RI3004A 3D Graphics Rendering

Discussion 3 (Answers)

For **Lecture 5** & **Lecture 6**

Please attempt the following questions before you go to your discussion class. Some of the questions may be quite open-ended and some may be even ambiguous. In those cases, you are encouraged to make your own (reasonable) assumptions.

(1) In many old CRT monitors, the pixels are not square. Let's assume the pixel width-to-height aspect ratio is 4:3. Suppose in the camera coordinate frame, there is a disc in the $z = 0$ plane, centered at (100, 200, 0), and has a radius of 10. You want to draw the entire disc as big as possible inside the window, and it should appear circular and not oval. **(A)** If the window size is 600 × 300 (width × height), how would you set up the viewport and the orthographic projection using OpenGL? **(B)** What if the window size is now 300 × 600? **(C)** What if the window size is now 300 × 320?

(A) & (B) & (C)

```
double pixelAspectRatio = 4.0 / 3.0;
int winWidth = 600;
int winHeight = 300;
double center[2] = {100.0, 200.0};
double radius = 10.0;
...
glViewport( 0, 0, winWidth, winHeight );
glMatrixMode( GL_PROJECTION );
glLoadIdentity();

double apparentWinHeight = winHeight / pixelAspectRatio;

if ( winWidth >= apparentWinHeight )
  gluOrtho2D(
    center[0] - radius * ( winWidth / apparentWinHeight ),
    center[0] + radius * ( winWidth / apparentWinHeight ),
    center[1] - radius, center[1] + radius );
else
  gluOrtho2D(
    center[0] - radius, center[0] + radius,
    center[1] - radius * ( apparentWinHeight / winWidth ),
    center[1] + radius * ( apparentWinHeight / winWidth ));
```

(2) We want to implement a new function `MyPerspective()`, which is the same as the original `gluPerspective()` function, but with an additional parameter `rollAngle` to specify how many degrees the view frustum is rotated about the *z*-axis of the camera frame. How would you implement it using the `glFrustum()` function and OpenGL geometric transformation functions (e.g. `glRotate*()`, `glTranslate*()`, and `glScale*()`)? You may assume the constant `PI` is already defined.

```
void MyPerspective( double fovy, double aspect,
                    double zNear, double zFar,
                    double rollAngle )


double fovyRadian = fovy * PI / 180.0;
double nearHalfHeight = zNear * tan( fovyRadian / 2 );
double nearHalfWidth = nearHalfHeight * aspect;

glFrustum( -nearHalfWidth, nearHalfWidth,
           -nearHalfHeight, nearHalfHeight,
           zNear, zFar );

glRotated( -rollAngle, 0.0, 0.0, 1.0 );
```

(3) We want to scan-convert (rasterize) the curve $y = x^2 / 100$ from the pixel locations (0, 0) to (200, 400). Assume there is a function `write_pixel(x, y, color)` to set the color of a pixel at location (*x*, *y*). Write a C program fragment to draw the curve. You are allowed to use floating-point operations and even the square-root function `sqrt()`.

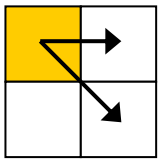
Differentiating the function $y = x^2 / 100$, we find that it has $0 \le$ gradient ≤ 1 when $x \le 50$, and gradient ≥ 1 when $x \ge 50$. So, we must rasterize it in two parts. For the first part, we increment x from 0 to 49, and find the value of y for each x. For the second part, we increment y from $50^2/100 = 25$ to 400 and find the value of x for each y.

```
for ( int x = 0; x <= 49; x++ )
    write_pixel( x, (int)round(x*x/100.0), color );

for ( int y = 50*50/100; y <= 400; y++ )
    write_pixel( (int)round(sqrt(100*y)), y, color );
```

(4) How do you scan convert a circle without using any floating-point operations? Assume that the circle center has integer pixel coordinates and its radius is an integer.

First, draw the top-most pixel on the circle. We now look at the pixel column to the right. We will draw either the pixel on the same row as the previous pixel, or the pixel below (see diagram).

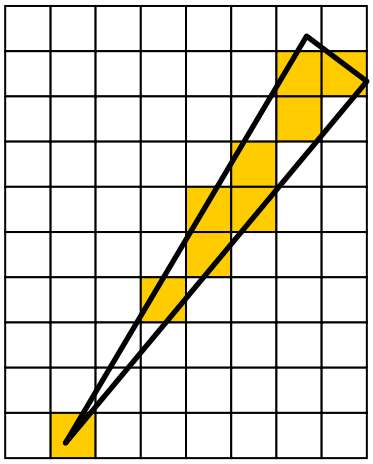


This can be easily decided by computing the squared distance of each pixel's center to the circle center. We choose to draw the pixel that has a squared distance closer to the squared radius of the circle.

The same process is continued for the next column on the right until we have draw 1/8 of the circle. The rest of the circle can be drawn by symmetry.

(5) What could be the problems with scan converting a very thin triangle?

If we draw only pixels whose centers are inside the triangle, then there may be gaps (see diagram for an example). Some very thin triangles may not turn on any pixel at all. This is not really a problem because usually the thin triangle is part of a triangle mesh, and the gap will be filled in by its neighboring triangles.



——— End of Document ———