідають цим проекціям. Існують алгоритми, працюють у тривимірному об'єктному просторі. Зокрема, для побудови тіней використовуються модифікації алгоритму Вейлера-Азертон, описаного в **лекції 7**. Модифікація полягає в тому, що, як і у випадку тіньового буфера, завдання видалення невидимих граней вирішується спочатку з позиції джерела світла, а потім отримана інформація про об'єкти використовується при побудові зображення з позиції спостерігача. У загальних рисах кроки алгоритму можна описати так:

1. Визначаються межі, видимі з точки розташування джерела світла. З метою підвищення ефективності запам'ятовується інформація тільки про видимі гранях. Оскільки аналіз виконується в системі координат, пов'язаної з джерелом світла, то отримані видимі багатокутники потім заново приводяться до вихідної системі координат. Багатокутники зв'язуються з гранями, яким вони належать (у результаті затінення одна грань може містити кілька багатокутників).

2. Сцена обробляється з положення спостерігача. При зображенні видимої грані враховуються тільки ті багатокутники, які входять в список, отриманий на першому етапі, тобто грань розглядається як сукупність таких багатокутників.

При наявності декількох джерел світла кількість освітлених ділянок природно збільшується.

Метод излучательной

У цій лекції вже говорилося, що освітленість поверхні визначається власним випромінюванням тіла і відбитими променями, що падають від інших тіл (джерел). Модель излучательной включає обидва ці чинника і заснована на рівняннях енергетичного балансу. При цьому виконуються розрахунки враховують тільки взаємне розташування елементів сцени і не залежать від положення спостерігача.

Уявімо сцену з N елементів (ділянок поверхонь). Освітленість будемо моделювати як кількість енергії, що випромінюється поверхнею. Для кожного елемента ця кількість енергії складається із власної енергії і відбитої частки енергії (E_k), отриманої від інших об'єктів. Передбачається, що для кожної пари елементів з номерами i, j можна визначити, яка частка енергії одного попадає на інший (w_{ij}). Нехай α_i - коефіцієнт відбиття енергії $i - \text{м}$ елементом.Тогда повна енергія, випромінювана цим елементом, буде визначатися рівнянням

$$U_i = E_i + \alpha_i \sum_{j=1}^{N} w_{ij} U_j .$$

Таким чином, ми отримуємо систему рівнянь для знаходження значень U_i, яка в матричному вигляді виглядає таким чином:

$$(I - W) \cdot U = E$$

де I - одинична матриця, U і T - вектори випромінюваної й власної енергій, а матриця W складається з елементів $(\alpha_i w_{ij})$. Оскільки частина випромінювання елемента може не попадати ні на один з решти, то

$$\sum_{i=1}^{N} w_{ij} \leq 1,$$

а це умова в поєднанні з тим, що $\alpha_i < 1$ (відображення не є повним), приводить до того, що матриця системи має так зване **діагональне переважання**, тобто діагональний елемент по абсолютній величині більше, ніж сума інших елементів рядка. У такому випадку система рівнянь має рішення, яке можна знайти за допомогою чисельних методів.

Отже, кроки алгоритму зображення сцени зводяться до наступних:

1. Сцена розбивається на окремі ділянки, для кожного з яких визначаються значення. $E_i, \alpha_i, w_{ij}, \quad j = 1, 2, \ldots, N$

2. Знаходяться значення U_i для кожної з трьох основних компонентів кольору.

3. Для обраної точки спостереження будується проекція з видаленням невидимих граней і здійснюється зафарбовування, що використовує значення U_i для завдання інтенсивності. При цьому можуть використовуватися небудь алгоритми, що дозволяють згладити зображення.

Складним моментом у моделі излучательной є розрахунок коефіцієнтів w_{ij}.

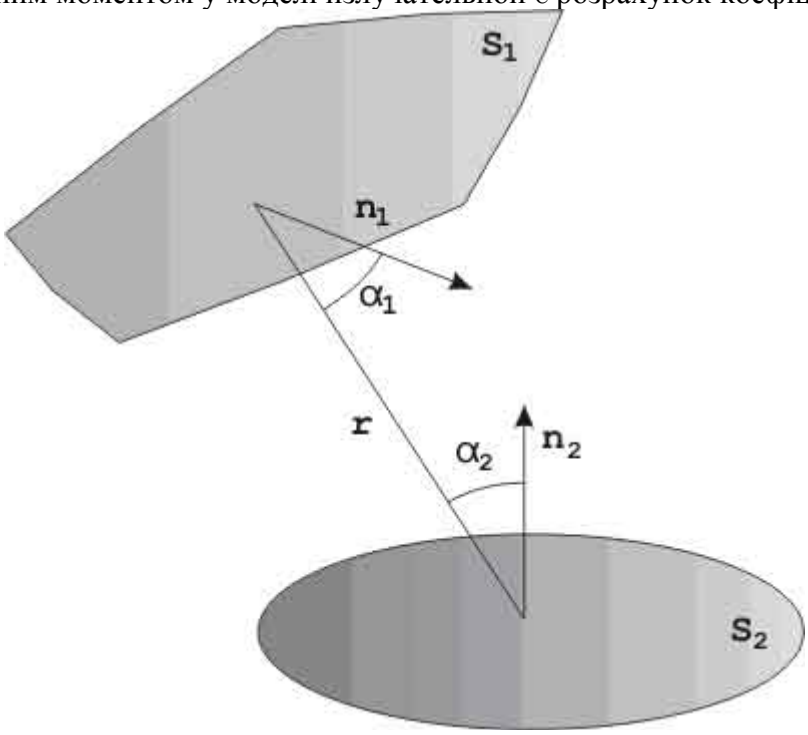

Рис. 10.2. Два елементи сцени

Розглянемо один приклад. Нехай є два елементи сцени S_1 і S_1 (рис. 10.2). Оскільки використовується дифузна модель освітлення, то частка енергії рисої ділянки dS_1 з норрислю $\overrightarrow{n}_1$, випромінювана під кутом α_1 до цієї норрисі, пропорційна косинусу кута. Отже, в напрямку елементарної ділянки dS_2 йде частка енергії, пропорційна косинусу кута між $\overrightarrow{n}_1$ і відрізком, який з'єднує ці ділянки. Відповідно, здобували другу ділянкою частка цієї енергії буде пропорційна косинусу кута між норрислю $\overrightarrow{n}_2$ і цим же відрізком. Отже, частка енергії, одержувана елементом dS_2 від елемента, dS_1 -

$dw_{21} = \cos(\alpha_1) \cdot \cos(\alpha_2)/\pi r^2$, де r - відстань між елементами. Крім того, необхідно врахувати, що випромінювана елементарним ділянкою енергія рівномірно розподілена по всіх напрямках. І, нарешті, в кожній сцені одні об'єкти можуть частково екранувати інші, тому треба ввести коефіцієнт, що визначає ступінь видимості об'єкта з позиції іншого. Далі отримане вираження інтегрується по S_1 й S_2, що також може бути складним завданням.

Звідси видно, наскільки трудомісткої може виявитися процедура обчислення коефіцієнтів w_{ij}. Тому, як правило, використовуються наближені методи їх обчислення. Зокрема, можна розглядати поверхні об'єктів як багатогранники, тоді елементами сцени будуть плоскі багатокутники, для яких формули кілька спрощуються.

Глобальна модель висвітлення із трасуванням променів

Ми вже торкалися раніше поняття трасування променів при описі алгоритмів видалення невидимих граней. Тепер розглянемо аналогічну процедуру в застосуванні до моделей освітлення. У попередній главі були описані моделі освітленості від деякого джерела світла без урахування того, що самі об'єкти сцени висвітлюють один одного за допомогою відбитих променів. Метод излучательной, розроблений для дифузної моделі освітленості, вже враховує цей фактор.
Глобальна модель освітленості здатна відтворювати ефекти дзеркального відображення і заломлення променів (прозорість і напівпрозорість), а також затінення. Вона є складовою частиною алгоритму видалення невидимих поверхонь методом трасування.

Якщо розглянути сцену, що містить у числі інших дзеркальні й напівпрозорі поверхні (**рис. 10.3**), то зображення буде включати, по-перше, проекції самих об'єктів, освітлених одним або кількома джерелами світла. У деяких своїх частинах ці об'єкти будуть перекручені за рахунок переломлення променів у прозорих і напівпрозорих тілах. По-друге, частина об'єктів буде відбиватися дзеркальними поверхнями, і ці відбиття з'являться на проекціях дзеркальних об'єктів. У зображеній на **рис. 10.3** сцені крапки на поверхні призми $C,\ D$ видні на картинній площині двічі: один раз - крізь напівпрозорий паралелепіпед у вигляді крапок $\widetilde{C},\ \widetilde{D}$, а другий раз - як двічі відбиті невидимою поверхнею паралелепіпеда й дзеркалом $C'',\ D''$. Паралелепіпед в даному випадку частково володіє дзеркальними властивостями.

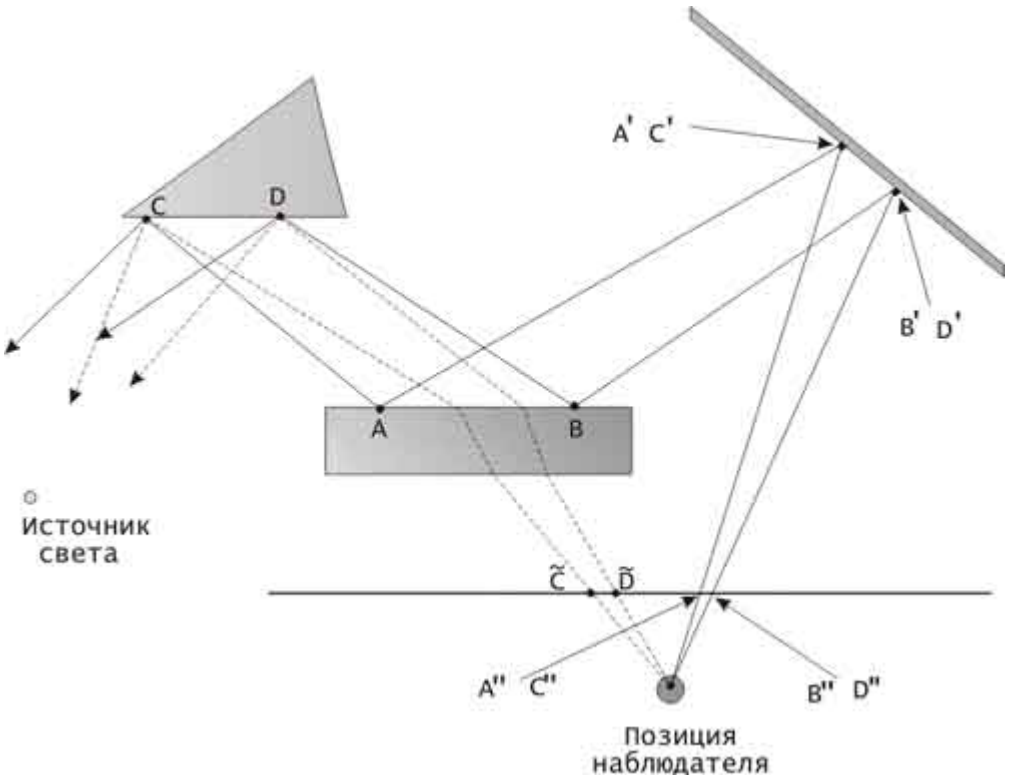

Рис. 10.3. Сцена, що містить дзеркальні й напівпрозорі поверхні

Глобальна модель висвітлення для кожного пікселя зображення визначає його інтенсивність. Будемо для простоти вважати, що всі джерела світла - точкові. Спочатку визначається безпосередня освітленість джерелами без урахування відображень від інших поверхонь (вторинна освітленість): відстежуються промені, спрямовані до всіх джерел. Тоді спостережувана інтенсивність (або відбита крапкою енергія) виражається наступним співвідношенням:

$$I = k_0 I_0 + k_d \sum_j I_j(\vec{n} \cdot \vec{l}_j) + k_r \sum_j I_j(\vec{s} \cdot \vec{r}_j)^\beta + k_r I_r + k_t I_t,$$

де k_0 - Коефіцієнт фонового (неуважного) висвітлення;

k_d - Коефіцієнт дифузного відбиття;

k_r - Коефіцієнт дзеркального відображення;

k_t - Коефіцієнт пропускання;

$\vec{n}$ - Одиничний вектор норрисі до поверхні в точці;

$\vec{l}_j$ - Одиничний вектор, спрямований к j -му джерелу світла;

$\vec{s}$ - Одиничний локальний вектор, спрямований у точку спостереження;

$\vec{r}_j$ - Відбитий вектор $\vec{l}_j$;

I_0 - Інтенсивність фонового освітлення;

I_j - Інтенсивність j -го джерела світла;

I_r - Інтенсивність, що приходить по дзеркально відбитого променя;

I_t - Інтенсивність, що приходить по заломлення променя.

В алгоритмі видалення невидимих ліній трасування променя тривала до першого перетину з поверхнею. У глобальній моделі висвітлення цим справа не обмежується: здійснюється подальша трасування відбитого і заломленого променів. Таким чином, відбувається розгалуження алгоритму у вигляді двійкового дерева. Процес продовжується до тих пір, поки чергові промені не залишаться без пересеченій.Отраженіе і переломлення

розраховуються за законами геометричної оптики, які вже розглядалися в попередньому розділі.

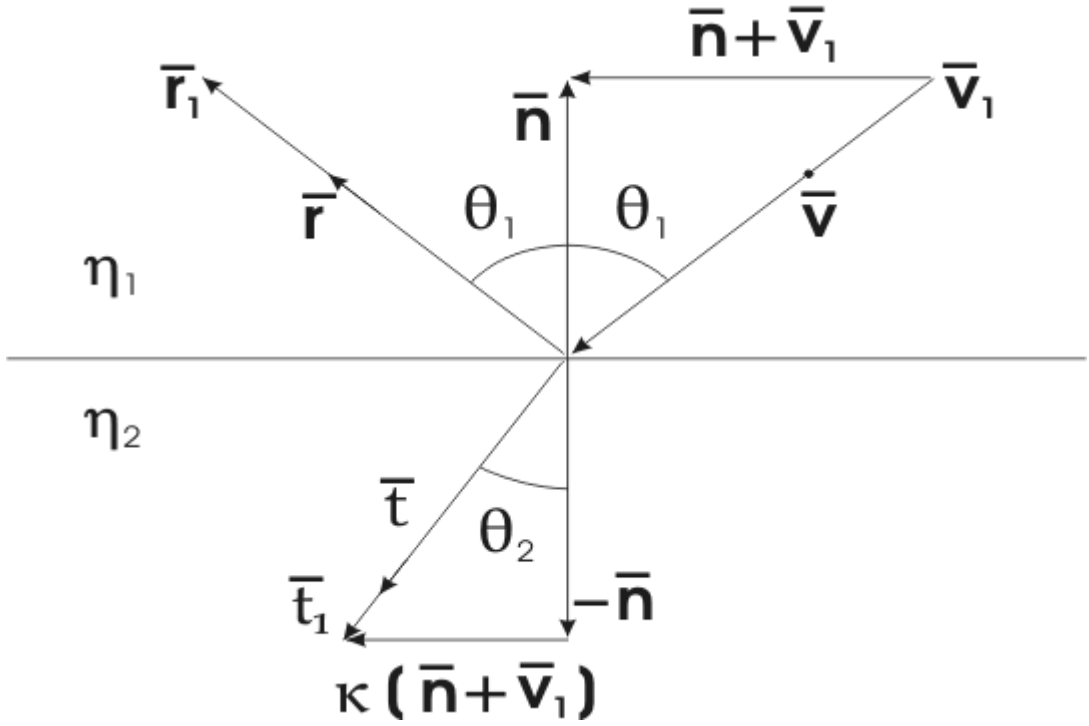

Рис. 10.4. Дзеркальне відображення і заломлення

Нехай $\vec{v}, \vec{r}, \vec{t}$ - напрямки падаючого, відбитого і заломленого променів (**рис. 10.4**), $\vec{v}_1 = \vec{v}/\cos(\theta_1)$, $\vec{n}$ - одинична зовнішня норрись, η_1, η_2 - коефіцієнти заломлення середовищ, розділених поверхнею. Тоді можна показати, що

$$\vec{r}_1 = \vec{v}_1 + 2 \cdot \vec{n}, \quad \vec{t}_1 = \kappa(\vec{n} + v_1) - \vec{n},$$
$$\kappa = (k_\eta^2 \cdot |\vec{v}_1|^2 - |\vec{v}|_1 + \vec{n}^2)^{-1/2}, \quad k_\eta = \frac{\eta_2}{\eta_1}.$$

Відповідні одиничні вектори отримати неважко.

Двійкове дерево променів можна будувати за принципом "ліве піддерево відповідає відбитому променю, а праве - заломлення". Після того як воно побудовано, можна обчислити інтенсивність у точці. Для цього здійснюється зворотний прохід від вершин до кореня, і при проходженні вузлів інтенсивність убуває

Теоретично дерево може виявитися нескінченним, тому при його побудові бажано задати максирисьну глибину, щоб уникнути переповнення пам'яті комп'ютера.

Оскільки значна частина променів, що виходить від джерел світла і інших поверхонь, не потрапляє в поле зору спостерігача, то відслідковувати їх все не має сенсу. Тому для формування зображення використовується зворотна трасування, тобто промені відслідковуються в зворотному порядку: від положення спостерігача через всі точки картинній площині до об'єктів і далі - по відбиття і заломлення променів.

Текстури

Текстура поверхні - це деталізація її будови, що враховує мікрорельєф і особливості забарвлення. По-перше, гладка поверхня може бути покрита яким-небудь візерунком, і тоді при її зображенні вирішується завдання відображення цього візерунка на проекції фрагментів поверхні (багатокутники). По-друге, поверхня може бути шорсткою, тому потрібні спеціальні прийоми імітації такого мікрорельєфу при зафарбовуванні.

Спочатку розглянемо методи відображення візерунків. Найчастіше візерунок

задається у вигляді зразка, заданого на прямокутнику в декартовій системі координат η, ξ в просторі текстури. Фрагмент поверхні може бути заданий у **параметричному вигляді** в тривимірній декартовій системі координат:

$$x = f(u,v), \quad y = g(u,v), \quad z = h(u,v), \quad a \le u \le b, \quad c \le v \le d.$$

Тепер досить побудувати відображення області в просторі текстури в область параметрів поверхні

$$u = \varphi(\eta,\xi), \quad v = \psi(\eta,\xi), \text{ або } \eta = \chi(u,v), \quad \xi = \theta(u,v),$$ і тим самим кожній точці поверхні буде відповідати точка зразка текстури. Нехай, наприклад, поверхня являє собою один октант сфери одиничного радіуса, заданий формулами

$$x = \sin\alpha \cdot \sin\beta, \quad y = \cos\beta, \quad z = \cos\alpha \cdot \sin\beta,$$
$$0 \le \alpha \le \pi/2, \quad \pi/4 \le \beta \le \pi/2$$

а зразок текстури заданий на квадраті $0 \le \eta \le 1, \quad 0 \le \xi \le 1$. Тоді можна скористатися лінійним відображенням виду $\alpha = a\eta + b, \quad \beta = c\eta + d.$ Якщо покласти $a = \pi/2, \quad b = 0, \quad c = -\pi/4, \quad d = \pi/2$, то кути зразка відобразяться в кути криволінійного чотирикутника, як це показано на рис.10.6.

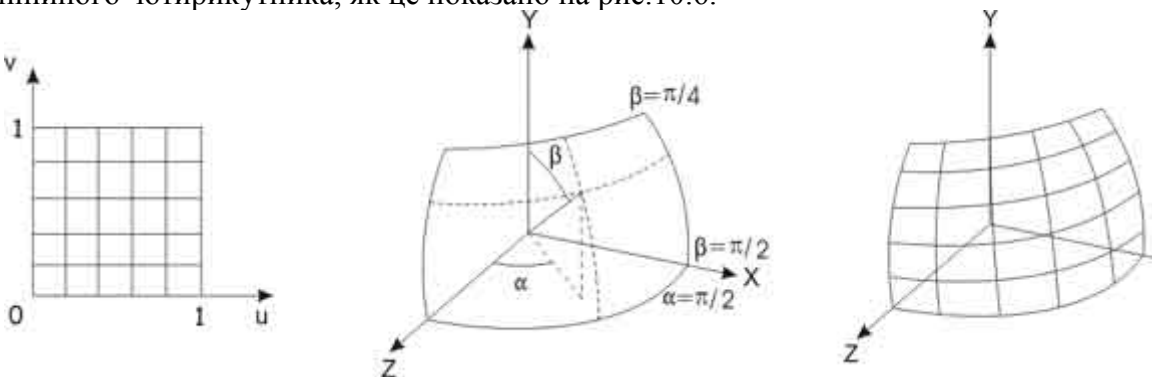

Рис. 10.5. Текстура на сферичній поверхні

Зворотне відображення має вигляд

$$\eta = \frac{\alpha}{\pi/2}, \quad \xi = \frac{\pi/2 - \beta}{\pi/4},$$

отже, вертикальні і горизонтальні лінії зразка відобразяться на окружності великого кола сфери.

Нехай тепер потрібно нанести текстуру при перспективному проектуванні довільно орієнтованої прямокутної грані. Грань задана в просторі набором своїх вершин A, B, C, D.

Побудуємо вектори $\vec{e}_1 = B - A$ і $\vec{e}_2 = D - A$, направлені вздовж сторін прямокутника. Яку точку прямокутника можна єдиним чином представити у вигляді

$$P = A + u\vec{e}_1 + v\vec{e}_2.$$

Будемо вважати, що використовується найпростіший випадок перспективного перетворення, що задається формулами

$$x' = \frac{x}{z}, \quad y' = yz.$$

Знайдемо образ точки P при такому перетворенні:

$$x' = \frac{A_x + u\vec{e}_{1x} + v\vec{e}_{2x}}{A_z + u\vec{e}_{1z} + v\vec{e}_{2z}}, \quad y' = \frac{A_y + u\vec{e}_{1y} + v\vec{e}_{2y}}{A_z + u\vec{e}_{1z} + v\vec{e}_{2x}},$$

або

$$\begin{cases} u(x'\vec{e}_{1z} - \vec{e}_{1x}) + v(x'\vec{e}_{2z} - \vec{e}_{2x}) = A_x - A_z x' \\ u(y'\vec{e}_{1z} - \vec{e}_{1y}) + v(y'\vec{e}_{2z} - \vec{e}_{2y}) = A_y - A_z y' \end{cases}.$$

Якщо тепер розглядати ці співвідношення як систему рівнянь для знаходження параметрів u, v, то, вирішивши її, одержимо необхідне зворотне перетворення. Для вирішення можна скористатися, наприклад, правилом Крамера:

$$u = \Delta_u/\Delta, \quad v = \Delta_v/\Delta$$

де

$$\Delta = (x'\vec{e}_{1z} - \vec{e}_{1x}) \cdot (y'\vec{e}_{2z} - \vec{e}_{2y}) - (x'\vec{e}_{2z} - \vec{e}_{2x}) \cdot (y'\vec{e}_{1z} - \vec{e}_{1y});$$
$$\Delta_1 = (A_x - A_z x') \cdot (y'\vec{e}_{2z} - \vec{e}_{2y}) - (x'\vec{e}_{2z} - \vec{e}_{2x}) \cdot (A_y - A_z y');$$
$$\Delta_2 = (x'\vec{e}_{1z} - \vec{e}_{1x}) \cdot (A_y - A_z y') - (A_x - A_z x') \cdot (y'\vec{e}_{1z} - \vec{e}_{1y}).$$

Знайдені параметри будуть визначати точку текстури, що відповідає крапці проекції.

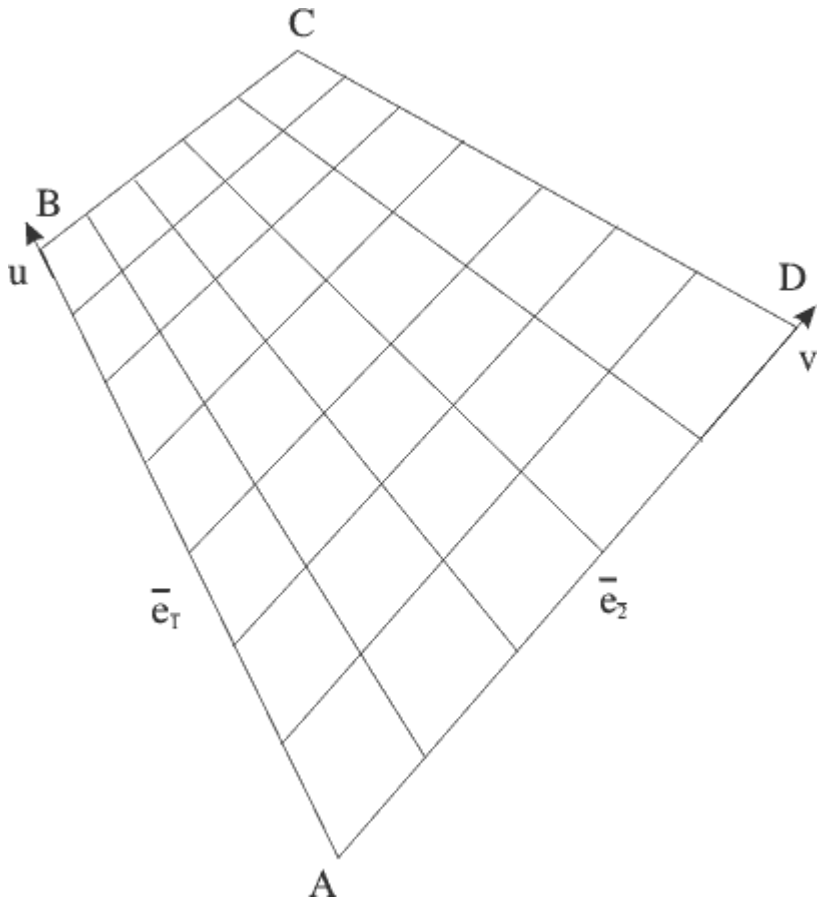

Рис. 10.6. Текстура при перспективній проекції

Можна розглянути більш загальний випадок перспективної проекції, що задається співвідношеннями

$$x' = \frac{x}{1 + \frac{z}{d}}, \quad y' = \frac{y}{1 + \frac{z}{d}}$$

Тоді рівняння для визначення u, v небагато ускладняться:

$$\begin{cases} u(x'\vec{e}_{1z}/d - \vec{e}_{1x}) + v(x'\vec{e}_{2z}/d - \vec{e}_{2x}) = A_x - (1 + A_z/d)x' \\ u(y'\vec{e}_{1z}/d - \vec{e}_{1y}) + v(y'\vec{e}_{2z}/d - \vec{e}_{2y}) = A_y - (1 + A_z/d)y' \end{cases}.$$

Відповідно зміниться й рішення:

$$\Delta = (x'\vec{e}_{1z}/d - \vec{e}_{1x}) \cdot (y'\vec{e}_{2z}/d - \vec{e}_{2y}) -$$
$$- (x'\vec{e}_{2z}/d - \vec{e}_{2x}) \cdot (y'\vec{e}_{1z}/d - \vec{e}_{1y});$$
$$\Delta_1 = (A_x - (1 + A_z/d)x') \cdot (y'\vec{e}_{2z}/d - \vec{e}_{2y}) -$$
$$- (x'\vec{e}_{2z}/d - \vec{e}_{2x}) \cdot (A_y - (1 + A_z/d)y')$$
$$\Delta_2 = (x'\vec{e}_{1z}/d - \vec{e}_{1x}) \cdot (A_y - (1 + A_z/d)y') -$$
$$- (A_x - (1 + A_z/d)x') \cdot (y'\vec{e}_{1z}/d - \vec{e}_{1y}).$$

У розглянутих прикладах ми риси справу з гладкими поверхностямі.Можна імітувати жорсткість шляхом вибору відповідного зразка нерегулярної текстури, але все одно зображення буде виглядати так, немов неоднорідності нанесені на **гладкій** поверхні. Для моделювання мікрорельєфу Дж.Блін запропонував метод, заснований на збуренні норрисі до поверхні.

Нехай, як і раніше, поверхня задана в параметричному виді за допомогою векторної функції $\vec{F}(u,v)$. У кожній її точці можна побудувати вектор норрисі, скориставшись частками похідними цієї функції.Відомо, що похідні $\vec{F}_u$ і $\vec{F}_v$ являють собою вектори, що лежать в дотичній площині даної поверхні. Тоді вектор норрисі може бути отриманий як векторний добуток цих двох векторів $\vec{n} = \vec{F}_u \times \vec{F}_v$. Після цього крапку поверхні можна відхилити від первісного положення в напрямку норрисі на деяку рису величину, що задається за допомогою функції збурювання $P(u,v)$:

$$\vec{F}'(u,v) = \vec{F}(u,v) + P(u,v) \cdot \vec{n}(u,v).$$

Можна показати, що норрись до нової обуреної поверхні буде визначатися виразом

$$\vec{n}' = \vec{n} + \frac{P_u \cdot (\vec{n} \times \vec{F}_u)}{|\vec{n}|} + \frac{P_v \cdot (\vec{n} \times \vec{F}_v)}{|\vec{n}|}.$$

Застосовуючи в моделі освітлення нову норрись, можна отримати ефект шорсткості поверхні. В якості функції збурювання можна використовувати довільну диференційовних по кожній із змінних функцію.

Питання та вправи

1. Які етапи виділяються в свето-тіньовому аналізі?
2. До якого типу відноситься алгоритм Аппеля: ітеративного або рекурсивному?
3. Чи можливе використання алгоритму Аппеля для сцен з неповним затіненням?
4. Що таке тіньовий буфер? Чим він відрізняється від традиційного Z-буфера?
5. У чому полягає модифікація алгоритму Вейлера-Азертон для виконання свето-тіньового аналізу?
6. У якій моделі освітленості можна використовувати метод излучательной?
7. Чим відрізняється трасування променів у глобальній моделі освітленості від методу видалення невидимих граней?
8. Які складові інтенсивності розглядаються в методі трасування?
9. Яким чином можна використовувати двійкові дерева в алгоритмі трасування?
10. Який спосіб завдання поверхні найбільш зручний для текстурирования?
11. У чому полягає ідея моделювання мікрорельєфу при нанесенні текстур?

ЛІТЕРАТУРНІ ДЖЕРЕЛА

1. Аббасов И.Б. Создаём чертежи на компьютере в AutoCAD. ДМК Пресс, 2011.-136 с.
2. Абрамов А.Е. Компьютерная графика. Учебно-методический комплекс. Ульяновская ГСХА, 2009.-50 с.

3. Гинзбург В.М. Формирование и обработка изображений в реальном времени. Радио и связь, 1986.- 232 с.
4. Голованов Н.Н. Геометрическое моделирование. Физматлит, 2002.-472 с.
5. Шмиг Р.А. та інші. Інженерна комп'ютерна графіка. Український бестселер, 2002.-600 с.
6. Shah M.B., Rana B.S. Engineering Drawing . Pearson Education, 2005.- 484 p.
7. **Компьютерная графика — Википедия**. **ru.wikipedia.org/wiki/Компьютерная_графика**
8. **Компьютерная графика** и графические редакторы. Adobe Photoshop, Adobe Illustrator, Image Ready, Image Styler . www.postroika.ru/drawing
9. http://www.intuit.ru/department/graphics/graphaly/1/print

Руслан Миколайович Літнарович, кандидат технічних наук, доцент
Іван Федорович Чернецький, магістрант інформаційних технологій
Дєдух Микола Ігорович , магістрант інформаційних технологій

Сучасні технології опрацюваня графічної інформації

Курс лекцій

Частина 1

Відповідальний редактор Й.В.Джунь, доктор фіз.-мат.наук, професор

Комп'ютерний набір, верстка, редагування і макетування та дизайн в редакторі Microsoft ® Offise ® Word 2007 Р.М.Літнарович, І.Ф.Чернецький, М.І.Дєдух

33027 Рівне , Україна
Вул.С.Дем'янчука, 4, корпус 1
Телефон : (+00380) 362 23 – 73 – 09
Факс :(+00380) 362 23 – 01 – 86
E-mail:mail@regi.rovno.ua
litnarovich@windowslive.com