Introduction to Computer Graphics and Animation

Exercise 1 of 5

Prof. Dr. Dennis Allerkamp – December 2, 2024

1.1 Hello Triangle

In the course material, you will find a first OpenGL program that displays a triangle. This program provides the basis for further practical exercises. In this task, you are to translate this program and fix any occurring problems.

If your program does not work immediately, you can find practical tips for troubleshooting at LearnOpenGL - Debugging.

If you want to know more about the OpenGL functions used, you will find what you are looking for in the OpenGL 4 Reference Pages. In the coming days, we will delve deeper into the various functions.

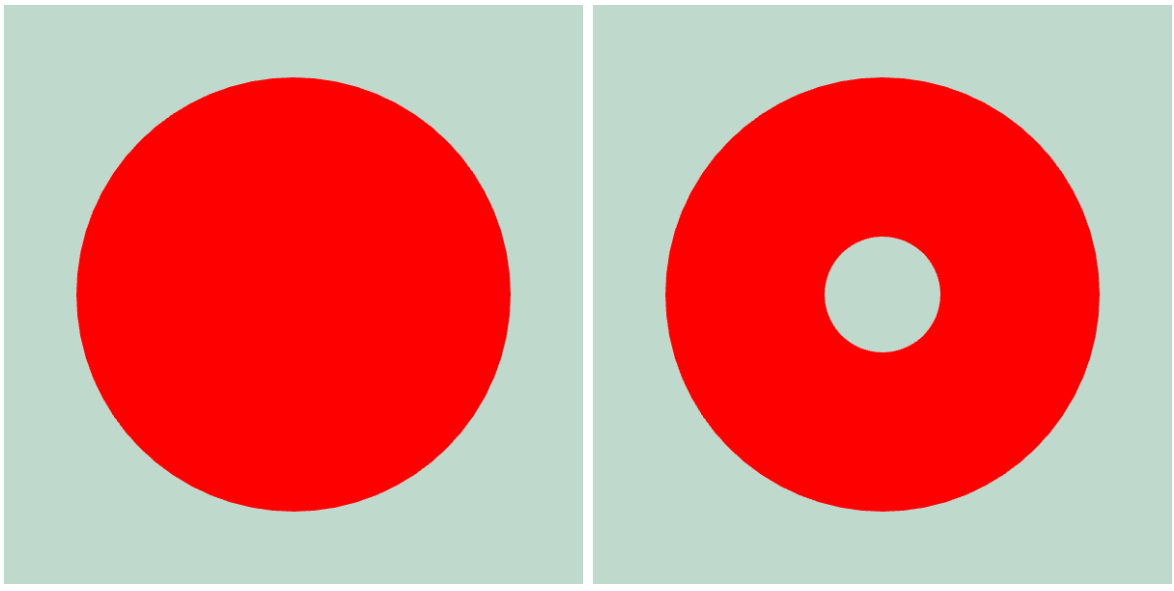
1.2 Disk

Instead of a simple triangle, you should now draw a disk. The OpenGL application from the last task serves as a base.

- The disk should be drawn as `GL_TRIANGLE_FAN`. For this, the center point (here the origin of the coordinate system) must be stored as the first pair of coordinates in the vertex buffer. All other coordinates result from the following formula, where r denotes the radius of the disk and α runs through values from 0 to 2π in equal steps:

$$\vec{x}(\alpha) = \begin{pmatrix} r \cdot \cos(\alpha) \\ r \cdot \sin(\alpha) \end{pmatrix}$$

  The result should look something like the illustration on the left.
- Can you, as shown in the illustration on the right, also draw a disk with a hole? Hint: Use `GL_TRIANGLE_STRIP` for this.



1.3 Uniform Variables

Uniform variables can be declared in the shader similar to attribute variables. However, they are set not per vertex but once per draw call.

- Add to the fragment shader a Uniform variable: `uniform vec3 color;`
- Assign in the shader the value of the uniform variable `color` to the variable `fragColor`.
- Determine with glGetUniformLocation(...) the location of the uniform variable `color`.

- Set with glUniform3f(...) the color `#dc3c05`. (This must first be converted appropriately.)

Now the object should appear in the correct color.

1.3.1 Animation of Color

- Modify your program so that a variable in the `draw` function gets a new value with each frame.
- Ensure now that the color of the object changes continuously.

1.3.2 Animation of Position

- Now not only the color but also the position of the object should be animated.
- Add another uniform variable to the shader.
- Add in the shader the value of the uniform variable to the position.
- Pass in the `draw` function a value to the new uniform variable. Calculate the value such that the object slowly moves across the screen.

1.4 Experiments with Shader Functions

In this task, you are to experiment a little with the fragment shader.

- Write an OpenGL application that draws a rectangle consisting of two triangles on the screen.
- In addition to the two-dimensional position, the vertices should receive another attribute `vertValue`, which should be of type `float`. This should receive the value 0 on the left side of the triangle and the value 1 on the right side. The values should be passed unchanged as variable `fragValue` to the fragment shader.
- Add to the fragment shader two uniform variables `fragColor1` and `fragColor2` of type `vec3`. These should be assigned the colors Red (1, 0, 0) and Green (0, 1, 0) by the application, respectively.

For the following points, really only the fragment shader should be changed. The rest of the application should remain unchanged. The solutions are not immediately obvious, in parts, you have to do a bit of experimenting.

- Use the `mix` function with parameters `fragColor1`, `fragColor2` and `fragValue` in the fragment shader to obtain the following picture:

- Use in the fragment shader a combination of the functions `mix` and `step`, to produce the following image:

- Use in the fragment shader a combination of the functions `mix` and `smoothstep`, to produce the following image:

- Can you generate a repeating pattern using the `mix` and `sin` functions, as well as multiplication and addition?

- Can you now also generate the following pattern?