1. Предмет комп’ютерної графіки. Основні поняття та визначення.

Дисципліни комп’ютерної графіки та завдання, що ними вирішуються.

Комп’ютерна графіка це наука предметом вивчення якої є створення, зберігання та обробка моделей та їх зображень за допомогою комп’ютерних засобів. Визначає спосіб введення/виведення інформації, перехід від символьного подання до графічного та навпаки. Під графічною формою розуміють креслення, ескізи, схеми, діаграми, графіки тощо.

Об’єднує декілька дисциплін:

Computer Vision(аналіз образів, дослідження абстрактних моделей, графічних об’єктів, взаємозв’язків), Машинна графіка(перетворює опис в зображення, Предметом машинної графіки є побудова моделі об’єкта генерування зображень та їх перетворення), обробка зображень(цифрове подання, підвищення якості, оцінка та розпізнавання зображень).

Основні області застосування:

Наукова графіка — дає можливість проводити обчислювальні експерименти з наочним поданням їх результатів.

Ділова графіка — область комп'ютерної графіки, призначена для наочного представлення різних показників роботи установ.

Конструкторська графіка використовується в роботі інженерів — конструкторів, архітекторів, винахідників нової техніки.

Ілюстративна графіка — це довільне малювання і креслення на екрані комп'ютера.

Художня і рекламна графіка — що стала популярною багато в чому завдяки телебаченню. За допомогою комп'ютера створюються рекламні ролики, мультфільми,комп'ютерні ігри, відео уроки, відео презентації.

Комп'ютерна анімація — це отримання рухомих зображень на екрані дисплея. Художник створює на екрані малюнки початкового і кінцевого положення рухомих об'єктів, всі проміжні стани розраховує і зображує комп'ютер.

Мультимедіа — це об'єднання високоякісного зображення на екрані комп'ютера зі звуковим супроводом. Найбільшого поширення системи мультимедіа отримали в галузі навчання, реклами, розваг.

2. Комп’ютерне представлення зображення. Подання інформації про колір у комп’ютерній графіці. Колірні моделі.

Класифікація зображень:

1. За кольоровістю
    a. монохромні
    b. відтінки сірого
    c. кольрові
2. За динамікою
    a. Статичні
    b. Динамічні
3. За внутрішнім поданням
    a. Растрові
    b. Векторні
4. За об’ємністю
    a. Двовимірні
    b. Тривимірні

Кольори в комп’ютерній графіці

Колір – суб’єктивна характеристика світла, яка відображає здатність людського зору розрізняти частоту електромагнітних коливань у області видимого світла.

Формування відчуття кольору залежить від таких складових: джерела освітлення та умов розповсюдження випромінювання, властивостей об’єктів що сприймаються, зорової системи людини.

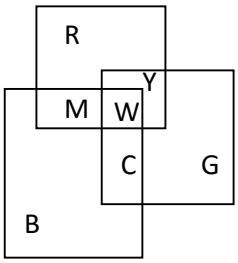
Характеристики кольору:

1. Яскравість - насичення кольору
2. Насиченість - віддалення сірого кольору
3. Світлота (ясність) – ступінь віддалення від білого кольору
4. Кольоровий тон

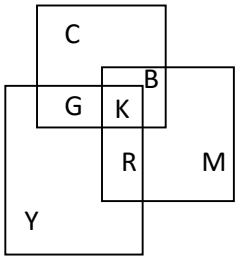
Фізичні принципи формування відтінків об’єктів

В комп’ютерній графіці розрізняють 2 види об’єктів: випромінюючі (монітори, світло-діодні матриці) та не випромінюючі (папір, світлофільтри).

Для випромінюючих об’єктів характерне адитивне формування відтінків, коли потрібний колір отримується як результат змішування трьох базових кольорів R G B.

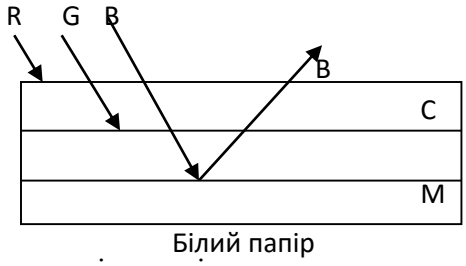


Адитивна модель



Субтрактивна модель

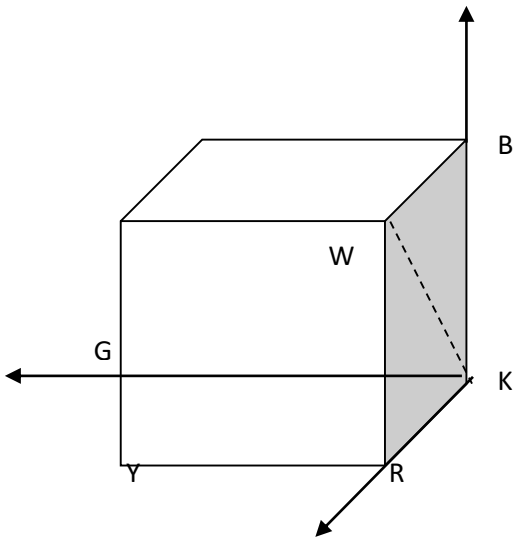
Кольори однієї моделі доповнюють кольори іншої моделі. Доповнюючий колір – це колір, який доповнює даний колір до білого.



Кольорові моделі

Призначення кольорової моделі - надати засіб опису відтінку кольору у межах деякої кольорової гамми, у тому числі і для виконання інтерполяції кольорів.

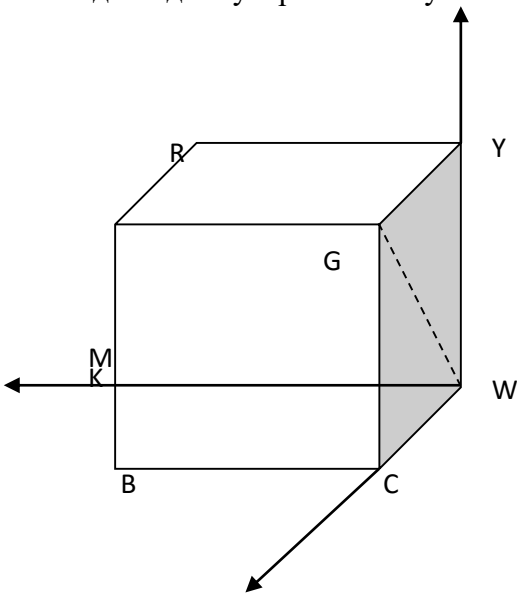
1. Модель RGB – апаратна орієнтована модель, яка використовується для адитивного формування відтінків зображень.



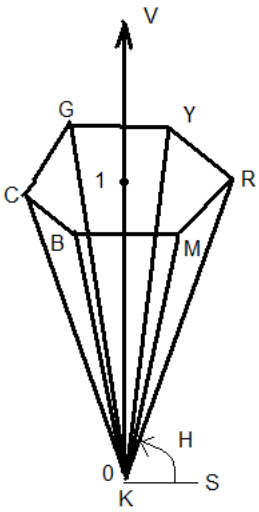
Трикутник паскаля об’єднує найбільш яскраві кольори. Він об’єднує точки R, G, B.

Кольори, що знаходяться біля штрихованої лінії – відтінки сірого.

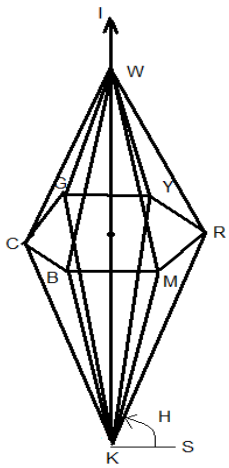
2. Модель CMY - апаратно орієнтована модель, яка використовується у поліграфії. Відповідає субтрактивному способу формування відтінків.



3. Модель HSV (Hue Saturation Value / Lightness)



4. Модель HSI (Hue Saturation Intensity)/HLS (Hue Lightness Saturation)/BHS (Brightness Hue Saturation)



Hue – кольоровий тон

Saturation - насиченість

$$I = \frac{R+G+B}{3}$$

$$S = 1 - \frac{3}{R+G+B} * \min(R,G,B)$$

$$H = \begin{cases} \theta \ \text{ при } B \le G \\ 360^{\circ} - \theta \ \text{ при } B > G \end{cases}$$

$$\theta = \arccos\left(\frac{(R-B)+(R-G)}{2\sqrt{(R-G)^2+(R-B)(G-B)}}\right)$$

- 0 <= H < 120

(RG сектор)

R=I*(1+(S*cos H)/cos(60-H))

G=3*I-(R+B)

B=I(1-S)

- 120 <= H < 270

(GB sector)

H’=H-120

R=I(1-S)

G=I*(1+(S*cos H)/cos(60-H’))

- 240 <= H < 360

(BR sector)

H’ = H – 240

R = 3*I – (G + B)

G = I (1 - S)

B = I(1 + (S*cos H’)/cos(60 - H))

$$I = \frac{1}{3}$$

$$S = 1$$

$$H = \theta = 0^{\circ}$$

5. Модель YIQ

Y – яскравість

I, Q – кольоро-різнісні сигнали

$$Y = Rr + Gg + Bb$$

Маленькі літери – еталони сприйняття людиною різних базових кольорів.

Аппаратно-оріентована модель яка використовується в телебаченні та слугує для скорочення смуги частот за рахунок врахування психофізіологічних особливостей зору людини.

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0,299 & 0,587 & 0,114 \\ 0,596 & -0,274 & -0,322 \\ 0,211 & -0,522 & 0,311 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1,0 & 0,956 & 0,623 \\ 1,0 & -0,272 & -0,648 \\ 1,0 & -1,105 & 0,705 \end{bmatrix} * \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

3. Растрова графіка. Векторна графіка. Особливості, переваги та недоліки цих видів комп'ютерного представлення зображення.

Ра́строва гра́фіка є частиною комп'ютерної графіки, яка має справу зі створенням, обробкою та зберіганням растрових зображень.
Растрове зображення є масивом кольорових точок (пікселів). Обробка растрової графіки здійснюється растровими графічними редакторами. Растрові зображення зберігаються у різних графічних форматах.

Інколи говорять про растрову графіку, маючи на увазі зображення представлене у растровому форматі.

Переваги:

- Растрова графіка дозволяє створити практично будь-яке зображення, незалежно від складності, на відміну від векторної, де неможливо точно передати ефект переходу від одного кольору до іншого без втрат у розмірі файлу.
- Поширеність — растрова графіка використовується зараз практично скрізь: від маленьких значків до плакатів.
- Висока швидкість обробки складних зображень, якщо не потрібно масштабування.
- Растрове представлення зображення природне для більшості пристроїв введення-виведення графічної інформації (за винятком векторних пристроїв виводу), таких як монітори, матричні та струменеві принтери, цифрові фотоапарати, сканери, а також стільникові телефони.
- Простота автоматизованого вводу (оцифрування) зображень, фотографій, слайдів, рисунків за допомогою сканерів, відеокамер, цифрових фотоапаратів;

Фотореалістичність. Можна отримувати різні ефекти, такі як туман, розмитість, тонко регулювати кольори, створювати глибину предметів.

Недоліки:

- Великий розмір файлів у простих зображень. Тому, що розмір файлу є пропорційним до площі зображення, роздільності і типу зображення, і, переважно, при хорошій якості є великим.
- Неможливість ідеального масштабування. Растрове зображення має визначену роздільність і глибину представлення кольорів. Ці параметри можна змінювати лише у визначених межах і, як правило, із втратою якості.
- Неможливість виведення на друк на векторний графічний пристрій.
- Складність управління окремими фрагментами зображення.

Через ці недоліки для зберігання простих малюнків рекомендують замість, навіть стиснутої, растрової графіки використовувати векторну графіку.

Ве́кторна гра́фіка — створення зображення в комп'ютерній графіці з сукупності геометричних примітивів — (точок, ліній, кривих, полігонів), тобто об'єктів, які можна описати математичними виразами.[1]

Векторна графіка для опису зображення використовує вектори, на відміну від растрової графіки, яка описує зображення як масив пікселів (точок).

Переваги:

- Розмір файла, який займає описова частина, не залежить від реальної величини об'єкта, що дозволяє, використовуючи мінімальну кількість інформації, описати достатньо великий об'єкт файлом мінімального розміру.
- У зв'язку з тим, що інформація про об'єкт зберігається в описовій формі, можна нескінченно збільшити графічний примітив, наприклад, дугу кола, і вона залишиться гладкою. З іншого боку, якщо крива представлена у вигляді ламаної лінії, збільшення покаже, що крива не є гладкою.
- Параметри об'єктів зберігаються і можуть бути легко змінені. Також це означає, що переміщення, масштабування, обертання та інше, не погіршує якості малюнка. Більш того, зазвичай вказують розміри в апаратно-незалежних одиницях (англ. device-independent unit), які ведуть до найкращої растеризації на растрових приладах.
- При збільшенні або зменшенні об'єктів товщина ліній може бути задана постійною величиною, незалежно від реального контуру.

Недоліки:

- Не кожен об'єкт може бути легко зображений у векторному вигляді — для того, щоб зображення було подібним до оригіналу може знадобитися дуже велика кількість об'єктів з високою складністю, що негативно впливає на кількість пам'яті, яку займатиме зображення та час для його відтворення.

4. Параметри растрових зображень.

Якість растрової графіки залежить від оптичної роздільності, яка вимірюється кількістю точок на одиницю довжини. Розрізняють:

- роздільність оригінала [dpi – dots per inch]
- роздільність екранного зображення [pixel]. Елементарну точку растра для екранного зображення називають пікселом. Розмір пікселу залежить від екранної роздільної здатності, роздільності оригіналу та масштабу відображення.
- Роздільність друкованого зображення [lpi – lines per inch, лініатура]
- Кількість використовуваних кольорів або глибина кольору (обсяг пам'яті в бітах, що використовуються для одного пікселя);
- 3.Колірний простір — RGB, CMYK, XYZ, YCbCr та ін;

В поліграфії при формуванні зображення можуть бути застосовані: амплітудна модуляція та частотна модуляція. Вважається що більш якісним виглядає зображення з частотною модуляцією.

Растр :

1. У поліграфії – подання графічної інформації за допомогою точок різної величини або таких що знаходяться на різній відстані.
2. У комп’ютерній графіці – порядок розташування точок, які називають растровими елементами. Розрізняють прямокутний (квадратний), трикутний та гексагональний (шестикутний) растр.

5. Класифікація алгоритмів комп’ютерної графіки.

5.1. Класифікація:

1.Алгоритми побудови відрізка - алгоритми для апроксимації відрізка на дискретній графічній поверхні: алгоритм Брезенхема, ЦДА, Ву

2.Алгоритм зафарбування з запалом – заповнює з’єднану область багатовимірного масиву заданим значенням

3.Алгоритм художника – визначає видимі частини тривимірної сцени

4.Трасування променів – рендерінг реалістичних зображень

5.Модель освітлення: затемнення по Фонгу, затемнення по Гуро, модель освітлення Блінна-Фонга, модель освітлення Кука-Торренса

6.Алгоритм скануючого рядка

7.Глобальне освітлення

8.Алгоритми інтерполяції: інтерполяція сплайнами

5.2.Алгоритми машинної графіки можна розділити на два рівні: нижній і верхній. Група алгоритмів нижнього рівня призначена для реалізації графічних примітивів (ліній, кіл, заповнень тощо).

Серед алгоритмів нижнього рівня можна виділити наступні групи:

Найпростіші в сенсі використовуваних математичних методів і ті, що відрізняються простотою реалізації. Як правило, такі алгоритми не є найкращими за обсягом виконуваних обчислень або необхідних ресурсів пам'яті.

Тому можна виділити другу групу алгоритмів, що використовують більш складні математичні передумови і відрізняються більшою ефективністю.

До третьої групи слід віднести алгоритми, які можуть бути реалізовані апаратно без великих труднощів.

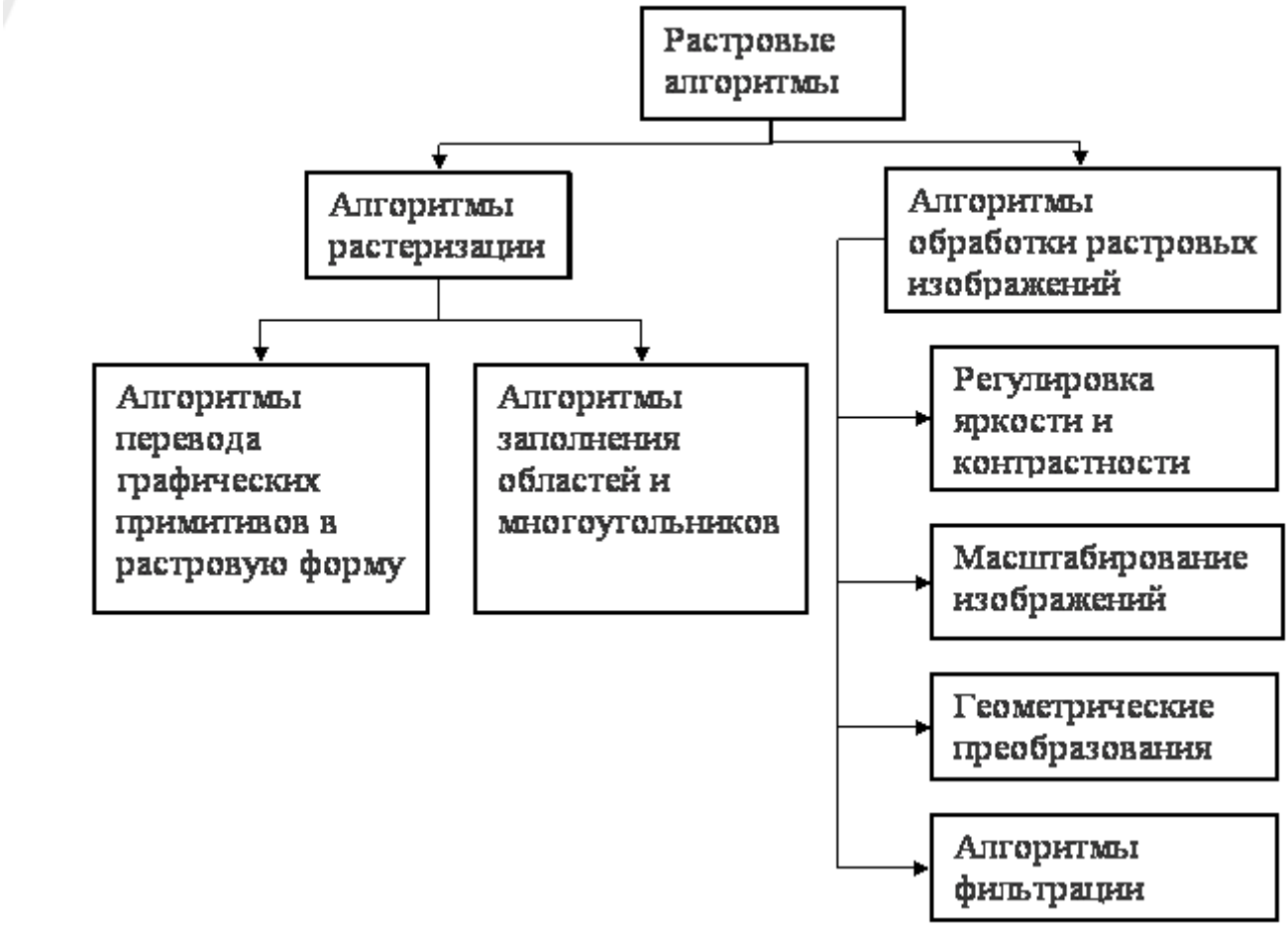
До четвертої групи можна віднести алгоритми зі спеціальним призначенням.

До алгоритмів верхнього рівня відносяться в першу чергу алгоритми видалення невидимих ліній і поверхонь. Від ефективності цих алгоритмів залежать якість і швидкість побудови тривимірного зображення.

До задачі видалення невидимих ліній і поверхонь додається завдання побудови (зафарбовування) напівтонових (реалістичних) зображень.

Однак при цьому не слід забувати, що виведення об'єктів в алгоритмах верхнього рівня забезпечується примітивами, що реалізують алгоритми нижнього рівня, тому не можна ігнорувати проблему вибору і розробки ефективних алгоритмів нижнього рівня.

6. Класифікація растрових алгоритмів. Поняття зв’язності.



Однією з переваг растрової графіки є можливість зображувати замальовані області. Область, що буде заповнюватися, може бути описана двома засобами: 1) у вигляді значень пікселів у відеобуфері; 2) у вигляді многокутника, що заданий координатами своїх вершина або ребер. Заповнення області, що задана значеннями пікселів у відео буфері. Область – це група прилеглих один до одного, ЗВ’ЯЗАНИХ пікселів. Область можна створювати та визначати: - присвоювачи деяке значення усім пікселям, що належать області; - визначаючи однаковим засобом пікселі, що оточують область.

Визначимо поняття зв’язної області, що створена прилеглими один до одного пікселями. Нас цікавлять 4-х та 8-ми зв’язні області. Алгоритми для 8-зв’язних областей працюють для 4- чотирьохзв’язаних областей, але не навпаки. Кожен піксель 4-чотирьохзвв’язаної області може бути досягнений з будь якого іншого пікселя цієї області за допомогою довільної послідовності переміщувань на один піксель з чотирьохелементного набору кроків: вгору, униз, вліво, вправо. Кожен піксель 8 - чотирьохзв’язаної області може бути досягнений з будь якого іншого пікселя цієї області за допомогою довільної послідовності переміщувань на один піксель з восьмиелементного набору кроків по вертикалі, горизонталі та діагоналям.

7.Растрові алгоритми генерування відрізків. Задачі, що вирішують за допомогою цих алгоритмів. Їхні переваги та недоліки.

Алгоритми побудови відрізка — графічні алгоритми апроксимації відрізка на дискретному графічному пристрої (растеризація).

Стандартними вимогами до алгоритмів є швидкість роботи, рівномірна яскравість та прямий вид отриманих відрізків, збіг початкових та кінцевих координат отриманої та ідеальної лінії.

Генерація відрізків

Загальні вимоги до зображення відрізків:

1. Кінці відрізка мають знаходитись в заданих точках
2. Відрізки повинні мати вигляд прямих
3. Яскравість уздовж відрізка повинна бути постійною та не залежати від довжини та нахилу

Жодна з цих умов не може бути повністю виконана на растровому моніторі:

1. Кінці відрізка у загальному випадку розташовуються на пік селах, які є лише наближенням до математичних координат відрізка
2. Відрізок апроксимується набором пік селів і лише в окремих випадках буде виглядати прямим
3. Яскравість для різних відрізків та уздовж того самого відрізка у загальному випадку різна тому що відстань між центрами пік селів буде відрізнятись

Растрові алгоритми генерування відрізків:

- ЦДА
- Алгоритм Брезенхема
- Алгоритм Ву

Цифровой дифференциальный анализатор

Отрезок описывается уравнением:

$$\frac{dY}{dX} = \frac{Py}{Px}$$

ЦДА формирует дискретную аппроксимацию решения этого дифференциального уравнения.

Задается N - количество узлов аппроксимации отрезка и за N циклов вычисляются очередные координаты:

$$X_0 = Xn; \quad X_{i+1} = X_i + Px/N.$$

$$Y_0 = Yn; \quad Y_{i+1} = Y_i + Py/N.$$

X_i, Y_i преобразуются в целочисленные значения.

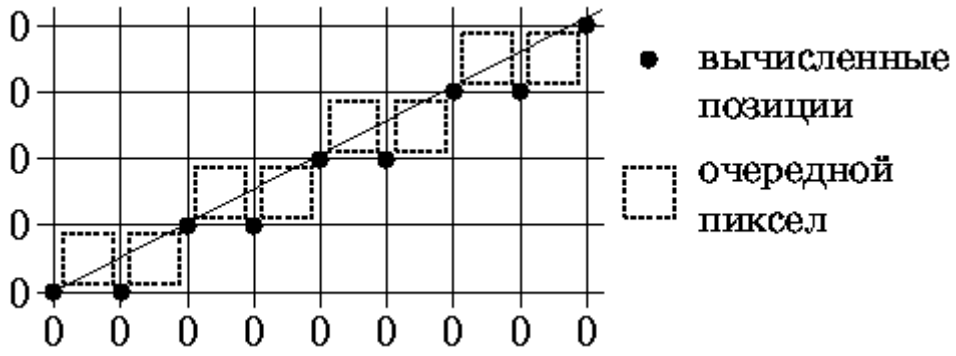
Недостатки:
· точки могут прописываться дважды,

· нет предпочтительных направлений и отрезки кажутся некрасивыми.

Несимметричный ЦДА

По большей координате делается единичный шаг.

Для Px > Py (Px, Py > 0) X-координата увеличиается на 1 Px раз, при этом Y-координата Px раз увеличивается на Py/Px.

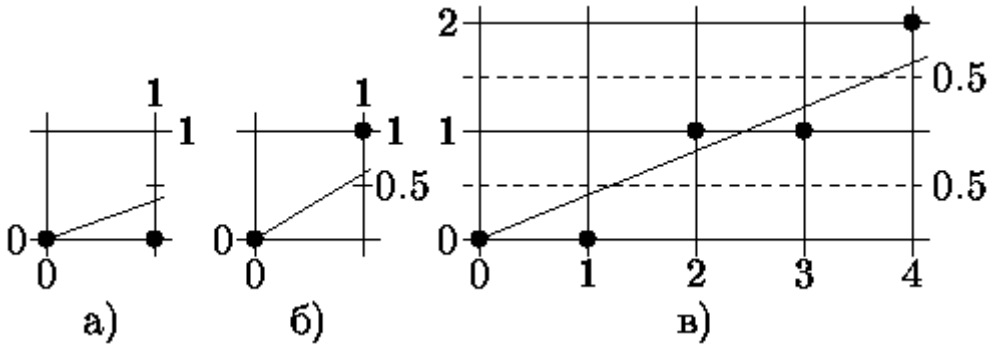


Проблема заключается в необходимости деления
(Py/Px) и сложения вещественных чисел.

Алгоритм Брезенхема построения векторов

Алгоритм средней точки. Вещественных операций нет.
Рис. а) и б). Пусть начальная точка (0,0).

Если P_Y / P_X J 0.5, то следующая точка (1,0).
Если P_Y / P_X > 0.5, то следующая точка (1,1).



$$X_0 = X_n; \quad Y_0 = Y_n; \quad X_1 = X_0 + 1; \quad E_1 = \frac{DY}{DX} - \frac{1}{2}.$$

Возможны случаи:

E_1 J 0 $E_1 > 0$

ближайшая точка есть:

$X_1 = X_0 + 1$; $X_1 = X_0 + 1$;

$Y_1 = Y_0$; $Y_1 = Y_0 + 1$;

$E_2 = E_1 + P_Y/P_X$; $E_2 = E_1 + P_Y/P_X - 1$.

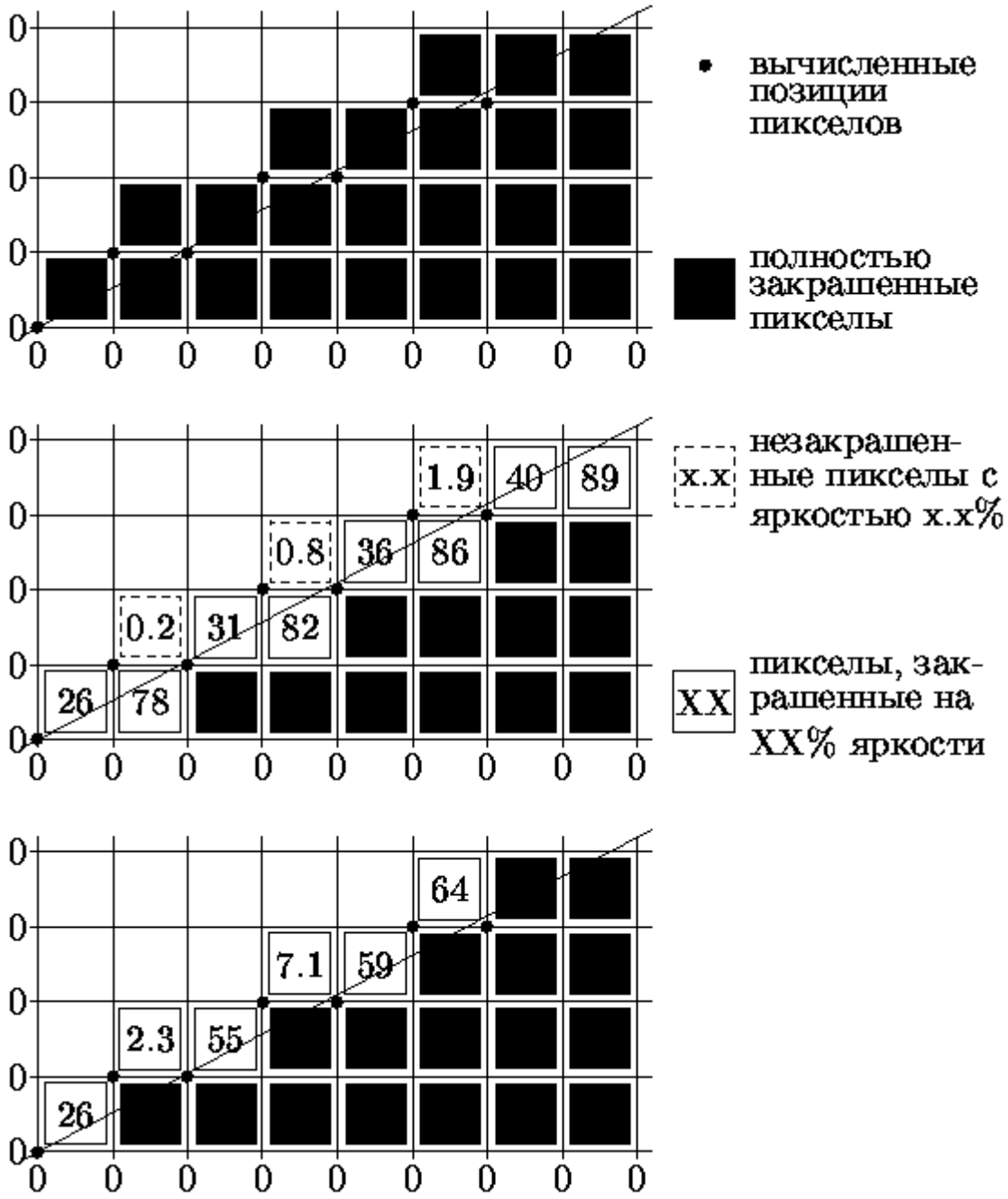
Важен только знак E, поэтому домножим E на $2 \times P_X$:

$E_1 = 2 \times P_Y - P_X$

E_1 J 0: $E_2 = E_1 + 2 \times P_Y$

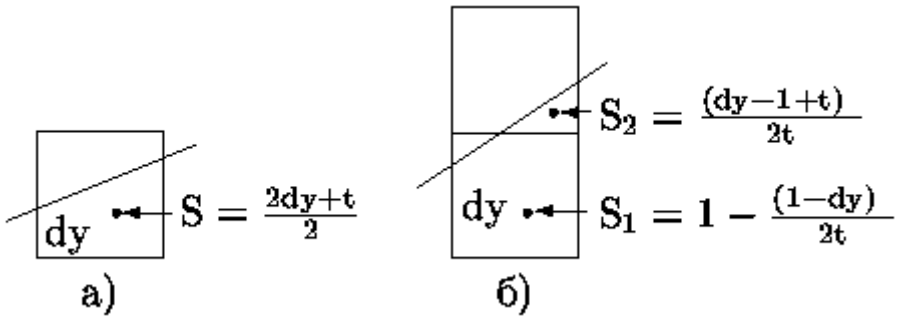
$E_1 > 0$: $E_2 = E_1 + 2 \times (P_Y - P_X)$

Модифицированный алгоритм Брезенхема



а) генерация ребер без устранения ступенчатости;
б) точное вычисление интенсивности пикселов;
в) пикселы границы по модифицированному методу Брезенхема.

Построение ребра заполненного моногугольника с устранением ступенчатости. Яркость пиксела ~ площади пиксела, попавшей внутрь многоугольника.



Алгоритм By — это алгоритм разложения отрезка в растр со сглаживанием. Алгоритм сочетает высококачественное устранение ступенчатости и скорость, близкую к скорости алгоритма Брезенхема без сглаживания.

Горизонтальные и вертикальные линии не требуют никакого сглаживания, поэтому их рисование выполняется отдельно. Для остальных линий алгоритм By проходит их вдоль основной оси, подбирая координаты по неосновной оси аналогично алгоритму Брезенхема. Отличие состоит в том, что в алгоритме By на каждом шаге устанавливается не одна, а две точки. Например, если основной осью является X, то рассматриваются точки с координатами (x, y) и (x, y+1). В зависимости от величины ошибки, которая показывает, как далеко ушли пиксели от идеальной линии по неосновной оси, распределяется интенсивность между этими двумя точками. Чем больше удалена точка от идеальной линии, тем меньше её интенсивность. Значения интенсивности двух пикселей всегда дают в сумме единицу, то есть это интенсивность одного пикселя, в точности попавшего на идеальную линию. Такое распределение придаст линии одинаковую интенсивность на всём её протяжении, создавая при этом иллюзию, что точки расположены вдоль линии не по две, а по одной.

8.Порівняльні характеристики алгоритмів растрування відрізків.

Алгоритм DDA-лінії : Застосування обчислень з дійсними числами і лише одноразове використання округлення для остаточного отримання значення растрової координати зумовлюють високу точність і низьку швидкодію алгоритму.

Алгоритм Брезенхейма малює відрізок дуже швидко, але він не виконує згладжування. Також він не може обробити ситуацію, коли кінцеві точки мають не цілочисельні координати. Алгоритм Брезенхейма використовується в графобудівниках і графічних процесорах сучасних відеокарт. Також його можна знайти в багатьох програмних графічних бібліотеках. Через свою простоту алгоритм часто реалізується або як вбудована програма або як апаратне забезпечення сучасних відеокарт.

Хоча алгоритм Ву також часто використовується в сучасній комп'ютерній графіці через підтримку згладжування, алгоритм Брезенхейма залишається вживаним завдяки його швидкісті і простоті. Наївна спроба малювання зі згладжуванням за допомогою алгоритму Брезенхейма вимагає багато часу, в той час як алгоритм Ву дуже швидкий (хоча й повільніший за алгоритм Брезенхейма).

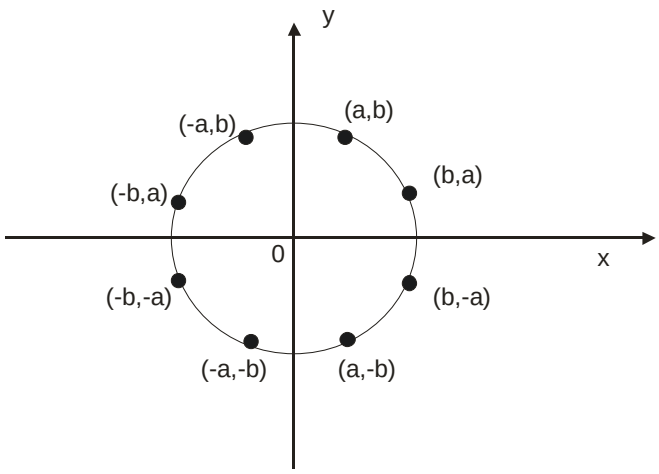
9.Задача растрування окружності та еліпсу. Алгоритми, що використовуються для вирішення цієї задачі.

Растрова розгортка кіл

1) $\begin{cases} x = R * \cos\alpha \\ y = R * \sin\alpha \end{cases}$

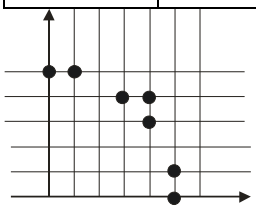
2) $x^2 + y^2 = R^2$

1:



$R = 5, \quad 0^\circ \le \alpha \le 90^\circ, \quad \Delta\alpha = 15^\circ$

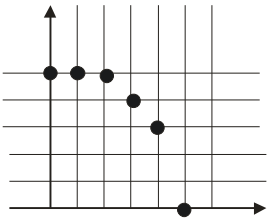
α	90°	75°	60°	45°	30°	15°	0°
X	0	]1.3[= 1	]2.5[= 3	]3.5[=4	4	5	5
y	5	]4.8[= 5	]4.3[= 4	]3.5[=4	3	1	0



2:

$y = \pm\sqrt{R^2 - x^2}$

X	0	1	2	3	4	5
Y	5	]4.8[=5	]4.6[=5	4	3	0



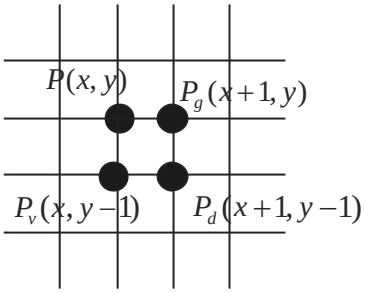
Алгоритм Брезенгхема

Коло не буде абсолютно ідеальним.

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

$$\varepsilon_i = x^2{}_i + y^2{}_i - R^2 = \min$$

$$P_i(x_i, y_i)$$



$$\varepsilon_g = (x+1)^2 + y^2 - R^2$$
$$\varepsilon_d = (x+1)^2 + (y+1)^2 - R^2$$
$$\varepsilon_v = x^2 + (y-1)^2 - R^2$$

Аналіз значень похибки починають зі значення ε_d

Якщо ε_d =0 то вибирають діагональну точку. Якщо ε_d <0 то точка знаходиться в середині кола. В цьому випадку вибирають з горизонтальної або діагональної точки. Для цього обчислюють d_i. Якщо d_i <=0 то вибирають точку P_g, інакше P_d.

$$d_i = \left|\varepsilon_g\right| - \left|\varepsilon_d\right| = \left|(x+1)^2 + y^2 - R^2\right| - \left|(x+1)^2 + (y-1)^2 - R^2\right|$$

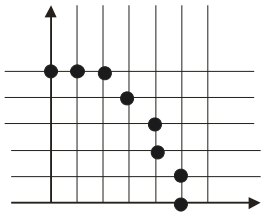
Якщо ε_d >0, тоді вибираємо з точок P_d або P_v.

$$d_i = \left|\varepsilon_d\right| - \left|\varepsilon_v\right| = \left|(x+1)^2 + (y-1)^2 - R^2\right| - \left|x^2 + (y-1)^2 - R^2\right|$$

Якщо d_i <=0 то вибираємо діагональну точку, інакше P_v.

$$(0,0), \quad k = 5, \quad P_i(0,k)$$

X	0	1	2	3	4	4	5	5
Y	5	5	5	4	3	2	1	0
ε_g	1+25-25=1	4	-------	-------	-------	-------	-------	-------
ε_d	1+16-25=-8 <0	-5 < 0	0	0	4 > 0	1 > 0	9 > 0	-------
ε_v	---------	-------	-------	-------	0	-8	0	-------
d_i	1-|-8|=-7 <0	-4 <0	-------	-------	4 > 0	-7 < 0	9 > 0	-------
		P_g	P_g	P_d	P_d	P_v	P_d	P_v



Алгоритм Ву растеризації кола з антиалісінгом:

Представляє собою змінений алгоритм растеризации лінії. Для точки кола вибираються два найближчих пікселя. Між ними пропорційно розподіляється прозорість кольору. При оптимізації даного алгоритму можна враховувати, що більш яскраві пікселі, що лежать на внутрішній частині лінії, надають угнутість лінії, а лежать на зовнішній стороні - опуклість.

Алгоритм Ву пастеризації еліпса з антиалісінгом:

Цей алгоритм дозволяє намалювати еліпс з використанням антиаліасінга, так що фігура не буде мати різких країв. Звичайні алгоритми малювання еліпса малюють всі крапки одним кольором, цей же алгоритм зафарбовує різні ділянки в різні кольори, і за рахунок цього "згладжує" нерівності.

Також слід зазначити, що цей алгоритм приймає не цілі розміри еліпса, а дійсні. При зміні координат еліпс, намальований алгоритмом Брезенхема, переміщується різко, стрибками. Еліпс по алгоритму Ву може переміщатися безперервно. За рахунок цього можна забезпечувати плавну анімацію при малюванні рухомих зображень.

10.Методи покращення якості зображення. Усунення ступінчатості.

Засади засобів усунення драбинного ефекту Основна причина виникнення драбинного ефекту полягає у тому, що відрізки, ребра многокутників, кольорові межі та інше мають безперервну природу, тоді як растровий пристрій дискретний. Існують три основні види спотворень, пов’язаних з дискретністю растрів екранів: 1) драбинність ребер; 2) некоректна візуалізація тонких деталей; 3) пов’язане з дуже дрібними об’єктами. Якщо об’єкт менший розміру пікселя або не перекриває внутрішню крапку, що служить для оцінки атрибутів пікселю, то він не буде враховуватись у результуючому зображенні. З іншого боку, якщо малий об’єм перекриває цю крапку, то він може дуже сильно впливати на атрибути пікселя.

Існуючі підходи щодо розв’язання завдань поліпшення цифрового зображення та

відновлення його структури поділяють на дві категорії:

1) методи обробки в просторовій області (просторові методи), які ґрунтуються на прямому

маніпулюванні пікселями зображення;

2) методи обробки в частотній області (частотні методи), які ґрунтуються на модифікації

(фільтрації) сигналу, що формується шляхом застосування до зображення перетворення

Фур'є.

Просторова обробка застосовується, коли єдиним джерелом викривлень є адитивний шум.

Частотна фільтрація може використовуватися для нечітких зображень з дефектами

освітлення, також вона враховує й шум. Тому частотна обробка є найбільш

універсальним і поширеним методом поліпшення якості цифрового зображення.

11. Цифрова фільтрація зображень.

Зазвичай зображення, сформовані різними інформаційними системами, спотворюються дією завад. Це ускладнює як їхній візуальний аналіз, так і автоматичну обробку. При вирішенні деяких завдань обробки зображень у ролі завад можуть виступати ті або інші компоненти самого зображення. Наприклад, при аналізі космічного знімка земної поверхні може стояти завдання визначення границь між її окремими ділянками - лісом і полем, водою й сушею тощо. З погляду цього завдання окремі деталі зображення всередині розділених областей є завадою.

Ослаблення дії завад досягається фільтрацією. При фільтрації яскравість (сигнал) кожної точки вихідного зображення, спотвореного завадою, замінюється деяким іншим значенням яскравості, яке в меншій мірі було спотворене завадою. Фільтрація зображень здійснюється в просторовій і частотній областях. При просторовій фільтрації зображень перетворення виконується безпосередньо над значеннями відліків зображення. Результатом фільтрації є оцінка корисного сигналу зображення. Це досягається завдяки тому, зображення часто являє собою двовимірну функцію просторових координат, що змінюється по цих координатах повільніше, ніж завада, що також є двовимірною функцією. Це дозволяє при оцінці корисного сигналу в кожній точці зображення взяти до уваги сусідні точки, скориставшись певною подібністю сигналу. В інших випадках, навпаки, ознакою корисного сигналу є різкі перепади яскравості. Однак, як правило, частота цих перепадів відносно невелика, так що на значних проміжках сигнал або постійний, або змінюється повільно. І в цьому випадку властивості сигналу проявляються при спостереженні не тільки його окремої точки, але й при аналізі її околиці.

12.Задача генерування кривих. Способи задання кривих.

Графічні пристрої, що використовуються в КГ, є в основному растровими, тому при синтезі графічних зображень, що складаються з окремих простих елементів, виникає необхідність у растрових алгоритмах.

Растровий алгоритм – це алгоритм, який для роботи програми, що реалізує графічні примітиви, враховує властивість растра.

Зобразити криву на дискретній поверхні (екрані) означає знайти таку її цілочислову апроксимацію, яка дасть можливість відтворити неперервну криву.

Методи побудови ліній на дискретній поверхні поділяються на 2 класи:

• числові методи (методи прямих обчислень координат)-базуються на числовому аналізі рівнянь кривих, тобто на обчисленні значень функцій.

• інкрементні методи. - виконуються як послідовність обчислень координат сусідніх точок шляхом додавання відповідних приростів так, щоб сусідня цілочислова точка була найближчою до неперервної кривої.

Прирости розраховуються на основі аналізу функції похибок. У циклі виконуються лише цілочислові операції порівняння, додавання та віднімання. У такий спосіб досягається більш висока швидкодія обчислень координат кожного пікселя порівняно з числовими методами

13.Форма Без'є параметричного подання кривих.

Це поліноміальні криві, форму яких визначають контрольні точки. Чим більша кількість контрольних точок тим більший ступінь поліноміального опису кривої і тим точніша його побудова. На практиці застосовують криві третього ступеня, які визначаються 4-ма контрольними точками. Рівняння кривої Без'є в параметричному вигляді :

$$B(t) = t^3 * P_0 + 3t(t-1)^2 * P_1 + 3t^2(t-1)P_2 + (t-1)^3 P_3$$

$$0 \le t \le 1$$

Для отримання кривих більш складного вигляду необхідно об'єднувати криві третього порядку. Необхідна умова для отримання безперервності – це розміщення двох останніх точок першої частини та двох перших точок другої частини уздовж тієї ж лінії, крім цього остання точка першої частини має бути першою точкою другої частини.

Властивості кривих:

- безперервність заповнення сегмента між початковою та кінцевою точками;
- крива завжди розташовується всередині фігури, утвореної лініями, що з'єднують контрольні точки;
- при наявності лише двох контрольних точок сегмент являє собою пряму лінію;
- пряма лінія утворюється лише тоді, коли контрольні точки розташовані на одній прямій;
- крива Безьє симетрична, тобто обмін місцями між початковою та кінцевою точками (зміна напрямку траєкторії) не впливає на форму кривої;
- масштабування та зміна пропорцій кривої Безьє не порушує її стабільності, оскільки вона з математичної точки зору «аффінно інваріантна»;
- зміна координат хоча б однієї з точок веде до зміни форми всієї кривої Безьє;
- будь який частковий відрізок кривої Безьє також є кривою Безьє;
- ступінь кривої завжди на один нижче кількості контрольних точок. Наприклад, при трьох контрольних точках форма кривої — парабола;
- коло не може бути описане параметричним рівнянням кривої Безьє;
- неможливо створити паралельні криві Безьє, за винятком тривіальних випадків (прямі лінії та співпадаючі криві), хоча існують алгоритми, що будують наближену паралельну криву Безьє з прийнятною для практики точністю.

14.Системи координат в комп'ютерній графіці.

У комп'ютерній графіці використовуються декілька різних систем

координат.

• Система координат об'єкта (СКО). Ця система координат жорстко зв'язана власне з об'єктом і в ній задаються координати точок об'єкта, наприклад координати вершин полігональної моделі. Це локальна система координат і координати в ній зберігаються в незмінній формі.

• Світова система координат (ССК). У цій системі координат можна представити координати всіх об'єктів сцени одночасно. Вона теж є правосторонньою системою координат.

• Видова система координат (ВСК). Це система координат спостерігача. Вважається, що спостерігач знаходиться в початку координат, а напрям його погляду на об'єкт збігається з віссю z, яка направлена на "центр" об'єкта. Вісь x направлена вправо від спостерігача, а вісь y – вверх, тобто видова система координат є лівосторонньою, а це відповідає вибору напряму осей екранної системи координат (вісь z направлена вглиб екрану).

• Система координат виведення зображення (СКВЗ). Ця система координат чотиривимірного простору, точки якого являють собою однорідні координати тривимірного простору в лівосторонній системі координат.

• Система координат області виведення Viewport (СКОВ). Область виведення – це прямокутна область вікна (екрану), в яку виводиться зображення, тобто вміст нормалізованого об'єму видимості (точки Pd).

• Віконна система координат (ВСК). Початок цієї системи координат розміщується в лівому нижньому куті вікна, вісь xw направлена горизонтально вправо, а вісь yw – вверх.

15.Геометричні перетворення на площині. Матрична форма запису основних двовимірних геометричних перетворень на площині в однорідних координатах. Композиція перетворень.

У сучасній комп'ютерній графіці досить широко використовується метод координат, оскільки графічне зображення складається з пікселів, які задаються координатами. Крім цього, координати використовуються для опису розміщення об'єктів та для створення зображень шляхом перетворень з однієї системи кординат в іншу. У прямокутній системі координат на площині точка M визначається своїми координатами – впорядкованою парою чисел (x, y) або матрицею розміром 1×2.

Якщо на площині ввести ще одну систему координат, то точці

M(x, y) ставиться у відповідність нова пара чисел (x′, y′). Перехід від

однієї системи координат на площині до іншої задається формулами

$x' = ax + by + m$, $y' = cx + dy + n$,

де a, b, c, d, m, n – довільні числа, $\Delta = ad - bc \neq 0$.

Перетворення, що задаються цими формулами , називаються афінними. Матриці основних елементарних перетворень:

• матриця повороту (rotation) точки відносно початку координат у додатному напрямку:

R =R(fi)= (cos fi sin fi 0
-sin fi cos fi 0
0 0 1)

• матриця розтягу (стиску) (dilation) відносно початку координат:

D =D(a,d)= (a 0 0
0 d 0
0 0 1)

• матриця перенесення (translation) точки на вектор (m, n):

T=T(m,n) = (1 0 0
0 1 0
m n 1)

• матриця дзеркального відображення (reflection) відносно осі x:

Ref = (1 0 0
0 -1 0
0 0 1)

Перетворення системи координат задається матрицями:

• R(–fi) – поворот системи координат на кут fi у додатному напрямі;

• T(–m, –n) – зсув системи координат на вектор (m, n).

16. Геометричні перетворення у просторі. Матрична форма запису основних тривимірних геометричних перетворень у просторі в однорідних координатах. Композиція перетворень.

У тривимірному випадку можна ввести однорідні координати так, що декартовим координатам (x, y, z) відповідатиме вектор (hx, hy, hz, h), $h \neq 0$, координати якого визначаються однозначно з точністю до множника h. Запропонований підхід (введення однорідних координат) дає можливість використовувати матричний запис всіх афінних перетворень у просторі, тобто у вигляді

(x′ y′ z′ 1) = (x y z 1) A,де A – матриця афінного перетворення, для якої $\det A \neq 0$.

У випадку $\det A = 0$ перетворення A називається проективним. матриці перетворень в тривимірному просторі

а) На відміну від 2D-випадку, в 3D-просторі визначаються три основні повороти: відносно осі x, відносно осі y, відносно осі z.матриця такого повороту одержується з матриці двовимірного повороту і має вигляд

Rz(χ) = (cosχ sinχ 0 0

-sinχ cosχ 0 0

0 0 1 0

0 0 0 1)

матрицю повороту відносно осі x

Rx(fi) = (1 0 0 0

0 cosfi sinfi 0

0 -sin fi cos fi 0

0 0 0 1)

Для побудови матриці повороту системи координат у додатному напрямку потрібно у формулах значення кутів змінити на їх протилежні значення, тобто поворот системи координат здійснюється інвертованими матрицями, а інверсія матриць повороту виконується шляхом транспонування.

б) Розтяг (стиск) вздовж осей x, y, z із коефіцієнтами α, β, γвідповідно задається матрицею

D = (a 0 0 0

0 b 0 0

0 0 y 0

0 0 0 1)

Якщо $\alpha = \beta = \gamma = s$, то маємо однорідне масштабування: при $s > 1$ –однорідне розтягування, при $s < 1$ – однорідний стиск.

17.Геометричні проекції в комп'ютерній графіці.
Для побудови графічних зображень тривимірних об'єктів на площині, екрані дисплея чи папері використовують проекції (від лат.projection – викидати вперед). Проекції відображають тривимірні об'єкти на двовимірну картинну площину , тобто перетворюють точки 3D-простору у 2D-простір. Проекції будуються за допомогою прямих (проекторів), що виходять з центра проектування.

Два класи проекцій: паралельна та центральна (перспективна) проекції.

У випадку центрального проектування всі прямі виходять з однієї точки – центру проектування, що знаходиться в скінченній точці тривимірного простору. Якщо центр проектування знаходиться на нескінченності, то всі проектори паралельні між собою і в результаті проектування одержується паралельна проекція .До важливих властивостей методу проектування належить ступінь достовірності сприйняття об'єкта спостерігачем за його проекціями, тобто розпізнавання об'єкта за його плоским зображенням. Центральні проекції умовно можна класифікувати за кількістю точок збігу зображення куба чи паралелепіпеда. Наприклад, проекції каркасних кубів мають одну, дві або три точки збігу і, відповідно, називаються одноточковими, двоточковими або триточковими перспективами.-

18.Задача заливки багатокутників та довільних областей зображення. Алгоритми заливки.

Область можна задавати двома способами: шляхом виведення ліній контуру (границі області) або зафарбувавши всі пікселі області Якщо область задається пікселями, що лежать усередині області, то всі пікселі з області мають одне й теж значення old і жоден піксель із границі такої області не має значення old. Такі області називаються внутрішньо заданими. Алгоритми, які працюють з такими областями так,щоб пікселі з атрибутом old перевести в пікселі зі значенням new, називаються внутрішньо заповнюючими алгоритмами.

Області, що задаються своїми замкнутими контурами (границями), називаються гранично визначеними. Пікселі, що належать границі, мають значення bound, а коди пікселів внутрішньої частини області відмінні від bound. Алгоритми заповнення таких областей називаються гранично заповнюючими алгоритмами. При цьому пікселі області потрібно зафарбувати в колір new, а колір контуру не повинен змінитися.

За способом доступу до сусідніх пікселів області поділяються на 4-зв'язні та 8-зв'язні.

4-зв'язними називають ті області, в яких кожен піксель області може бути досягнутий з іншого пікселя області за допомогою комбінацій переміщень на один піксель у чотирьох напрямках: горизонтальних (вліво,вправо), або вертикальних (вверх, вниз), тобто якщо координати сусідніх пікселів задовольняють умову

$|x1 - x2| + |y1 - y2| \le 1.$

Такі пікселі називають ще околом фон Неймана.

У 8-зв'язних областях до цих напрямків додаються ще і діагональні, тобто 8-зв'язними називаються області, в яких кожний піксель може бути досягнутий з іншого пікселя цієї ж області за допомогою послідовності переміщень на один піксель у восьми напрямках У цьому випадку сусідніми вважаються пікселі, якщо їх координати відрізняються відповідно не більше ніж на одиницю, тобто

$|x1 - x2| \le 1$ і $|y1 - y2| \le 1$.рекурсивний алгоритм зафарбовування

4-зв'язної гранично заданої області .

Pixel_Fill (x, y, bound, new);

// (x,y) – координати початкової точки, з якої почнеться зафарбовування, bound – колір границі, new – новий колір області,якщо точка (x, y) не належить контуру і не зафарбована в колір new, то початок

Зафарбувати точку (x, y) у колір new;

Pixel_Fill (x + 1, y, bound, new);
Pixel_Fill (x, y + 1, bound, new);
Pixel_Fill (x, y – 1, bound, new);
Pixel_Fill (x – 1, y, bound, new);
кінець.

19.Задача відтинання відрізків та багатокутників. Алгоритми відтинання.

Відтинання або клиппинг (англ. clipping) - метод оптимізації рендерінгу і комп'ютерній графіці, коли комп'ютер промальовує тільки ту частину сцени, яка може знаходитися у полі зору користувача. У двомірній графіці, якщо користувач збільшив зображення і на екрані залишилася видно тільки невелика частина, програма може заощадити процесорний час і пам'ять і не промальовувати ті частини зображення, які залишилися "за кадром". Аналогічно, в тривимірній графіці, сцена може складатися з об'єктів (зазвичай трикутників), розташованих з усіх боків віртуальної камери, але програмі достатньо рендерити тільки ті об'єкти, які знаходяться в полі зору. У тривимірній графіці, це нетривіальна задача. Для кожного трикутника в сцені вимагається визначити, входить він в полі зору або ні. Якщо трикутник частково входить в поле зору, то частина його доведеться відсікти.

За способом розв’язування цієї задачі всі алгоритми відсікання поділяються на 2 класи:

– алгоритми, що використовують коди кінців відрізка або самого відрізка;

– алгоритми, що використовують параметричне задання самих відрізків і сторін вікна.

До першого класу алгоритмів належить алгоритми Сазерленда-Коена, FC-алгоритм, до другого класу – алгоритм Кіруса-Бека, Ейлера-Азертона і більш пізній алгоритм Ліанга-Барскі та ін.Алгоритми з кодуванням застосовуються до прямокутних вікон,сторони яких паралельні осям координат, а алгоритми з параметричним заданням – до будь-яких вікон.

20.Двувимірний алгоритм Коена-Сазерленда.

Алгоритм Сазерленда-Коена достатньо просто й ефективно дозволяє прийняти або відсікти відрізки цілком відносно довільного прямокутника. У цьому алгоритмі для двох кінців відрізка будуються 4-бітні коди (зліва направо):

біт 0 визначає чи точка лежить лівіше вікна;

біт 1 визначає чи точка лежить вище вікна;

біт 2 визначає чи точка лежить правіше вікна;

біт 3 визначає чи точка лежить нижче вікна.

Біт набуває одиничного значення, якщо умова істинна, і нуль, якщо умова хибна. Тобто площина з вікном чотирма прямими розбивається на 9 областей, у кожній з яких є свій код (рис. 9.8). Ці коди використовуються для визначення чи знаходиться відрізок повністю у вікні чи зовні вікна.

Отже, маємо такий алгоритм відсікання відрізків.

• Формуємо коди (код1 та код2) для кінців відрізка.

• Якщо два коди для кінців відрізка дорівнюють 0000, то відрізок

повністю знаходиться у вікні.

• Відрізок цілком відсікається, якщо код1 and код2 ≠ 0000.

Якщо код1 and код2 = 0000,то відрізок не можна ні відсікти, ні прийняти. У цьому випадку необхідно шукати точки перетину відрізка з ребрами вікна, причому оскільки відомо, в які області попадають кінці відрізка, то, аналізуючи коди кінців відрізка, легко зрозуміти, з якими саме ребрами вікна може перетинатися відрізок.

22. Двувимірний алгоритм Кіруса-Бека.

Все возможные варианты расположения концов отрезка кодируются восьмибитным кодом (по четыре бита для каждого конца), который интерпретируется одним большим switch с 72 вариантами (всего 92=81, из них девять симметричных). Для каждой ситуации реализуется свой, наиболее оптимальный алгоритм нахождения ответа.

Отрезок можно представить в векторно-параметрическом виде: $V(t) = v0 + t(v1 – v0)$, $t \in [0,1]$, или, покоординатно, $x(t) = x0 + t(x1 – x0) = x0 + t\Delta x$ $y(t) = y0 + t(y1 – y0) = y0 + t\Delta y$ Потребуем также, чтобы $x0 \le x1$ $y0 \le y1$

Тогда задача заключается в нахождении t0 и t1, которые будут соответствовать началу и концу видимой части отрезка. Поставим условие видимости:

$xЛ \le x(t) \le xП$

$yН \le y(t) \le yВ$

или

$xЛ \le (x0 + t\Delta x) \le xП$

$yН \le (y0 + t\Delta y) \le yВ$

что равносильно системе из четырёх неравенств:

$-t\Delta x \le x0 - xЛ$

$t\Delta x \le xП - x0$

$-t\Delta y \le y0 - yН$

$t\Delta y \le yВ - y0$

Общий вид этих неравенств:

$Pit \le Qi$ $i=1..4$

где

$P = \{-\Delta x; \Delta x; -\Delta y; \Delta y\}$

$Q = \{x0 - xЛ; xП - x0; y0 - yН; yВ - y0\}$

Каждое из них соответствует одной из сторон отсекающего окна. Инициализируем t0=0 и t1=1 (что означает, что отрезок полностью виден) и последовательно проверяем все условия:

● Если $Pi > 0$, то $t \le Qi / Pi \Rightarrow t1 \le Qi / Pi \Rightarrow t1 := \min(t1, Qi/Pi)$

● Если $Pi < 0$, то $t \ge Qi / Pi \Rightarrow t0 \ge Qi / Pi \Rightarrow t0 := \max(t0, Qi/Pi)$

● Если $Pi = 0$, то прямая параллельна стороне отсекающего окна. В этом случае:

○ если $Qi \ge 0$, то переходим к следующему неравенству,

○ если $Qi < 0$, то отрезок полностью не виден. break.

23. Задачі візуалізації тривимірних об’єктів. Алгоритми видалення невидимих ліній та поверхонь (класифікація, основна ідея, особливості).

Нехай задано тривимірний об’єкт і видові параметри, що визначають тип проекції та картинну площину. Потрібно визначити, які ребра та грані поверхні об’єкта видимі, якщо на об’єкт дивитися ззовні з деякого центру проектування (для центральних проекцій) або вздовж напрямку проектування (для паралельних проекцій). Виводити на екран необхідно тільки видимі ребра та грані.. Складність цієї задачі привела до появи великої кількості різних методів її розв’язання, в тому числі й апаратних.

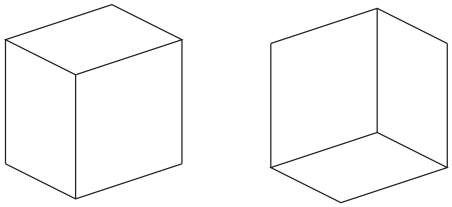
Не існує універсального алгоритму розв’язання цієї задачі, однаково придатного для різних типів об’єктів та сцен.

При побудові каркасних зображень (об’єкти в цій моделі задаються сукупністю ребер) використовуються методи усунення невидимих ліній, для візуалізації суцільних тіл (тут об’єкти задаються сукупністю видимих граней) використовуються методи усунення невидимих поверхонь.

Всі алгоритми усунення невидимих ліній та поверхонь містять сортування, тому їх ефективність істотно залежить від ефективності процесу сортування.

Видалення невидимих ліній та поверхонь

Алгоритми видалення невидимих ліній та поверхонь слугують для визначення ліній, ребер, поверхонь та об’ємів які можуть бути видимими, невидимими або частково видимими спостерігачу який знаходиться в заданій точці простору. Від видимості або невидимості певних ліній залежить інтерпретація об’єкту спостерігачем.



Розглянемо класифікацію методів видалення невидимих частин об’єктів. Їх можна поділити за такими ознаками:

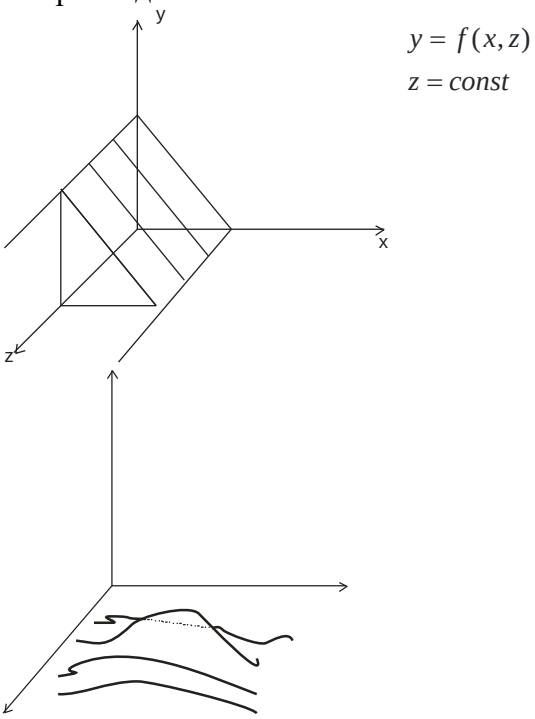
1. За вибором частин, які видаляються (видалення ребер, поверхонь, об’ємів)
2. За порядком обробки елементів сцени (видалення у довільному порядку, видалення у порядку, що визначається процесом візуалізації)
3. За системою координат (Алгоритми які виконуються у просторі зображення коли для кожного піксела зображення визначається яку з n граней об’єкта видно. Для цього кожна з граней порівнюється з позиціями пік селів в системі координат екрану. Прикладом такого алгоритму є алгоритм плаваючого горизонту; Алгоритми які виконуються у просторі об’єктів, коли кожна з n граней об’єкта порівнюється з n-1 гранню що залишились. Прикладом є алгоритм Робертса.)

24. Алгоритм плаваючого горизонту.

Алгоритм плаваючого горизонту частіше за все використовується для видалення невидимих ліній тривимірного представлення функцій, які описують поверхню у вигляді: $F(x, y, z) = 0$.

Основна ідея даного алгоритму полягає у зведенні тривимірної задачі до двовимірної шляхом перетину вихідної поверхні послідовністю паралельних січних площин, які мають сталі значення координат x, y або z.

Наприклад:

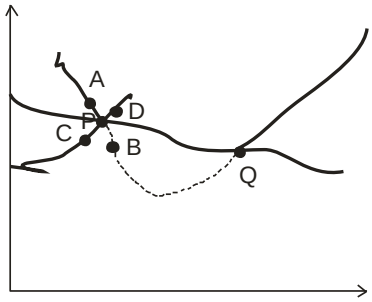


$y = f(x, z)$

$z = const$

Критерій видимості: для кожного значення координати x кожної кривої визначимо відповідне значення, тоді критерій видимості ліній, що спроецьовані на певну площину полягає в наступному: якщо на заданій площині при певному значенні x, відповідне значення y на кривій, видимість якої аналізується більше максимуму (виявлення «гір»), або менше мінімуму (виявлення «ям») по y для всіх попередніх кривих для цього x, то поточна крива є видимою.

Для того щоб візуалізувати отриманий горизонт необхідно визначити точки перетину ліній, які інтерполюють поточну та попередню криві.



$$\begin{cases} x_i = \dfrac{\Delta x_2(y_{n1}\Delta x_2 + x_{n1}\Delta y_1) - \Delta x_2(y_{n2}\Delta x_2 + x_{n2}\Delta y_2)}{\Delta y_1\Delta x_2 - \Delta y_2\Delta x_2} \\ y_i = y_{n1} + \dfrac{\Delta y_1(x_i - x_{n1})}{\Delta x_1} \end{cases}$$

$$\Delta x_i = x_{(n+1)j} - x_{nj}$$

$$\Delta y_i = y_{(n+1)j} - y_{nj}$$

У загальному вигляді алгоритм плаваючого горизонту зводиться до таких кроків:

1. Якщо на поточній площині при певному заданому значенні x відповідне значення y на кривій видимість якої аналізується більше максимуму, або менше мінімуму по y для всіх попередніх кривих для цього x, то поточна крива візуалізується
2. Якщо на проміжку від попереднього (x_n) до поточного (x_{n+k}) значення x видимість кривої змінюється, то обчислюється точка перетину x_i
3. Якщо на проміжку (x_n , x_{n+k}) сегмент кривої повністю видимий, то він візуалізується цілком. Якщо він став невидимим, то візуалізується фрагмент (x_n , x_i). Якщо він став видимим, то візуалізується фрагмент (x_i , x_{n+k}).
4. Якщо крива цілком невидима то виконується перехід до наступної кривої.

25.Алгоритм Робертса.

Цей алгоритм працює безпосередньо в просторі самих об’єктів і застосовується у випадку сцен, що складаються з многогранних об’єктів, кожна грань яких є опуклим многокутником, тобто для опуклих об’єктів. Неопуклі тіла потрібно розбивати на опуклі частини. Крім цього, грані об’єктів не повинні між собою перетинатися.

В алгоритмі Робертса вимагається щоб всі об’єкти були опуклими. Опуклий багатогранний об’єкт з пласкими гранями має задаватись набором площин, які перетинаються. В алгоритмі передбачається що об’єкти складаються з пласких полігональних граней, які складаються з ребер, а ті в свою чергу задаються набором вершин.

Алгоритм Робертса складається з трьох основних етапів:

1. З кожного об’єкта вилучаються ті ребра та грані, які екрануються самим об’єктом
2. Перевіряється чи екрануються ребра та грані, що залишились, іншими об’єктами
3. Обчислюються відрізки, які утворюють нові ребра, якщо об’єкти протикають один-одного

В загальному вигляді алгоритм Робертса зводиться до наступного:

1. Видалення невидимих площин для кожного об’єкта в сцені.
    a. Сформувати багатокутники, виходячи зі списку вершин об’єкта.
    b. Обчислити рівняння площини для кожної полігональної грані об’єкта.
    c. Перевірити знак рівняння площини
        i. Взяти будь-яку точку в середині об’єкта
        ii. Обчислити скалярний добуток рівняння площини та точки в середині об’єкта
        iii. Якщо цей скалярний добуток від’ємний, то змінити знак рівняння площини
        iv. Сформувати матрицю об’єкту

26. Алгоритм z-буфера.

Z-буферизація — алгоритм який відповідає за створення зображень 3D-об'єктів спираючись на глибину елементів зображення. Зазвичай реалізується наапаратному рівні, іноді в програмному забезпеченні. Є одним з рішень проблеми видимості об'єктів, є попіксельним узагальненням алгоритму художника.

При створенні (візуалізації) 3D-об'єкту, його глибина генерується на осі Z-координат і зберігається у Z-буфері. Сам буфер, зазвичай, являє собою двомірний масив X-Y координат із одним елементом для кожного екранного пікселю. Коли інший об'єкт сцени повинен бути відображений у цьому пікселі зараз, тоді порівнюється дві глибини та перекривається поточний піксель, якщо об'єкт знаходиться ближче до спостерігача. Обрана глибина зберігається в Z-буфері і замінює попередню. Зрештою, Z-буфер дозволяє правильно відтворювати звичне для нас сприйняття глибини: ближчий до нас об'єкт перекриває наступні, що розташовуються за ним — невидимих поверхонь.

Для Z-буферизації є критичним роздільна здатність буферу. Чим вона вища, тим краще відображаються глибина об'єктів сцени. Так, 16-розрядний буфер дозволяє реалізувати лише 64 тис. точок. Можуть з'явитися так звані артефакти— коли два об'єкти знаходяться дуже близько один до одного. Тоді як 24 біт Z-буфер, має роздільну здатність 16 млн, 32bit — вже два мільярди.

Z-буферизація дозволяє поєднувати 2D елементи у тривимірному просторі, створюючи декорації, а також складні ефекти такі як відображення різних типів поверхонь.

27.Алгоритми художника та трасування променів (основна ідея, особливості).
Як художник спочатку малює більш далекі об’єкти, а потім поверх них більш близькі, так і метод сортування за глибиною спочатку впорядковує лицьові грані наближенням до спостерігача, а потім виводить їх у цьому ж порядку. Спочатку виводяться більш далекі грані, а потім більш близькі, тобто та грань, яка при проектуванні може закривати іншу, буде виводитися пізніше. Зауважимо, що дві опуклі грані, які не мають спільних точок, завжди можна впорядкувати (для неопуклих граней це в загальному випадку неправильно).
Цей алгоритм базується на такому факті: якщо для двох граней A і B найдальша точка грані A знаходиться ближче до спостерігача, ніж найближча точка грані B (назвемо це умовою AB), то грань B не може закрити грань A . У випадку, коли проектування ведеться вздовж осі z, це означає, що їхні z-оболонки не перетинаються. Тому, якщо для двох лицьових граней виконується умова AB, то для впорядкування таких граней достатньо просто відсортувати їх за відстанню до спостерігача (картинної площини).

Трасування променів у комп'ютерній графіці є способом створення зображення тривимірних об'єктів чи сцени за допомогою відстеження ходу променя світла крізь точку екрану і симуляції взаємодії цього променя з уявними об'єктами, що підлягають відображенню. Для кожного об'єкта існує геометричний опис його форми і оптичних властивостей поверхні та речовини, з якої він зроблений. Зазвичай, кожен промінь має бути перевірено на перетин із певною підмножиною об'єктів сцени. Спочатку слід визначити перший об'єкт, на який падає промінь, підрахувати чи досить точно припустити освітленість об'єкта в точці падіння променя, виходячи з відомих даних про джерела світла у сцені, а потім, враховуючи відомості про оптичні властивості матеріалу об'єкта у точці перетину з променем, обрахувати остаточний колір та яскравість променя-піксела. В разі, якщо матеріал об'єкта в точці падіння променя відбиває промінь, або ж заломлює його, знадобиться додати більше променів до сцени, щоб відстежити колір цього піксела.

28. Моделювання віддзеркалення та переломлення світла. Модель Фонга.

Луч, который приходит непосредственно от источника света, называется**первичным**. Луч, который претерпел одно или несколько переотражений, называется**вторичным**. Физические модели, которые не учитывают перенос света между поверхностями (не используют вторичное освещение), называются **локальными**. В противном случае модели называются **глобальными** или моделями глобального освещения.

Физически обоснованные и эмпирические модели освещения

Физически обоснованные модели материалов. Физически обоснованные модели стараются аппроксимировать свойства некоторого реального материала. Такие модели учитывают особенности поверхности материала, например слои материала (моделирование кожи или тонких пленок) или же поведение частиц материала (моделирование снега, песка, различных жидкостей).

Эмпирические модели материалов. Эмпирические модели устроены несколько иначе, чем физически обоснованные. Как правило, данные модели подразумевают некий набор параметров, не имеющих физической интерпретации, но позволяющих с помощью подгона получить нужный вид конечной модели. Иногда такие модели дают более качественный результат за счет большего контроля за выразительностью, чем за точностью.

Освещение по Фонгу включает в себя также и модель освещения Фонга, т.е. алгоритм расчёта освещения в заданной точке. Это локальная модель освещения, т.е. она учитывает только свойства заданной точки и источников освещения, игнорируя эффекты рассеивания, линзирования, отражения от соседних тел.

Затенение по Фонгу требует сравнительно мало ресурсов, но большинство оптических явлений игнорируются либо рассчитываются с грубым приближением.

Другие модели освещения могут лучше учитывать свойства материала (локальные модели Орена-Наяра, Кука-Торренса, анизотропные модели) или сложные оптические явления (глобальные модели), но ведут к росту накладных расходов.

29. Моделювання зафарбування поверхонь об’єктів. Однотонне зафарбування.

Однотонная закраска полигональной сетки

Существует три основных способа закраски объектов, заданных полигональными сетками. В порядке возрастания сложности ими являются:

1) однотонная закраска;

2) метод Гуро (основан на интерполяции значений интенсивности);

3) метод Фонга (основан на интерполяции векторов нормали).

В каждом из этих случаев может быть использована любая из моделей закраски, описанная выше.

При однотонной закраске вычисляют один уровень интенсивности, который используется для закраски всего многоугольника. При этом предполагается, что:

1. Источник света расположен в бесконечности, поэтому произведение ($\overline{L} \cdot \overline{N}$) постоянно на всей полигональной грани.
2. Наблюдатель находится в бесконечности, поэтому произведение ($\overline{N} \cdot \overline{V}$) постоянно на всей полигональной грани.
3. Многоугольник представляет реальную моделируемую поверхность, а не является аппроксимацией криволинейной поверхности. Если какое-либо из первых двух предположений оказывается неприемлемым, можно воспользоваться усредненными значениями $\overline{L}$ и $\overline{V}$, вычисленными, например, в центре многоугольника.

Последнее предположение в большинстве случаев не выполняется, но оказывает существенно большее влияние на получаемое изображение, чем два других. Влияние состоит в том, что каждая из видимых полигональных граней аппроксимированной поверхности хорошо отличима от других, поскольку интенсивность каждой из этих граней отличается от интенсивности соседних граней. Различие в окраске соседних граней хорошо заметно вследствие эффекта полос Маха.

13.3. Закраска полигональных сеток

Объект, представленный или аппроксимированный гранями, с учетом освещения точечным источником света может быть закрашен разными способами. Существуют три способа закраски объектов, заданных полигональными сетками:

- однотонная закраска;
- интерполяция интенсивностей (метод Гуро);
- интерполяция векторов нормали (метод Фонга).

Однотонная закраска

Данный вариант закраски самый простой. Вычисляется один уровень интенсивности, который используется для закраски всего полигонального многоугольника. Это чаще всего не соответствует реальной картине, так как освещенность в рамках грани меняется. Если из некоторой точки направить один луч к источнику света, а второй — перпендикулярно поверхности, то угол между этими лучами не будет постоянен в рамках полигональной грани. Исключение составляет случай, когда источник света находится в бесконечности. Чем ближе источник света расположен к поверхности, тем больше будет различие в освещенности разных точек грани.

При однотонной закраске предполагается, что:

- источник света расположен в бесконечности ($\cos\theta$ = const на всей полигональной грани);
- наблюдатель находится в бесконечности ($\cos\alpha$ = const на всей полигональной грани);
- многоугольник представляет собой реальную моделируемую поверхность, а не является аппроксимацией криволинейной поверхности.

Рис. 13.12. Пример однотонной закраски

Если первое или второе условие неприемлемо, можно использовать усредненные значения $\cos\theta$, $\cos\alpha$, вычисленные в центре многоугольника.

Третье предположение тоже часто не выполняется, но оно оказывает большое влияние на результат: каждая из видимых граней аппроксимированной поверхности хорошо отличима от других, так как интенсивность этих граней отличается от интенсивности соседних граней, то есть наблюдается эффект полос Маха (рис. 13.12).

30.Моделювання зафарбування поверхонь об’єктів. Зафарбування методом Гуро та Фонга.

Шейдинг — використання затемнення або просвітлення окремих ділянок при створенні зображення. Використовується художниками для створення зображень і в різноманітних графічних програмах.

Заснований на сприйнятті глибини зором в залежності від рівня затемнення зображення.

Затемнення по Гуро це інтерполяційний метод комп'ютерної графіки, який використовується для побудови неперервного градуйованого освітлення поверхонь, описаних у вигляді багатогранників абополігональної сітки з пласкими гранями.

Якщо кожна пласка грань має один постійний колір, визначений з урахуванням відображення, то різні кольори сусідніх граней дуже помітні та поверхня виглядає саме як багатогранник. Здавалося б, цей дефект можна замаскувати за рахунок збільшення числа граней при апроксимації поверхні. Але зір людини має здатність підкреслювати перепади яскравості на кордонах суміжних граней — такий ефект називається ефектом смуг Маху. Тому для створення ілюзії гладкості потрібно набагато збільшити число граней, що призводить до істотного уповільнення візуалізації — чим більше граней, тим менше швидкість малювання об'єктів.

Затемнення по Фонгу це інтерполяційний метод комп'ютерної графіки, який використовується для побудови неперервного градуйованого освітлення поверхонь в 3D комп'ютерній графіці. Також називається інтерполяцією Фонга або нормально-векторною інтерполяціює затінення.Метод засновано наінтерполяції нормалей поверхні по растерізованим полігонам та обчислює колір пікселів на основі інтерпольованої нормалі та моделі відбиття світла. Затемнення по Фонгу може також відноситися до поєднання інтерполяції по Фонгу та моделі відбиття Фонга.На відміну від затемнення Гуро, яке інтерполює кольори через полігони, в затемненні Фонга вектор нормалі до поверхнілінійно інтерполюється на багатокутник по нормалям в вершинах багатокутника. Нормаль поверхні інтерполюється і нормалізується в кожному пікселі, а потім використовується в моделі відбиття Фонга, для отримання кінцевого кольору пікселя. Затемнення по Фонгу потребує більше обчислювальних ресурсів, ніж у затемненні Гуро, оскільки модель відбиття має бути обчисленною в кожному пікселі, а не лише в вершинах

31.Моделювання затіненості об’єктів.

Виділяють три класи алгоритмів побудови тіней:

1. Обчислення затінення в процесі перетворення в растровий вигляд;
2. Поділ поверхонь об'єкта на тіньові та нетеневие площі, що передує перетворенню в растровий вигляд;
3. Включення значення тіней в дані, що описують об'єкт.

У першій групі використовуються алгоритми, засновані на принципі відкидання променів, і сполучають з алгоритмами трасування променів.

Друга група алгоритмів використовує відомі алгоритми визначення видимих поверхонь. При цьому використовуються двопрохідний реалізації алгоритмів прогресивного сканування або z-буфера. Один прохід тут виконується відносно спостерігача, другий - щодо джерела світла.

Побудова тіней в алгоритмі трасування променів:

З кожної точки перетину променя трасування з поверхнею будуються додаткові промені у напрямку до кожного джерела світла. Якщо такий промінь перетинає на своєму шляху яку-небудь поверхню, то на точку, з якої був випущений промінь, падає тінь від цієї поверхні. Таким чином, всі ці промені разом з даними про фізичні характеристики об'єктів моделі (колір, прозорість, дзеркальність і т.д.) дозволяють визначити колір і його інтенсивність для кожної точки зображення.

Побудова тіней з використанням алгоритму z-буфера.

При побудові тіней з використанням алгоритму z-буфера виконується два проходи: один - відносно джерела світла, другий - відносно спостерігача. Для цього виділяється окремий "тіньової" z-буфер. Перший прохід необхідний для того, щоб визначити, які точки видимі з джерела світла. При другому проході сцена візуалізується з положення спостерігача з урахуванням того, що точки, які опинилися невидимі з джерела світла, перебувають у тіні. Потім сцена будується з точки, в якій перебуває спостерігач. При скануванні кожної поверхні значення її глибини в кожному пікселі порівнюється зі значенням глибини в z-буфері. Якщо поверхня видима, то необхідно перевірити, видима чи дана точка з положення джерела світла. Для цього координати точки x, y, z з виду спостерігача лінійно перетворюються в координати x ', y', z 'на вигляді з джерела світла. Перевірка на видимість здійснюється шляхом порівняння значення, яке зберігається в "тіньовому" z-буфері для координат x ', y', і значення z '. Якщо точка невидима з положення джерела світла (значення в "тіньовому" z-буфері більше значення z '), то вона перебуває в тіні і зображається відповідно до правила розрахунку інтенсивності з урахуванням затінення. Якщо ж точка видима з положення джерела світла, то вона зображується без змін.

Включення тіней в алгоритми порядкового сканування.

Додавання тіней в алгоритми прогресивного сканування, як і в алгоритмі z-буфера, здійснюється у два етапи.

На першому етапі для кожного багатокутника сцени і кожного джерела визначаються самозатененние ділянки (що знаходяться у власній тіні) і проекційні (падаючі) тіні. Другий етап полягає в обробці сцени щодо положення спостерігача. Він складається з двох

процесів сканування. Спочатку перший процес визначає, які відрізки видимі на інтервалі, потім другий процес за допомогою сформованого раніше списку тіньових багатокутників знаходить, падає тінь на багатокутник, який створює видимий відрізок на даному інтервалі.

32.Задача деталізації поверхонь. Текстури. Методи накладання текстур.

Деталізація поверхонь

Існують два способи деталізації поверхні: кольором і фактурою. В результаті застосування до гладкої поверхні деталізації кольором форма поверхні не змінюється, якщо ж проводиться деталізація фактурою - поверхня стає шорсткою.

Деталізація кольором

Деталізацію кольором на глибокому рівні легко здійснити шляхом введення багатокутників деталізації поверхні, щоб виділити особливості на основному. Багатокутники деталізації поверхні лежать в одній площині з основними багатокутниками і так помічені в структурі даних, щоб алгоритм видалення схованих поверхонь міг присвоїти їм більш високі пріоритети, ніж основним багатокутників.

У міру того як деталізація кольором стає більш тонкою і складною, безпосереднє моделювання за допомогою багатокутників стає менш практичним.

Деталізація фактурою

Ідея деталізації фактурою полягає у відображенні масиву візерунка, що представляє собою оцифроване зображення, на плоску або криволінійну поверхню. Значення з масиву візерунка використовуються для масштабування дифузної компоненти інтенсивності.

Один піксель на екрані може покривати кілька елементів масиву візерунка. Щоб уникнути проблем, пов'язаних з сходовим ефектом, необхідно враховувати всі зачіпають піксель елементи. Для цього визначаються чотири точки в масиві візерунка, які відповідають чотирьом кутках пікселя. Ці точки в масиві візерунка утворюють чотирикутник. Значення потрапляють в нього елементів зважуються з урахуванням частки кожного елемента, а потім сумуються.

Відображення при такій деталізації проводиться в два етапи:

1. Фіксоване відображення малюнка на поверхню об'єкта. 2.

Видове перетворення об'єкта на екран.

Відображення масиву візерунка впливає на забарвлення поверхні, однак поверхню продовжує здаватися геометрично гладкою. Існує два способи нанесення на поверхню ***деталей фактури.*** У першому з них безпосередню геометричне моделювання фактури не виробляється, і тим не менш виходить хороший візуальний ефект. Для цього вноситься обурення в нормаль до поверхні до її використання в моделі зафарбовування. Ці обурення моделюють невеликі нерівності на поверхні.

Другий спосіб ґрунтується на використанні фрактальних поверхонь, т. Е. Класу нерегулярних форм, що задаються імовірнісним чином і добре описують багато реальні форми, такі, як рельєфи місцевості, берегові лінії, мережі річок, пластівці снігу і гілки дерев.

Текстура (англ. *Texture mapping*) — це спосіб надання поверхні 3D деталі — полігону: кольору, фактури, блиску, матовості та інших фізичних властивостей (для імітації найчастіше якогось природного матеріалу, наприклад: паперу, дерева, каменю, металу тощо). Текстури бувають шовні та безшовні (патерни або візерунки). Головна відмінність останніх в тому, що при поєднанні одних і тих же фрагментів поверхня залишається цілісною.

Якість поверхні текстури визначається текселями — кількістю пікселів на мінімальну одиницю текстури. Оскільки сама по собі текстура є зображенням, роздільність текстури і її формат відіграють велику роль, яка згодом позначається на загальному враженні від якості графіки у 3D-додатку.

Спосіб накладання текстури (texture wrapping) визначає, яким чином текстура буде з'єднуватися з об'єктом. Найпростіший спосіб накладення припускає, що текстура потрапляє на об'єкт як вистріленим з гармати. У цьому випадку кольору текстури проходять об'єкт наскрізь і з'являються з іншого його боку. Такий метод зазвичай називається плоским накладенням. Цей метод часто застосовується для великих об'єктів, особливо коли глядач може бачити тільки одну сторону об'єкта. Плоске накладення є простим у використанні, оскільки вимагає тільки завдання напряму накладення текстури на об'єкт. Так як при

плоскому накладення текстура накладається на об'єкт тільки в одному напрямку, бічні сторони об'єкта зазвичай виходять смугастими.

Інший метод накладення називається циліндричним. При циліндричному накладення текстура згортається в циліндр, усередині якого знаходиться об'єкт.

Також є сферичний метод накладання. При цьому методі текстура згортається в сферу, всередині якої знаходиться об’єкт.

33.Використання, особливості, властивості та види фракталів.
В широкому розумінні **фрактал** означає фігуру, малі частини якої в довільному збільшенні є подібними до неї самої.
Фрактальна графіка — технологія створення зображень на основі фракталів. Фрактальна графіка базується на фрактальній геометрії.

Поява нових елементів меншого розміру відбувається за певним алгоритмом. Основна особливість – описати подібні об'єкти можна всього лише декількома математичними рівняннями.
Однією з основних властивостей фракталів є самоподібність. У самому простому випадку невелика частина фракталу містить інформацію про весь фрактал. Фрактал – структура, яка складається з частин, які в якомусь розумінні подібні цілому.

Існують такі види фракталів:

1) Геометричні. Фрактали цього класу самі наочні. У двомірному випадку їх отримують за допомогою деякої ломаної (чи поверхні в трьохмірному випадку). У результаті безкінечного повторення цієї процедури, отримується геометричний фрактал. Для побудови геометричних фрактальних кривих використовуються рекурсивні алгоритми. Приклад: Крива Коха, сніжинка Коха, лист, трикутник Серпинського, криві Гільберта, криві Серпинського, трикутник Серпинського. Геометричні фрактали застосовуються для отримання зображень дерев, кущів, берегових ліній тощо.

2) Алгебраїчні. Це сама значна група фракталів. їх будують на основі алгебраїчних формул. Методів отримання алгебраїчних фракталів декілька. Один із методів являє собою багатократний (ітераційний) розрахунок функції $Zn+1=f(zn)$, де Z – комплексне число, а f - деяка функція. Розрахунок даної функції продовжується до виконання певної умови. І коли ця умова виконається – на екран виводиться точка. Приклад: Множина Мандельброта.

3) Стохастичні. Стохастичні фрактали, отримуються в тому випадку, коли в ітераційному процесі випадковим чином змінювати будь-які його параметри. При цьому отримуються об'єкти дуже схожі на природні – несиметричні дерева, зрізані берегові лінії і т.д. Алгебричні та стохастичні використовують для побудови ландшафтів, поверхні морів, моделей біологічних та інших об'єктів.

4) **Системи ітеруючих функцій *(IFS – Iterated Function Systems)*.** Кодуваня зображень за допомогою фракталів. Це дозволяло добитися високих ступенів стиснення. Оскільки зображення закодоване за допомогою формул, то його можна збільшити до будь-яких розмірів і при цьому з'являтимуться нові деталі, а не просто збільшиться розмір пікселів. В IFS в ході кожної ітерації якийсь полігон (квадрат, трикутник, круг) замінюється на набір полігонів, кожен із яких піддадуть аффінним перетворенням. При аффінних перетвореннях початкове зображення змінює масштаб, паралельно переноситься уздовж кожної з осей і обертається на деякий кут.

Фракталами добре описуються такі процеси та явища, що стосуються механіки рідин і газів:
- динаміка та турбулентність складних потоків;
- моделювання полум'я;
- пористі матеріали, у тому числі в нафтохімії.

Біологія:
- Моделювання популяцій;
- біосенсорні взаємодії;
- процеси всередині організму, наприклад, биття серця.

34.Зберігання та способи стиснення графічної інформації. Алгоритми стиснення інформації. Формати файлів зображення.

Стиснення зображень — використання алгоритмів стиснення даних до зображень, що зберігаються в цифровому виді. В результаті стиснення зменшується розмір зображення, що зменшує час передачі зображення по мережі і економить простір для зберігання. Стиснення зображень розділяють на стиснення з втратами якості істиснення без втрат. Стиснення без втрат більш підходить для штучно побудованих зображень, таких як графіки, іконки програм, або для спеціальних випадків, наприклад, якщо зображення призначені для подальшої обробки алгоритмами розпізнавання зображень. Алгоритми стиснення з втратами при збільшенні степені стиснення, як правило, породжують добре помітні людському оку артефакти.

Алгоритми архівації без втрат:

Алгоритм RLE - один з найстаріших і найпростіших алгоритмів архівації графіки. Зображення в ньому витягується в ланцюжок байт по рядках растра.

Алгоритм LZW - Стиснення в ньому, на відміну від RLE, здійснюється за рахунокоднакових ланцюжків байт. Ми зчитуємо послідовно символи вхідного потоку і перевіряємо, чи є у створеній нами таблиці рядків такий рядок. Якщо рядок є, то ми зчитуємо наступний символ, а якщо рядка немає, то ми заносимо в потік код для попередньої знайденої рядки, заносимо рядок в таблицю і починаємо пошук знову.

Алгоритм Хаффмана - Використовує тільки частоту появи однакових байт в зображенні. Зіставляє символам вхідного потоку, які зустрічаються більше число раз, ланцюжок біт меншої довжини. І, навпаки, тим, які зустрічається рідко — ланцюжок більшої довжини.

Алгоритм JBIG - для стиснення однобітних чорно-білих зображень. При цьому алгоритм розбиває їх на окремі бітові площини.

Алгоритм Lossless JPEG. На відміну від JBIG, Lossless JPEG орієнтований на повнокольорові 24-бітні або 8-бітові зображення в градаціях сірого без палітри. Він являє собою спеціальну реалізацію JPEG без втрат.

Алгоритми архівації з втратами

Алгоритм JPEG - один з найновіших і досить потужних алгоритмів. В цілому алгоритм заснований на дискретному косинусному перетворенні, що застосовується до матриці зображення для отримання деякої нової матриці коефіцієнтів. Для отримання початкового зображення застосовується зворотне перетворення. Тобто стиснення засноване на усередненні кольору сусідніх пікселів (інформація про яскравість при цьому не усереднюється) і відкиданні високочастотних складових в просторовому спектрі фрагмента зображення.

Фрактальний алгоритм - заснована на тому, що зображення представляється в більш компактній формі — з допомогою коефіцієнтів системи ітерованих функцій (Iterated Function System - IFS). IFS являє собою набір тривимірних афінних перетворень, які переводять одне зображення в інше. Перетворенню підлягають точки в тривимірному просторі (x_координата, y_координата, яскравість).

Рекурсивний (хвильовий) алгоритм - безпосередньо виходить з ідеї використання когерентності областей.

Деякі растрові формати
1.BMP — зазвичай використовується без стиснення, хоча можливо використання алгоритму RLE.
2.GIF — формат, який витісняється PNG та підтримує не більше 256 кольорів одночасно.
3. PSD — стандартний формат пакету Adobe Photoshop і відрізняється від більшості звичайних растрових форматів можливістю зберігання шарів (layers). Формат підтримує альфаканали, шари, контури, прозорість, векторні написи тощо
4.TIFF підтримує великий діапазон зміни глибини кольору, різні колірні простору, різні настройки стиснення.
5.RAW зберігає інформацію, безпосередньо одержувану з матриці цифрового фотоапарата або аналогічного пристрою без застосування до неї будь-яких перетворень.
6.JPEG (Joint Photographic Experts Group) — растровий формат збереження графічної інформації, що використовує Алгоритм JPEG.
7.PNG (Portable Network Graphics) — растровий формат збереження графічної інформації, що використовує стиснення без втрат.

Деякі векторні формати
2D:
Corel Draw Bitmap — основний формат векторного графічного редактора CorelDRAW. Формат CDR став універсальним для інших програм завдяки використанню окремої компресії для векторних і растрових зображень, можливості вбудовувати шрифти, величезному робочому полю 45x45 метрів, підтримці багатосторінковості.
Scalable Vector Graphics (скорочено SVG)— формат файлів для двовимірної векторної графіки, як статичної, так і анімованої та інтерактивної.
3D:
STL — формат для стереолитографии
COLLADA — формат, разработанный для обмена между 3D приложениями.
U3D — Universal 3D
VRML — Virtual Reality Modeling Language
Комплексные форматы:
Djvu, PDF

35.Апроксимація складних поверхонь за допомогою полігональної сітки. Задача тріангуляції та напрями її вирішення. Алгоритми тріангуляції полігонів.
Апроксима́ція — наближене вираження одних математичних об'єктів іншими, простішими, наприклад, кривих ліній — ламаними, ірраціональних чисел — раціональними, неперервних функцій — многочленами.

Полігональна сітка — це набір вершин, ребер, та граней, що описують форму багатогранного об'єкта в тривимірній графіці та твердотільному моделюванні. Грані зазвичай складаються з трикутників, чотирикутників, чи інших опуклих многокутників, що спрощує їх рендеринг, хоча можуть використовуватись і загальніші, неопуклі багатогранники, чи багатогранники з дірками.

Полігональні сітки можуть бути представлені безліччю способів, використовуючи різні способи зберігання вершин, ребер і граней. У них входять:

- Список граней: опис граней відбувається за допомогою покажчиків в список вершин.
- «Крилате» представлення: у ньому кожна точка ребра вказує на дві вершини, дві грані і чотири (за годинниковою стрілкою і проти годинникової) ребра, які її стосуються
- Півреберні сітки: спосіб схожий на «крилате» представлення, за винятком того, що використовується інформація обходу лише половини грані.
- Чотиреберні сітки, які зберігають ребра, півребра і вершини без будь-якої вказівки полігонів. Полігони прямо не виражені в поданні, і можуть бути знайдені обходом структури. Вимоги по пам'яті аналогічні півреберним сіткам.
- Таблиця кутів, які зберігають вершини в зумовленій таблиці, такій що обхід таблиці неявно задає полігони.
- Вершинне представлення: представлені лише вершини, що вказують на інші вершини.

В геометрії, **тріангуляція** в найзагальнішому значенні — це розбиття геометричного об'єкта на симплекси. Наприклад, на площині це розбиття на трикутники, звідки й назва. Тріангуляція тривимірного об'єкта містить розбиття на тетрагедрони («піраміди» разноманітних форм та розмірів), що лежать один до одного.

Тріангуляція *T* простору $\mathbb{R}^{n+1}$ — це підрозбиття $\mathbb{R}^{n+1}$ на (*n* + 1)-вимірні симплекс такі що:

1. будь-які два симплекси в *T* перетинаються в спільній грані ребру чи вершині, або взагалі не перетинаються;

2. будь-яка обмежена множина в $\mathbb{R}^{n+1}$ перетинає скінченну кількість симплексів з *T*.

- Тріангуляція множини точок, тобто, тріангуляція дискретної множини точок $P \subset \mathbb{R}^{n+1}$ — це розбиття опуклої оболонки точок на симплекси так що виконується перша умова з попереднього означення, та множина точок що є вершинами симплексів розбиття збігається з P. Тріангуляція Делоне є найвідомішим видом тріангуляції множини точок.

- Тріангуляція многокутника — це розбиття многокутника на трикутники, що мають спільні ребра з умовою, що множина вершин трикутників співпадає з множиною вершин многокутника. Гранична тріангуляція Делоне — це адаптація тріангуляції Делоне від множин точок до многокутників, у загальнішому — до планарних графів.

- В методі скінченних елементів тріангуляція використовується в якості сітки, що є основою обчислення. В цьому випадку, трикутники повинні утворювати множину в області визначення функції. Для того щоб бути придатними для обчислення, тріангуляція має мати у кожному випадку різні типи трикутників, що залежать від критеріїв звичайно-елементного

маделювання. Наприклад, деякі методі потребують гострокутні чи прямокутні трикутники, що формують нетупокутову сітку. Відомі багато сіточних технік, що містять уточнення Делоне, наприклад другий алгоритм Чу та алгоритм Руперта.

 В більш загальних топологічних просторах, тріангуляція — це розбиття на простіші комплекси, що гомеоморфні простору.

Триангуляція полігону - декомпозиція багатокутника P на безліч трикутників, внутрішні області яких попарно не перетинаються і об'єднання яких в сукупності становить P. У строгому сенсі слова, вершини цих трикутників повинні збігатися з вершинами вихідного багатокутника. Триангуляція будь-якого багатокутника не єдина.

Примітивний алгоритм

У загальному випадку в довільному n-косинці всього n ^ 2 можливих варіантів побудови діагоналей. За $\mathcal{O}(n)$ перевіримо кожний з них. Для цього з'ясуємо:

 перетинає дана діагональ багатокутник - знаходиться за лінійний час перевіркою по всіх ребрах
 чи належить діагональ внутрішній області багатокутника.

Щоб побудувати тріангуляцію потрібно знайти n - 3 діагоналей. У результаті виходить оцінка
$\mathcal{O}(n^4)$.

Для деяких класів багатокутників попередню оцінку можна поліпшити. Наприклад, якщо багатокутник опуклий, то достатньо лише вибирати одну його вершину і з'єднувати з усіма
іншими, крім його сусідів. У підсумку оцінка $\mathcal{O}(n)$.

Монотонний метод

Суть даного методу полягає в тому, щоб розбити багатокутник на монотонні частини, а потім тріангулювати кожну з них.

Простий багатокутник P називається монотонним відносно прямої l, якщо будь-яка l ', така що $l' \perp l$, перетинає сторони P не більше двох разів (результатом перетину l 'і P може бути тільки один відрізок або точка).

Тріангулювати будемо таким методом: проходити зверху вниз по вершинах багатокутника проводячи діагоналі де це можливо.

«Вушний» метод

Вершина v_i называется **ухом**, если диагональ $v_{i-1}v_{i+1}$ лежит строго во внутренней области многоугольника P.

Рассмотрим все вершины многоугольника P, и где возможно, будем отрезать уши до тех пор, пока P не станет треугольником.

Будем рассматривать вершины многоугольника в порядке обхода. Индексирование вершин для удобства будем вести по модулю n, т.е. $v_{-1} = v_{n-1}$ и $v_0 = v_n$. Если вершина v_i является ухом, построим диагональ $v_{i+1}v_{i-1}$ и отрежем треугольник $\Delta v_{i-1}v_iv_{i+1}$ от P. В противном случае переходим к следующей вершине v_{i+1} в порядке обхода.

36. Тріангуляція Делоне.

Тріангуля́ція Делоне́ для множини точок P на площині — це така тріангуляція DT(P), що жодна точка множини P не знаходиться всередині описаних довкола трикутників кіл в множині DT(P). Тріангуляція Делоне дозволяє якомога зменшити кількість малих кутів. Цей спосіб тріангуляції був винайдений Борисом Делоне в 1934 році.

Базуючись на визначенні Делоне, коло описане навколо трикутника утворене трьома точками з вихідної множини точок називається *пустим*, якщо воно не містить вершин трикутника інших ніж ті три, що його задають (інші точки допускаються тільки на периметрі кола, але не всередині)

Умова Делоне стверджує, що мережа трикутників є тріангуляцією Делоне, якщо всі описані кола трикутників пусті. Це є початкове визначення для двовимірного простору. Його можна використовувати для тривимірного простору, якщо використовувати описані сфери замість описаних кіл.

Для множини точок на одній лінії тріангуляції Делоне не існує (фактично, поняття тріангуляції для такого випадку невизначене). Для чотирьох точок на одному колі (наприклад прямокутник) тріангуляція Делоне має два випадки, тобто можна розділити цей чотирикутник двома способами, які задовольняють умови Делоне.

Властивості:
Нехай n — кількість точок, а d — розмірність.

- Об'єднання всіх симплексів в тріангуляції — опукла оболонка точок.
- Тріангуляція Делоне містить щонайбільше $O(n^{\lceil d/2 \rceil})$ симплексів.
- Якщо на площині ($d = 2$), b вершин входять до опуклої оболонки, то будь-яка тріангуляція точок має щонайбільше $2n - 2 - b$ трикутників, і ще одну зовнішню «грань» (див. характеристика Ейлера).
- На площині, кожна вершина має в середньому шість інцидентних трикутників.
- На площині, тріангуляція Делоне максимізує найменший кут. Найменший кут в тріангуляції Делоне, буде не меншим ніж в будь-якій іншій тріангуляції. Правда тріангуляція Делоне не обов'язково мінімізує максимальний кут.
- Коло описане навколо довільного трикутника тріангуляції не містить всередині жодних інших вхідних вершин.
- Якщо коло що проходить через дві вхідні точки не містить всередині жодних інших, тоді сегмент що з'єднує ці дві точки є ребром тріангуляції Делоне цих точок.
- Тріангуляція Делоне множини точок в d-вимірному просторі є проекцією точок опуклої оболонки на $(d + 1)$-мірний параболоїд.
- Найближчий сусід b довільної точки p лежить на ребрі bp тріангуляції Делоне, бо граф найближчих сусідів — підграф тріангуляції Делоне.

Багато алгоритмів для обчислення тріангуляції Делоне спираються на швидкі операції для визначення того, чи точка знаходиться всередині описаного навколо трикутника кола, і ефективних структур даних для зберігання трикутників та ребер. В двовимірному випадку, якщо D знаходиться всередині кола описаного навколо A, B, C треба перевірити чи визначник:

$$\begin{vmatrix} A_x & A_y, & A_x^2 + A_y^2, & 1 \\ B_x & B_y, & B_x^2 + B_y^2, & 1 \\ C_x & C_y, & C_x^2 + C_y^2, & 1 \\ D_x & D_y, & D_x^2 + D_y^2, & 1 \end{vmatrix} = \begin{vmatrix} A_x - D_x, & A_y - D_y, & (A_x^2 - D_x^2) + (A_y^2 - D_y^2) \\ B_x - D_x, & B_y - D_y, & (B_x^2 - D_x^2) + (B_y^2 - D_y^2) \\ C_x - D_x, & C_y - D_y, & (C_x^2 - D_x^2) + (C_y^2 - D_y^2) \end{vmatrix} >$$

Коли *A*, *B* та *C* впорядковані проти годинникової стрілки визначник додатній тоді і тільки тоді, коли *D* знаходиться всередині описаного кола.

Алгоритм заміни ребра

Як вже згадувалось вище, якщо трикутники не задовольняють умові Делоне, ми можемо замінити ребро. З цього можна вивести очевидний алгоритм: побудувати хоч якусь тріангуляцію, а потім замінювати ребра аж поки вона не буде задовольняти умові Делоне. На жаль, це може зайняти O(n^2) замін ребер, і не узагальнюється на три виміри і більші.

Інкрементний

Також досить прямим алгоритмом побудови тріангуляції Делоне є додавання до неї по одній вершині, постійно перебудовуючи тріангуляцію. Коли ми додаємо вершину *v*, ми розбиваємо на три частини трикутник що містить *v*, а потім застосовуємо алгоритм заміни ребра. При повному переборі всіх трикутників які можуть містити *v* ми витратимо час O(n), потім нам треба буде перевірити трикутники на відповідність умові Делоне. Загальний час робота алгоритму O(n^2).

Якщо ми будемо додавати вершини в випадковому порядку, виявиться (через дещо складне доведення), що кожна вставка буде замінювати в середньому лише O(1) трикутників, хоча іноді й більше. Це залишає можливість оптимізувати час виявлення точки. Ми можемо зберігати історію поділів та замін ребер: кожен трикутник зберігає вказівник на два, чи три трикутники що його замінили. Щоб виявити який трикутник містить *v*, ми починаємо з кореневого трикутника, і ідемо по вказівнику який показує на менший трикутник що містить*v*, поки не знайдемо трикутника якого ще не замінювали. В середньому, це займе час O(log n). Загалом для всіх вершин це займе час O(n log n).

Алгоритм Бов'єра-Ватсона (en:Bowyer–Watson algorithm) описує інший підхід до інкрементної побудови, без необхідності заміни ребер.

Розділяй і володарюй

В цьому алгоритмі, рекурсивно проводять пряму, щоб розділити вершини на дві множини. Для кожної з них будується триангуляція Делоне, і потім дві триангуляції зливаються вздовж прямої що їх розділювала. Використовуючи деякі хитрі трюки, операцію злиття можна здійснити за O(n), і загальний час побудови буде O(n log n)

Для деяких видів множин точок, наприклад випадково рівномірно розподілених, розумний вибір розділювальних прямих може зменшити середній час роботи до O(n log log n), при цьому залишаючи час в найгіршому випадку таким самим.

Парадигма «розділяй і володарюй» для виконання тріангуляцій в *d* вимірах описується в «DeWall: A fast divide and conquer Delaunay triangulation algorithm in E^d».

Показано що «розділяй та володарюй» є найшвидшою технікою генерації триангуляції Делоне.

Замітання

Алгоритм Форчуна використовує прийом замітаючої прямої щоб досягти часу O(n log n) у плоскому випадку.

Замітальна оболонка

Замітальна оболонка[є швидкою гібридною технікою побудови двовимірної тріангуляції Делоне що використовує радіально поширювану замітаючу оболонку (яка послідовно утворюється з радіально відсортованої множини точок), в парі з наступним ітеративним заміщенням ребер.

Емпіричні результати показують що алгоритм працює приблизно вдвічі швидше ніж Qhull.