COMPUTER GRAPHICS

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
UNIVERSITY OF WEST ATTICA

DEPARTMENT OF INFORMATICS AND COMPUTER ENGINEERING

SEMESTER PROJECT

BUILDING A 3D GRAPHICS SCENE USING WEBGL

TEAM DETAILS AND DIFFERENCE

DETAILS OF STUDENT 1: ATHANASIOU VASILEIOS EVANGELOS (UNIWA-19390005)
DETAILS OF STUDENT 2: TATSIS PANTELIS (PADA-20390226)

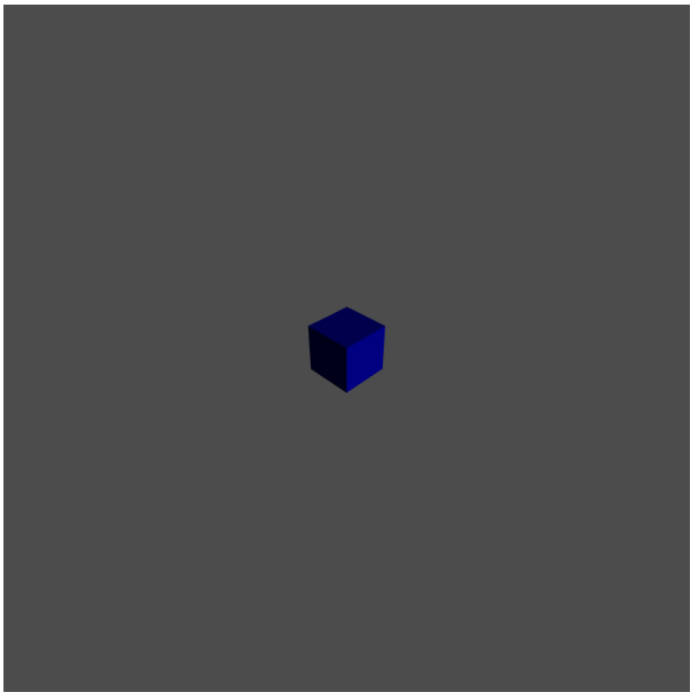
COURSE RESPONSIBLE: BARDIS GEORGIOS
SUBMISSION DATE: 30/07/2024
DEADLINE DATE: 30/07/2024

COMPUTER GRAPHICS

CONTENTS

Scene 1

The source code is in the file 1st_scene.html



Step 1 - Cube Formation

In the first step using the gl.clearcolor() command, inside main we gave the background the requested color. In this case, giving the variables R, G, B the same value results in a shade of gray. To draw the cube as a set of squares we had to set a buffer for the vertices, one for the colors and finally give them the necessary values. The difficulty was inevitable as it was the beginning of the task, however taking ideas from the workshop task 6, which was the first in which we dealt with 3D shapes we managed to complete the first step.

Step 2 - Camera Placement

In the second step of the work, it was necessary to use the functions look A t() and perspective() in order to be able to define the necessary settings on the camera to look properly the cube we implemented in step 1. There was no difficulty in this particular step, as the values of the variables used by our functions were given by the utterance.

Step 3 - View Angle / Rectangular Camera Distance

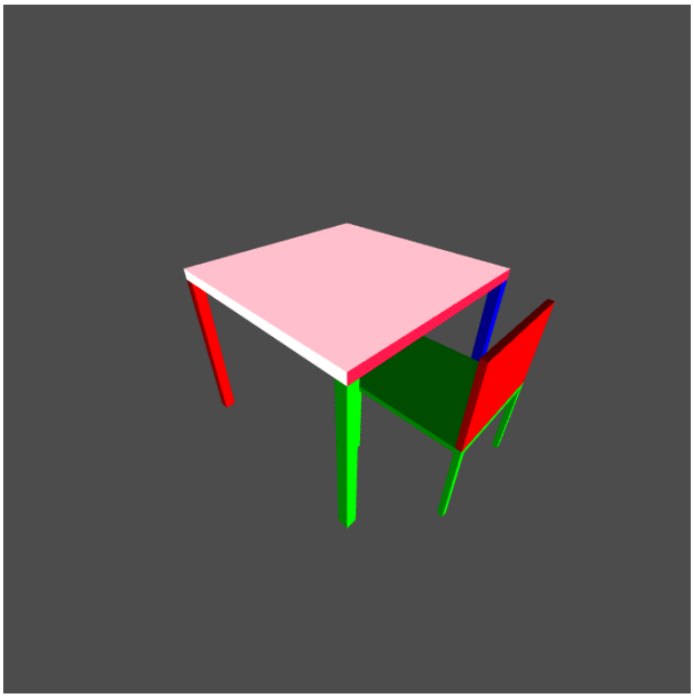
In the third step it was necessary to place text boxes for the viewing angle and for entering the orthogonal distance. We have succeeded in changing the camera position according to the user's choice. How the coordinates change was clear from the pronunciation, so no serious difficulty was encountered in this question either.

Step 4 - Redraw Scene Button

In order to implement step 3, however, some button is also required that will execute the function that redesigns our cube according to the user's desire. The data from which the cube will be recreated is retrieved from the 2 text boxes we have made. The only thing we had to be careful enough with was that we had to set a flag so that the previous commands regarding the placement of the camera would not run.

Scene 2

The source code is in the file 2nd_scene.html



Step 5 - Transform Cube into Table with colors

To implement step 5 we pretty much used the code from scene 1 as a base. With the use of the functions fromTranslation(), fromScaling() and multiply() we achieved respectively moving, enlarging/shrinking and multiplying the tables. The assembly of the table was essentially done piece by piece, so each piece had a different treatment to achieve it. The difficulty encountered in this particular step was that we had to change several values in order for all the pieces of the table to be connected to each other, without one "detaching" from the rest.

Step 6 - Transform Cube into Stool with colors

To transform the final chair, it was first necessary to create a stool which is essentially a table with smaller dimensions. For this reason, it was not necessary to create any new buffers for the sides or the colors, since we already had them available for the construction of the table. There was therefore no difficulty beyond finding the correct stool displacement values at this particular step as all the tools were available.

Step 7 - Transforming a Stool into a Chair with colors

Essentially all that is required from this step is the addition of a new piece as the back of the chair, whose dimensions we have from the speech. Again because they needed to fit a new piece onto an existing object, several tests were needed to get the piece to "click" together with the rest of the construction.

In the program, the initial frame may not be displayed properly and some interaction from the user is needed.

Scene 3

The source code is in the file 3rd_scene.html

Step 8 - Add Animation

Placing the start and pause buttons for the animation is something simple that we had seen in the workshops. Essentially it again required experimenting with the values of the lookat() function to rotate the camera and the Math.cos() and Math.sin() functions to implement the object. The implementation of this particular step was a bit demanding but with proper testing and planning as to how the functions will be called together, we achieved the desired result.

Step 9 - Add Textures instead of colors to the objects

For step 9, we consulted exercise 9 of the workshop, which has to do with textures and how we give the objects a more "realistic" display. So far all the buffers we had were for colors, so we needed to create a new buffer for textures to replace the color buffers with them. The dimensions of the images were chosen to be 512x512 pixels (power of 2) and JPEG format (.jpg) so that we could do pre-processing techniques.

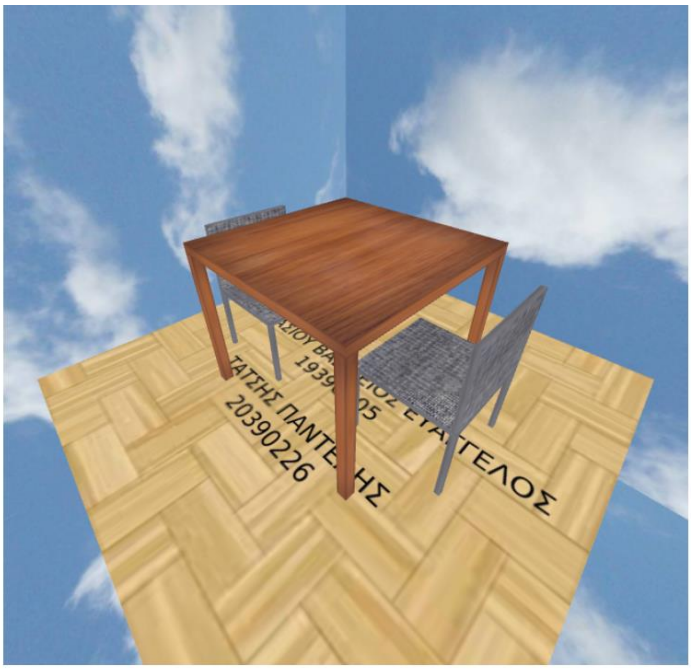
The problem we faced was again with the correct values of the texture buffers, since at first the textures were not loading on the scene. On the contrary, the table and the chair were black in color.

Another problem we faced was in the pre-processing of the images, but with the proper advice of exercise 9 of the workshop and the selection of power dimensions of 2, we were able to overcome this.

For some reason there is no image in the first frame, in order to display the desired result some interaction from the user is required.

Scene 4

The source code is in the file 4th_scene.html



Γωνία θέασης (σε μοίρες): 90 Οριζόντια απόσταση της κάμερας: 20

Θέση κάμερας: Left-Front-Top Left-Front-Bottom Left-Back-Top Left-Back-Bottom Right-Front-Top Right-Front-Bottom Right-Back-Top Right-Back-Bottom

Επανασχεδίασε

Ξεκίνα Σταμάτα

Step 10 - Add Skybox and 2D floor

For this step 10 we had to deal with the sky and the floor. First we placed the objects on the scene by first activating the sky textures and we draw the box based on the instructions given by the speech. Next we dealt with the floor where it was necessary to create and use new buffers such as an indexBuffer and a textureBuffer because it is 2D. Since the floor is 50x50, we didn't deal with the 3rd dimension and just stuck to the design of a flat square.

There was no problem, as due to the texture buffer we created the names were written on the floor correctly and clearly.

Step 11 - Control animation with the mouse

The mouse control in step 11 gave us quite a bit of trouble as we skipped several important steps throughout the creation of this function. Especially regarding mouse control while the animation is stopped. Appropriate changes to the code overcame these issues. By calling the function

animationStep() it helps us refresh the moment. What we're essentially doing is every time the mouse moves is to refresh the scene with any changes.

Step 12 - Overturning a chair with the mouse wheel

In this step we used elements from the last lab exercise and basically with the fromYRotation() method we rotated the chair on the y axis. We also made changes to the movement table with the fromTranslation() method, with changes to the x and z axis values of the chair. The rotation angle is changed by the variable wheelRadiusFactor, where it determines the pitch of the mouse wheel.

Problems we encountered were in the chair movement and mouse wheel pitch values, as we limited the chair to rotate from 0 to 90 degrees. The back of the chair either didn't touch the floor or went through it.

Another problem we encountered was that the chair moved on all 3 axes, even though we were changing 2 of the 3 axes in the translate panel. The problem was solved without changing the y-axis value in the chair's translate array, as the wheelRadiusFactor was already affecting the y-axis with the fromYRotation() method.

Step 13 - Easter Egg

In this step we count with two simple variables how many times the chair has fallen and risen. That is, in the event that it is noticed that the chair has fallen completely (it has rotated 90 degrees) the counter increases by 1, when this counter is equal to the counter that counts the number of times the chair has fallen. This way it is possible to avoid any further increase of the counter. The counter that counts the number of times the chair is raised and is incremented only if the initial position of the chair is down and chairDown is greater. For designing an extra chair for Easter Egg , we used the same transformations to create the chair with reversed signs on the x and y axis values so that it is drawn exactly opposite the already existing chair. The statement set is contained in an if that checks if the chair has been tipped over 3 times.

No problem was encountered in this step, as the implementation steps are repeated from previous steps.

COMPUTER GRAPHICS

Thank you for your attention.