Trường đại học Khoa học Tự nhiên

Khoa Công nghệ Thông tin

BÁO CÁO ĐỒ ÁN VẼ ĐỐI TƯỢNG 2D

Môn: Đồ họa máy tính

Họ tên	: Trần Kiều Minh Lâm
Mã số sinh viên	: 20120018
Lớp	: Cử nhân Tài năng 2020
Giáo viên phụ trách	: Trần Thái Sơn, Võ Hoài Việt
Năm học	: 2022 - 2023

MỤC LỤC

1. **Cài đặt OpenGL**

   Đối với hệ điều hành Linux (Ubuntu) có công cụ quản lý phần mềm `apt`:

   - Các công cụ cần phải có để chạy được C++ là: `g++` và `make` (không bắt buộc)
   - Để cài đặt thư viên OpenGL và GLUT thì chạy các câu lệnh sau trên terminal.

   Câu lệnh sau để update hệ thống:

   ```
   sudo apt update -y
   ```

   Cài đặt OpenGL bằng câu lệnh sau:

   ```
   sudo apt install libglu1-mesa-dev freeglut3-dev mesa-common-dev
   ```

   - Để sử dụng được thư viện trong chương trình C++, thêm dòng code

   ```
   #include <GL/glut.h>
   ```

   vào đầu chương trình.

   - Chạy chương trình bằng lệnh sau trên terminal

   ```
   g++ main.cpp -o main -lglut -lGLU -lGL
   ```

   sẽ biên dịch file `main.cpp` thành file biên dichj `main`. Gõ lệnh `./main` để chạy.

   Đối với hệ điều hành window, có thể tham khảo tại đây.

2. **Tạo màn hình bằng GLUT**

   Khởi tạo glut bằng lệnh

   ```
   glutInit(&argc, argv);
   ```

với &argc, argv là các tham số chương trình.

Khởi tạo màn hình kích thước `WINDOW_WIDTH` × `WINDOW_HEIGHT` bằng lệnh

```
glutInitWindowSize(WINDOW_WIDTH, WINDOW_HEIGHT);
```

Khởi tạo vị trí của màn hình tại tọa độ `(WINDOW_X, WINDOW_Y)` bằng lệnh

```
glutInitWindowPosition(WINDOW_X, WINDOW_Y);
```

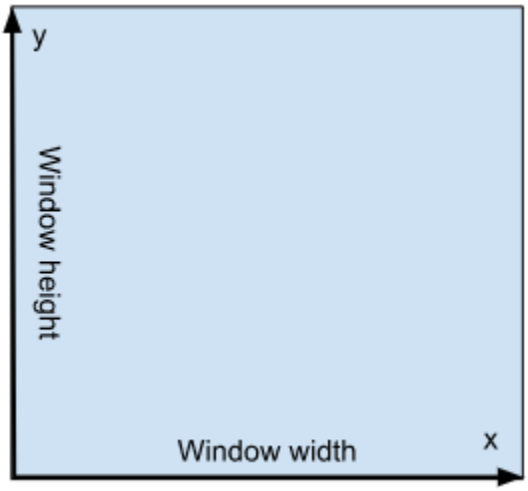
Tạo một màn hình với tiêu đề `Computer Graphics` bằng lệnh

```
glutCreateWindow("Computer Graphics");
```

Sau khi khởi tạo màn hình, nếu chạy chương trình, ta sẽ được một màn hình màu đen với các thông số trên.

Tọa độ các pixel trên màn hình được xác định bằng một hình vuông có kích thước là 2x2 với gốc tọa độ nằm chính giữa.

Để dễ dàng tính toán và mô phỏng đúng bản chất của màn hình là gồm `WINDOW_WIDTH` × `WINDOW_HEIGHT` pixel, ta sẽ chuyển đổi tọa độ trên về dạng



bằng khối lệnh

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0, WINDOW_WIDTH, 0, WINDOW_HEIGHT);
```

Tiếp theo, tạo một hàm `void display()` dùng để vẽ các đối tượng lên màn hình. Sau đó, thêm hàm vừa tạo vào glut để glut gọi đến hàm đó mỗi lần cần vẽ màn hình, bằng lệnh

```
glutDisplayFunc(display);
```

Ngoài ra, tạo một hàm `void reshape(GLsizei width, GLsizei height)` gọi đến hàm `glClear(GL_COLOR_BUFFER_BIT);` dùng để xóa màn hình mỗi khi màn hình bị thay đổi kích thước bởi người dùng và thêm hàm vào glut bằng lệnh

```
glutReshapeFunc(reshape);
```

Sau cùng, tạo một vòng lặp chính (main loop) để glut hiện lên màn hình cho đến khi nào ta tắt màn hình hoặc dừng chương trình, bằng lệnh:

```
glutMainLoop();
```

3. Vẽ các đối tượng 2D

Trước hết cần phải biết, trong OpenGL có hỗ trợ để vẽ các hình cơ bản như điểm, đường thẳng, đa giác,...

Để vẽ tập hợp pixel tại các vị trí `(x1, y1), (x2, y2)...(xn, yn)`

```
glBegin(GL_POINTS);
    glVertex2i(x1, y1);
    glVertex2i(x2, y2);
    …
    glVertex2i(xn, yn);
glEnd();
glFlush();
```

Hàm `glBegin()` xác định hình cơ bản được vẽ. Ở đây là `GL_POINTS` là tập hợp các điểm.

Hàm `glVertex()` xác định các đỉnh (vertex) của hình đó.

Hàm `glEnd()` để báo kết thúc cho `glBegin()`.

Khi gọi hàm `glFlush()` hình mới được thực sự được vẽ lên màn hình.

Ngoài ra, để tô màu cho vertex, gọi hàm `glColor3f(r, g, b)` trước khi gọi `glVertex()` để tô màu vertex đó theo hệ thống màu RGB với giá trị là (r, b, g).

Cụ thể, mỗi lần gọi hàm `glVertex()`, màu của vertex đó được xác định bằng màu của hàm `glColor()` gần nhất, được gọi trước đó. Nếu không tồn tại, màu trắng mặc định sẽ được chọn.

3.1. Đường thẳng

Vẽ đường thẳng đi qua 2 điểm (x_1, y_1) và (x_2, y_2).

- Vẽ bằng thuật toán Digital Differential Analyzer (DDA)

  Trên màn hình máy tính, tọa độ các pixel là các số nguyên. Khi vẽ đường thẳng lên màn hình, sẽ xảy ra trường hợp đường thẳng đi qua các vị trí có tọa độ không nguyên. DDA khắc phục việc đó bằng cách làm tròn tọa độ các điểm trên đường thẳng. Chính vì việc làm tròn tọa độ dẫn đến việc đường thẳng sau khi vẽ bị gấp khúc, không được mịn. Để khắc phục điểm này, ta sẽ cố gắng vẽ càng nhiều điểm càng tốt.

  Gọi dx=x2-x2, dy=y2-y1 là chênh lệnh hoành độ và tung độ của 2 điểm đầu mút.

  Nếu abs(dx) > abs(dy) nghĩa là

  Thuật toán DDA sẽ chọn giá trị bước nhảy đơn vị (bằng 1) dọc theo một trục tọa độ và giá trị bước nhảy tương ứng ở trục tọa độ còn lại được tính sau đó.

  Giá trị bước nhảy đơn vị luôn ở trục tọa độ có khoảng cách giữa 2 điểm lớn hơn. Ví dụ: Nếu Dx = 10 và Dy = 5 thì ta sẽ chọn giá trị bước nhảy đơn vị ở trục hoành x và tính toán các giá trị bước nhảy dọc theo trục tung y.

  Đoạn thẳng được vẽ bắt đầu từ điểm có tọa độ thấp hơn (dựa theo trục đơn vị) và từng bước vẽ điểm mới cho đến khi chạm đầu mút còn lại. Ở mỗi bước, các điểm được thêm các giá trị như nhau - chính là bước nhảy đã được tính ở trên.

Mã giả của thuật toán DDA cụ thể như sau:

Tính các giá trị $D_x = x_2 - x_1$ và $D_y = y_2 - y_1$

Gọi step là số bước cần nhảy

Tính giá trị trong mỗi bước nhảy của tung độ và hoành độ bằng cách lấy khoảng cách tương ứng chia cho số bước nhảy

// Vẽ điểm ảnh qua từng bước nhảy bắt đầu từ (x0, y0)

Lặp lại `step` lần:

Vẽ điểm ảnh sau khi làm tròn hai giá trị tung độ và hoành độ

Tung độ y tăng lên một giá trị bước nhảy tương ứng

Hoành độ x tăng lên một giá trị bước nhảy tương ứng.

- Vẽ bằng thuật toán Bresenham

// vì nó chỉ sử dụng số nguyên, phép trừ và dịch chuyển bit, tất cả đều là các hoạt động rất rẻ trong các kiến trúc máy tính tiêu chuẩn

- Vẽ bằng thư viện OpenGL

Vẽ đường thẳng đi qua 2 điểm `(x1, y1)` và `(x2, y2)` trong OpenGL bằng hàm `glBegin(GL_LINES)` và gọi 2 hàm glVertex cho 2 điểm của đường thẳng trước khi gọi hàm `glEnd()`. Cụ thể:

```
glBegin(GL_LINES);
    glVertex2i(x1, y1);
    glVertex2i(x2, y2);
glEnd();
glFlush();
```

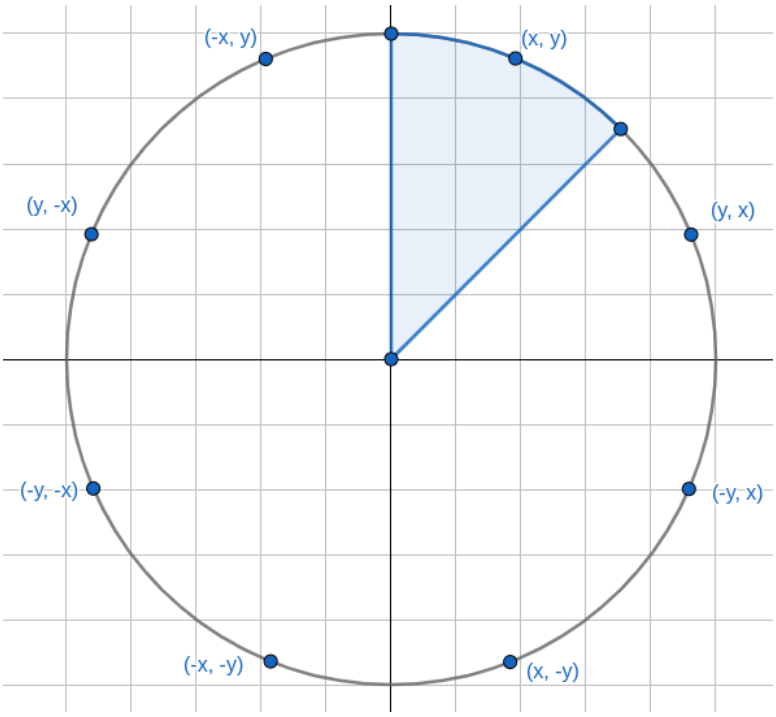
3.2. Đường tròn

Xét đường tròn tâm $O(0,\ 0)$ có bán kính là r có phương trình đường tròn là:

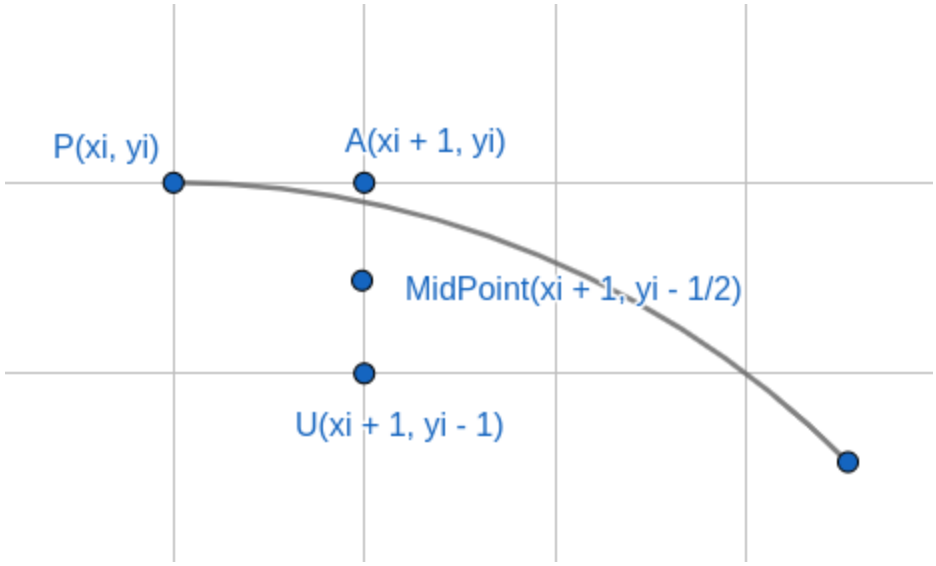
$$f(x,y) = x^2 + y^2 - r^2$$

- ❖ Nếu $f(x,y) < 0$ thì điểm (x,y) nằm trong đường tròn.
- ❖ Nếu $f(x,y) = 0$ thì điểm (x,y) nằm trên đường tròn.
- ❖ Nếu $f(x,y) > 0$ thì điểm (x,y) nằm ngoài đường tròn.

- Vẽ bằng thuật toán MidPoint



Xét đường tròn có tâm là gốc tọa độ. Đường tròn có thể chia thành các cung bằng nhau và bằng $\frac{1}{8}$ đường tròn. Nghĩa là, từ điểm $(x,\ y)$ thuộc đường tròn, có thể dễ dàng xác định đường 8 điểm cách đều nhau trên đường tròn. Do đó, chỉ cần tìm các điểm trên cung $\frac{1}{8}$ là có thể xác định được toàn bộ đường tròn (phần được tô đậm ở trên).



Giả sử đã tìm điểm nguyên $P(x_i, y_i)$, điểm tiếp theo cần phải vẽ có 2 trường hợp xảy ra hoặc là điểm $A(x_i + 1, y_i)$ hoặc là điểm $U(x_i + 1, y_i - 1)$.

Để xác định được điểm tiếp theo là A hay là U, xét điểm $MidPoint(x_i + 1, y_i - \frac{1}{2})$.

- Nếu điểm $MidPoint$ nằm trong đường tròn thì điểm A gần đường tròn hơn. Chọn điểm A.
- Nếu điểm $MidPoint$ nằm trên hoặc ngoài đường tròn thì điểm U gần đường tròn hơn. Chọn điểm U.

Xét $f_i = f(x_i + 1, y_i - \frac{1}{2}) = (x_i + 1)^2 + (y_i - \frac{1}{2})^2 - r^2$

➔ $f_i = x_i^2 + 2x_i + y_i^2 - y_i + \frac{5}{4} - r^2$

Nếu $f_i < 0$ thì chọn điểm A, do đó $y_{i+1} = y_i$

Ngược lại, nếu $f_i \geq 0$ thì chọn điểm U, do đó $y_{i+1} = y_i - 1$

Để tránh việc tính toán với số thập phân của hàm f, ta sẽ tìm cách để tính f_{i+1} dựa trên f_i

Xét $f_{i+1} = f(x_{i+1} + 1, y_{i+1} - \frac{1}{2}) = (x_{i+1} + 1)^2 + (y_{i+1} - \frac{1}{2})^2 - r^2$

Nếu $f_i < 0$ thì $x_{i+1} = x_i + 1, y_{i+1} = y_i$

➔ $f_{i+1} = x_i^2 + 4x_i + 4 + y_i^2 - y_i + \frac{1}{4} - r^2$

➔ $f_{i+1} - f_i = 2x_i + 3$

➔ $f_{i+1} = f_i + 2x_i + 3$

Nếu $f_i \geq 0$ thì $x_{i+1} = x_i + 1,\ y_{i+1} = y_i - 1$

➔ $f_{i+1} = x_i^2 + 4x_i + 4 + y_i^2 - 3y_i + \frac{9}{4} - r^2$

➔ $f_{i+1} - f_i = 2x_i + 3 - 2y_i + 2$

➔ $f_{i+1} = f_i + 2x_i + 3 - 2y_i + 2$

Mã giả thuật toán MidPoint trên đường tròn như sau:

- Vẽ bằng thư viện OpenGL

Trong OpenGL, đường tròn không được xem là một hình cơ bản. Nhưng bản chất của đường tròn là tập hợp các đường thẳng nối bởi các điểm trên đường tròn.

Gọi p là chu vi của đường tròn (làm tròn thành số nguyên). Nếu vẽ $p - 1$ điểm pixel cách đều nhau trên đường tròn, sau đó nối lại với nhau thì chia đường tròn thành p góc quay bằng nhau và mỗi góc quay bằng $\Delta = \frac{2\pi}{p}\ (rad)$.

Điểm đầu tiên có góc quay bằng 0, là điểm cao nhất trong đường tròn. Điểm tiếp theo (theo chiều kim đồng hồ) có góc quay bằng góc quay của điểm trước cộng với Δ.

Từ đó, ta có tọa độ của điểm có góc quay là α là:

$$(x + r \times cos(\alpha),\ y + r \times sin(\alpha))$$

Sau cùng ta vẽ các đường thẳng được nối bởi các điểm thành đường tròn bằng hàm `glBegin(GL_LINE_LOOP)` trong OpenGL.

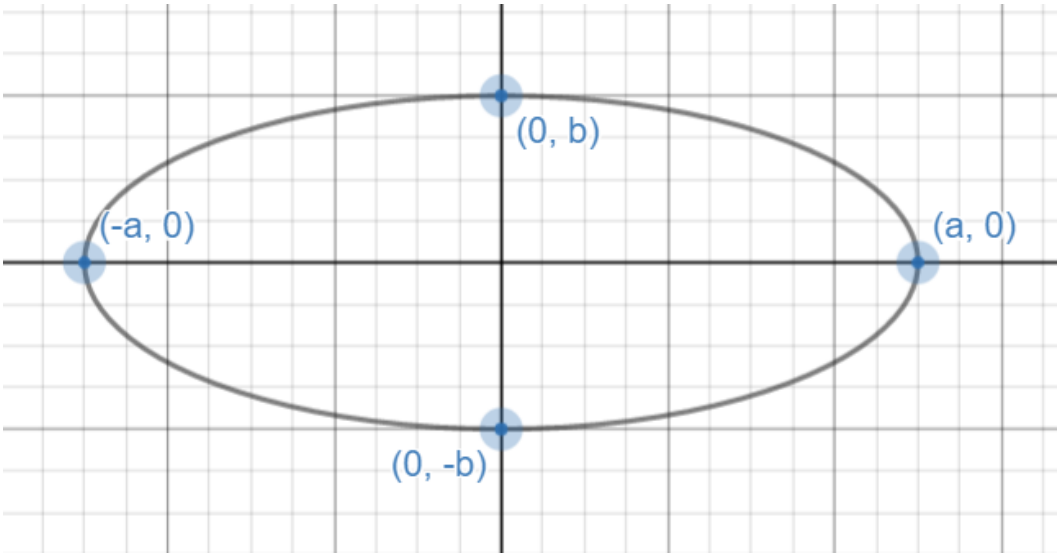
3.3. Đường elip

Xét hình elip có tâm ở gốc tọa độ $O(0,\ 0)$ và chiều dài 2a, chiều rộng 2b có phương trình là:

$$f(x, y) = b^2x^2 + a^2y^2 - a^2b^2$$

- ❖ Nếu $f(x, y) < 0$ thì điểm (x, y) nằm trong elip.
- ❖ Nếu $f(x, y) = 0$ thì điểm (x, y) nằm trên elip.
- ❖ Nếu $f(x, y) > 0$ thì điểm (x, y) nằm ngoài elip.

- Vẽ bằng thuật toán MidPoint



Xét elip có tâm là gốc tọa độ. Đường elip có thể chia thành các cung bằng nhau và bằng $\frac{1}{4}$ đường elip. Do đó, bằng cách thay đổi thông số 2 điểm đầu mút của một cung, ta có thể vẽ được toàn độ đường elip.

¼ đường elip

(Hình ảnh)

Tại một cung elip, ta chia cung thành 2 phần, điểm giao của 2 phần là nơi mà tốc độ biến thiên của x và y có sự đảo chiều hay tại đó 2 đạo hàm riêng của $f(x, y)$ theo trục x và trục y là bằng nhau $f_x = f_y$

- Trong phần thứ nhất (x biến thiên nhanh hơn y):

Giả sử đã tìm điểm nguyên $P(x_i, y_i)$, điểm tiếp theo cần phải vẽ có 2 trường hợp xảy ra hoặc là điểm $A(x_i + 1, y_i)$ hoặc là điểm $U(x_i + 1, y_i - 1)$.

Để xác định được điểm tiếp theo là A hay là U, xét điểm $MidPoint(x_i + 1, y_i - \frac{1}{2})$.

- ❖ Nếu điểm $MidPoint$ nằm trong đường tròn thì điểm A gần đường tròn hơn. Chọn điểm A.
- ❖ Nếu điểm $MidPoint$ nằm trên hoặc ngoài đường tròn thì điểm U gần đường tròn hơn. Chọn điểm U.

Xét $f_i = f(x_i + 1,\ y_i - \frac{1}{2}) = b^2(x_i + 1)^2 + a^2(y_i - \frac{1}{2})^2 - a^2b^2$

Nếu $f_i < 0$ thì chọn điểm A, do đó $y_{i+1} = y_i$

Ngược lại, nếu $f_i \geq 0$ thì chọn điểm U, do đó $y_{i+1} = y_i - 1$

Để tránh việc tính toán với số thập phân của hàm f, ta sẽ tìm cách để tính f_{i+1} dựa trên f_i

Xét

$f_{i+1} = f(x_{i+1} + 1,\ y_{i+1} - \frac{1}{2}) = b^2(x_{i+1} + 1)^2 + a^2(y_{i+1} - \frac{1}{2})^2 - a^2b^2$

Nếu $f_i < 0$ thì $x_{i+1} = x_i + 1,\ y_{i+1} = y_i$

- ➔ $f_{i+1} = b^2(x_i + 1 + 1)^2 + a^2(y_i - \frac{1}{2})^2 - a^2b^2$
- ➔ $f_{i+1} - f_i = b^2[(x_i + 2)^2 - (x_i + 1)^2]$
- ➔ $f_{i+1} - f_i = b^2(2x_i + 3)$
- ➔ $f_{i+1} = f_i + b^2(2x_i + 3)$

Nếu $f_i \geq 0$ thì $x_{i+1} = x_i + 1,\ y_{i+1} = y_i - 1$

- ➔ $f_{i+1} = b^2(x_i + 1 + 1) + a^2(y_i - 1 - \frac{1}{2})^2 - a^2b^2$
- ➔ $f_{i+1} - f_i = b^2[(x_i + 2)^2 - (x_i + 1)^2] + a^2[(y_i - 1 - \frac{1}{2})^2 - (y_i - \frac{1}{2})^2]$
- ➔ $f_{i+1} = f_i + b^2(2x_i + 3) + a^2(-2y_i + 2)$
- ➔ $f_{i+1} = f_i + b^2(2x_i + 3) + a^2(-2y_i + 2)$

- Trong phần thứ hai (y biến thiên nhanh hơn x):

Cách tính tương tự phần thứ nhất, ta có:

Để xác định được điểm tiếp theo là L hay là R, xét điểm $MidPoint(x_i + \frac{1}{2},\ y_i - 1)$.

- ❖ Nếu điểm $MidPoint$ nằm trong đường tròn thì điểm R gần đường tròn hơn. Chọn điểm R.
- ❖ Nếu điểm $MidPoint$ nằm trên hoặc ngoài đường tròn thì điểm R gần đường tròn hơn. Chọn điểm R.

Xét $f_i = f(x_i + \frac{1}{2}, y_i - 1) = b^2(x_i + \frac{1}{2}) + a^2(y_i - 1)^2 - a^2b^2$

Nếu $f_i < 0$ thì $x_{i+1} = x_i + 1, y_{i+1} = y_i - 1$

➔ $f_{i+1} = f_i + b^2(2x_i + 2) + a^2(3 - 2y_i)$

Nếu $f_i \geq 0$ thì $x_{i+1} = x_i, y_{i+1} = y_i - 1$

➔ $f_{i+1} = f_i - a^2(3 - 2y_i)$

- Vẽ bằng thư viện OpenGL

Trong OpenGL, đường elip không được xem là một hình cơ bản. Nhưng bản chất của đường elip là tập hợp các đường thẳng nối bởi các điểm trên đường elip.

Gọi $p \approx 2\pi\sqrt{\frac{a^2+b^2}{2}}$ là chu vi của đường elip (làm tròn tới số nguyên). Tâm đường elip là (x, y) có độ dài 2 trục là a và b.

Nếu vẽ $p - 1$ điểm pixel cách đều nhau trên đường elip, sau đó nối lại với nhau thì chia đường tròn thành p góc quay gần bằng nhau và mỗi góc quay bằng $\Delta \approx \frac{2\pi}{p} (rad)$.

Điểm đầu tiên có góc quay bằng 0, là điểm cao nhất trong đường elip. Điểm tiếp theo (theo chiều kim đồng hồ) có góc quay bằng góc quay của điểm trước cộng với Δ.

Từ đó, ta có tọa độ của điểm có góc quay là α là:

$$(x + a \times cos(\alpha), y + b \times sin(\alpha))$$

Sau cùng ta vẽ các đường thẳng được nối bởi các điểm thành đường elip bằng hàm `glBegin(GL_LINE_LOOP)` trong OpenGL.

3.4. Đường parabol

Xét parabol có đỉnh trùng với gốc tọa độ, có phương trình:

$$f(x, y) = x^2 - 4ay^2$$

- ❖ Nếu $f(x, y) < 0$ thì điểm (x, y) nằm trong parabol.
- ❖ Nếu $f(x, y) = 0$ thì điểm (x, y) nằm trên parabol.
- ❖ Nếu $f(x, y) > 0$ thì điểm (x, y) nằm ngoài parabol.

- Vẽ bằng thuật toán MidPoint

Xét parabol có đỉnh là gốc tọa độ. Parabol có thể chia thành 2 phần bằng nhau. Để thuận tiện, ta chỉ xét trên phần nằm bên trục dương x

(Hình ảnh)

Ta tiếp tục chia thành 2 phần, điểm giao của 2 phần là nơi mà tốc độ biến thiên của x và y có sự đảo chiều hay nói cách khác là tại đó 2 đạo hàm riêng của $f(x, y)$ theo trục x và trục y bằng nhau

($f_x = f_y$ hay $x = -8a\ y$)

- Trong phần thứ nhất (x biến thiên nhanh hơn y):

Giả sử đã tìm điểm nguyên $P(x_i, y_i)$, điểm tiếp theo cần phải vẽ có 2 trường hợp xảy ra hoặc là điểm $L(x_i, y_i + 1)$ hoặc là điểm $R(x_i + 1, y_i + 1)$.

Để xác định được điểm tiếp theo là A hay là U, xét điểm $MidPoint(x_i + \frac{1}{2}, y_i + \frac{1}{2})$.

- ❖ Nếu điểm $MidPoint$ nằm trong đường tròn thì điểm R gần đường tròn hơn. Chọn điểm R.
- ❖ Nếu điểm $MidPoint$ nằm trên hoặc ngoài đường tròn thì điểm L gần đường tròn hơn. Chọn điểm L.

Xét $f_i = f(x_i + \frac{1}{2}, y_i + 1) = (x_i + \frac{1}{2})^2 - 4a(y_i + 1)^2$

Nếu $f_i < 0$ thì chọn điểm R, do đó $x_{i+1} = x_i + 1$

Ngược lại, nếu $f_i \geq 0$ thì chọn điểm U, do đó $x_{i+1} = x_i$

Để tránh việc tính toán với số thập phân của hàm f, ta sẽ tìm cách để tính f_{i+1} dựa trên f_i

Xét $f_{i+1} = f(x_{i+1} + \frac{1}{2}, y_{i+1} + 1) = (x_i + \frac{1}{2})^2 - 4a(y_i + 1)^2$

Nếu $f_i < 0$ thì $x_{i+1} = x_i + 1$, $y_{i+1} = y_i + 1$

Nếu $f_i \geq 0$ thì $x_{i+1} = x_i$, $y_{i+1} = y_i + 1$

- Vẽ bằng thư viện OpenGL

Trong OpenGL, đường parabol không được xem là một hình cơ bản. Nhưng ta có thể dùng công thức của parabol để tính tọa độ các điểm cần vẽ.

Trong đoạn thứ nhất, ta vẽ parabol theo x và y được tính theo x bằng công thức $y = \frac{x^2}{4a}$

Trong đoạn thứ nhất, ta vẽ parabol theo y và x được tính theo y bằng công thức $x = \sqrt{4ay}$

Sau cùng ta làm tròn và vẽ pixel hàm `glBegin(GL_POINTS)` trong OpenGL.

3.5. Đường hyperbol

Xét hyperbol có tâm trùng với gốc tọa độ, có phương trình:

$$f(x, y) = x^2 + y^2 - r^2$$

- ❖ Nếu $f(x, y) < 0$ thì điểm (x, y) nằm trong đường tròn.
- ❖ Nếu $f(x, y) = 0$ thì điểm (x, y) nằm trên đường tròn.
- ❖ Nếu $f(x, y) > 0$ thì điểm (x, y) nằm ngoài đường tròn.

- Vẽ bằng thuật toán MidPoint

Xét hyperbol có tâm là gốc tọa độ. Đường hyperbol có thể chia thành các cung bằng nhau và bằng $\frac{1}{4}$ đường hyperbol. Do đó, chỉ cần vẽ một cung ta sẽ vẽ được toàn bộ hyperbol bằng....

(Hình ảnh)

Tại một cung hyperbol, ta chia cung thành 2 phần, điểm giao của 2 phần là nơi mà tốc độ biến thiên của x và y có sự đảo chiều hay nói cách khác tại đó 2 đạo hàm riêng của $f(x, y)$ theo trục x và trục y là bằng nhau

($f_x = f_y$ hay $2b^2x = -2a^2y$)

- Trong phần thứ nhất (x biến thiên nhanh hơn y):

Giả sử đã tìm điểm nguyên $P(x_i, y_i)$, điểm tiếp theo cần phải vẽ có 2 trường hợp xảy ra hoặc là điểm $A(x_i + 1, y_i + 1)$ hoặc là điểm $U(x_i + 1, y_i)$.

Để xác định được điểm tiếp theo là A hay là U, xét điểm $MidPoint(x_i + 1, y_i + \frac{1}{2})$.

- ❖ Nếu điểm $MidPoint$ nằm trong đường tròn thì điểm A gần đường tròn hơn. Chọn điểm A.
- ❖ Nếu điểm $MidPoint$ nằm trên hoặc ngoài đường tròn thì điểm U gần đường tròn hơn. Chọn điểm U.

Xét $f_i = f(x_i + 1, y_i + \frac{1}{2}) = b^2(x_i + 1)^2 - a^2(y_i + \frac{1}{2})^2 - a^2b^2$

Nếu $f_i < 0$ thì chọn điểm A, do đó $y_{i+1} = y_i + 1$

Ngược lại, nếu $f_i \geq 0$ thì chọn điểm U, do đó $y_{i+1} = y_i$

Để tránh việc tính toán với số thập phân của hàm f, ta sẽ tìm cách để tính f_{i+1} dựa trên f_i

Xét

$f_{i+1} = f(x_{i+1} + 1, y_{i+1} + \frac{1}{2}) = b^2(x_{i+1} + 1)^2 - a^2(y_{i+1} + \frac{1}{2})^2 - a^2b^2$

Nếu $f_i < 0$ thì $x_{i+1} = x_i + 1, y_{i+1} = y_i + 1$

- ➔ $f_{i+1} = b^2(x_i + 1 + 1)^2 - a^2(y_i + 1 + \frac{1}{2})^2 - a^2b^2$
- ➔ $f_{i+1} - f_i = b^2[(x_i + 2)^2 - (x_i + 1)^2] - a^2[(y_i + 1 + \frac{1}{2})^2 - (y_i + \frac{1}{2})^2]$
- ➔ $f_{i+1} - f_i = b^2(2x_i + 3) - a^2(2y_i + 2)$
- ➔ $f_{i+1} = f_i + b^2(2x_i + 3) - a^2(2y_i + 2)$

Nếu $f_i \geq 0$ thì $x_{i+1} = x_i + 1,\ y_{i+1} = y_i$

➔ $f_{i+1} = b^2(x_i + 1 + 1) - a^2(y_i + \frac{1}{2})^2 - a^2b^2$

➔ $f_{i+1} - f_i = b^2[(x_i + 2)^2 - (x_i + 1)^2]$

➔ $f_{i+1} - f_i = b^2(2x_i + 3)$

➔ $f_{i+1} = f_i + b^2(2x_i + 3)$

- Vẽ bằng thư viện OpenGL

4. So sánh và đánh giá