

CSC418 - Tutorial 2

Drawing 2D Primitives

Primitive Representation

A preliminary step to drawing lines and curves is choosing a suitable representation for them. There are three possible choices which are potentially useful.

Explicit Equations

Basic form: $y = f(x)$

One dimension is specified as a function of another. Therefore, given the location of a point along one dimension of the primitive, the other dimension can be easily found.

Examples:

Line: $y = \frac{dy}{dx}(x - x_0) + y_0$

Circle: $y = \pm\sqrt{r^2 - x^2}, |x| \leq r$

Parametric Equations

Basic form: $x = f(t), y = g(t)$

Each dimension is parameterized with a common variable. Therefore, by traversing the range of the parameter, you can get all points on the primitive. Parametric equations are convenient for describing curves in higher-dimensional spaces.

This way of expressing curves is practical as well as efficient; for example, one can integrate and differentiate such curves for each term. Therefore, finding the tangent to a curve is relatively easy.

Examples:

$$\begin{aligned}x(t) &= x_0 + t(x_1 - x_0) \\y(t) &= y_0 + t(y_1 - y_0)\end{aligned}$$

Line: $t \in [0,1]$

$$\begin{aligned}x(\theta) &= r \cos(\theta) \\y(\theta) &= r \sin(\theta)\end{aligned}$$

Circle: $\theta \in [0,2\pi]$

Implicit Equations

Basic form: $f(x,y) = 0$

Implicit functions can often be useful in situations where it is inconvenient to solve explicitly an equation of the form $F(x,y) = 0$ for y in terms of x . Even if it is possible to rearrange this equation to obtain y as an explicit function $f(x)$, it may not be desirable to do so since the expression of f may be much more complicated than the expression of F .

Implicit equations are also useful if we want to easily classify the position of arbitrary points with respect to the primitive shape. See examples below.

Examples:

Line: $F(x, y) = (x - x_0)dy - (y - y_0)dx$

if $F(x,y) = 0$, then (x,y) is on the line

if $F(x,y) > 0$, then (x,y) is below the line

if $F(x,y) < 0$, then (x,y) is above the line

Circle: $F(x, y) = x^2 + y^2 - r^2$

if $F(x,y) = 0$, then (x,y) is on the circle

if $F(x,y) > 0$, then (x,y) is outside the circle

if $F(x,y) < 0$, then (x,y) is inside the circle

Line Drawing (Bresenham/Midpoint Algorithm)

See separate notes.

OpenGL versions of Lines, Circles, etc.

Lines

Line segments can be drawn easily using the following simple code:

```
glBegin(GL_LINES);
```

```
glVertex3f(10, 15, 20); // Start the line from the 3D point (10, 15, 20)
```

```
glVertex3f(40, 50, 60); // End the line at the 3D point (40, 50, 60)
```

```
glEnd();
```

Each pair of calls to `glVertex3f` will draw a new line segment between the specified start and end points.

Specifying `GL_LINESTRIP` instead of `GL_LINES` allows you to create a connected sequence of line segments. `GL_LINELOOP` is similar to `GL_LINESTRIP`, except that the a final line segment will be drawn from the last point you specified to the first point you specified (thereby closing the loop).

Circles

OpenGL does not have a primitive type for drawing circles. So how would this be done? Use a bunch of line segments!

```
void draw2DCircle(double center_x, double center_y, double radius, int sides)
{
    double increment = PI*2 / sides;
    double current = 0;

    glBegin(GL_LINELOOP);

    for(int i = 0;i < sides;i++)
    {
        glVertex2d(center_x + cos(current)*radius,
                   center_y + sin(current)*radius);

        current += increment;
    }

    glEnd();
}
```

Other Primitives

The same idea can be used to draw different types of shapes as well, such as curves, ellipses, parabolas, etc. (but the code might be more complex).

Polygons

Concave vs Convex

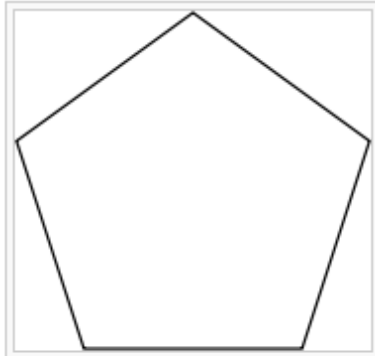
A *simple polygon* is a polygon whose sides do not intersect.

A *convex polygon* is a *simple polygon* whose interior is a convex set (for every pair of points inside the polygon, the line segment that joins them is also completely inside of the polygon). The following properties of a simple polygon are all equivalent to convexity:

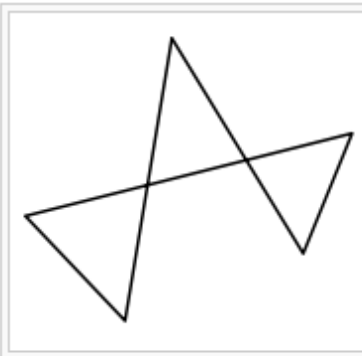
- Every internal angle is at most 180 degrees.
- Every line segment between two vertices of the polygon does not go exterior to the polygon (i.e., it remains inside or on the boundary of the polygon).

If a simple polygon is not convex, it is called *concave*. At least one internal angle of a concave polygon is larger than 180 degrees.

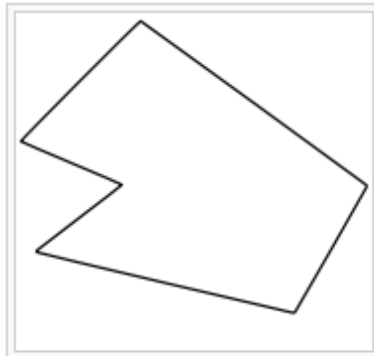
A triangle is a strictly convex polygon.



A convex pentagon



A non-simple (complex) polygon.



A simple concave hexagon

Triangulating polygons

Most graphics hardware is only capable of rendering triangles. A polygon with 4 or more sides is therefore typically made of a number of smaller triangles. *Triangulating* refers to this conversion of arbitrary sized polygons into triangles.