Task 1 Readme

In this exercise, we have the following functions:

* drawTriangle() – sets the color of the triangle to red and creates its vertexes.
* drawSquare() – same as before, but the color is green.
* drawCircle(float cx, float cy, float r, int num\_segments) – builds a circle with the center in the point (cx, cy) and the radius r.
* changeSize(int w, int h) – is responsible for changing the perspective when the window size is changed, as w and h are the width and height of the window.
* renderScene() – this function is responsible for the camera’s position and where it’s looking at, plus building the actual objects we create with key presses. Firstly, the function creates an “arena” of sorts, which stands as a point of reference for our objects’ coordinates. After that, it checks which object needs to be drawn, then draws it.
* processNormalKeys(unsigned char key, int xs, int ys) – MARIA EXPLICA TU CE FACE FUNCTIA CU BUFFERII SI MAT4 SI VEC3 SI PULA MEA CE MAI GASESTI PE ACOLO PRETTY PLEASE ♥♥♥♥♥♥♥♥♥. Then, we can do different operations depending on which key we press. The ‘a’ and ‘d’ keys change the horizontal rotation angle of the camera, causing it to turn left or right, while ‘c’ and ‘v’ change the vertical rotation angle, causing it to rotate upwards or downwards (not perfectly, though, as the camera stops at one point and starts going in the opposite direction if either key is held long enough. This probably has to do with the fact that we did not do anything with the camera’s “Up” vector). The ‘w’ and ‘s’ keys cause the camera to go in the direction it is facing, forward and backward, respectively. The ‘r’ and ‘f’ keys cause the camera to go up or down. The ‘1’ key specifies that the program should draw a triangle, through the bool variable triangle. It also calculates the position of the object in space. As with ‘1’, the ‘2’ and ‘3’ keys specify the object we want to draw and calculate its coordinates in space.

Task 2 Readme

In this exercise, we have the following functions:

* drawCube() – sets the color of the cube to light blue, rotates it according to the rotationAngle (initially 0.0), draws a solid cube according to the scale variable (initially 1.0), sets the color to white, then draws a wire cube according to the scale + 0.005. The wire cube serves as an outline to the solid cube.
* changeSize() – same as in Task 1.
* renderScene() – same as in Task 1, only instead of shapes, we have operations (translation, rotation and scaling). If none of them are currently selected, the function just draws the cubes we have created. In order for scaling and translation to be done automatically, a bool variable r (which stands for reverse) that reverses the scaling and translation process, creating the illusion that the cubes are increasing/decreasing in size or going left to right, then right to left, and so on.
* processNormalKeys(unsignes char key, int xs, int ys) – same as in Task 1, with a few exceptions. The camera keys remain unmodified. The changes come with the operation keys. Key ‘1’ randomizes the position of the cube on the screen and increments the noCubes variable by 1. This means we can build as many cubes as we want (up to 1000, we hardcoded it for memory safety). Key ‘2’ sets the program up for rotation, key ‘3’ sets it for scaling and key ‘4’ sets it for translation.