Computer Graphics and Human-Computer Interaction

Professor: Eng. Roman Balbuena Carlos Aldair

Semester 2025-2

Technical Manual

Group: 05

Student:

319033993

Table of Contents

---

1. *Objectives*

- **Develop an interactive virtual walkthrough using OpenGL 3.3,** featuring a realistic representation of a building façade and two interior rooms, based on carefully selected reference images.

- **Integrate a real-time controllable synthetic camera** that allows free and natural navigation through the virtual space, providing an immersive user experience.

- **Create and implement four animations within the walkthrough** to enhance dynamism and realism in the environment, showcasing meaningful interactions with objects and elements in the scene.

2. *Project Scope*

Included Features

This project involved the complete development of an interactive virtual walkthrough using OpenGL 3.3, based on the façade and two interior rooms of the location “Random Play.”

The walkthrough successfully integrates detailed and textured 3D models, along with various light sources including directional and spotlight lighting, which enhance depth and atmosphere in the virtual environment. User-controlled animations were implemented, enabling dynamic interactions such as the opening of doors and windows, further enriching the immersive experience.

The camera movement was implemented in real time, allowing free exploration of the space from multiple angles and perspectives, improving navigation and engagement within the scene.

The project covers every stage from modeling and texturing of architectural elements to the programming of interaction logic and animations, concluding with the delivery of a functional executable and comprehensive technical documentation in both Spanish and English.

Project Limitations

Despite the achievements, the project faced certain limitations that impacted its development and scope:

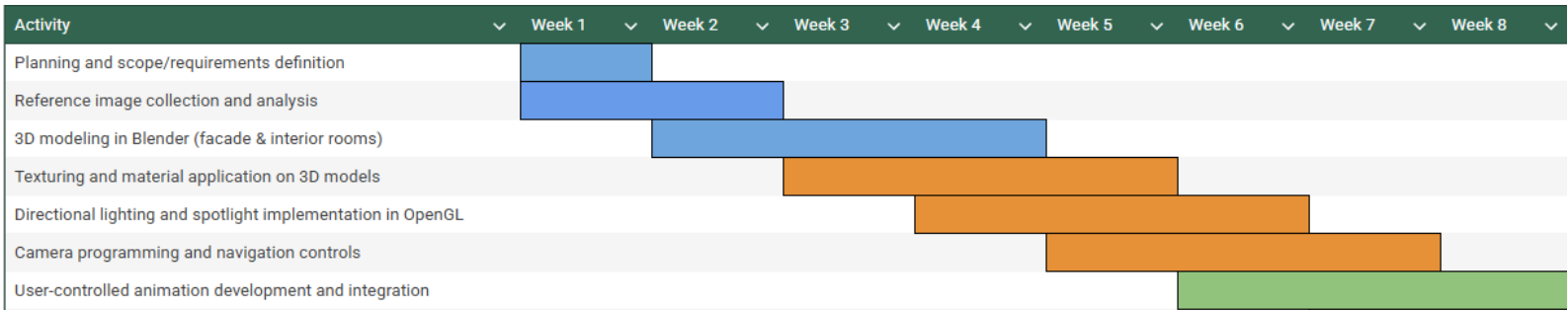
- **Technical Resources**: The available processing power and memory on the workstations limited the complexity of textures and the number of dynamic lights that could be included without compromising real-time performance.

- **Tools and Environment**: Although Blender and OpenGL 3.3 offered broad capabilities, advanced features such as custom shaders or complex visual effects were not fully explored due to the learning curve and time constraints.

- **Functional Scope**: Interaction was limited to four user-controlled animations. Additional features like advanced physics, sound, or artificial intelligence were not included in order to maintain a manageable project within the given resources and timeframe.

These limitations were considered from the planning phase to define a realistic scope that would ensure the delivery of a functional and high-quality product, while respecting the constraints imposed by the context of the project.

3. *Gantt Chart*



4. *Function and Variable Dictionary*

Main Functions

int main()

Main function where GLFW and GLEW are initialized, the application window is created, shaders are configured, 3D models are loaded, and the main rendering and animation loop is executed. It controls the complete execution cycle of the program and event handling.

void DoMovement()

Manages the camera's movement based on user input (W, A, S, D, Q, E keys). Allows for 3D movement and lateral rotation, enabling a dynamic and fluid exploration of the virtual environment.

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

Keyboard event handler that detects key presses and releases. Updates the internal state of keys and triggers specific actions, such as closing the window (ESC) or starting animations with the V, M, B, and N keys, enhancing user interactivity.

void Animation()

Updates the position and rotation angles of animated objects (window, door, laptop) based on predefined states and limits. Implements smooth, oscillating movements.

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

Captures mouse movement to control the camera's orientation. Calculates the displacement between successive mouse positions to rotate the view, improving immersion and intuitive control.

Variables

Shader lightingShader

Shader object responsible for managing lighting in the scene. Executes GPU-based calculations to apply lighting effects that enhance the realism of the 3D models.

Directional Light (dirLight)

A directional light that simulates a distant light source with a fixed direction (-0.2, -1.0, -0.3). Provides uniform lighting, similar to sunlight. Its ambient, diffuse, and specular components define how light interacts with materials to provide depth and volume.

Camera camera(glm::vec3(0.0f, 0.0f, 3.0f))

Object representing the 3D camera, initially positioned on the Z-axis. Controls the view and projection of the scene, allowing for dynamic framing and navigation.

bool keys[1024]

Array that stores the state (pressed or released) of each key, used for efficient and continuous user input handling.

GLfloat deltaTime

Variable that stores the time elapsed between the current and the previous frame. It is used to normalize movement and animation speed, ensuring consistent behavior regardless of frame rate.

Animation variables

Variables such as `ventcMovingRight`, `ventcPos`, `ventRotationAngle`, etc., control the state and parameters of specific object animations, such as position and rotation of windows, doors, and the laptop. They allow for natural and time-synchronized movements.

Shader lightingShader, Shader lampShader

Shaders responsible for general lighting calculations and rendering effects in the scene.

Model compu, vent, ventc, puert, ran;

3D models loaded and rendered in the scene, each with its own geometry, textures, and transformations contributing to the visual composition of the virtual space.

bool firstMouse = true;

Flag that indicates the first mouse movement read, used to prevent abrupt jumps in the camera orientation.

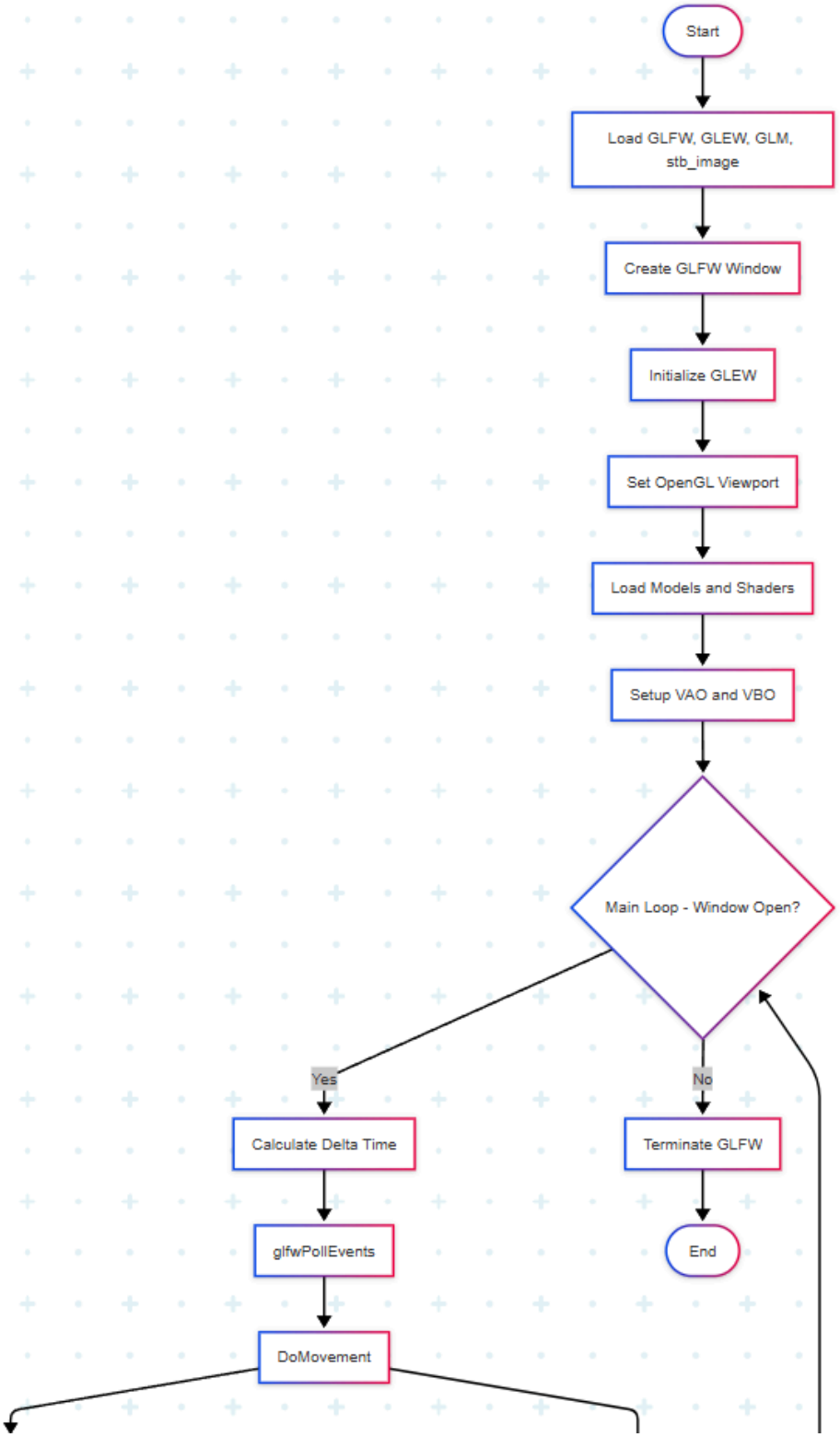
float vertices[]

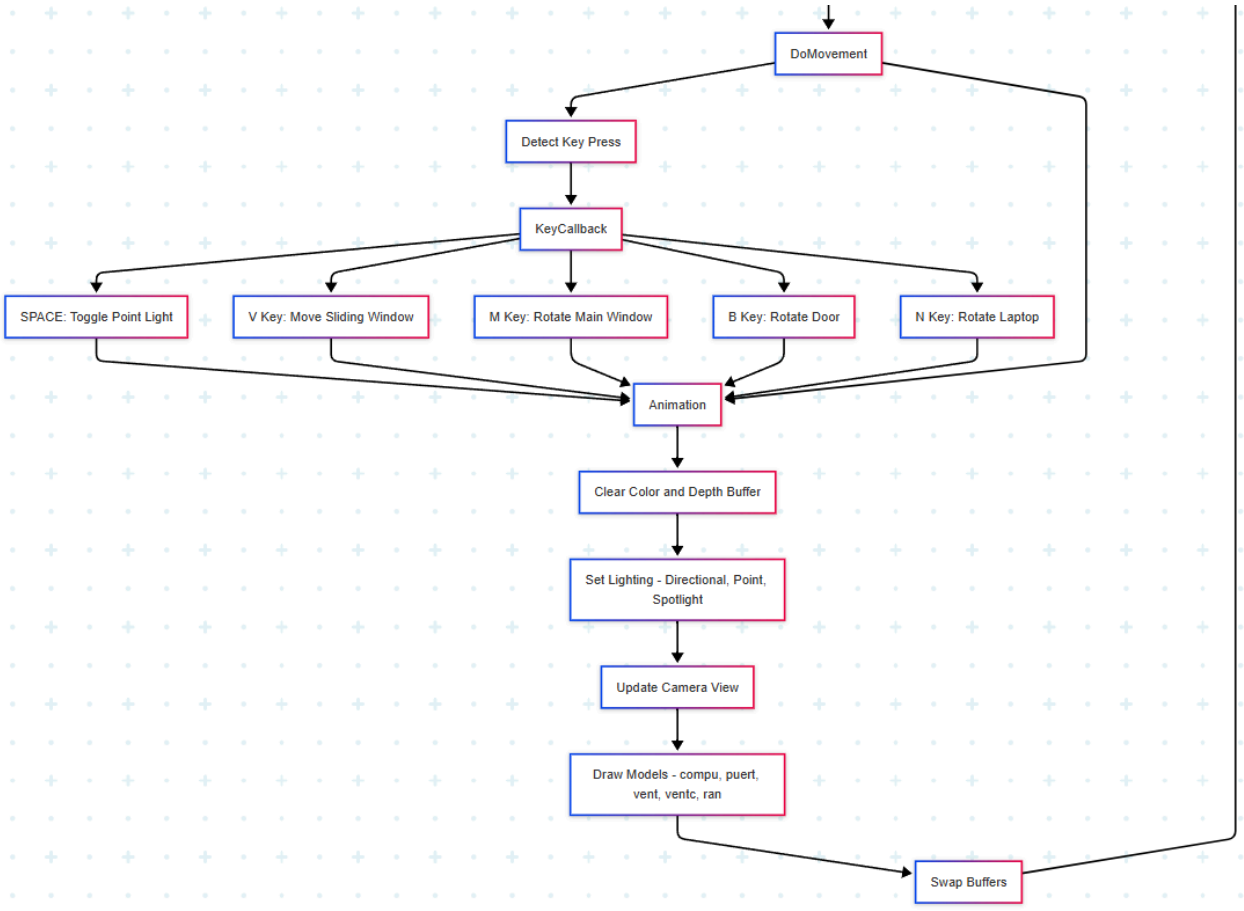
Array that defines the base geometry of a 3D cube, including positions and normals for accurate lighting calculations. Serves as the foundational structure for building objects in the scene.

Transformation matrices (glm::mat4 modelCompu, modelPuert, etc.)

Matrices that manage the position, rotation, and scale of each 3D model. They are essential for animating and placing objects within the 3D space, enabling visual changes and movement.

5. Flowchart


Start
Load GLFW, GLEW, GLM, stb_image
Create GLFW Window
Initialize GLEW
Set OpenGL Viewport
Load Models and Shaders
Setup VAO and VBO
Main Loop - Window Open?
Yes
No
Calculate Delta Time
Terminate GLFW
glfwPollEvents
End
DoMovement



6. Cost Analysis

For the development of the interactive virtual tour using OpenGL 3.3, a detailed cost analysis was conducted, considering human resources, tools, estimated time, risks, and the necessary margins to ensure the quality and sustainability of the project.

Project Objective

Develop an interactive virtual tour using OpenGL 3.3 that includes:

- Realistic 3D modeling of a façade and two interior rooms.
- A real-time interactive camera for free exploration.
- Four animations (doors, windows, interactive objects).
- Dynamic lighting (directional light + spotlight).
- A fully functional executable and technical documentation in both English and Spanish.

Desglose de Costos

Concept	Description	Estimated Hours	Cost (USD)
1. 3D Modeling	Creation of 3D models (facade, rooms, objects) in Blender.	60 hours	$2,400
2. Texturing	Application of realistic textures (walls, furniture, floors).	30 hours	$1,200
3. OpenGL Programming	Implementation of rendering, shaders, camera, lighting, and animations.	100 hours	$5,000
4. Animation Integration	Programming of 4 animations (doors, windows, laptop, etc.).	40 hours	$2,000
5. Optimization	Performance tuning (memory management, LOD, frustum culling).	20 hours	$1,000
6. Testing & Debugging	Validation of interactions and bug fixing.	30 hours	$1,500
7. Documentation	Technical manual (Spanish + English) and user guide.	20 hours	$800
8. Licenses & Tools	Blender (free), Visual Studio (free), possible asset licenses.	-	$300
9. Contingencies (10%)	Unforeseen issues (design changes, technical problems).	-	$1,520
Total		300 hours	$15,720.00

Concept	Description	Price (USD)
Total Cost	Sum of all resources and activities.	$15,720
Profit Margin (30%)	Expected profit.	$4,716
Base Price	Total cost + profit margin.	$20,436
VAT/Taxes (16%)	Value Added Tax (16%)	$3,270
Income Tax (10% on profit)	Income Tax (Corporate Tax)	$472
Final Sale Price	Total invoice amount to client.	$24,178

- Total Development Cost $(15{,}720)+\textit{Profit Margin}(4{,}716)$ =$20,436.
- VAT (16% applied to $20,436) $20{,}436 \rightarrow 3{,}270$.
- Income Tax (10% applied to profit of $4,716) $(4{,}716) \rightarrow 472$.
- **Final Sale Price (Total to Invoice)**: $20{,}436 + 3{,}270 + 472 = 24{,}178$

Observations and Justification

Use of Free Tools:

The development relied on open-source tools such as Blender and Visual Studio Code, eliminating licensing costs and allowing greater investment in technical and visual development.

Free or Custom Textures:
Public domain or manually created textures were used, minimizing expenses related to asset libraries.

Contingencies:
A 10% contingency was included in the base cost to cover unforeseen events such as redesigns, complex bugs, or additional client requests.

Projected Profit Margin:
The 30% profit margin ensures fair compensation for the technical specialization, time investment, and minor post-delivery support and maintenance.

Potential Additional Costs:

- Purchase of premium 3D models for increased visual detail.
- Use of paid tools such as Substance Painter for professional-grade texturing.
- Post-sale support or future feature updates upon client request.

This analysis not only provides transparency regarding the resources used, but also establishes a fair and competitive price that ensures the economic viability of the project for both the developer and the end client.

7. Conclusion

The development of the interactive virtual tour using OpenGL 3.3 successfully met the proposed objectives, delivering an immersive and dynamic experience based on the realistic representation of a façade and two interior rooms of the "Random Play" site.

Detailed and accurately textured 3D models were implemented, along with directional lighting and spotlight techniques. The integration of a real-time controllable synthetic camera enabled smooth navigation throughout the virtual environment.

A structured software development methodology was followed during the process, allowing for organized planning, implementation, testing, and documentation of each phase. As a result, a fully functional product was delivered. The comprehensive technical documentation, available in both Spanish and English, enhances understanding and usability for users and future developers.

The cost analysis presents a realistic and detailed estimation that includes all resources, materials, tools, licenses, and indirect expenses, as well as taxes and profit margins. This ensures the project's economic viability and competitive positioning in the market.

Despite certain technical and functional limitations—such as the absence of advanced visual effects and hardware constraints—these were accounted for during the planning phase, allowing for a realistic and manageable scope. Thus, a coherent product was delivered within the available time and resource constraints, without compromising quality or requirements compliance.

In conclusion, this project stands as a solid practical application of advanced concepts and techniques in computer graphics and human-computer interaction, demonstrating proficiency in modeling, animation, and graphics programming. Furthermore, the entire process—from design to documentation—reflects a professional and well-managed approach to software development.