1 Before You Start

The platform that we are using for this project, http://glslsandbox.com is using WebGL in the background. Because WebGL is designed to run on multiple devices, it does not support doubles. In this project, use all floats.

2 Starter Code

Easy to copy Gist link: https://gist.github.com/applekey/c563e941d4ea562ab9e72aecefe12ebe8
paste to here http://glslsandbox.com/e

```glsl
precision mediump float;

varying vec2 surfacePosition;

void main ( void ) {
    vec3 color = vec3(0.8, 0.8, 0.8);
    vec2 h = surfacePosition;

    gl_FragColor = vec4( color , 1.0 );
}
```

3 Glsl Syntax Needed

vec2, vec3, vec4

A convenience struct that contains x, y, z components of a vector. For example, you can instantiate a vec3 simply by writing

```
vec3 color = vec3(0.8, 0.8, 0.8);
```

or access or modify the vec by using dot X, Y, Z.
vec3 color = vec3(0.8, 0.8, 0.8);
color.x = 0.5;

Distance
Returns the absolute distance between 2 points.

float distance(float p0, float p1)
float distance(vec2 p0, vec2 p1)
float distance(vec3 p0, vec3 p1)

Surface Position
The surface position is a parameter supplied by glslsandbox, so if you do any shader programming in the future on other platforms, this value may not be available to you. The surface position is the x, y position of screen, it takes into account screen size, zoom and aspect ratio to ensure that if you try to draw a rectangle, it will always look the right size / aspect ratio with any browser zoom or size. The surface position defines the center point (0, 0) in the middle of your screen.

![Surface Position Centered Grid](Figure 1: Surface Position Centered Grid)

glFragColor
This is how you set the color of pixel that the current main program belongs to. The first 3 values are the r, g, b values and the last 1 is alpha. For this project
just keep alpha at 1.0. Try running this program,

```cpp
if (surfacePosition.x > 0.0) {
    gl_FragColor = vec4( 1.0, 1.0, 1.0, 1.0 );
} else {
    gl_FragColor = vec4( 0.0, 0.0, 0.0, 1.0 );
}
```

Others

normalize → normalizes vec2, vec3, vec4...,
dot → dot product,
cross → cross product,
clamp(d, 0.0, 1.0); → clamps a float between 0.0 and 1.0.

4 First Step: Ray March A Circle

Your first step is to draw a red 2D ball with radius 0.5. Remember the ray marching concept that Nicole taught in lecture. Also, there is no camera definition here per say, but you can image that we are looking directly in front of the 2D ball without any perspective. Hint: the equation of a circle centered at (h,k) is \((x - h)^2 + (y - k)^2 = r^2\), how do you determine if a pixel is inside or outside of this ball?

![Figure 2: Centered Red Ball With Radius 0.5](image-url)
5 Second Step: Calculating Normals

If you recall from lecture, in order to calculate the shading of a surface, we need to determine the normal on the surface. Now, if we were rendering a 2D ball, then we would be done, all the normals would be just facing you out of the screen, \((0, 0, 1)\). But that would be too trivial and I am sure Hank wouldn’t be too pleased either. So let’s try to think in 3 dimensions, we can say that the center of the sphere has a normal that is directly facing out at us. On the other hand, the edges of the sphere have normals facing tangential to the screen. In order to figure out the normal in between those points, we will use the lerping we learned from project 1. Hint: distance and \texttt{vec2}, \texttt{vec3}, are really useful.

Figure 3: Calculating Normals

6 Third Step: Calculating Shading

In this step you will yourself define a light direction and dot product the normal that you calculated in the previous step to get diffuse shading. You can also try
to calculate your view direction to get specular shading but that’s not required. The code here is trivial, its pretty much the same code you used in project 1.

Figure 4: Final Result

7 What To Turn In

1. Your code

2. A screen shot of your shaded ball, it must look shaded, so set your light directions accordingly.

8 Debugging Tips

- glslsandbox is interactive, so make changes, see what they do and experiment.

- glsl does not like setting a float as 1, it does not recognize that 1 is a float and will complain, be specific and write 1.0 instead.
• if you have a compile error, i.e. a line appears red. Go to the page source, right click and "inspect", the compile error will be printed in the console log.