Homework 2, Computer Graphics 2019

Dr. Szymon Krupiński, Jacobs University Bremen

Handed out 04.04.2019, due 12.04.2019 at 11:15

This is a programming assignment. You can program in C, C++ or python. Please only submit the necessary files, i.e. one source code file (.c, .cpp or .py) per assignment, compressed together in a *.zip file. If you need library functions, only use the libraries mentioned in the assignment or in lecture slides "OpenGL - Basics" (most up to date version of lecture slides is always available on moodle). To qualify for full points, your code should

- compile (if necessary) and execute cleanly
- have readable, clean code with no unnecessary operations
- produce results which are evident (= set the initial position/view/projection right, add lighting, etc)
- comply with the corresponding problem requirements defined below.

Please sign each file with an appropriate comment block in the beginning:

```
/*
Computer Graphics 2019
Problem X.Y
<First_name> <Last_name>
<your_email>@jacobs-university.de
*/
```

Please upload the assignment submission on moodle (`http://moodle.jacobs-university.de`) of our course. In case of technical problem with moodle submission, notify us and send it to the TAs (m.thanasi@jacobs-university.de, muh.hassan@jacobs-university.de) **before** the deadline.

Problem 2.1 *Using primitives* (5 points)

Construct a simple 3-D model of a human being. It should consist (at least) of a torso, a head with a nose, a pair of arms and legs. For the body parts, you can use transformed GLUT objects such as glutSolidSphere, glutSolidCube, glutSolidCone, etc. The body parts shall be attached to each other in a realistic way. Place the human on a square representing the floor. For the pond simply use two aligned triangles. Place at least one light in the scene and give the human and the ground different materials (different colors, different reflection coefficients).

[5pt bonus] write the name of your college slightly above and in the plane of the ground using one of the ttf fonts available in OpenGL. Make sure it's easy to read.

Use perspective projection to view the entire scene. Note: Use `glutInitDisplayMode(GLUT RGBA | GLUT DOUBLE | GLUT DEPTH)` and `glEnable(GL DEPTH TEST)` in your OpenGL initialization and `glClear(GL COLOR BUFFER BIT | GL DEPTH BUFFER BIT)` at the beginning of your display function for a correct rendering. Don't forget to swap the rendering buffers at the end of display callback: `glutSwapBuffers()`.

Problem 2.2 *User interaction* (10 points)

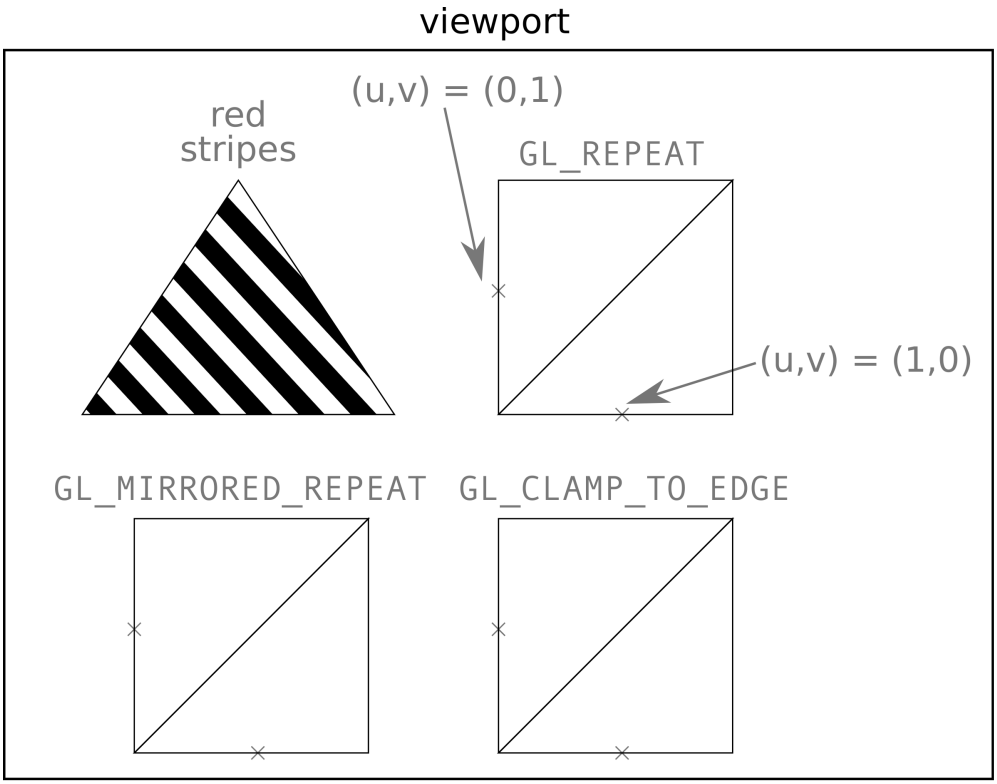
Take the example code provided on moodle called "mypurecode.c" and modify it so that it is interactive for the user. Use the keyboard event callback mechanism to move your geometry and affect the way the image is rendered.

Program the interaction of the cube (just the cube, not the pyramid) in the scene in the following way:

- move it along the three global axes using buttons "w" and "s" (up and down), "a" and "d" (left and right) and "r" and "f" (away/towards the camera)
- rotate it about the three global axes in both directions using buttons "1" and "2" (x axis), "3" and "4" (y axis) and "5" and "6" (z axis)
- button "p" should change the projection from panoramic to orthogonal and back
- Escape button should close the program.

Problem 2.3 *Checkerboard texture* (10 points)

Pick up the "hello_texture.cpp" sample program and modify it so that



- the upper left shape is a triangle. Texture it with an oblique (30-40 degrees) red stripe pattern created through procedural texture (hint: the texture will still be square)
- the upper right shape is a square created of two triangles. The UV coordinates should take their maximal values around the midpoints of the sides. Texture this shape using `GL_REPEAT` parameter
- the lower left and the lower right shapes should be identical but textured using `GL_MIRRORED_REPEAT` and `GL_CLAMP_TO_EDGE` respectively
- feel free to experiment with interpolation parameters to see what impact it has on your rendering.

The illustration above shows a scheme of how the scene should appear in the viewport.

[5pt bonus] Use a simple square image texture for the squares. You can load it from a file using SOIL (C/C++) or PIL (python) libraries. Don't forget to submit it through moodle (use a *.zip file).