Міністерство освіти і науки України Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» Факультет інформатики та обчислювальної техніки Кафедра обчислювальної техніки

Контрольна робота

з дисципліни

«Комп’ютерна графіка та мультимедіа»

Виконав:

студент групи ІП-05

Гапій Денис Едуардович

номер залікової : 0504

варіант у групі : 4

Перевірив:

Родіонов П. Ю.

Київ 2022

Питання 1 (Фрактали у комп’ютерній графіці. Класифікація фракталів):

Частіше за все у комп'ютерній графіці фрактали використовують для стиснення даних. Основна перевага фрактального стиснення зображень - дуже маленький розмір упакованого файлу і малий час відновлення картинки. Фрактально упаковані картинки можна масштабувати без появи пікселізації. Але процес стиснення займає тривалий час і іноді триває годинами. Декомпресія алгоритму фрактального стиснення проводиться за кілька ітерацій тривимірних афінних перетворень..

Основним елементом фрактальної графіки є рівносторонній трикутник. За допомогою фрактальної графіки можна створити абстрактні зображення, де можна реалізувати такі прийоми як, горизонталі та вертикалі, діагональні напрямки, симетрію і асиметрію. На сьогоднішній день фрактальна графіка є одним з найшвидших і перспективних видів графіки. Варто зазначити, що при збільшенні / приближені фрактального зображення його форма залишається незмінною.

Стосовно класифікації фракталів, їх можна поділити відповідно до їхньої самоподібності (схожості певних частин об'єкта одна на одну). Розрізняють три типи самоподібності у фракталах:

1. Точна самоподібність — фрактал виглядає однаково при різних збільшеннях. Приклад — згенеровані за допомоги ітераційних / повторних функцій.
2. Близька самоподібність — фрактал виглядає приблизно (але не точно) самоподібним при різних збільшеннях. Майже самоподібні фрактали містять малі копії цілого фрактала у перекручених та вироджених формах. Приклад — згенеровані з використанням рекурентних відношень.

3. Статистична самоподібність — фрактал має чисельні або статистичні міри (розмірність , як числова міра і тому подібне), що зберігаються при збільшенні. Приклад — ймовірнісні фрактали.

Згідно інших джререл, альтернативно фрактали можна класифікувати за їх методом генерації / побудови:

1. геометричні — отримують шляхом простих геометричних побудов
2. алгебраїчні — отримуються за допомогою нелінійних процесів у n-мірних просторах
3. стохастичні — утворюються у тому випадку, якщо коли в ітераційному процесі випадковим чином міняти які-небудь його параметри
4. системи ітеруючих функцій — фрактальне зображення закодоване за допомогою формул.

Підсумувати можна тезою, що фрактали — випадкові на перший погляд форми, які насправді є складними геометричними фігурами, що складаються з менших фігур, які точно повторюють більшу.

Питання 2 (Поняття та алгоритми екранного згладжування):

Алгоритм екранного згладжування - зручний прийом / технологія, яка дозволяє ‘згладити’ зображення від зазубрин (зробити повільніший перехід між двома різними точками / пікселями екрану). Зазубрини виникають через «ефект сходів». Зазвичай це виникає через те, що лінія, представлена в растровому режимі, апроксимується послідовністю пікселів. Нерівності можуть виникати з різних причин, але найпоширенішою є те, що пристрій виведення (монітор чи принтер) не має достатньої роздільної здатності для зображення гладкої лінії і в результаті утворюються сходинки. На допомогу і приходить Anti-Aliasing, що за розраховує усереднені значення / відсоток пікселя, що займає дана область у

векторній графіці - у нашому випадку квадрат розміром з піксель, можливо транспонований на кілька пікселів - і використання цього відсотка як кольору. Простіше кажучи відображення значень кольору сусідніх пікселів. Але для більш складних форм алгоритм може бути узагальнений як візуалізація форми в піксельну сітку з вищою роздільною здатністю, ніж цільова поверхня дисплея, а потім використання інтерполяції (бі-кубічної чи іншої зручної) для визначення середньої інтенсивності кожного реального пікселя на поверхні екрану.

Питання 3 (Моделі відбиття світла. Дифузне відбиття):

Направлене освітлення передбачає, що світло надходить рівномірно з одного боку. Для обрахунки відбиття вже використовуються вектори та тригонометричні формули. Головне це знати ‘направлення’ променів світла та поверхню на яку / від якої має падати / відбиватися світло. Найпростіша модель освітлення базується на обчисленні інтенсивності відбитого об’єктом світла точкового джерела. Відбиття світла об’єктом може бути дифузним або дзеркальним.

Дифузне відбиття має місце при рівномірному розсіюванні світла, за рахунок чого створюється ілюзія, що поверхня має однакову яскравість незалежно від кута огляду. Розсіяне світло практично завжди присутнє в реальній обстановці. Світло точкового джерела відбивається від ідеального розсіювача по закону косинусів Ламберта:

$$I = I_l \, k_d \, Cos\theta,$$

де Il – інтенсивність джерела світла, I – інтенсивність відбитого світла, kd – коефіцієнт дифузного відбиття, q – кут між направленням світла та нормаллю до поверхні.

При освітленні об’єкта точковим джерелом на нього падає також і світло від сусідніх об’єктів. Ці світло буде розсіяним. Для обчислення інтенсивності розсіяного світла використовується формула виду:

$$I = I_a k_a + I_l k_d Cos\theta,$$

де Ia – інтенсивність розсіяного світла, ka – коефіцієнт дифузного відображення розсіяного світла (0≤ka≤1).

Питання 4 (Афінні перетворення на площині):

Найбільш відомі два зображення, отримані за допомогою IFS (системи інтегруючих функцій: «трикутник Серпінського» і «папоротник Барнслі», що є умовним фундаментом афінних перетворень. Кожне
перетворення кодується малою кількістю байт, що дозволяє генерувати зображення, які займатимуть менше пам’яті в той час, як зображення без використання перетворень, але такого ж розміру може займати в рази більшу пам’ять.

Визначенням терміну афінного перетворення (affine transformations) можна вважати: системне одноманітне відтворення ліній / форм у дочірніх частинах тієї чи іншої частини зображення. Основними видами таких перетворень є: поворот, відображення, стиснення, розтягнення та переміщення / трансляція. Фрактали тісно пов’язані з афіннами, тому ми можемо зазначити, що самоподібне / взаємнооднозначне перетворення переводить кожну точку однієї площини в іншу точку другої площини, таким чином, що кожній точці 1 відповідає якась точка 2.

Питання 5 (Інкрементний алгоритм виведення еліпса):

Алгоритм Брезенхейма, який можна назвати інверсією алгоритму згладжування — це алгоритм, який визначає які точки в n-вимірному растрові

мають бути накреслені для формування близького наближення для прямої лінії між двома заданими точками (тобто з прямої лінії у ступінчасту).
Окрім алгоритмів побудови прямих ліній, Брезенхем запропонував також алгоритми побудови деяких кривих: кола чи окружності, еліпса та ін.
Наведемо інкрементний алгоритм виведення кола з центром у точці та радіусом, при цьому будемо вважати, що в нас є визначена така функція, що малює точки симетрично відносно точки xc, yc та прямих, що проходять через цю точку паралельно осям координат. В цьому алгоритмі використано симетрію кола – в основному циклі обчислюються координати точок кола тільки для одного октанта (між променями OA і OB) і одразу малюються вісім симетрично розташованих пікселів. Але на відміну від окружності, для якої було достатньо побудувати одну восьму її частину, а потім скористатися властивостями симметрии, еліпс має тільки дві осі симетрії, тому доведеться побудувати одну четверть всієї фігури. Вздовж всієї дуги координати Y є монотонно спадною, але в першій частині дуги вона спадає повільніше, чим росте X, а у другій швидше. Тому в першій частині дуги будемо спочатку збільшити значення X, а у другій спочатку зменшити значення Y.
Псевдокод у додатку 1.

Додаток1:

```
xErr = 0; yErr = 0;
dx = x2 – x1; dy = y2 – y1;
Якщо dx > 0 то incX = 1;
 dx = 0 то incX = 0;
 dx < 0 то incX = -1;
Якщо dy > 0 то incY = 1;
 dy = 0 то incY = 0;
 dy < 0 то incY = -1;
dx = |dx|; dy = |dy|;
Якщо dx > dy
 то d = dx
 інакше d = dy;
x = x1; y = y1;
малюватиПіксел(x,y);
Виконати цикл d разів {
xErr += dx;
yErr += dy;
Якщо xErr > d, то {
xErr -= d; x += incX; }
 Якщо yErr > d, то {
yErr -= d; y += incY;
 }
 малюватиПіксел(x,y);
}
sim(int x, int y, Color col) {
 малюватиПіксел(x+xc, y+yc, col);
 малюватиПіксел(x+xc, -y+yc, col);
 малюватиПіксел(-x+xc, -y+yc, col);
 малюватиПіксел(-x+xc, y+yc, col);
 малюватиПіксел(y+xc, x+yc, col);
 малюватиПіксел(y+xc, -x+yc, col);
 малюватиПіксел(-y+xc, -x+yc, col);
 малюватиПіксел(-y+xc, x+yc, col);
}
 d = 3 – 2 * y;
 x = 0;
 y = r;
 поки (x <= y) {
 sim(x,y);
 якщо (d<0)
 то {
 d = d + 4 * x + 6;
 }
 інакше {
 d = d + 4 * (x - y) + 10;
 dec(y);}
x++;}
```

Використані джерела:

https://classroom.google.com/u/0/c/NTI3MTE5MjU0ODAz/m/NTI3MTE5NTY3MzIz/details

https://ukrbukva.net/99224-Fraktal-naya-komp-yuternaya-grafika.html

https://uk.wikipedia.org/wiki/Фрактал

https://sites.google.com/site/fraktalioneua/home/klasifikacia-fraktaliv

https://en.wikipedia.org/wiki/Spatial_anti-aliasing

https://classroom.google.com/u/0/c/NTI3MTE5MjU0ODAz/m/NTQ2NTA5ODA5Mjk5/details

http://www.berkut.mk.ua/download/pdf/cg_lab_2.pdf

https://web.posibnyky.vntu.edu.ua/fitki/8romanyuk_komp_grafika/zmg1/zmg/43.htm