Universidad Nacional Autónoma de México

Facultad de ingeniería

Final Project
Technical Manual

Computer Graphics and Human-Computer Interaction

Group 5 Theory Semester: 2026-1

By:

320010941

Objective:

Apply and demonstrate the knowledge learned during the course by recreating a 3D facade and its interior spaces in OpenGL

- **Recreate:** A facade and two interior rooms. Each room must be furnished with several objects based on the reference image.
- **Interactivity and Navigation:** Implement a synthetic camera that allows the user to move freely through the stage, simulating a first-person virtual tour.

Dynamism and Animation: Integrate 5 animations generated by code: 3 simple and 2 complex.

These animations must be consistent with the context of the stage and not limited to basic transformations.

Technologies Used

Visual Studio Community:

A free and complete IDE from Microsoft. It allows individual developers and students to create apps for Android, iOS, Windows, and the web. It offers robust tools and is compatible with C++

OpenGL

A multi-platform and open-source API used to render 2D and 3D graphics. It interacts with the graphics hardware (GPU) to create complex visualizations. It is included in the graphics card drivers.

Paint

Basic graphic editing app included in Windows. Used to create and edit simple images, color, resize, draw, cut, and adding text.

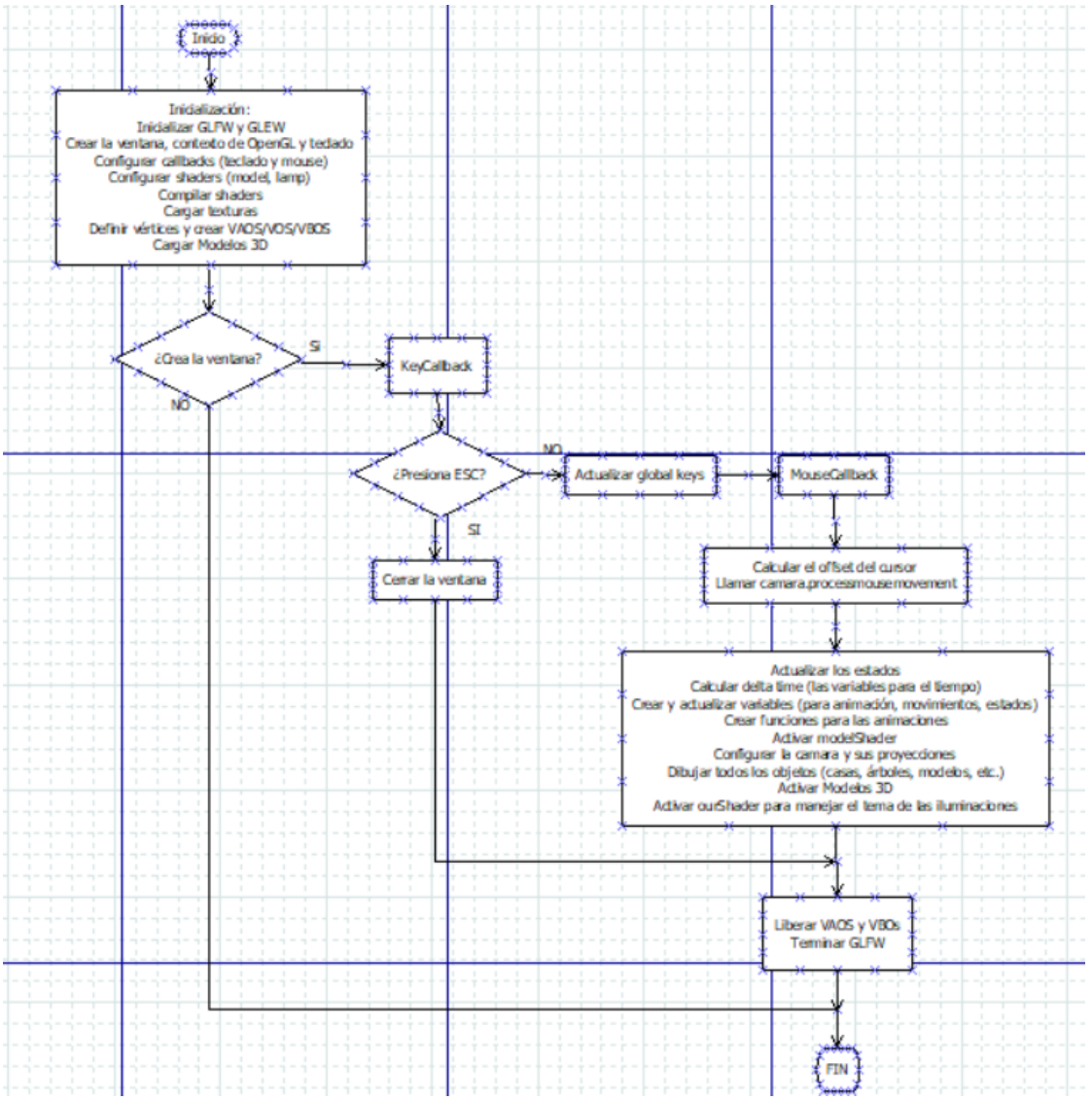
Gimp

Free and open-source image editor. Ideal for handling the images that will be the textures.

Autodesk Maya

Professional 3D software. It is mentioned here just as a reference of what is *not* used as the main tool, but it is a standard in the industry.

Software Flowchart



Gantt Chart

Actividad	Inicio	Final	10/10/2025	11/10/2025	12/10/2025	13/10/2025	14/10/2025	15/10/2025	16/10/2025	17/10/2025	18/10/2025	19/10/2025	20/10/2025	21/10/2025	22/10/2025	23/10/2025	24/10/2025
Investigación	10/10/2025	15/10/2025															
Planificación	10/10/2025	16/10/2025															
Codificación	10/10/2025	14/11/2025															
Documentación	30/10/2025	06/11/2025															

Project Scope

To start, I think it is very important to mention that I will not use specialized 3D modeling software (like 3ds Max, Blender, Maya, etc.). This is because the spaces for the project were requested in the OpenGL environment using the C++ code we used throughout the semester. Any 3D object imported from these apps will be counted as "extra".

Defining the limits of the Project about we are going to do and what we do not, and taking mind the sources we need to this project:

- Computer for all the project (my laptop in this case)
- Dedicated Software: Visual Studio Community, OpenGL.
- Excel for diagrams.
- Work team: 1 member (just me).

The project will recreate the scenario of Pallet Town, specific from the remakes: Pokémon FireRed and LeafGreen, within Red's house and Professor Oak's laboratory.

Reference Images:

Pallet town:

Reds's living and room:

Scenario: Pallet Town with 2 houses, the in-house (living room and bedroom) of Red's house (the protagonist), and Oak's lab

Modeling: Modeling and texturing of 7 different objects located inside the houses.

Navigation: A synthetic camera that allows the user to move freely

Animations: Integration of 5 code-generated animations (3 simple, 2 complex). (The Theory document asks for 4, but the Lab one is more specific with 5, so I will use that number)

Deliverables:

- Source code in C++.
- Functional project in a GitHub repository
- Executable file of the scenario.
- Documentation (Technical and User manual)

Out of Scope:

Como ya se mencionó, no usaremos ninguna aplicación dedicada al modelado 3D como blender o maya, y si se usan elementos diseñados de estas aplicaciones, serán contadas como un extra:

- **Physics:** No physics engine. The user can pass through objects (no collision detection).
- **Advanced Interactivity:** Interaction is limited to navigation. The user cannot pick up or move objects.
- **Sound:** No sound effects or music. I tried to use the irrklang library, but now they charge for it (it used to be for non-commercial use).

Project Costs.

Taking in mind the requirements in base scopes, time, dedication, as my work and knowledge that I took for this.

1. Production Cost:

To not undervaluing my labor, I am taking into account the cost of production, working hours, and the software used.

Formula: Production Cost = (Work Hours × Cost Rate/Hour) + Indirect Costs.

- **Time dedicated:**
    - Total estimate of hours dedicated to the project
        - Research and Planning: 1 week (10/10/2025 to 16/10/2025)
        - 3D Modeling and Texturing: 2 months (16/10/2025 to 25/10/2025)
        - Coding: 2 months (16/10/2025 to 25/10/2025)
        - Documentation: 5 days
        - **Total hours:** 150 hours
    - Cost Rate: $150 MXN / hour (Junior developer rate).
- **Indirect Costs:**
        - (Electricity, Internet): $700 MXN

Calculation Costs:

- Labor: 150 hours * $150 = $22,500 MXN
- Indirect: $700 MXN
- Total Production Cost: $23,200 MXN

I am valuing my production cost at MXN $23,200. This amount reflects the 150 hours dedicated to the project in all its aspects, from the initial research on the scenario to be recreated to the development, including coding, research, and use of resources, libraries, code, etc. I have established a cost rate of MXN $150/hour, covering the time of a junior developer. Additionally, it includes MXN $700 in indirect service costs.

2. Sale Prce

To sell my Project, I am contemplating 2 methods due the last analyze:

Cost-Plus

Take the established cost plus a profit margin

- Production Cost: $ 23,200 MXN
- Market Rate of 40%: $23,200 * 0.40 = $9280 MXN
- **Sale price: $32,480 MXN**

Market Rate

Considering my knowledge and experience in the field, as well as the complete management of the project throughout its execution, the hourly rate is higher.

- Total of Hours: 150 hours
- Professional Service Rate ($350 MXN / hour):
- **Sale Price: 150 * $350 = $52,250 MXN**

This is a competitive price for an app 3D development due the requested features.

Limitations

The only limitation was not using Blender or Maya. This makes the graphic quality not super realistic, but it feels more like the original Pokémon games.

Software Methodology Applied

For this project, I see the **Waterfall Methodology** as the most viable option. Why? Because the requirements are fixed and defined from the beginning (the specifications in Classroom). They won't change. Also, it is an individual project, so Scrum roles are unnecessary (and this way I have more freedom to define what part of my mind does each thing).

Waterfall Methodology

A linear and sequential model where each phase must be completed before the next can begin. This model is perfectly suited to a project like this because the requirements are fixed, allowing the process to be described in an orderly manner.

Development phases due methodology:

1. **Analyze and requeriment phase:**
    - A virtual enviroment in OpenGL, 7 minimum, and animations, 5 animations (3 simple, 2 comples), and the full documentation.
    - The theme for the facade and rooms (video game theme) was selected, and reference images were presented for approval.
2. **Design phase:**
    - Planned the software structure and base code. Choosing the code base provided. Defined lighting and camera. Created Flowcharts and Gantt charts.
3. **Implementation (Coding):**
    - The 3D modeling and texturing of the 7 objects was carried out.
    - The code was writting on C++ y OpenGL to:
        - Render the scene and objects.
        - Implement the synthetic camera for the virtual tour.
        - Add the animations
4. **Testing: (Verification):**
    - The .exe were tested to ensure work correctly.
    - The requirements was verified that were met (such as texturing and the number of objects)
5. **Maintenance (Delivery):**
    - Compiled the final version of the .exe.
    - Se redactaron los manuales de usuario y técnico.
    - The project was uploaded to the GitHub repository, ensuring the use of dynamic paths.

Code Documentation

There are some changes I made to the code compared to the semester practices:

For class Camera:

- Implements an FPS (First Person Shooter) camera using Euler Angles.
- Yaw: Rotation on Y axis.
- Pitch: Rotation on X axis.
- Vectors (Front, Right, Up) are recalculated using basic trigonometry so movement is relative to where the player looks (making WASD controls friendly)

Day/Night Changes:

- Uses Linear Interpolation (Lerp) to transition smoothly between colors.
- float transitionSpeed = 0.5f; controls how fast it gets dark.

For textures:

- GL_NEAREST: "Pixel Art" filtered. When zoomed in, the pixels appear hard instead of blurred.
- Two different textures (water.png and water2.png) are loaded into different OpenGL IDs.
- They are not mixed in the shader, but will alternate in the main loop to create a "flipbook" or frame animation effect.

For illumination:

- Point Lights (which are the light poles next to the houses): They have a specific position and dimming capability.
- Linear and Quadratic: Coefficients that cause light intensity to decrease with distance (real physics).
- Equation: Intensity = 1.0 / (Constant + Linear*Dist + Quadratic*Dist^2)

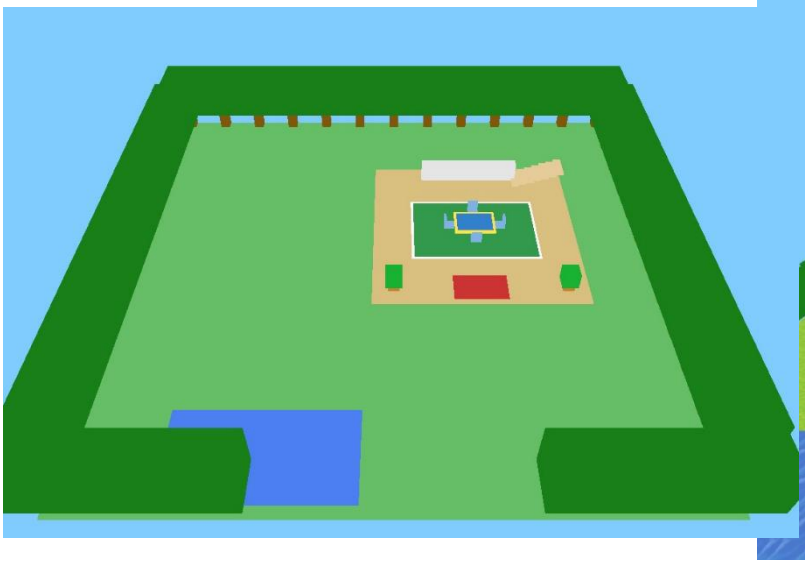
For animation:

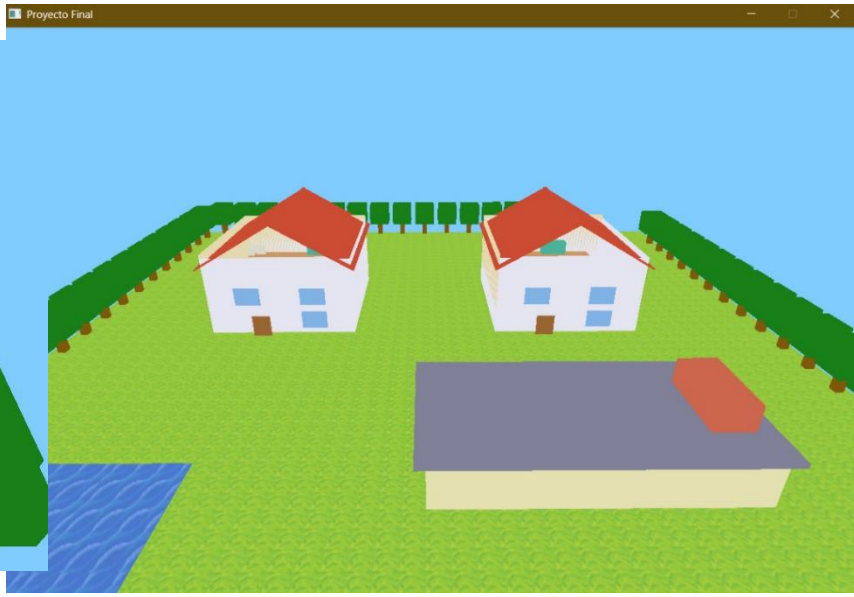
- Target Detection: `glm::distance` checks if the Pokémon is close to its target. If it is (< 2.0f), a new random coordinate (rand) is chosen within a defined range.
- Curved Trajectory Calculation (Simple Steering Behavior): glm::cross(direction, UP)` : Obtains a vector perpendicular to the flight direction (for more "natural" movement for Ho-Oh and Mew, since Mew is known for roaming in Pokémon lore and also hides in the games).
- sin(time)` : Generates an oscillating value (-1 to 1).
- Position += Direction * Speed * DeltaTime` : Ensures that it flies at the same speed in a fast or slow PC.

First code versión, it Will be not put all the code versions for document optimization

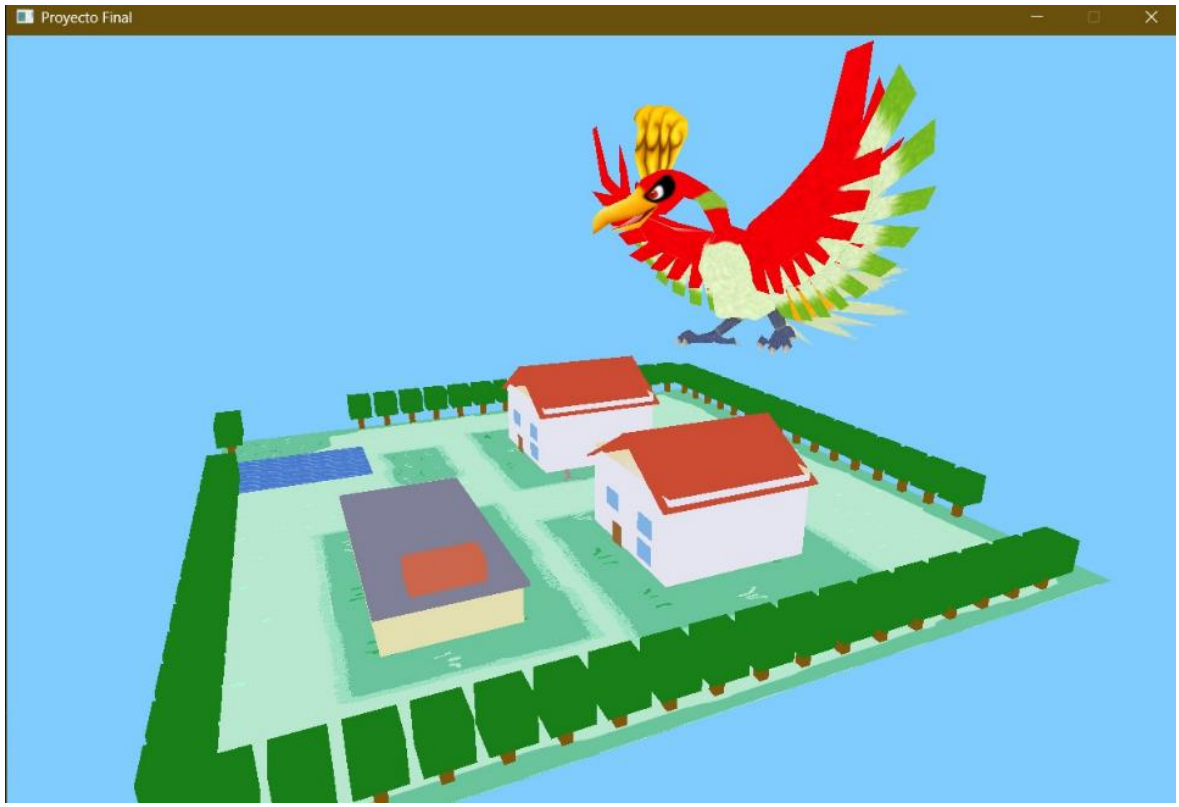
```
hola1                                                    (Ámbito global)
273             glfwSetWindowShouldClose(window, true);
274         if (key >= 0 && key < 1024) {
275             if (action == GLFW_PRESS)
276                 keys[key] = true;
277             else if (action == GLFW_RELEASE)
278                 keys[key] = false;
279         }
280     }
281
282     // Callback para el mouse
283     void MouseCallback(GLFWwindow* window, double xPos, double yPos) {
284         if (firstMouse) {
285             lastX = xPos;
286             lastY = yPos;
287             firstMouse = false;
288         }
289
290         GLfloat xOffset = xPos - lastX;
291         GLfloat yOffset = lastY - yPos;
292
293         lastX = xPos;
294         lastY = yPos;
295
296         camera.ProcessMouseMovement(xOffset, yOffset);
297     }
```

Visual Process:





```
808             model = houseBaseModel;
809             model = glm::translate(model, glm::vec3(blue_X, blue_Y, blue_Z));
810             model = glm::scale(model, glm::vec3(blue_W - (frameThick * 2), blue_H, blue_D - (frameThick * 2)));
811             glUniform3fv(uniformColor, 1, glm::value_ptr(tableTopColor));
812             glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
813             glDrawArrays(GL_TRIANGLES, 0, 36);
814
815             // 2b. Marco Negro (4 piezas, plano)
816             // Arriba (Z+)
817             model = houseBaseModel;
818             model = glm::translate(model, glm::vec3(blue_X, frameY_Blue, blue_Z + (blue_D / 2.0f) - (frameThick / 2.0f)));
819             model = glm::scale(model, glm::vec3(blue_W, blue_H, frameThick));
820             glUniform3fv(uniformColor, 1, glm::value_ptr(blackColor));
821             glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
822             glDrawArrays(GL_TRIANGLES, 0, 36);
823             // Abajo (Z-)
824             model = houseBaseModel;
825             model = glm::translate(model, glm::vec3(blue_X, frameY_Blue, blue_Z - (blue_D / 2.0f) + (frameThick / 2.0f)));
826             model = glm::scale(model, glm::vec3(blue_W, blue_H, frameThick));
827             glUniform3fv(uniformColor, 1, glm::value_ptr(blackColor));
828             glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
829             glDrawArrays(GL_TRIANGLES, 0, 36);
830             // Izquierda (X-)
831             model = houseBaseModel;
832             model = glm::translate(model, glm::vec3(blue_X - (blue_W / 2.0f) + (frameThick / 2.0f), frameY_Blue, blue_Z));
833             model = glm::scale(model, glm::vec3(frameThick, blue_H, blue_D - (frameThick * 2)));
834             glUniform3fv(uniformColor, 1, glm::value_ptr(blackColor));
```

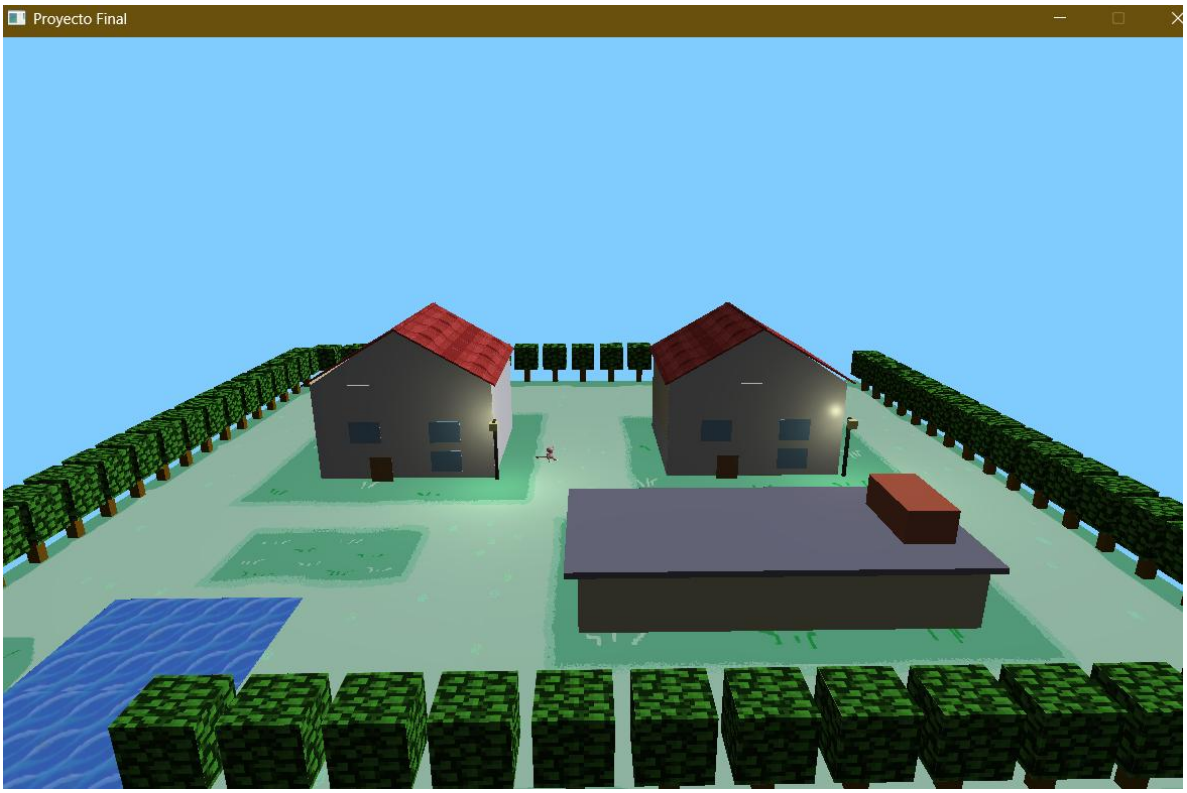


Final version (2351 code lines):

Model.h ProyectoFinal.cpp

hola1 (Ámbito global) main()

```
290     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
291     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
292     // Configurar filtrado (pixelado)
293     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
294     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
295
296     // Cargar la imagen
297     unsigned char* agua2 = stbi_load("images/agua2.png", &textureWidth, &textureHeight, &nrChannels, 0); // <- CAMBIO AQUÍ
298
299     if (agua2) // <- CAMBIO AQUÍ
300     {
301         // Detectar formato
302         GLenum format;
303         if (nrChannels == 3)
304             format = GL_RGB;
305         else if (nrChannels == 4)
306             format = GL_RGBA;
307         else
308             format = GL_RGB;
309
310         glTexImage2D(GL_TEXTURE_2D, 0, format, textureWidth, textureHeight, 0, format, GL_UNSIGNED_BYTE, agua2); // <- CAMBIO AQUÍ
311         glGenerateMipmap(GL_TEXTURE_2D);
312     }
313     else
314     {
315         std::cout << "Failed to load texture: images/agua2.png" << std::endl; // <- CAMBIO AQUÍ
```



Conclusions

Doing this project was a complete journey that went beyond the initial objective of "recreating a stage". The true learning didn't come from just building shapes, but from overcoming software architecture challenges to go from a simple static render to a dynamic, unified, and illuminated 3D scene.

It was a full simulation of the development cycle in computer graphics: from planning, re-architecting the software (handling files, codes, shaders), debugging at a low level, and finally implementing dynamic systems. The result is tangible proof of deep learning, from how to draw a cube to orchestrating a coherent world.

I also want to say as a "final thank you" that I liked this subject a lot. It is the only one where you have almost complete creative control. It is very interactive and visual, which fosters better learning for people with these learning methods. I really enjoyed the whole process from the practices to the culmination of the project

References

- Kessenich, J., Sellers, G., & Shreiner, D. (2016). *OpenGL programming guide: The official guide to learning OpenGL, version 4.5 with SPIR-V* (9ª ed.). Addison-Wesley Professional.
- Foley, J. D., van Dam, A., Feiner, S. K., & Hughes, J. F. (1995). *Computer graphics: Principles and practice* (2ª ed. en C). Addison-Wesley.
- Lelis, B. (2020). *Learn OpenGL: Learn modern OpenGL graphics programming in a step-by-step, example-oriented way*. Packt Publishing.
- Lelis, B. (s.f.). *Learn OpenGL*. Recuperado el 15 de noviembre de 2025, de https://learnopengl.com
- Khronos Group. (s.f.). *OpenGL registry*. Recuperado el 15 de noviembre de 2025, de https://registry.khronos.org/OpenGL/
- Royce, W. W. (1970). Managing the development of large software systems. *Proceedings of IEEE WESCON 26*, 328-388.
- Tutorials Point. (s.f.). *SDLC - Waterfall model*. Recuperado el 15 de noviembre de 2025, de https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm