UNIVERSIDAD NACIONAL
AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

COMPUTACIÓN GRÁFICA E INTERACCIÓN HUMANO-COMPUTADORA

GROUP: 5

TECHNICAL MANUAL

STUDENT ID: 319162538

PROFFESOR: CARLOS ALDAIR ROMAN BALBUENA

SEMESTER: 2026 - 1

Index

Introduction

The goal of this project is to develop a virtual walkthrough of the Teen Titans Tower using graphical programming techniques in OpenGL. Throughout the process, a fully explorable 3D environment was built, integrating models, textures, lighting, and animations to faithfully recreate the interior space shown in the series. To maintain visual consistency, reference images were used to guide the scene's composition and the placement of objects.

The creation of the environment involved a complete workflow, ranging from building models based on modular structures to integrating materials, transformations, and textures. Several objects were included, such as furniture, lamps, appliances, and decorative elements, as well as programmed animations that add movement and make the exploration more engaging. The first-person camera and free user control allow the environment to be explored interactively and in real time.

This project serves as an opportunity to apply the concepts learned in the course in a practical way, covering topics such as rendering, buffer handling, lighting, 3D transformations, and scene control. It also helps strengthen skills related to mathematical logic, code organization, and technical documentation. The final result demonstrates the ability to combine creativity and programming to build a functional virtual environment, using modern development tools and an approach focused on creating interactive visual experiences.

Current State of the Art of the Project

Virtual walkthroughs have evolved from simple static visualizations into fully immersive real-time experiences, driven by advances in 3D graphics, rendering engines, and specialized hardware. Originally, these types of walkthroughs were mainly used in architecture to present digital models before construction; however, their applications have now expanded to fields such as entertainment, education, museums, training simulations, video games, and virtual reality.

Current trends focus on creating interactive environments that allow users to explore spaces freely, perceive depth, experience dynamic lighting, and observe ambient animations, resulting in a much more realistic experience. Technologies such as real-time rendering, physically-based lighting (PBR), dynamic shadows, FPS-style navigation, and modular modeling have become standard in professional visualization projects. Likewise, frameworks and engines like Unreal Engine, Unity, and OpenGL have made it easier to develop specialized virtual walkthroughs, offering efficient tools for both artistic and technical implementations.

In this context, the presented project—an interactive recreation of the interior of the Teen Titans Tower—is based on these principles, applying modern techniques in rendering, animation, and camera control to simulate a realistic walkthrough inside a fictional environment. The use of modular modeling with textured cubes, directional and point lighting, custom shaders, and real-time animations places this work within the same development line used in contemporary prototypes of video games and virtual experiences. Additionally, incorporating reference images helps maintain visual consistency and aesthetic coherence with the original material, following professional practices in digital design.

This project not only aims to visually reconstruct the space, but also to apply current concepts from the state of the art in 3D simulation, integrating interactivity, free navigation, and animated elements that enrich the user's experience. In doing so, it aligns with modern standards of virtual walkthroughs and demonstrates how these technologies can be used to recreate both real and fictional spaces with a modern, technical, and functional approach.

Objectives

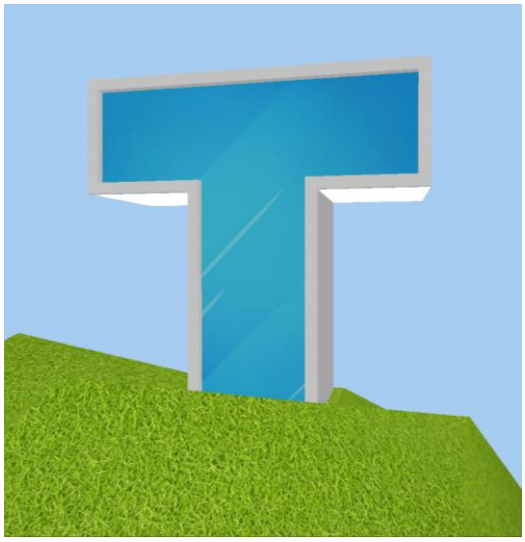
- Apply the knowledge learned in the course to develop a 3D recreation of the façade and two interior rooms of the Teen Titans Go Tower, using OpenGL to build the scenes. The virtual representation should be as faithful as possible to the reference images, including the atmosphere and details of each space.
- Integrate an external modeling workflow using Blender together with the rendering process in OpenGL, modeling a minimum of 10 objects, with 5 objects for each room created between OpenGL and Blender.
- Implement a set of coherent, code-generated animations to bring the scene to life, developing the 4 required animations plus 1 additional one.

References Images

Below are the visual references that served as the main guide to ensure that the environment and architectural structure were recreated as faithfully as possible to the original series.

Facade Reference

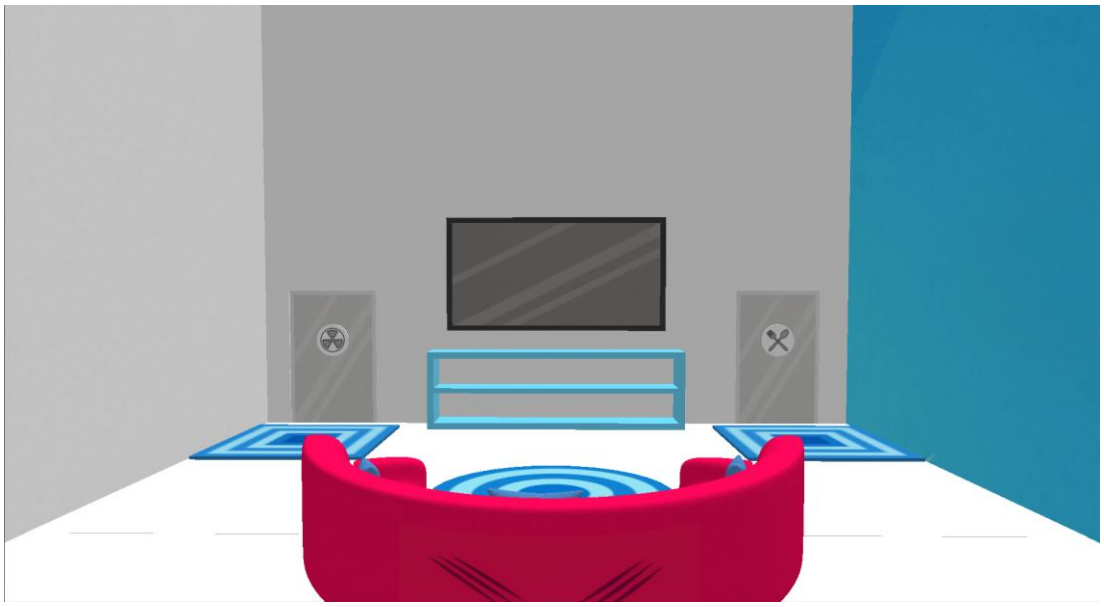
The exterior of the project focuses on the characteristic "T"-shaped structure of the building, located on top of a hill. This image served as the main basis for modeling the external architecture and the surrounding environment.



References for the Rooms to Recreate

For the interior design, the scope of the project was limited to two main rooms, recreating the layout of the furniture and the specific lighting of each area.

Living room



Recreated objects:

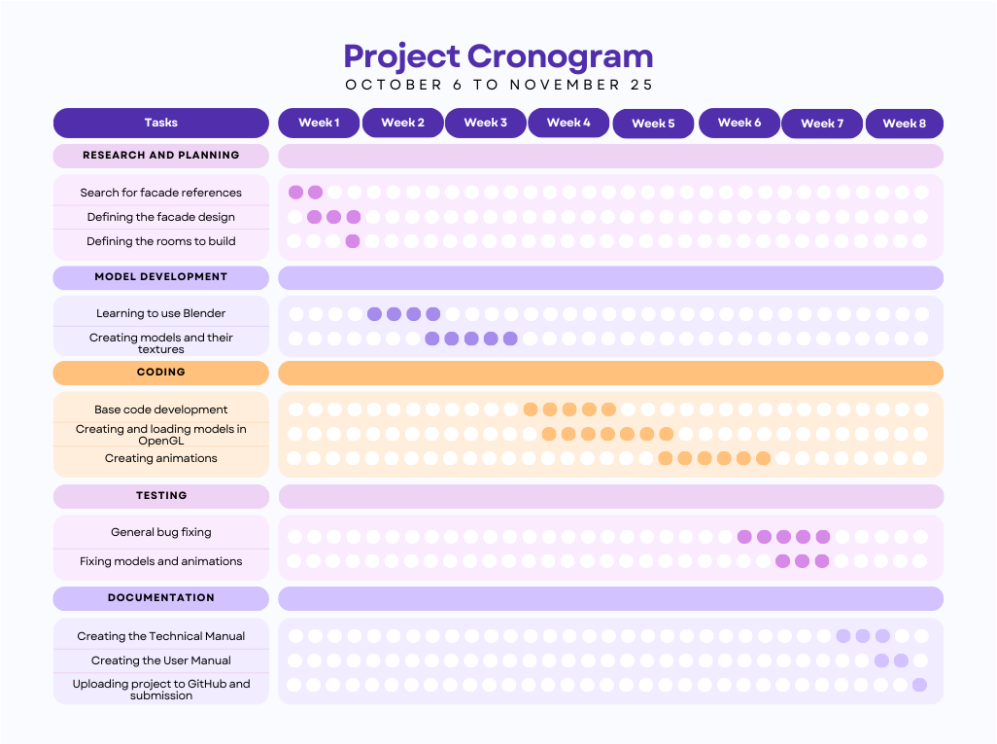
1. **TV table**
2. **Television**
3. **Carpets**
4. **Pillows**
5. **Sofa**

Dining room or kitchen

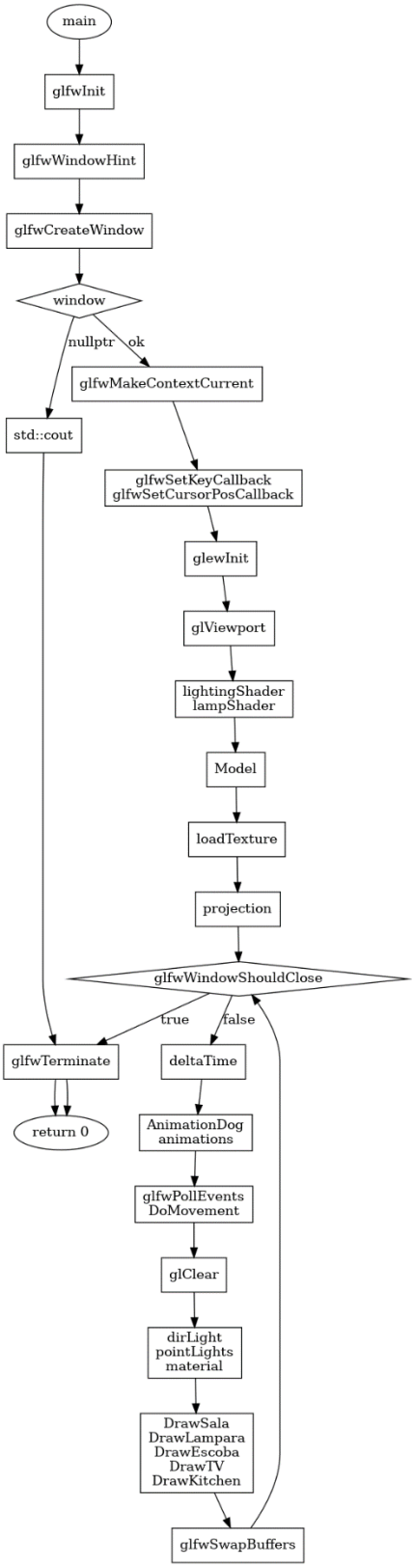
Recreated Items:

1. **Seats**
2. **Table**
3. **Stove**
4. **Refrigerator**
5. **Kitchen cabinet**

Gantt Chart



Software Flowchart



Project Scope

The scope of this project is the development of a functional graphic application in OpenGL that recreates a 3D environment based on the Teen Titans Go series, limited to the following:

1. **Development of the Environment:** The creation of a 3D scene that faithfully represents the exterior façade and two interior rooms of the Tower of the Teen Titans, based on reference images.

2. **Modeling and Texturing**: The creation and integration of at least 10 unique and evaluable 3D objects modeled in OpenGL, along with Blender and appropriately textured.

3. **Implementation of Animations:** The programming of 4 code-generated animations that are coherent with the environment.

4. **Functional Software:** The delivery of a functional executable file built on top of the codebase provided by the teacher.

5. **Camera and view:** Camera movement was implemented seamlessly and in real-time, offering freedom to explore the space from different angles and perspectives, improving navigation and user interaction with the tour.

Technologies used

In the development of the project, various technologies and tools were used to create an interactive virtual tour with 3D models, animations and a coherent visual environment. Below is a description of each one and its function within the production process:

OpenGL (Open Graphics Library)

OpenGL was the main technology of the project, used to implement real-time rendering, buffer handling, camera, dynamic lighting, and shader processing. Its low-level approach allowed for fine-grained control over the graphics pipeline, which was essential for the creation of animations, light effects, and three-dimensional structures.

Modeling and texturing in blender

Blender was used to create the visual references and practice modeling the interior structure. Although the final project uses modularly built models with OpenGL cubes, Blender functioned as a support tool for understanding proportions, volumes, and spatial composition based on videos and tutorials. It does count as a technical tool of the project.

ChatGPT (Technical Support, Documentation, and Texture Generation)

ChatGPT was used as an auxiliary tool to:

- Tips on shader correction
- Lighting debugging
- Optimizing Functions and Animations
- Generating or editing simple textures
- OpenGL Concepts Explained

Its function is part of modern development support tools, similar to an intelligent IDE or technical assistant.

Reference material (YouTube images and videos)

Using reference videos and images to model the Tower of the Teen Titans is standard practice in 3D and animation environments. It is not a "technology", but it is a methodological tool that supports the visual coherence of the project.

Limiting

Despite the achievements made, the implementation of this faced certain limitations which affected to some extent the development until the end:

1. **Technical resources:** The processing power and memory available in the workstations conditioned the complexity of the textures and the number of dynamic lights that could be implemented without affecting real-time performance.

2. **Tools and environment:** While Blender and OpenGL offered extensive possibilities, some advanced features like custom shaders or complex visual effects weren't fully explored due to the learning curve and time available.

3. **Animation Limitations:** All implemented animations must be generated exclusively by code. The use of pre-rendered animations or animations imported from Blender is not allowed.

4. **Scalability:** The project was limited and adjusted to the requirements established in the evaluation, so adding or implementing more concepts caused errors or errors in programming and a lot of graphic demand for the computer.

Applied software methodology

For the management of this project, the use of the incremental development methodology was chosen, involving initial planning of a waterfall model and flexibility of construction in stages. Among the stages we have the following:

- **Phase 1: Research and Planning (Weeks 1-2)** This initial phase defined the scope and conceptual foundations of the project. The tasks included the search for visual references, the definition of the façade of the Tower of the Teen Titans and the selection of the two rooms to be recreated.

- **Phase 2: Model Development (Weeks 2-4)** This corresponds to the first tangible increment of the project: the creation of the 3D assets. This stage

focused on learning Blender and making and texturing all the models required for the scene.

- **Phase 3: Coding (Weeks 4-6)** In this phase, the "Models" increment was taken and brought to life in the graphical environment. It included the development on the codebase, the implementation of the logic for loading the models into OpenGL and the creation of the programmed animations.

- **Phase 4: Testing (Weeks 6-7)** This phase was dedicated to refining and debugging the "Coding" increment. He focused on fixing general bugs in the app, as well as visual and functional adjustments to models and animations.

- **Phase 5: Documentation and Delivery (Weeks 7-8)** The final increment consisted of packaging the product for delivery. It included the creation of the technical and user manuals, and the final preparation of the GitHub repository for project delivery.

This incremental model ensured an orderly workflow, where each stage (Modeling, Code, Testing) was built on the success of the previous one, making it easier to detect problems and ensuring that all components were ready for the delivery date.

Dictionary of functions and variables

Main functions

int main()

Parameters: None.

Description: Entry point of the program. Initialize GLFW/GLEW, create the window, set up the camera and lights, load models and textures, define animations, and run the main render loop until the window closes.

void MouseCallback(GLFWwindow* window, double xPos, double yPos)

Parameters:

- window (GLFWwindow*): A window associated with the event.
- xPos (double): X position of the mouse.
- yPos (double): Y position of the mouse.

Description:

Update the camera orientation using the mouse movement (yaw/pitch), using lastX, lastY and firstMouse.

void KeyCallback(GLFWwindow* window, int key, int scancode, int action, int mode)

Parameters:

- window (GLFWwindow*): Window that receives the event.
- Key (int): The code of the key.
- Scancode (INT): Scancode of the key.
- action (int): action (GLFW_PRESS, GLFW_RELEASE, etc.).
- mode (int): modifiers (Shift, Ctrl, etc.).

Description: Handles keyboard input. Update the global keys[] array, enable/disable animations (clock, broomstick, disk, dog, lamp), and control flags such as active, gClockAnimating, AnimDog, etc.

void DoMovement()

Parameters: None.

Description: Reads the global array keys[] and moves the camera. ProcessKeyboard(...)) according to WASD/arrows, using deltaTime to make the movement independent of the framerate.

void AnimationDog()

Parameters: None.

Description: Controls the dog's animation state using global variables (dogPos, dogRot, FLegs, RLegs, head, tail, earL, earR, dogAnim, step, AnimDog). Change the position and rotations of the dog to simulate walking, wagging tail, head, ears, etc.

void DrawMueble(Shader& shader, GLint modelLoc, GLuint VAO, const glm::vec3& pos, const glm::vec3& rot)

Parameters:

- shader (Shader&): lighting shader to use.
- modelLoc (GLint): The location of the uniform model in the shader.
- VAO (GLuint): VAO of the base cube.
- POS (const GLM::VEC3&): Position of the furniture.
- ROT (const GLM::VEC3&): Rotation (in degrees) of the furniture.

Description: Build and draw the TV cabinet using several scaled/moved boxes with the TV Cabinet texture.

void DrawMesa(Shader& shader, GLint modelLoc, GLuint VAO, const glm::vec3& pos, const glm::vec3& rot)

Parameters:

- shader (Shader&): lighting shader to use.
- modelLoc (GLint): The location of the uniform model in the shader.
- VAO (GLuint): VAO of the base cube.
- POS (const GLM::VEC3&): Position of the furniture.
- ROT (const GLM::VEC3&): Rotation (in degrees) of the furniture.

Description: Draw the dining/living room table using the cube and textured table for board and legs.

void DrawReloj(Shader& shader, GLint modelLoc, GLuint VAO,const glm::vec3& pos, const glm::vec3& rot,float animTime,float minutePeriodSec, float hourPeriodSec);

Parameters:

- shader (Shader&): lighting shader.
- modelLoc (GLint): uniform model.
- VAO (GLuint): VAO of the cube.pos (const glm::vec3&): base position of the clock.
- ROT (const GLM::VEC3&): Base rotation.
- animTime (float): Cumulative animation time (e.g. gClockTime).
- minutePeriodSec (float): Period in seconds of the minute hand.
- hourPeriodSec (float): Period in seconds of the hour hand.

Description: Draw the clock (base and hands) using cube VAO, BaseClock and Hand textures, calculating the rotation of the hands based on animTime.

void DrawEscoba(Shader& shader, GLint modelLoc, GLuint VAO,const glm::vec3& pos, const glm::vec3& rot,float angleFall);

Parameters:

- shader (Shader&): lighting shader.
- modelLoc (GLint): uniform model.
- VAO (GLuint): VAO of the cube.pos (const glm::vec3&): base position of the clock.
- ROT (const GLM::VEC3&): Base rotation.
- angleFloat: angle of inclination/fall of the broom.

Description: Draw the broom (handle + bristles) using FurnitureTV/BroomBristle textures and apply an additional rotation to simulate the animation of falling.

void DrawSillaSimple(Shader& shader, GLint modelLoc, GLuint VAO,const glm::vec3& pos, const glm::vec3& rot);

Parameters:

- shader (Shader&): lighting shader to use.
- modelLoc (GLint): The location of the uniform model in the shader.
- VAO (GLuint): VAO of the base cube.
- POS (const GLM::VEC3&): Position of the furniture.
- ROT (const GLM::VEC3&): Rotation (in degrees) of the furniture.

Description: Draw a simple chair (seat, legs, backrest) with the cube HOV and furniture textures.

void DrawLampara(Shader& shader, Shader& lampShader,GLint modelLoc, GLuint VAO,const glm::vec3& pos, const glm::vec3& rot,const glm::mat4& view, const glm::mat4& projection);

Parameters:

- shader (Shader&): general lighting shader (base + tube).
- lampShader (Shader&): simple shader for screens/spotlights.
- modelLoc (GLint): uniform model (for the currently active shader).
- HOV (GLuint): HOV of the cube.
- POS (const GLM::VEC3&): Lamp position.
- ROT (const GLM::VEC3&): Base rotation.
- view (const glm::mat4&): current view array (for lampShader).
- projection (const glm::mat4&): current projection matrix (for lampShader).

Description: Draw the floor lamp: base, tube (with shader) and the shades/spotlights (with lampShader), changing shader halfway through the function.

void DrawDoorHinged(Shader& shader, GLint modelLoc,Model& model, const Animation& d);

Parameters:

- shader (Shader&): lighting shader.
- modelLoc (GLint): uniform model.model (Model&): 3D model of the door.

- d (const Animation&): Door animation data (angle, pivot, etc.).

Description: Applies a hinged door transform (pivot on hingeLocal) to the Model and draws it according to the Animation state (used for dining room door).

void DrawDiscoBall(Shader& shader, GLint modelLoc,Model& sphere, Model& tube,const Animation& anim);

Parameters:

- shader (Shader&): lighting shader.
- modelLoc (GLint): uniform model.sphere (Model&): model of the dial of the disc.
- tube (Model&): model of the tube/chain.
- anim (const Animation&): animation (position, rotation, currentLength, etc.).

Description: Draw the disco ball from the ceiling using two models (tube + sphere) and use the Animation fields for tube extension and sphere rotation.

Variables

Variable	Description	Associated values or variables
WIDTH, HEIGHT	OpenGL Window Initial Dimensions	1920, 1080
SCREEN_WIDTH, SCREEN_HEIGHT	Actual Framebuffer Size	Set by glfwGetFramebufferSize()
camera	Camera object that controls position and orientation	Initialized in glm::vec3(2.0f, 47.0f, 40.0f)
lastX, lastY	Last recorded mouse position in X/Y	WIDTH / 2, HEIGHT / 2
firstMouse	Flag to detect the first mouse movement	true / false
keys[1024]	Fixing Key States for Input	true (pressed), false (released)
deltaTime	Time between the current and previous frame	currentFrame - lastFrame
lastFrame	Time of the last rendered frame	Updated every iteration
gClockTime	Cumulative time to animate the clock	Incremented using deltaTime
gClockAnimating	Indicates if the watch is animating its hands	true / false

gDoorDining Room	Animation structure controlling dining room door	Contains: angle, target, speed, hingeLocal, etc.
gEscoba	Structure Animation for the broom	Controls Drop/Rotation
gDisco	Animation structure for disco ball	Contains currentLength, maxLength, extendSpeed, etc.
Active	Flag that enables/deactivates the lamp (pointLight[0])	true / false
pointLightPositions[4]	Positions of the point lights in the scene	[0] is the lamp; Other (0,0,0)
TVscreen, MarcoTV, TVfurniture, Table, Handle, BaseClock, BristleBroom	Loaded texture IDs	Generated by loadTexture()
dogPos	Current Dog Position	GLM::VC3
dogRot	Dog Y Rotation	float
FLegs, RLegs	Rotation of front and rear legs	float
head, tail, earL, earR	Head, Tail, and Ear Animation Variables	float controlled by AnimationDog()
dogAnim	Dog animation status	Int
Step	Alternator for Dog Walk Cycle	true / false
AnimDog	General flag to activate the dog animation	true / false
rotDog	Additional Dog Rotation	float
VAO, VBO	Base bucket buffers used in various models	Generated at the beginning
lightPos	Base position of a general light (sometimes not used)	GLM::VC3
dirLight *(uniform)*	Directional light data (ambient, diffuse, specular)	Shipped with lightingShader.Use()
pointLights[4] *(uniform)*	Scene point light data	ambient, diffuse, specular, attenuation

Project Cost Analysis

This section presents a detailed evaluation of the resources invested in the project, including development time, materials and operating expenses, the objective is to determine the real cost of production and estimate an adequate sales price and this

analysis allows to visualize the project as a professional product, justifying both its value and the investment made.

Project Cost Breakdown

Concept	Description	Estimated Hours	Cost (MXN)
1. OpenGL Programming	Implementation of rendering, shaders, camera, lighting, inputs, base animations.	40 h	$5,600
2. Modular 3D Modeling	Creation of furniture, room, objects and structures using cubes + transformations.	8 p.m.	$2,800
3. Integration of Animations	Dog, broom, door, clock, lamp, disco ball.	8 p.m.	$2,800
4. Optimization	Performance adjustments, calculation reduction, visual improvements.	10 am	$1,400
5. Testing and Debugging	Fixes, lighting bugs, clipping, camera, broken textures.	3 p.m.	$2,100
6. Technical Documentation	Dictionary of functions, variables, diagrams, analysis, justification.	10 am	$1,400
7. Basic Texturing	Integration of TV, furniture, table and broom textures.	6 h	$840
8. Final Lighting Settings	Review of brightness, diffuse, point lights, shading cartoon.	6 h	$840

Total direct hours: 127 hrs.

Estimated hourly cost: $140 MXN/h.

Fixed costs

Concept	Description	Cost (MXN)
Electricity	120 hours of work × 150 W × CFE rate $2.3 kWh	$41
PC depreciation	$20,000 MXN PC / 36 months → prorated	$185
Internet	Proportional use (1 month / 30 working days)	$80
Peripheral	Mouse + keyboard (1,200 MXN for 30 months)	$40
Software	All open source → $0	$0

Fixed Cost Subtotal: $346 MXN

Variable costs

Concept	Cost
Labor	$17,780
Using textures or assets	$0 (own resources)
Additional Software	$0

Contingencies (10%)

- It is recommended to cover:
- Design Changes
- Visual adjustments requested
- Unexpected bugs
- Extra time per integration

$$(17,780+346) \times 0.10 = 2,212 \text{ MXN}$$

Total project cost and price at profit

$$\text{Total cost} = 17,780+346 + 2,212 = 20,338 \text{ MXN}$$

$$\text{Suggested Profit Margin: } 30\%$$

$$20,338 \times 0.30 = 6,101$$

$$\text{Base Selling Price:}$$

$$20,338 + 6,101 = 26,439 \text{ MXN}$$

Taxes and final price to the customer

Tax	Calculation	Amount
VAT (16%)	26,439 × 0.16	4,230 MXN
ISR (10% profit)	6,101 × 0.10	610 MXN

$$26,439 + 4,230 + 610 = 31,279 \text{ MXN}$$

Final price to the customer: 31,280 MXN

Observations and justification

The project is developed with open source tools such as OpenGL, GLFW, GLEW, Blender and Visual Studio Code, reducing licensing costs and allowing greater investment in programming and optimization.

Technical complexity

The system includes:

- Low-level programming with C++ and OpenGL
- Using shaders (vertex + fragment)
- Manual animations with matrices
- Complete lighting system (directional + point lights)
- Management of models, VAOs, VBOs
- FPS camera + callbacks + input control
- Integration of textures and materials

This justifies a higher final price than a standard project.

Contingencies (10%)

They allow you to cover unexpected changes, delays, and extra requirements without affecting profitability.

Projected Profit

The 30% margin is standard on visual development or 3D simulation projects, covering support, maintenance, and unbilled hours.

Conclusions

The development of this project allowed for the practical application of course fundamentals, culminating in a functional 3D recreation of the Teen Titans Go Tower in OpenGL. A complete workflow was successfully integrated, from modeling and texturing the seven required objects in Blender to their rendering and setting in the scene. The implementation of the four code-generated animations, including two complex ones, was crucial to add dynamism to the environment and demonstrate the handling of transformations.

The main technical challenge was the correct import of Blender models into the OpenGL base code, ensuring the visual fidelity of textures and scales. Likewise, the programming of complex animations that were coherent with the context required a solid understanding of mathematical logic. This project not only meets the technical requirements but also consolidates skills in creating interactive graphic environments, from conceptualization to documented delivery.

References

- SH design. (2024, January 15). How to make a curved sofa with a boucle fabric in Blender | Blender Tutorial for Beginners [Video]. YouTube. https://www.youtube.com/watch?v=5ZHDL1K6jq8
- Blend nothing. (2023, February 11). How to make stylized Gas Stove in blender [Video]. YouTube. https://www.youtube.com/watch?v=WIjNJMSq7iU
- Nisakai. (2024, 15 de April). ¿Cómo usar Blender 3D? 😈 | Tutorial para principiantes DESDE CERO en español 👌 [Video]. YouTube. https://www.youtube.com/watch?v=HliwF-iTTWo
- Nisakai. (2024, 31 de May). GUÍA RÁPIDA: ¿Cómo modelar en Blender 3D? 😳 | Tutorial de blender en español 👌 [Video]. YouTube. https://www.youtube.com/watch?v=_f_6QN_G3bk
- davidgagul2005. (2018, August 16). Blender Robot Dog [3D model]. Free3D. https://free3d.com/3d-model/blender-robot-dog-319865.html