Lab 01: Basics

Tasks

Task 1: Construct a vertex buffer object that will contain 3 different colors. Assign this VBO attribute "1". Extend the vertex shader to accept the new attribute:

```
layout(location=1) in vec3 vertexColor;
```

Add an output variable:

```
out vec3 color;
```

Propagate the vertexColor to the output (in main):

```
// task propagate color to fragment shader
color = vertexColor;
```

Inside the fragment shader add an input variable that will accept the color from the vertex shader and assign the input color to the fragmentColor:

```
in vec3 color;
```

Task 2: Use the following command and test the two modes.

```
// Draw wire frame triangles or fill: GL_LINE, or GL_FILL
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
```

Task 3: Try drawing with GL_LINE_STRIP or GL_LINES or GL_POINTS instead of triangles. Are the lines drawn as you expect? How big are the points by default?

Hint: Make use of:

```
glDrawArrays(GL_POINTS, 0, 3);

glEnable(GL_PROGRAM_POINT_SIZE);

// point size in vertex shader (reserved attribute)
gl_PointSize = 10;
```

Task 4: Color modulation

Go back to drawing triangles instead of points. Add the following line to the bottom of the main() function inside the fragment shader:

```
fragmentColor *= vec3(abs(cos(length(color)*100.0)));
```

Replace the factor "100.0" with a much larger number. Try different values. What happens when the factor changes?

Hints:

- The *length* function creates the circular shapes (because the locus of all points having the same distance to a constant point is a circle).
- The *cos* function creates a periodic sinusoidal behavior featuring increase followed by decrease in color.
- The *abs* function shifts the values of the cosine from [-1, 1] to [0, 1], as color is bound by those values.