Homework 1: Rasterization and Transformation

Wei Xia (wei4@stanford.edu)

July 7, 2016

1. Problem 1. Why is this approach bad? Provide at least two reasons.

   Here are the reasons:
   (a) multiply and square-root operations are very expensive
   (b) the circle will have large gaps for values of x close to R, because the slope of the circle becomes infinite there.

2. Problem 2

   The corresponding symmetric point is of coordinate (y, x)

3. Problem 3.
   (a)Show that the slope of the circle in the 45° arc we are drawing satisfies -1 $\leq$ dy/dx $\leq$ 0

   In the mentioned region, we have x $\geq$ 0, y $\geq$ 0 and y $\geq$ x.

$$\begin{aligned} y &= \sqrt{R^2 - x^2} \\ \frac{dy}{dx} &= -\frac{x}{\sqrt{R^2 - x^2}} \\ &= -\frac{x}{y} \end{aligned} \tag{1}$$

   since we know x $\geq$ 0, y $\geq$ 0 and y $\geq$ x, then

$$-1 \leq \frac{dy}{dx} \leq 0 \tag{2}$$

   (b) Use this fact to justify why if we trace pixels in increasing x along the circle, it is sufficient to consider only the two neighbors.

$$\begin{aligned} y_{next} &= y_{current} + \Delta y \\ y_{next} &= y_{current} + \frac{\Delta y}{\Delta x} * (x_next - x_current) \\ y_{next} &= y_{current} + \frac{\Delta y}{\Delta x} * 1 \end{aligned} \tag{3}$$

   As we know $-1 \leq \frac{dy}{dx} \leq 0$, so $(y_{current} - 1) \leq y_{next} \leq (y_{current})$, which means that y_{next} could be $(x+1, y)$ or $(x+1, y-1)$.

4. Problem 4. Give a rule using d for choosing between $(x_P + 1, y_P)$ and $(x_P + 1, y_P - 1)$ using the property explained above.

$$\begin{aligned} f(x, y) &= x^2 + y^2 - R^2 \\ f(x_P + 1, y_P - \frac{1}{2}) &= (x_P + 1)^2 + (y_P - \frac{1}{2})^2 - R^2 \\ &= x_P^2 + y_P^2 - R^2 + (2x_P + 1) + (-y_P + 0.25) \end{aligned} \tag{4}$$

   if $f(x_P + 1, y_P - \frac{1}{2}) \leq 0$, we choose $(x+1, y)$, else, we choose $(x+1, y-1)$.

5. Problem 5. Provide update rules of the form $d_new = d_old + \Delta_E$ and $d_new = d_old + \Delta_{SE}$ to be applied depending on which pixel you choose.

   If $(x_P + 1, y_P)$ is chosen as E, then here is the way to calculate Δ_E

$$\begin{aligned} \Delta_E &= f(x_P + 2, y_P - \frac{1}{2}) - f(x_P + 1, y_P - \frac{1}{2}) \\ &= (2x_P + 3) \end{aligned} \tag{5}$$

   If $(x_P + 1, y_P - 1)$ is chosen as SE, then here is the way to calculate Δ_{SE}

$$\begin{aligned} \Delta_{SE} &= f(x_P + 1, y_P - \frac{3}{2}) - f(x_P, y_P - \frac{1}{2}) \\ &= (2x_P + 3) + (-2y_P + 2) \\ &= 2x_P - 2y_P + 5 \end{aligned} \tag{6}$$

6. Problem 6. Set $h = d - c$

$$\begin{aligned} d_{initial} &= f(1, R - \frac{1}{2}) \\ &= 1 + R^2 - R + 1/4 \\ &= 1.25 - R \\ h_{initial} &= d_{initial} - c \\ &= 1.25 - R - c \end{aligned} \tag{7}$$

   As we can see, $d_{initial}$ is not integer.

   Set $c = 0.25$, then $h_{initial} = 1 - R$ then when $h \leq -0.25$, choose E, otherwise, choose SE. As we know, the value of $h_{initial}$ is integer and all the updates of Δ_E and Δ_{SE} are all integers, we can have the the criteria as when $h \leq 0$, choose E, otherwise, choose SE.

7. Problem 7. What kind of transformation is the following matrix ? - describe it precisely for full point. What does the eigen vector, with non-zero eigenvalue, of this matrix correspond to ?

   The matrix will rotate by $(-\theta)$ degree in the X-Z plane.

   The rotation matrix has 3 eigenvalues and they are: 1, $e^{i\theta}$ and $e^{-i\theta}$. Correspondingly, the eigen vecters are: $\begin{pmatrix} 0 & \sqrt{2} & \sqrt{2} \\ 1 & 0 & 0 \\ 0 & \sqrt{2}i & -\sqrt{2}i \end{pmatrix}$

   As we can see, because it roate in X-Z plane, so the vector on the y-axis remains the same, so the eigenvalues has 1 and the corresponding eigen vector is [0, 1, 0]. The other two eigen values are the rotation by θ in the complex plane.

8. Problem 8. Write down the projection of a vector $\vec{v}$ on another vector $\vec{u}$.

$$\begin{aligned} projection &= \frac{\vec{v} \cdot \vec{u}}{|u|} * \frac{\vec{u}}{|u|} \\ &= |v| cos\theta * \frac{\vec{u}}{|u|} \end{aligned} \tag{8}$$

   where θ is the angle between vector $\vec{v}$ and vector $\vec{u}$.

9. Problem 9.

(a) Write down the vector that point to p with respect to the camera position b.

Denote O as the origin in the global coordinate system. Then $b\vec{P} = O\vec{P} - O\vec{b} = (x - b_x, y - b_y)$

(b) Write down the projection of the above vector on $\vec{u}$ and $\vec{v}$ separately

$$\begin{aligned}
Projection_on_\vec{u} &= b\vec{P} \cdot \vec{u} \\
&= \frac{(x - b_x, y - b_y) \cdot (u_x, u_y)}{|u|} \\
&= u_x x + u_y y + (-b_x u_x - b_y u_y) \\
Projection_on_\vec{v} &= b\vec{P} \cdot \vec{v} \\
&= \frac{(x - b_x, y - b_y) \cdot (v_x, v_y)}{|v|} \\
&= v_x x + v_y y + (-b_x v_x - b_y v_y)
\end{aligned} \tag{9}$$

(c) What are these projections in terms of the camera frame and the local coordinate $p\prime$? Write these projections in form a homogenous transformation.
From the above project, we can easily get the transform as below.

$$\begin{pmatrix} x\prime \\ y\prime \\ 1 \end{pmatrix} = \begin{pmatrix} u_x & u_y & -(b_x v_x + b_y v_y) \\ v_x & v_y & -(b_x v_x + b_y v_y) \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{10}$$

(d) What special properties of the basis vectors?
$\vec{u}$ and $\vec{v}$ has to be orthogonal.

10. Problem 10.

As we know, the following stands: $\vec{v\prime} = \vec{v}$ - (Projection of $\vec{v}$ on $\vec{n}$) $\vec{n}$

$$\begin{aligned}
\vec{v\prime} &= \vec{v} - \frac{\vec{v} \cdot \vec{n}}{|n|}\vec{n} \\
&= (1 - \vec{n}\vec{n})\vec{v} \\
&= (1 - nn^T)\vec{v}
\end{aligned} \tag{11}$$

11. Rasterize an ellipse that is described the following equation
$b^2x^2 + a^2y^2 = a^2b^2$

Because of the symmetry, so we only needs to find a way to raster in the arc of [0, b] to [a, 0], then we can figure out of how to raster the whole ellipse.
We can assume $a > b$.

Firstly, we will find the slope of a point on the ellipse.

$$\begin{aligned}
y &= \frac{\sqrt{a^2b^2 - b^2x^2}}{a} \\
\frac{dy}{dx} &= \frac{-b^2x}{a\sqrt{a^2b^2 - b^2x^2}}
\end{aligned} \tag{12}$$

when $\frac{dy}{dx} == 1$, then we can get $x = \frac{a^2}{\sqrt{a^2+b^2}}$ and $y = \frac{b^2}{\sqrt{a^2+b^2}}$, here is the criteria for the slope:

- when $x * b^2 <= y * a^2$, then $-1 \leq \frac{dy}{dx} \leq 0$
- when $x * b^2 > y * a^2$, then $\frac{dy}{dx} < -1$

We will derive the update function for $x * b^2 <= y * a^2$ and $x * b^2 > y * a^2$ respectively.

In case $x * b^2 <= y * a^2$, $F(x, y) = b^2x^2 + a^2y^2 - a^2b^2$, then
$d_{initial} = F(1, b - 1/2) = b^2 + a^2/4 - a^2b$, in order to have integer of $d_{initial}$, we can reset the function to be $F(x, y) = 4b^2x^2 + 4a^2y^2 - 4a^2b^2$, then we have $d_{initial} = F(1, b - 1/2) = 4b^2 + a^2 - 4a^2b$.

Update function:
If $d_old < 0$, then E is chosen, here is the update function

$$\begin{aligned} d_{old} &= F(x + 1, y - 1/2) \\ d_{new} &= F(x + 2, y - 1/2) \\ &= d_{old} + (8x + 12)b^2 \\ \Delta_E &= (8x + 12)b^2 \end{aligned} \tag{13}$$

Similarly, If $d_old > 0$, then SE is chosen, the update function is $\Delta_{SE} = F(x+2, y-3/2) - F(x+1, y-1/2) = (8x + 12)b^2 - (8y - 4)a^2$

In case $x * b^2 > y * a^2$, let the $F(x, y) = 4b^2x^2 + 4a^2y^2 - 4a^2b^2$.

- If $d_{old} < 0$, then [x+1, y-1](SE) is chose, here is the update function:

$$\begin{aligned} d_{old} &= F(x + 1/2, y - 1) \\ d_{new} &= F(x + 3/2, y - 2) \\ \Delta_{SE2} &= F(x + 3/2, y - 2) - F(x + 1/2, y - 1) \\ &= (8x + 8)b^2 - (8y - 12)a^2 \end{aligned} \tag{14}$$

- If $d_{old} >= 0$, then [x, y-1](E) is chose, here is the update function:

$$\begin{aligned} \Delta_E &= F(x + 1/2, y - 2) - F(x + 1/2, y - 1) \\ &= -(8y - 12)a^2 \end{aligned} \tag{15}$$

Here is the pseudo code:

```
void pointEllipse(int a, int b, int value) {
    // assume a > b
    int x = 0;
    int y = b;
    int d = 4*b*b + a*a - 4*a*a*b;
    Point(x, y, value);

    while(x*b*b <= y*a*a) { // -1<= dy/dx <=0
        if(d<0) {
            d += (8*x + 12)*b*b; // E is chosen
        } else {                 // SE is chosen
            d += (8*x+12)*b*b - (8*y-12)*a*a;
            y--;
        }
        x++;
        Point(x, y, value);
    }
```

```
        while(x*b*b > y*a*a) { // dy/dx < -1
            if(d<0) { // SE2 is chosen
                d += (8*x+8)*b*b - (8*y-12)*a*a;
                x++;
            } else { // E is chosen
                d += - (8*y-12)*a*a;
            }
            y--;
            Point(x, y, value)
        }
    }
```

Here is the CPP code:

```
#include <iostream>
#include <fstream>
#include <cstdio>
#include <cassert>
using namespace std;

// We'll store image info as globals; not great programming practice
// but ok for this short program.
int a;
int b;
bool **image;

void renderPixel(int x, int y) {
assert(x >= 0 && y >= 0 && x <= 2*a && y <= 2*b);
image[y][x] = 1;

// TODO:  light up the pixel's symmetric counterpart
  image[2*b - y][x] = 1;
  image[2*b - y][2*a - x] = 1;
  image[y][2*a - x] = 1;
}

void rasterizeArc() {
// TODO:  rasterize the arc using renderPixel to light up pixels
  int x = a;
  int y = 2*b;
  int d = 4*b*b+a*a-4*a*a*b;

  renderPixel(x, y);

  while((x-a)*b*b <= (y-b)*a*a) {
    if(d < 0) {
      d += (8*(x-a)+12)*b*b; // select E
    } else {
      d += (8*(x-a) + 12)*b*b - (8*(y-b)-12)*a*a; // select SE
      y--;
    }
    x++;
    renderPixel(x, y);
  }
```

```
  while((x-a)*b*b > (y-b)*a*a) {
    if(d < 0) {
      d += (8*(x-a)+8)*b*b - (8*(y-b)-12)*a*a; // select SE2
      x++;
    } else {
      d += -(8*(y-b)-12)*a*a; // select SE
    }
    if(y > b+1)
      y--;

    if(x >= 2*a && y >= b)
      break;
    renderPixel(x, y);
  }

}

// You shouldn't need to change anything below this point.

int main(int argc, char *argv[]) {
if (argc != 3) {
cout << "Usage: " << argv[0] << " a  b\n";
return 0;
}

#ifdef _WIN32
sscanf_s(argv[1], "%d", &a);
sscanf_s(argv[2], "%d", &b);
#else
sscanf(argv[1], "%d", &a);
sscanf(argv[2], "%d", &b);
#endif
if (a <= 0 || b<=0) {
cout << "Image must be of positive size.\n";
return 0;
}

  if(a <= b) {
cout << "specify ellipse with a > b\n";
return 0;
  }

// reserve image as 2d array
image = new bool*[2*a+1];
for (int i = 0; i <= 2*b; i++) image[i] = new bool[2*a+1];

rasterizeArc();

char filename[50];
#ifdef _WIN32
sprintf_s(filename, 50, "ellispe%dx%d.ppm", a, b);
#else
sprintf(filename, "circle%dx%d.ppm", a, b);
```

```
#endif

ofstream outfile(filename);
outfile << "P3\n# " << filename << "\n";
outfile << 2*a+1 << ' ' << 2*b+1 << ' ' << 1 << endl;

for (int i = 0; i <= 2*b; i++)
for (int j = 0; j <= 2*a; j++) {
outfile << image[2*b-i][j] << " 0 0\n";
  }

// delete image data
for (int i = 0; i <= 2*b; i++) delete [] image[i];
delete [] image;

return 0;
}
```

Here is the result using the above code: