NATIONAL AUTONOMOUS UNIVERSITY OF MEXICO

FACULTY OF ENGINEERING

ELECTRICAL ENGINEERING DIVISION

COMPUTER ENGINEERING

COMPUTER GRAPHICS AND HUMAN-COMPUTER INTERACTION

TECHNICAL MANUAL

"FAIR"

MEMBERS:

319156715

LABORATORY GROUP: 03

THEORY GROUP: 05

SEMESTER 2025-2

DEADLINE: 20/05/2025

GRADE:________

Content

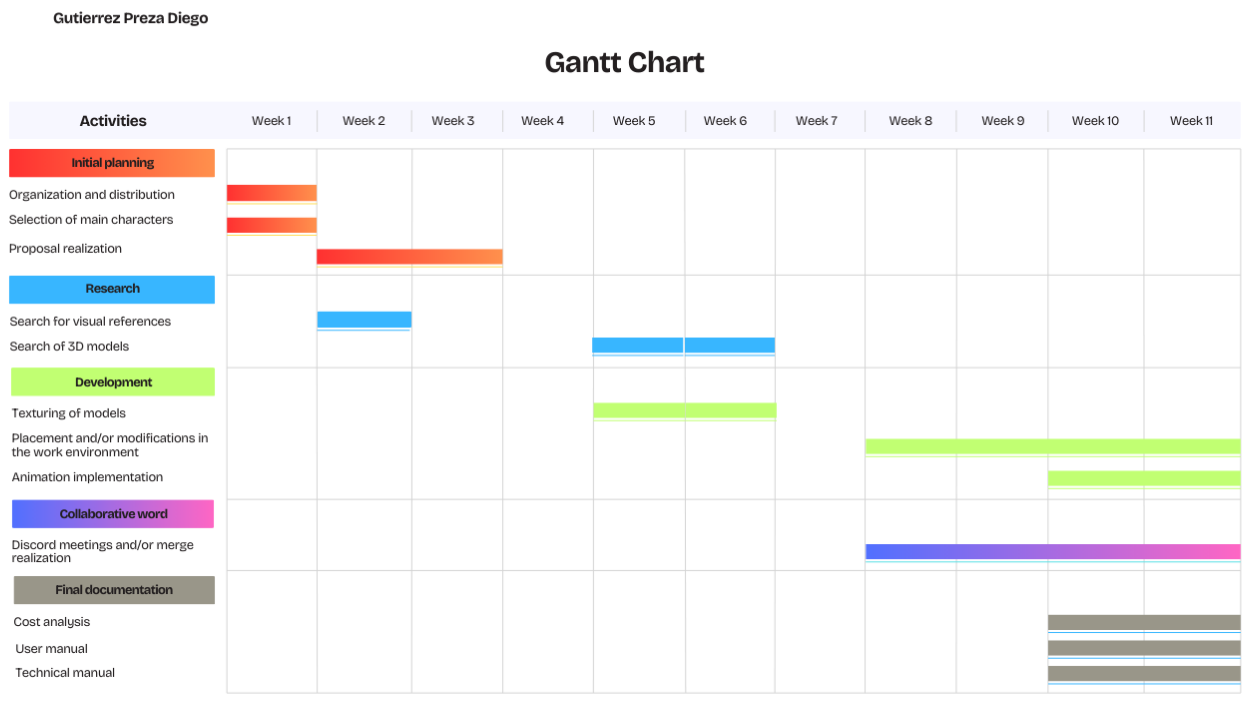
Objective

Through tools oriented to the development of computer-generated graphic elements such as C++ language, OpenGL and Blender, a fair was developed with different elements of the universes of Mario Bros, Snoopy and the incredible world of Gumball, this project was proposed by Professor Roque, where there are 6 different stands which correspond to different stalls, which are bowling, kissing ball, darts, moles, axe throwing and dice each of these positions has different animations which make use of basic transformations, these being scaling, rotation and translation.

In turn, among the objectives we find the correct use of texturing and with it, the correct use of UV mapping to adjust the images of the textures. As for lighting, a day and night cycle was developed in which the lighting decreases from time to time and automatic lights are turned on, it was also implemented that in the day and night cycle the effect of sunlight advancing on the elements was seen, simulating the effect of the sun, In turn, each station has lighting that is activated or deactivated by means of the keyboard.

Finally, 3 different cameras were implemented, which are an aerial camera that allows us to see the entire fair, a third-person camera to tour the fair with the avatar, and finally a camera that focuses on each of the stalls to be able to visualize both the stall and the animation corresponding to each one.

Work schedule



Project Scope

1.Modeling of the fair

- Exterior structure of each of the stands following the visual style of each of the universes.
- Decoration of each of the stands, which is different in all of them, making the fair have a unique aspect for each attraction.
- Modeling of each of the exterior elements, as well as benches, trees, posts, NPCs, etc.

2. Texturing

- Apply custom textures to each model, where those textures are created were edited in GIMP.
- Editing and retouching of images to keep the textures in the specified format, this being PNG and scale in some base 2 number.
- Modification of the UV maps to each 3D model, to ensure that the textures align correctly with the geometry of the model and that there are no unwanted details.

3. Animation

- Implementation of both simple and complex animations. The animations of the stands are so that the user can activate to interact with the environment and animations that are running all the time on the main characters were also implemented

4. Lighting

- implementation of DirectonalLight for the sun, PointLight for the lamp posts, and SpotLight for the spotlights of each stand.
- Adjustment of the lighting system so that it is automatic according to the day and night cycle.
- Creating ambient light that is present throughout the space, like how the sun illuminates in real life.

5. Cameras

- Implementation of 3 different cameras for the different visualization of the fair and in this way have a better appreciation of the details.

Costs

In order to make the development of costs clearer and more understandable, this section will be divided into the different types:

Cost of Human Resources

The development of this project was carried out independently, with a total workload of 200 hours. Considering an intermediate level of experience in C++ language and taking as a reference an average rate of $110.00 MXN per hour, the cost for human resources amounts to:

Human Resources Cost = $22,000.00 MXN

Fixed costs

Fixed costs are those expenses that are not affected by the amount of work performed:

- **Electricity:** A desktop computer, a monitor and a stabilizer were used. Estimated monthly cost: $430.00 MXN.
- **Internet:** The internet service provided by Totalplay was used, with a monthly cost of $420.00 MXN. Since the project lasted 11 weeks, a period of 2.75 months is considered, resulting in a total cost of $1,155.00 MXN.
- **Office:** It is considered an office located in the Roma Norte neighborhood of Mexico City. The workspace had a monthly cost of $5,800.00 MXN. Therefore, the total for 2.75 months is $15,950.00 MXN.
- **Equipment depreciation:** An Asus TUF Gaming A15 laptop was used with an estimated cost of $26,000.00 MXN. Considering a useful life of five years (60 months), the monthly depreciation is $433.33 MXN. Multiplied by the project period, a depreciation cost of $1,191.67 MXN is obtained.

Total, Fixed Costs = 430 + 1,155 + 15,950 + 1,191.67 = $18,726.67 MXN

Variable Costs

Variable costs correspond to the inputs and tools used whose need depends directly on the work performed:

- **Software:** Free development tools were used: Visual Studio 2022 Community Edition and GitHub, so no additional costs were incurred.

Total, Variable Costs = $0.00 MXN

Net Project Cost

The net cost considers the sum of all the above items:

- Human Resources: $22,000.00
- Fixed Costs: $18,726.67
- Variable Costs: $0.00

Net Cost = $40,726.67 MXN

Cost with Profit Margin

A profit margin of 30% is considered to ensure the profitability of the project:

- Profit: $40,726.67 x 0.30 = $12,218.00
- Subtotal with profit = $52,944.67 MXN

Final Cost to Customer

For the calculation of the total cost to the customer, taxes and commissions for payment gateways are considered:

- VAT (16%) = $8,471.15
- Subtotal with VAT = $61,415.82
- Card Payment Fee (5.5%) = $3,377.87

Total Cost to Customer = $64,793.69 MXN

The final price of this project, considering operating costs, inputs, professional fees, projected profit and tax obligations, amounts to **$64,793.69 MXN**. This amount guarantees both the coverage of expenses and adequate remuneration for the work performed.

Documentation

Main Archive

- **batting (...)**

Render the models that make up the batting cage game. It includes the stand, the bat, the ball, the target to be hit and decorative posters. This function ensures that all objects are displayed with appropriate lighting and transformations.

Parameters:

GLM::MAT4& Model: Applied transformation matrix.

GLuint uniformModel: Location of the uniform model in the shader.

std::vector<Model*>& objectsBatting: vector with the models of the batting game.

- **bowling (...)**

Render the elements of the bowling stand, such as the bowling pins, the ball, the stand, the table, and decorative signs. The function applies the corresponding transformations and sends each model to the rendering pipeline.

Parameters:

GLM::MAT4& Model: General transformational matrix.

GLuint uniformModel: Identifier of the model uniform in the shader.

std::vector<Model*>& objectsBowling: models of the bowling stand.

- **setting(...)**

Render all the objects of general atmosphere of the fair: trees, lights, benches, trash cans, lagoon, among others. It serves to bring life and visual coherence to the non-playable environment.

Parameters:

GLM::MAT4& Model: Global transformation matrix.

GLuint uniformModel: Transform uniform in the shader.

std::vector<Model*>& objectsSetting: static scenario models.

- **food (...)**

Draw the food stalls and their respective elements such as popcorn, ice cream, cotton candy. It adds visual appeal and thematic realism to the fair.

Parameters:

GLM::MAT4& Model: Matrix used to locate models.

GLuint uniformModel: Uniform location in the shader.

std::vector<Model*>& objectsFood: models that represent food and stalls.

- **NPCs (...)**

Render the models of non-playable characters (NPCs) such as Teri, Carrie, etc. They are distributed around the fair for decorative and atmospheric purposes.

Parameters:

GLM::MAT4& Model: Common transformation matrix.

GLuint uniformModel: reference to the uniform of the model.

std::vector<Model*>& charactersNPCs: list of characters to render.

- **areaPhotos(...)**

Render a themed section designed as a photo zone, including posing characters and decorative settings. Animated transformations are used for certain elements (wings, legs, etc.).

Parameters:

GLM::MAT4& Model: Base matrix for transformations.

GLuint uniformModel: Location of the uniform model.

std::vector<Model*>& objectsAreaPhotos: models that make up the scenario.

float deltaTime: Time between frames, used for animations.

- **UpdateBatting(float speed)**

Controls the movement of the ball and bat in the batting game. It is responsible for the continuous animation of both objects during the game (before or after the impact).

Parameter:

Speed: Delta Time multiplied by a factor, controls animation speed.

- **UpgradeBowling(float speed)**

He is in charge of the animation of the bowling ball, making it advance until it impacts the pins. It can also handle bounces or ball reset.

Parameter:

Speed: Time control to animate the ball.

- **PitchBatting()**

The animation cycle begins where the ball is thrown and the bat moves to hit it. It binds to the Y key on the keyboard.

- **Throwball()**

Activate the bowling ball to be thrown, controlling its movement until it hits the pins. Respond to the O key.

Functions given by Eng. Jose Roque RG

- **mainWindow.Initialise()**

It initializes the main OpenGL window with GLFW and GLEW, setting the parameters for graphical context, buffer size, and capturing events.

Parameters:

It doesn't receive parameters directly (it uses the values defined when building mainWindow).

- **CreateObjects()**

It creates several primitive geometric meshes (pyramid, floor, vegetation, sign) with their vertices and indexes, and stores them in meshList.

- **CreateShaders()**

Loads the shader files (vertex and fragment) from the vShader and fShader paths, and adds them to the shaderList vector.

- **LoadModel(...)**

Loads a .obj model from a specific file path to the Model object. It is called many times for each 3D object of the fair (stalls, setting, characters, etc.).

Parameters:

const char* fileName: Path relative to the model's .obj file.

- **Skybox(...)**

Create a sky box with 6 images that renders in the background to simulate an immersive environment. Two are created: one for the day and one for the night.

Parameters:

std::vector<std::string> faceLocations: Image paths for each face of the cube (right, left, bottom, up, front, back).

- **Material(...)**

It creates a material with reflectance properties to be applied in specular lighting.

Parameters:

float specIntensity: Intensity of the mirror reflex.

Float Shininess: Brightness of the material (higher value = smaller, brighter reflection).

- **DirectionalLight(...)**

A constructor that initializes a directional light, which simulates a distant light source such as the sun. It has no position, only direction, and it evenly affects all objects in the scene.

Parameters:

GLfloat red, green, blue: components of the color of light.

GLfloat ambientIntensity: The intensity with which light affects areas that are not directly illuminated (ambient light).

GLfloat diffuseIntensity: Intensity of direct light on objects.

GLfloat xDir, yDir, zDir: direction in which the light points.

- **PointLight(...)**

Builder of a point light, which emits light in all directions from a specific position. This type of light dims with distance, simulating a lamp or lantern.

Parameters:

GLfloat red, green, blue: light colour components.

GLfloat ambientIntensity: ambient intensity.

GLfloat diffuseIntensity: Diffuse intensity.

GLfloat xPos, yPos, zPos: position of light in the 3D world.

GLfloat constant, linear, exponent: light attenuation coefficients. These determine how the intensity decreases with distance (standard dimming model).

- **SpotLight(...)**

Builder of a spotlight light, which is like a point light but with a defined direction and an opening angle. It is useful for simulating flashlights, headlights, or reflectors.

Parameters:

GLfloat red, green, blue: color components.

GLfloat ambientIntensity: ambient light emitted.

GLfloat diffuseIntensity: Diffuse intensity.

GLfloat xPos, yPos, zPos: position in space.

GLfloat xDir, yDir, zDir: direction in which the focus is pointing.

GLfloat constant, linear, exponent: attenuation coefficients.

GLfloat edge: angle of aperture of the light cone (in degrees), defines the illumination limit of the spotlight.

- **getActiveCamera()**

Returns the integer ID of the camera currently selected by the user (0 = aerial, 1 = third person, 2 = post camera).

- **mouseControl(x, y)**

It receives mouse shifts in X and Y, and adjusts the orientation (yaw, pitch) of the active camera.

Parameters:

GLfloat x: Horizontal mouse scrolling.

GLfloat and: vertical mouse scrolling.

- **calculateViewMatrix()**

Returns the view array (glm::mat4) calculated from the camera's position and direction. Used to transform the scene from the user's point of view.

- **UseShader(...)**

Enable the pre-compiled shader to start using it in drawing operations.

- **SetSpotLights(...)**

Send a spotlight array of lights to the active shader. Each light contains color, direction, dimming, intensity, and cutoff angle information. It is used to illuminate specific areas of the scene such as individual games (batting, bowling, etc.).

Parameters:

SpotLight* lights: Pointer to an array of active SpotLight lights.

unsigned int lightCount: Number of SpotLight lights that need to be sent to the shader

- **SetPointLights(...)**

Bind a set of point lights, such as lanterns or poles, to the shader. These lights emit in all directions from a point and dim with distance.

Parameters:

PointLight* lights: PointLight light array.

unsigned int lightCount: number of lights to activate and switch to the shader.

- **SetDirectionalLight(...)**

Assign a single directional light to the shader, which simulates a light source such as the sun. This light affects the entire scene from a fixed direction.

Parameters:

DirectionalLight* dLight: Pointer to a directional light instance with the parameters already defined.

- **RenderModel()**

Draw a model previously loaded in OpenGL on the screen.

- **swapBuffers()**

Swap the drawing buffer in the window to display the rendered image. It's part of the real-time rendering cycle.

Separate Files

File: ZonaFotos.cpp

void zonePhotos (glm::mat4 model, GLuint uniformModel, std::vector<Model*> objectsZonePhotos, float deltaTime)

Render and animate the characters Snoopy, Woodstock, Gumball, and Yoshi within a specific area.

- Model: Transformation matrix.
- uniformModel: Uniform location in the shader for the model array.
- objectsAreaPhotos: 3D models used (Snoopy, Gumball, Woodstock, Yoshi and his parts).
- deltaTime: Time between frames, used for smooth animations.

Gumball Animation

- **Main variables:**
  - angulovaria, timeGumball: accumulators of time and angles.
  - movGumball: Offset on the Z-axis.
  - AdvanceGumball: controls direction of movement.
  - angleLimbsG: basis for calculating animations of arms, legs and tail.
  - rotationContinuousGumball: rotation of the body while walking.

Behavior: Gumball moves back and forth on the Z-axis, spins when he changes direction, jumps gently, and moves limbs and tail with a clapping effect and opening and closing his legs.

File: NPCs.cpp

void NPCs(glm::mat4 model, GLuint uniformModel, std::vector<Model*> charactersNPCs)

Render multiple NPCs (static characters) located in different positions on the map.

- Model: Base matrix of transformation.
- uniformModel: Identifier of the model uniform.

- charactersNPCs: list of 3D models of characters.

Auxiliary functions per character:

All of these functions use the same structure:

void render[Name](glm::mat4 model, GLuint uniformModel, Model& character, glm::vec3 position, int degrees)

Render a character model at the specified position and rotation.

- Position: Vector of position in the world.
- degrees: rotation on the Y axis in degrees.

Applies to: renderCarrie, renderTeri

File: Tierra.cpp

void earth(glm::mat4 model, GLuint uniformModel, Texture& earth, std::vector<Mesh*> meshList)

Draw paths and floor areas for each section of the map.

- Earth: texture to be applied.
- meshList: list of meshes; meshList[5] is used for terrain.

void renderEarth(glm::mat4 model, GLuint uniformModel, Texture& ground, Mesh& floor, glm::vec3 position, glm::vec3 scale)

Renders a portion of textured terrain with custom scaling. *

- Floor: mesh to be drawn.
- Position: Global position.
- Scale: Scaling vector.

File: Window.cpp

Window::Window() / Window::Window(GLint windowWidth, GLint windowHeight)

Constructors of the Window class. They initialize size, keyboard and camera status.

- int Window::Initialise(): Initializes the window, GLEW, and the OpenGL context.
- void Window::createCallbacks(): Assigns keyboard and mouse handling functions.

- void Window::HandleKeyboard(GLFWwindow* window, int key, int code, int action, int mode): Handles keyboard events for camera and light control.
    - Key: A key pressed.
    - Action: Type of event (press, release).
- void Window::MouseHandle(GLFWwindow* window, double xPos, double yPos): Calculates the difference in mouse movement for the camera.
- GLfloat getXChange() / GLfloat getYChange(): Return the mouse position change in X or Y.

Important variables:

- CameraActive: Select between aerial camera, third-person or by seats.
- Spot[MAX_SPOT_LIGHTS lights: Controls on/off lights per seat.
- PositionCurrent: Index of the active position for animation or lighting.
- Muevex: General horizontal motion modifier.

File: Ambientacion.cpp

void setting(glm::mat4 model, GLuint uniformModel, std::vector<Model*> objectsSetting)

Render decorations such as benches, trees, boats, shelves, etc.

- objectsSetting: list of environmental models.

Render auxiliary functions:

void render[Object](glm::mat4 model, GLuint uniformModel, Model& model, glm::vec3 position[, int degrees])

- They apply transformations and render the model.
- Versions with degrees allow rotation (e.g. benches, boats).
- They include: renderBanking, renderLuminaire1/2/3, renderTree1/2/3, renderBote1/2, renderDecorativePlant, renderRack, renderLuminaireCeiling, renderLaguna.

File: Bateo.cpp

void batting(glm::mat4 model, GLuint uniformModel, std:vector<Model*> objectsBatting)

Render the batting station: cage, fences, poster, ball and bats.

- void renderStand2(...): Renders the main stand.
- void renderBardas(...): Draw three fences for the game environment.
- void renderballBatting(...): Renders two balls: one fixed and one animated that moves and rotates.
- void renderCartelBatting(...): Draws the poster on the back of the stage.
- void renderBats(...): Renders two bats: an animated one that rotates and rises, and a decorative static one.
- void updateFloat deltaTime: Manages ball progress, bat impact, and return animation for both.
- void throwBatting (): Activates the ball toss if no other animation is active.

Important variables:

- PositionBall: animated position of the ball.
- rotationBat, heightBat: state of the bat.
- ballInMotion, batHitting, ballReturning, batReturning: control flags.
- VELOCIDAD_BOLA_ORIGINAL: base ball speed.

File: Bolos.cpp

void bolos(glm::mat4 model, GLuint uniformModel, std::vector<Model*> objectsBowling)

Render the bowling stand with stand, balls, pins, poster and table.

- void renderStand5(...): Renders the bowling stand.
- void renderPines(...): Draw the 10 pines in a triangular formation. Simulates falls with rotations.
    - Base: starting position.
    - AngleAnimated: progressive angle for simulated fall.
- void renderBall(...): Renders balls. If animated, it is dynamically rotated.

    - position: 3D coordinates.
    - Animated: If true, turn on rotation.
- void renderTable(...): Renders a long table behind the play area.
- void renderCartelBoliche(...): Renders the back sign.
- void updateBowling(float deltaTime): Simulates the throwing and rotation of the ball, progressive knocking down of pins, and restarting positions after falling.
- void throwball(): Activates the ball throw.

Key variables:

- positionBall, rotationBall: the animated state of the ball.
- pinsKnocked down, angleAnimation: logic of the fall of the pines.

Conclusions

The development of this project represented a comprehensive experience in which technical knowledge, design skills, planning and resource management were combined. Throughout the 11 weeks of work, it was possible to build a functional, aesthetically attractive and technically sound product, meeting the objectives set from the beginning.

The correct distribution of time and the application of tools such as Visual Studio 2022 and GitHub allowed to maintain an efficient and orderly workflow. This project is of considerable complexity due to the number of models, animations, lighting logic, etc. that was implemented according to Professor Roque's specifications. This project not only allows the visualization and interaction of different elements but also allows us a more realistic and interactive tour due to the different functions that the user has for the user to interact. This demonstrates mastery of fundamental concepts in graphical programming and modular code structuring.

On the economic side, a detailed cost analysis was carried out that included both human resources and inputs and operating expenses. This breakdown made it possible to establish a fair sale price, which considers not only the investment made, but also a reasonable profit margin and the tax commitments associated with the commercialization of the project. a schedule was also established through a Gantt chart which allows us to see the workflow, as well as the documentation made allows us to know the details of the project for different people.

In short, the project not only managed to meet the technical and functional requirements requested, but also represents a clear example of professional and self-directed work.

Model Links:

Cotton candy: https://www.cgtrader.com/items/4617534/download-page

Table: https://sketchfab.com/3d-models/cutting-board-61eadc1d646c47ac892ce418677dfb76#download

Snoopy: https://sketchfab.com/3d-models/snoopy-72116d2e288f4c45a11f323d76142a6c#download

Luminaires: https://www.turbosquid.com/es/3d-models/free-max-model-classical-street-light/965356

Trash cans: https://www.turbosquid.com/es/3d-models/free-obj-model-street-bin/1106299

Food stall: https://free3d.com/es/modelo-3d/temporary-food-stalls-80503.html

Gumball waterson: https://sketchfab.com/3d-models/gumball-fusionfall-heroes-39c568e694e1478bb0e3ed81cde9c359

Bookshop Awards: https://sketchfab.com/3d-models/libreria-vuota-098205aa418f4237b361328a282e21e3

Texture Links:

Pictures:

https://www.arte-mio.mx/MLM-763335879-set-nintendo-super-mario-bros-4-cuadros-en-tela-canvas-list-_JM

The Amazing World Of Gumball Clown Mystery Explained

peanuts-the-art-of-snoopy-history-snoopy-flying-ace-snoopy-come-home

Grass: https://img.freepik.com/fotos-premium/textura-hierba-verde-que-esta-hecha-empresa-empresa_612834-258.jpg

Skybox: https://opengameart.org/content/sky-box-sunny-day

Table: https://www.istockphoto.com/es/fotos/madera-clara-textura

Earth:https://png.pngtree.com/thumb_back/fw800/background/20220711/pngtree-brown-paper-texture-ground-parcel-photo-image_999914.jpg