UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE INGENIERÍA

Final Project Documentation

MADE BY
Gómez Luna, Alejandro
316034946

PROFESSOR
ING. CARLOS ALDAIR ROMAN BALBUENA

SUBJECT
Laboratorio de Computación Gráfica

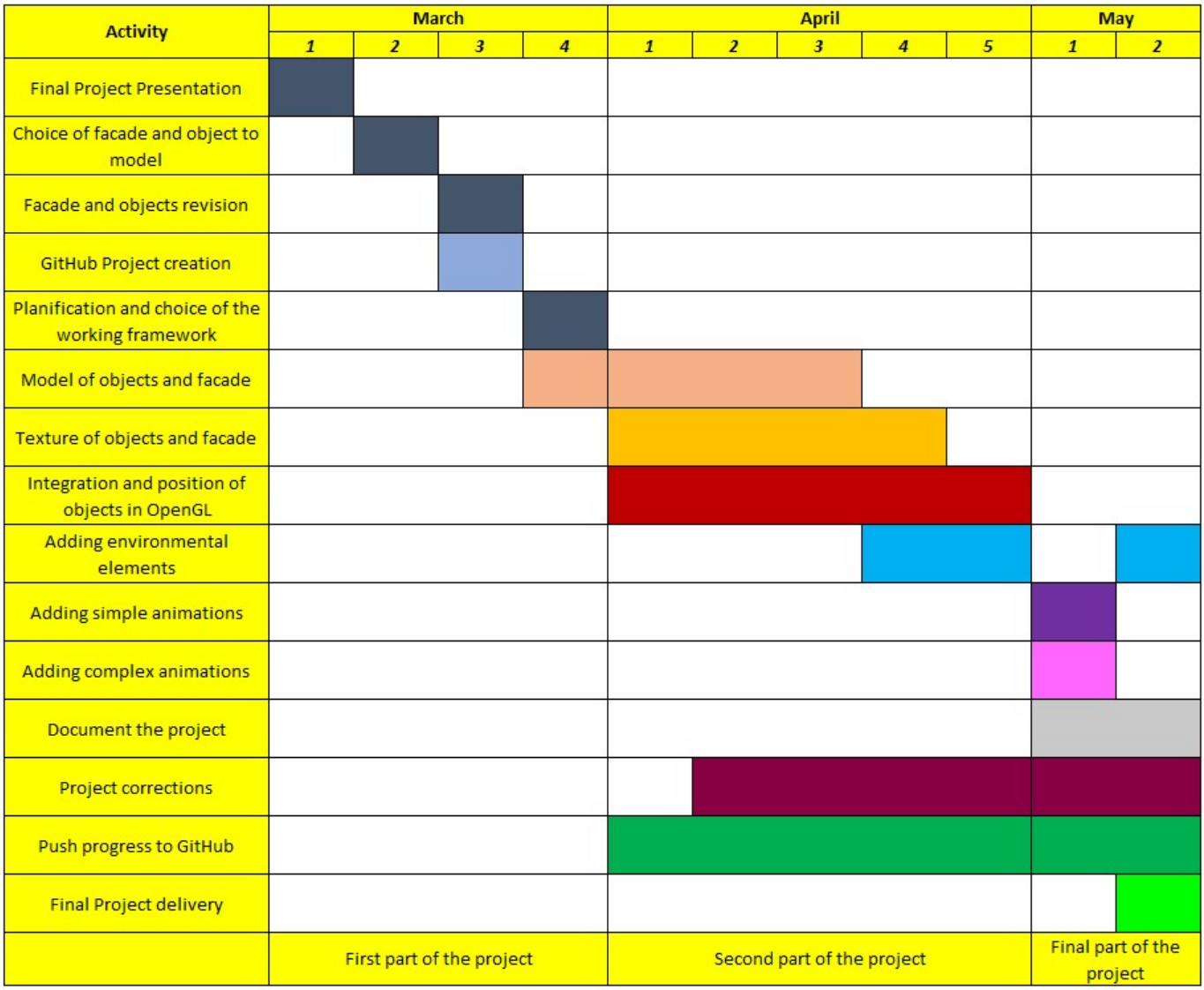
THEORY GROUP **No. 04**
LABORATORY GROUP **No. 04**

DEADLINE
11/Mayo/2022

Objectives

- The student must apply and prove all the knowledge acquired throughout the course.
- The student will use tools such as Maya and GIMP to model and texture all of his objects.
- The student will learn the basis of OpenGL, in addition with the utility and application of GLFW and GLEW.
- The student will analyze and comprehend the most important features of the graphic pipeline.

Gantt Diagram



The methodology used for the development of this project was based on Scrum, where three sprints were set, with the first one with a duration of 4 weeks, the second with a duration of 5 weeks, and the last with 2 weeks. For the first sprint the goal was to establish the basis and deadlines to do the project, also having the theoretical fundamentals to satisfy the requirements for the project.

Subsequently, with the first sprint completed, second sprint came, whose objective was to model all the proposed objects as a whole in addition with the facade, besides of texturing them correctly, to finally integrate them in the OpenGL environment, taking in account the project given by the professor. In this sprint there was also a search to create the proper environmental components for the virtual surroundings.
Finally the last sprint, the last part of the project, where the goal was to create the requested animations, adding the last environmental component of the sky box, document the project, and make general corrections.
For each Sprint there was a development and testing phase, where each change in the project was verified with the execution of the code in Visual Studio, therefore there was a constant feedback, allowing the code to be modified in a flexible way in view of any error that could arise.

Project Scope and Limitations

The recreation was made with the most resemblance of the facde, basement and seven objects in the home of Eren Jaeger of the series “Attack on Titan", following a Japanese animation style, where in this case try to imitate the reality, thus using textures that could be seen in a real environment.
The objects were modeled and textured using the software Maya LT 2020, also the images used for the textures were adapted using GIMP 2.10.30
In addition to OpenGL, the libraries GLEW and GLFW were used, simplifying the work of detecting inputs and managing extensions in OpenGL.
The entire project was made using the Microsoft Visual Studio 2022 IDE. In order to download Visual Studio 2022, you need to access the following link `https://visualstudio.microsoft.com/es/vs/` where Visual Studio Community 2022 must be downloaded. The file is saved somewhere on our computer, from where the entire installation process will take place. After it is installed we open it and select the Create New Project option, then Blank Project, we put the name of our project and its location, finally creating the project.
Together with the objects, environmental elements were added, including the following:

- Two point lights, which provided the major illumination source in the facade main room and in the basement. In conjunction with the illumination there was a general directional light that allowed all the objects to be illuminated in the environment.
- One sky box to represent the surroundings of the home.
- The use of transparencies in the house windows and the hanging lamp in the basement to accomplish a crystal effect in them.

Out of the scope of the project are:

- The use of collisions to physically delimit the rooms, objects, etc.
- The use of advanced techniques in graphic computation such as Ray Tracing or Aliasing.
- Complex textures, as the applied textures were obtained from an image, thus not having any associated map.
- The interactivity with a character that could move throughout the virtual environment.
- More immersive environmental elements, like sound or object physics.

Development

This project was based on the following functional requirements

- Correctly model and texture minimum 7 objects showed on a reference image, which must be approved by the professor.
- Correctly model and texture any other objects like windows and doors showed on the reference image, which must be approved by the professor.
- Correctly model and texture a facade showed on a reference image, which must be approved by the professor.
- Integrate all the objects and facade in OpenGL, where basic transformations are used to recreate the chosen space in third dimension.
  All of the integration will be based on a project and code given by the professor, which will be executed using the Visual Studio tool.
- The modeled objects must have a file extension .obj. These could be totally or partially modeled, indicating the source where there were retrieved to lately be modified.
- Using all the modeled and integrated object in OpenGL, create 5 animations, where 3 of them are simple and 2 are complex. The simple animations solely use basic transformations, like linear movements or rotations along one axis. The complex animations can be done using animation techniques such as KeyFrames or with more complex movements, such as parabolic shot, free fall, compound paths, paths based on trigonometrical functions, among others. All of the animations must have a context in the space where they are developed, meaning that they must have a logic according to their surroundings.
- Make the technical documentation of the project, along with a user manual where the user can know how to interact with the environment.
- Use Git and GitHub to upload all the progress done throughout the project.

According to the no functional requirements, which supply the project with more quality, there are

- Avoid the use of scaling in OpenGL, as this basic transformation consume plenty of computational resources during execution time.
- The use of environmental components that will accomplish a better recreation of the virtual space. These components include illumination, Sky box, Bumping maps, transparencies, among others.
  In illumination is the existence of directional, point and spot lights.
- Let the user have a fluid interaction with the recreated environment, allowing him to intuitively manipulate the camera, also easily play the animations.

Finally, about the code, the following glossary of variables and functions is presented

Variables Glossary

1. `WIDTH, HEIGHT, SCREEN_WIDTH Y SCREEN_HEIGHT`: Define the width and height of the window with which we are working.
2. `camera, lastX, lastY, firstMouse`: Used for camera mouse manipulation.
3. `keys`: This array of boolean variables store which key has been pressed.
4. `deltaTime y lastFrame`: Know the times between the current frame and the last one.
5. `anim_PP, regreso_PP y rot_PP`: These three variables are used for the simple animation of the main facade door. The anim_PP variable has the task of knowing if the animation is active or not, the two other variables purpose is detailed in the DoMovement() function.
6. `anim_PC, regreso_PC y rot_PC`: These three variables are used for the simple animation of the hallway door. The anim_PC variable has the task of knowing if the animation is active or not, the two other variables purpose is detailed in the DoMovement() function.
7. `anim_PS, regreso_PS y rot_PS`: These three variables are used for the simple animation of the basement door. The anim_PS variable has the task of knowing if the animation is active or not, the two other variables purpose is detailed in the DoMovement() function.
8. `anim_C, regreso_C, tras_caj1, tras_caj2, tras_caj3 y tras_caj4`: These six variables are used for the simple animation of the drawers of the desk in the basement. The anim_C variable has the task of knowing if the animation is active or not, the other variables purpose is detailed in the DoMovement() function.
9. `anim_L, espiral_L, regreso_C, posX, posY, posY2, posZ, rotX, rotX2, rotY, rotZ, theta`: These variables are used for the complex animation of the hanging lamp on the basement ceiling. The variable anim_L has the task of knowing if the animation is active or not, the other variables purpose is detailed in the DoMovement() function.
10. `anim_T, regreso_T, puntoX, puntoY, giroX, giroZ, desA, t`: These variables are used for the complex animation of the cup that is positioned over the desk in the basement. The variable anim_T has the task of knowing if the animation is active or not, the other variables purpose is detailed in the DoMovement() function.
11. `lightPos`: 3 position vector that define where the light originates from respect to the lighting objects that are created.
12. `pointLightPositions`: 3 position vector that define where the absolute positions to place the two point lights.
13. `Light1 y Light2`: Define the color of the light for the previously created point lights, where the first one will have an orange color, while the second will be totally a white light.
14. `vertices`: Array of positions and normals, with which the geometric figure of a cube is defined, which will be used to draw the point lights. In each row, the first three elements correspond to coordinates where the necessary vertices will be positioned to draw a face of the cube, while the following three elements define the normal of that cube face.
15. `skyboxVertices`: Defines the positions vertices position that conform the cube and will be used as the Sky box.

Functions glossary

1. `main()`: Main function of the project
   First is necessary to setup what is necessary to be able to work, so the window must be initialized and verify that it is created correctly. This window will have dimensions of 800 x 600 pixels.
   Once the window is loaded, it is setup to bind the key and mouse callback functions mouse, besides of disabling the pointer.
   Finally we define the viewport, which corresponds to the window dimensions.
   With the window loaded and setup correctly, the shaders are located and compiled, which in this case are the lightning, lamp and skybox, coming in pairs, since they are made up of a vertex and a fragment shader, where the information is first received in the vertex for processing it and later send it to the fragment shader.
   Then the defined models are loaded and compiled, specifying the relative path to where the .obj files are located. With the necessary files compiled and located the VBO and VAO are used, which are pointers used to load the previously defined vertices to draw a cube and the sky box, where the number of elements that correspond to positions must be specified. In the case of the sky box it is also required to specify the relative path of the faces that will be loaded for the skybox.
   Finally, the type of projection to be used is defined, which in this case is perspective, which keep the proportion of the objects in third dimensions, since the projection lines converge in leak points.
   Once all the initial configuration is done, a endless loop will be executed as long as the windows is not closed. In this loop, everything related to lights and model loading is defined. The lights and models will be affected by the lightning shader, which is used to achieve the desired lighting, which is being triggered by a directional light (environmental that is equal on all objects in the scene) and two point lights, with their respective illumination distances.
   With all the lighting in the scene, all the models are loaded, to whom basic transformations are applied first such as translations or rotations to place them correctly in the scene, according to the reference image. In some objects, translations or rotations are performed according to a variable, because they will perform an animation later on. After all the models have been loaded the lamp shader is used to draw the point lights that were used and the skybox shader to create the sky box. Finally, all the resources stored in memory are freed and the function ends.

2. `DoMovement()`: The is the function where all the animations and camera movement are done.
   In addition to camera movement, which is done with the keys W, S, A and D, there is all the animations.

   - The simple animation of the front door is triggered whenever the variable anim_PP is active.
     This animation consists of three parts. The first one is when the door rotates from 0 to 90 degrees about the Y axis. Once this trajectory is completed then the second part comes, which is the return, where the door goes from 90 degrees to -90 degrees on the same Y axis.
     After this is completed, there is the last part, where the door return to its initial position.
     All these rotations are controlled with the variable rot_PP, where the door rotations are stored. Likewise, the return_PP variable is used to know in which part of the animation the door is.
   - The simple animation of the door leading to the hallway is activated whenever the variable anim_PC is active. This animation consists of two parts. The first one is when the door rotates from 0 to -110 degrees about the Z axis. Once this boundary is reached, it returns to its initial position.
     All these rotations are controlled with the variable rot_PC, where the door rotations are stored. Likewise, the return_PC variable is used to know in which part of the animation the door is.

- The simple animation of the basement door is triggered whenever the anim_PS variable is active. This animation is the same as the one of the main door of the facade.
- The simple animation of the drawers of the desk in the basement is triggered whenever the variable anim_C is active. The drawers will be opened one by one, starting with the fourth bottom one until the top. first is reached. Once all of them have been opened, then they will be closed from the first at the top to the fourth at the bottom.
  The drawers movement are controlled with the variables tras_caj 1 to 4, where each variable stores the translation in Z of each one of the drawers.
  Likewise, the variable return_C is used to know in which part of the animation the drawers are.
- The complex animation of the hanging lamp on the basement ceiling is triggered whenever the variable anim_L is active. The lamp will describe three trajectories. The first one is a tilt of the lamp about its Z axis, where it will be moved to the point where the second trajectory begins, which is a spiral where the lamp will be moving along, to finally tilt again and return to its starting position.
  In the first trajectory, everything is taking account of the posZ variable, where the position in the Z axis of the ceiling lamp will be stored. This position ranges from 2 to 6 units, with increments of 0.05, having a total of 80 increments.
  With these increases in mind we have the two other variables augmentation, which are rotX and posY, where rotX is expected to reach 40 and posY 1, so $\frac{rotX}{80} = \frac{40}{80} = 0.5$, and $\frac{posY}{80} = \frac{1}{80} = 0.0125$
  With rotX the lamp is rotated about the X axis to give this tilt effect. The variable posY is used to modify the position of the lamp in Y, because at the moment of applying the rotation it creates a displacement in Y of approximately one unit, so this movement must be countered to improve the animation.
  For the second trajectory, it is an Archimedes spiral trajectory, which is described with the following equation in polar coordinates

$$r = a + b\theta$$

  Where a is equal to 0 and b is equal to 0.1, thus making $r = 0.1\theta$. Because the needed coordinates are rectangular the transformation equations are applied

$$x = rcos(\theta)$$
$$y = rsen(\theta)$$

  Replacing everything with θ

$$x = 0.1\theta cos(\theta)$$
$$y = 0.1\theta sen(\theta)$$

  For this case the X axis corresponds to the Z axis in OpenGL and the Y axis corresponds to the X axis in OpenGL. In the Z axis of OpenGL there is a displacement of 0.4 units, while in the X axis from OpenGL there is a displacement of 37.5 units.

With all these calculations done, the corresponding lamp can be moved to describe the ellipse trajectory, but at the same time it is necessary to rotate about a Y axis, which corresponds to a pivot that is outside the lamp and is not tilted, so another tilted lamp needed to be created with these necessary characteristics, and which is only visible when there is this part of the animation.
Theta variable is the one that corresponds to the parameter on which the position of Z and X of the lamp depends, so it will vary from -15.7 to 0.0, with an increase of 0.02. At the same time this change is made, it is required to rotate the lamp about the Y axis, which is done with the variable rotY. It needs to rotate 360 degrees every one cycle of the spiral, which happens every interval of 6.3 in theta, so if you have an increment of 0.02 $\frac{rotY}{\frac{6.3}{0.02}} = \frac{360}{315} = 1.1428$
Finally we have the last trajectory, in which the tilted lamp will return to the starting position. In this case, the rotation about the X axis is stored in the variable rotX2, the position in Z in posZ and the position in Y in posY2. As for the first route, there will be 80 increments, since rotX2 will be done every 0.5 units until it reaches 40.0, which is when the animation ends and all the variables return to their initial state. In this case posZ goes up to 6 because the tilted lamp has its pivot 4 units moved in Z. The correction of position on the Y axis is also done. In addition to all this movement, there is point light, for the spiral only follows a circular movement, but in the other two trajectories it does follow the same position in Z as the lamp.

- The complex animation of the cup over the desk in the basement is triggered whenever the variable anim_T is active. In this animation there are three trajectories. The first one is only a lineal changing of the cup on the X axis towards the edge of the desktop. The second trajectory is a parabolic shot defined by the following equations in X and Y

$$x = x_0 + v * t$$
$$y = H - \frac{1}{2} * g * t^2$$

In this case, the horizontal speed is 3.0 and the gravity is 0.96 were to have a larger offset in X and a lighter offset in Y. The starting position in X is 2.0 and the height is 4.8. With this solely replace and the corresponding offsets are obtained, which are stored in pointX and pointY, corresponding to the axes X and Y, respectively.
When the Y value is 0.4 then the base of the cup already touches the ground. At the same time the cup is rotated about the Z axis, which must be up to 90 degrees, to simulate that it is falling correctly. Making the corresponding variable isolations, it is known that at t = 3.027 the floor is reached, and Considering an increase of t of 0.05, there are $\frac{3.027}{0.05} = 60.54$ intervals, making the rotational increment of $\frac{90}{60.54} = 1.487$.
In the last part of the animation we need to rotate the cup about its X axis, to simulate that it rolls a little on the floor due to the accumulated kinetic energy, suffering a slight deceleration. This is done until the threshold of 180 degrees is reached, and that is when the entire animation is reset again.
The last trajectory is only to return the cup to its initial state and re-initialize all the used variables.

- `KeyCallback()`: This function is called whenever a key is pressed.
If the key corresponds to Esc, then the window is closed.
With the C key the state of the variable anim_PP is changed, activating or deactivating the animation of the main door.
With the V key, the state of the anim_PC variable is changed, activating or deactivating the animation animation of the hallway door.
With the B key the state of the variable anim_PS is changed, activating or deactivating the animation of the basement door.
With the N key the state of the variable anim_C is changed, activating or deactivating the animation of the drawers of the desk in the basement.

With the M key, the state of the variable anim_L is changed, activating or deactivating the animation of the lamp.
With the L key, the state of the anim_T variable is changed, activating or deactivating the animation of the cup.

- `MouseCallBack()`: This function is called whenever there is movement of the mouse cursor, which will change the orientation of the camera.

The results obtained are shown, where they are compared with the respective reference images.

Figura 1: Reference image for the facade

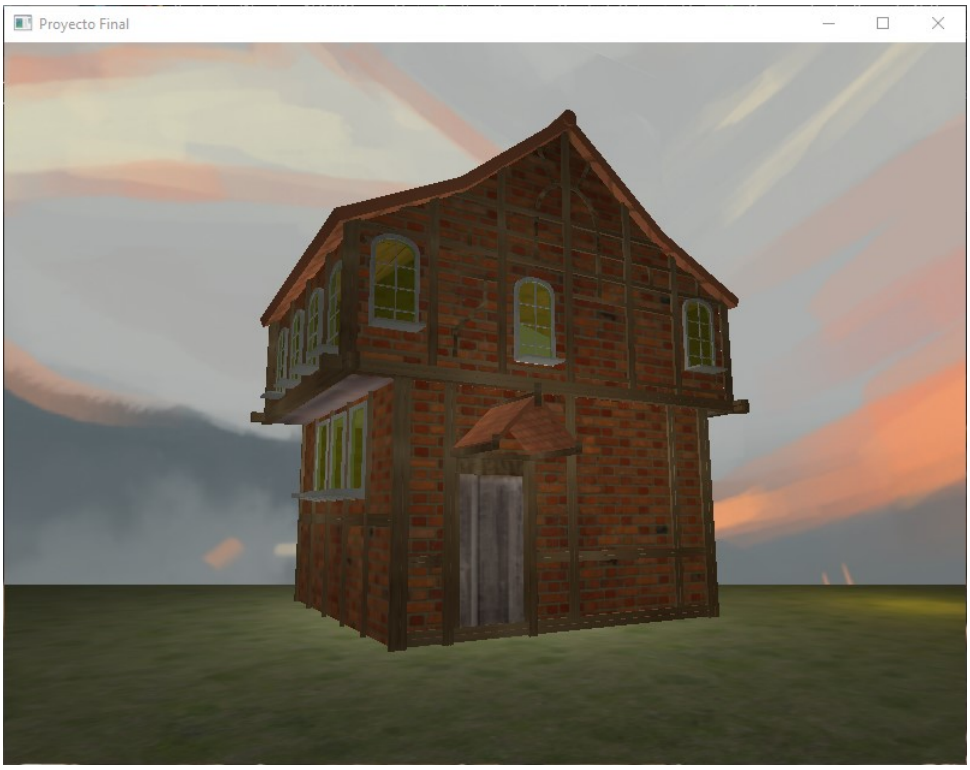


Figura 2: Facade recreated and integrated in OpenGL

Figura 3: Reference image for the hallway

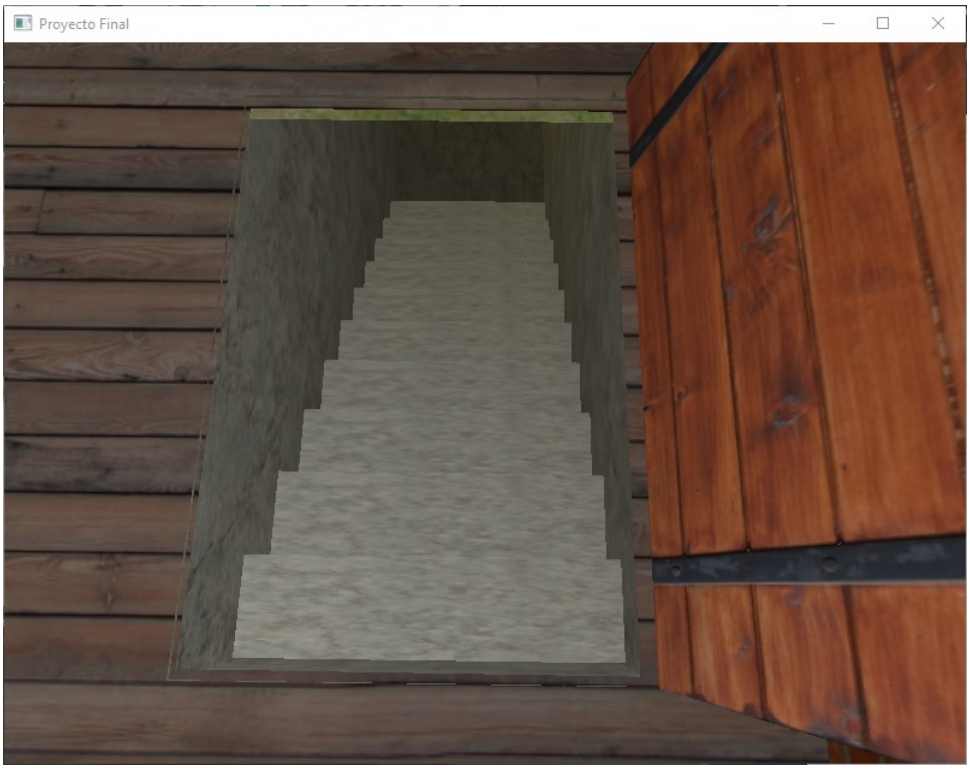


Figura 4: Hallway recreated and integrated in OpenGL

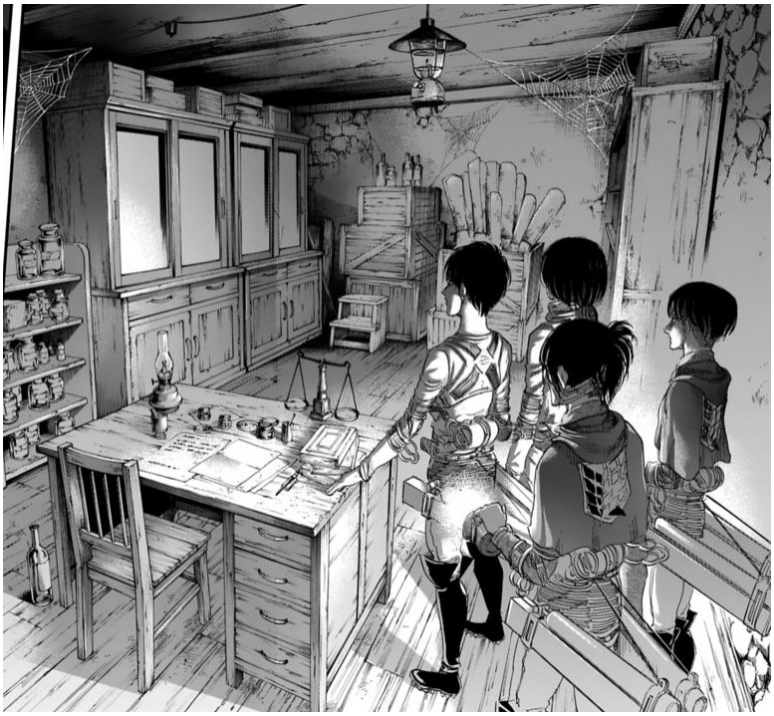


Figura 5: Reference image for the basement

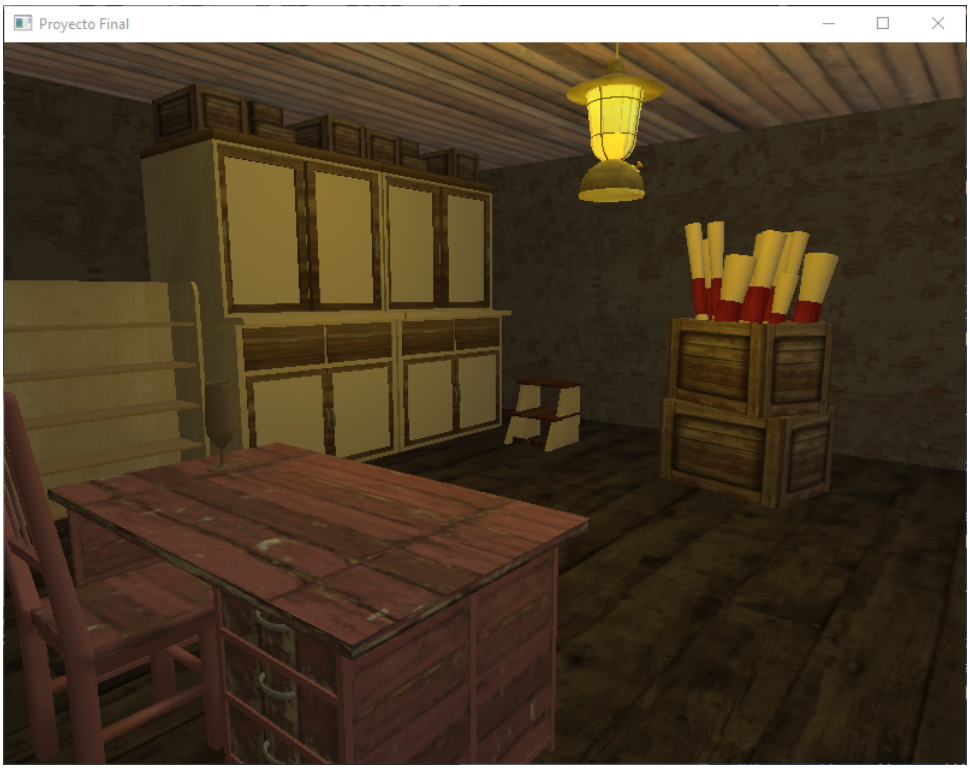


Figura 6: Basement recreated and integrated in OpenGL

Conclusions

For this project, all the proposed objectives were accomplished, since everything learned during the course was applied to the recreation of the proposed virtual environment, which, although it is not a completely reliable representation, does show a good degree of kinship respect to the reference image. Throughout this project, what was learned theoretically was applied, like the initial learning on how to model objects in OpenGL, solely defining all the geometry from vertices from which lines, loops, traingles, etc. were drawn. Although this was a directly way for modeling objects in OpenGL, it became an extremely complex and tedious task, so the Maya modeling software was used, which also makes use of basic transformations but allows the user to create all of his models in a much more intuitive way by having a graphical interface. Likewise, the texturing process of the models was done with the help of GIMP, to resize images that would later be used as textures, as well as to be able to create purely solid colors. Something important to remark is that whenever texturing was done, it was necessary to correctly orient the texture in Maya, occupying the projection on the corresponding axis.
After all the objects were modeled and textured, they were brought over to OpenGL. Although it was not a complicated task, it required understanding the code to know how to place the objects, translate them, rotate them, add lighting, transparency, etc. All this had already been reviewed previously in class, which made this whole process quite easy.
Finally, the simple and complex animations were made, where the complex ones required most of the time and understanding of mathematical equations, since the movements were based on trigonometric functions such as sine and cosine in the case of the spiral, or based on physical equations like the parabolic shot. Despite the difficulties aroused during the creation of these animations, the objects were animated correctly.

Without the completion of this final project, all the theoretical and practical knowledge would not have been enough to achieve a better understanding of computer graphics, because it is until everything learned is applied when a more significant learning is really achieved. While there are still many more advanced computer graphics topics with higher levels of complexity, being able to understand the basics of modeling, texturing, and lighting allows for a broad overview of how computer graphics works.
With everything learned during the course and this project, there is a better notion of the applications of computer graphics, which is currently highly bind to any technological advance, since it allows users to interact with a digital system in a much simpler and more fluid manner, as well as paving the way for the creation of increasingly realistic and detailed virtual environments.

References

3D Models

- The basement chair, customized from `https://www.turbosquid.com/3d-models/3d-set-16-medieval-wooden-model-1422733`, TurboSquid, made by leon017.
- The basement lamp, customized from `https://www.turbosquid.com/3d-models/obj-ceiling-lamp/532153`, TurboSquid, made by odio.
- The cup above the desk, obtained from `https://www.turbosquid.com/3d-models/3d-glass-champagne-model-1465324`, TurboSquid, made by Lazaro Bustos.

Textures

- Wood textures for the desk, obtained from `https://www.turbosquid.com/FullPreview/Index.cfm/ID/922096` and `https://www.turbosquid.com/FullPreview/Index.cfm/ID/922092`, TurboSquid, made by James Brody.
- Metallic texture used in every metallic handle of the furniture, obtained from `https://www.turbosquid.com/FullPreview/Index.cfm/ID/342167`, TurboSquid, made by flumpe.
- Whitened wood texture, obtained from `https://www.turbosquid.com/FullPreview/Index.cfm/ID/141341`,TurboSquid, made by Ace.
- Plank textures for the shelves, obtained from `https://www.turbosquid.com/FullPreview/Index.cfm/ID/509041`,TurboSquid, made by Ghostman56.
- Dark wood texture for the furniture and planks of the facade, obtained from `https://www.turbosquid.com/FullPreview/Index.cfm/ID/1143570`,TurboSquid, made by orange 3d.
- Wood crate texture, obtained from `https://www.turbosquid.com/FullPreview/Index.cfm/ID/350667`, TurboSquid, made by zaphad1.
- Dark metallic texture, obtained from `https://www.turbosquid.com/FullPreview/Index.cfm/ID/162363`, TurboSquid, made by alexkemmier.
- Red texture to roll up the scrolls, obtained from `https://www.texturepalace.com/red-color-textile-texture-free-download/`, TexturePalace, made by Sab.

- Concrete texture, obtained from https://www.turbosquid.com/FullPreview/Index.cfm/ID/595616, TurboSquid, made by Game Design Planet.
- Wood planks texture for the ceiling and floors of the rooms, obtained from https://www.turbosquid.com/FullPreview/Index.cfm/ID/555186, TurboSquid, made by ringhino.
- Wood planks texture for the basement door, obtained from https://www.turbosquid.com/FullPreview/Index.cfm/ID/232531, TurboSquid, made by bupaje.
- Wood texture for the hallway door, obtained from https://www.textures.com/download/WindowsShutters0187/54583, Textures.
- Wood texture for the basement floor, obtained from https://www.textures.com/download/WoodPlanksOld0244/121294, Textures.
- Roof tiles texture for the facade, obtained from https://www.vecteezy.com/vector-art/171678-roof-tile-background, Vecteezy, made by foxarthappy.
- Bricks texture for the facade, obtained from https://www.turbosquid.com/FullPreview/Index.cfm/ID/243350, TurboSquid, rdone by InfinityStudio.
- Wood texture for the main door of the facade, obtained from https://www.textures.com/download/WoodPlanksBare0354/110860, Textures.
- Grass texture for the surroundings of the facade, obtained from https://www.textures.com/download/Grass0169/51506, Textures.
- Bricks texture for the basement, obtained from https://www.textures.com/download/BrickSmallPlaster0119/134802, Textures.
- Sky Box texture, obtained from https://www.cleanpng.com/png-heat-sky-plc-skybox-3444567/download-png.html, Cleanpng, made by Alishu.