NATIONAL AUTONOMOUS UNIVERSITY OF MEXICO

FACULTY OF ENGINEERING

ELECTRICAL ENGINEERING DIVISION

COMPUTER ENGINEERING

COMPUTER GRAPHICS AND HUMAN-COMPUTER INTERACTION

TECHNICAL MANUAL

"FAIR"

319094479

LABORATORY GROUP: 03

THEORY GROUP: 05

SEMESTER 2025-2

SUBMISSION DEADLINE: May 20th , 2025

GRADE: _____

Index

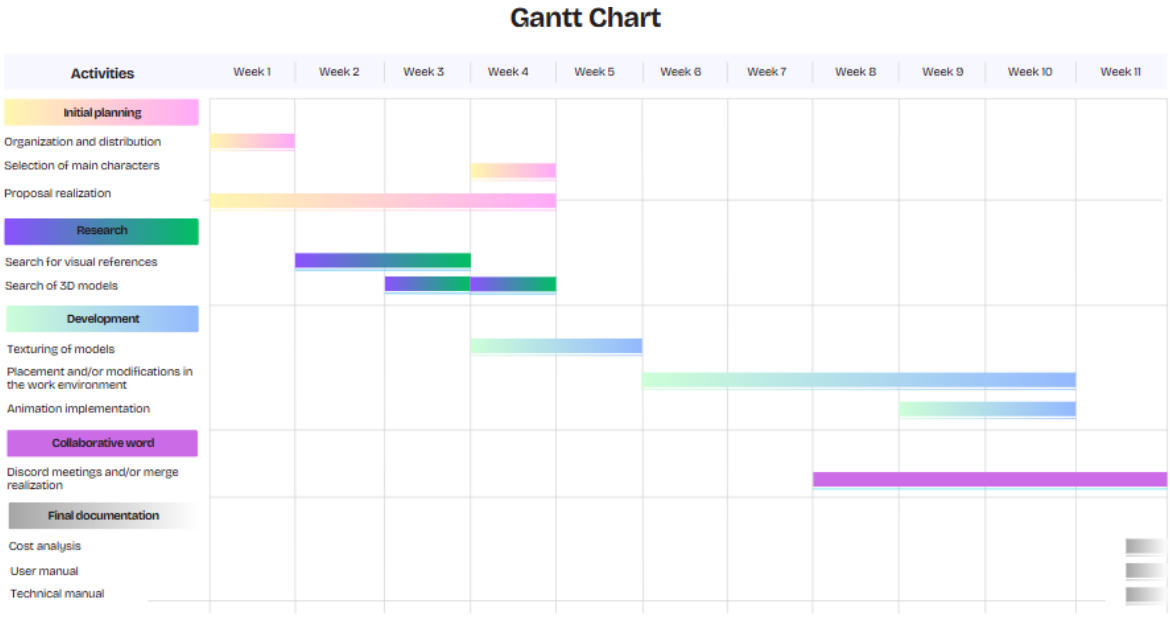
Objective

This project, proposed by my laboratory professor, Eng. José Roque Román, will develop a virtual environment through which you can navigate and will be inspired by a game fair set with elements of the universes of *Mario Bros*, *Snoopy* and *The Incredible World of Gumball*; this fair will be centered on 6 game stands which are: axe throwing, dart throwing, batting cage, dice throwing, bowling line, and "hit the mole". The project will be developed using specialized tools in working with graphic elements, such as *OpenGL* with *C++* in *VisualStudio*, *3dsMax*, *Blender* and *GIMP.*

The elements that are visualized in the scenario will be mostly textured 3D models with correct UV mapping; In each of the booths of the games there will be a characteristic animation of the game, as well as for each of the main characters included. As an additional setting, there will be lights that are activated by means of the keyboard, a light that works as a day and night cycle, and automatic lights that are activated when it is dark, and deactivated when it is daytime.

There will also be 3 different cameras: a camera that follows the avatar that travels through the fair in third person, 6 cameras that will be located in each game where the implemented animations can be seen from them, and an aerial camera that visualizes the entire fair from above.

Schedule



Scope

1.Modeling

- Assembly of each game stand following the visual style of each of the mentioned universes.
- Different decoration for each stand so that there is a unique look for each attraction.
- Correct positioning of each of the exterior elements: benches, trees, lampposts, NPC's, entrance arch, fountain, and food stalls.

2. Texturing

- Image editing to keep textures in the required format (.png), and scalding on both axes at the same power of 2.
- Application and modification of the UV maps for each 3D model, so that the textures are correctly aligned with the geometry of the model taking care of the details.

3. Animation

- The games' signature animations are implemented in a way that the user can activate them to interact with the environment, and there are also animations that are running all the time on the main characters (Yoshi, Gumball, and Snoopy), as well as the animated sign that is on the entrance arch.

4. Lighting

- A DirectonalLight *is used* to simulate the effect of the sun, *PointLight* for the lamp posts, and *SpotLight* for the spotlights inside each stand.
- Inclusion of a lighting system so that the lights on poles are automatic according to the day and night cycle.

5. Cameras

- Implementation of 3 different cameras for different viewing points of the fair and observation of all the details.

Costs

The cost of developing this project is broken down as follows:

Cost of Human Resources

This project was carried out with a total workload of 110 hours. Considering an intermediate level of experience in C++ and *OpenGL language,* and taking as a reference an average rate of $200.00 MXN per hour, the cost for human resources amounts to:

Human Resources Cost = $22,000.00 MXN

Fixed costs

These 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.
- **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 + 1,191.67 = $2,776.67 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

Total Project Cost

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

- Human Resources: $22,000.00
- Fixed Costs: $2,776.67
- Variable Costs: $0.00

Total Cost = $24,726.67 MXN

Cost with Profit Margin

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

- Profit: $24,726.67 x 0.30 = $7,432.80
- Subtotal with profit = $32,159.47 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%) = $5,145.51
- Subtotal with VAT = $37,304.98
- Card Payment Fee (5.5%) = $2,051.77

Total Cost to Customer = $39,356.75 MXN

It should be considered that the project is a team effort of 3 people, so the cost shown must be multiplied by 3.

This amount guarantees both the coverage of expenses and adequate remuneration for the work performed.

Documentation

Main file

- **darts(...)**

Render all the visual objects of the darts game, such as the board, decorative balloons, the base, and the darts themselves. Visual charts related to the game's theme are also included.

Parameters:

- GLM::MAT4& Model: Matrix for model transformations.
- GLuint uniformModel: Location of the uniform model in the shader.
- std::vector<Model*>& objectsDarts: Pointers to models in the darts game environment.

- **moles (...)**

In charge of rendering the game environment "hit the mole". Draw objects such as the table, spawn tubes, pop-up characters (such as carnivorous plants), and additional details.

Parameters:

- GLM::MAT4& Model: Transformation matrix.
- GLuint uniformModel: Uniform location in the active shader.
- std::vector<Model*>& objectsTopos: models of the interactive topos post.

- **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 Snoopy, Charlie Brown, Bowser, Donkey Kong, 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.

- **photoArea(...)**

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.

- **UpdateDarts(float speed)**

Updates the position of the dart in flight, simulating its trajectory towards the board. Increases the realism of the cast by moving it frame by frame.

Parameter:

- Speed: Time between frames, adjust the animation of the dart.

- **UpdateTopos(float speed)**

Encourage the movement of moles or emerging figures in the game "hit the mole". Alternate between going up and down based on time.

Parameter:

- Speed: Regulates the frequency of mole appearance.

- **throw dart()**

Fire the dart toward the board, initiating a linear or parabolic flight animation. It is executed by pressing the I key.

- **activaTopos()**

Trigger the appearance of the game's moles, initiating their vertical upward movement to be hit. Associated with the P key

Topos File

- **void topos(glm::mat4 model, GLuint uniformModel, std::vector<Model*> objectsTopos)**

Render all elements of the mole game, including the main booth and the machines from which the poles emerge. Use multiple renderMachine functions to place them in different positions.

- **void renderStand6(glm::mat4 model, GLuint uniformModel, Model& stand, Model& picture, glm::vec3 position)**

Draw the main structure of the stand and a decorative painting. Performs position, rotation, and scale transformations on models.

- **void renderMachine(...), renderMachine2(...), renderMachine3(...)**

They render variants of the mechanism from which the moles come out. They include a table, several tubes, and plants. renderMachine3 includes vertical animation, rotation, and dynamic scaling for the fifth top.

- **void activateTopos()**

Initializes the animation variables of the moles. Start by activating the first mole.

- **void updateTopos(float deltaTime)**

Update the polka dot animations. Control up/down, rotation, and sequential appearance. The last mole (mole 5) is scaled at the end of the animation.

Important variables for animation:

- translationTopos[5]: Controls the vertical movement of each individual mole. Increases as you go up, decreases as you go down.
- rotationTopos[5]: Defines the angle of rotation of each mole on the Y-axis while it is active.
- TopsRising[5]: Indicates whether a mole is in an ascending phase (true) or descending (false).
- animationTopos On[5]: Topos activates or disables the animation of each individual mole.
- topoCurrent: Index of the mole that is currently active in the stream.

- delayTopos: A timeout between the activation of one mole and the next.
- Cumulative time: A time accumulator that, when passing delayTopos, activates the next mole.
- scaleTopo5: Scale applied to the fifth mole. It is progressively reduced until it disappears.
- topo5Scaling: Activates the phase in which the fifth mole begins to scale.
- TimerTopo5: Controls how long the fifth mole remains invisible before respawning.

Darts File

- **void darts(glm::mat4 model, GLuint uniformModel, std::vector<Model*> objectsDarts)**

Render the darts game: stand, balloons, and multiple darts (one animated).

- **void renderDarts(...)**

Draw a row of fixed darts and an animated dart that advances towards the rotating balloons.

- **void renderGlobos(...)**

Show balloons organized on bases. The last balloon climbs on impact.

- **void updateDarts(float deltaTime)**

Manage the animation of the dart throw:

- **Void throw dart()**

Restart the variables to start a new launch.

Important variables for animation:

- Dart animation: Represents the progress of the dart animation over time. Control their position and rotation.
- ThrowingDart: Activates the animation sequence of the dart. Only if it is set to true are the values updated.
- balloonImpacted: Marks the moment when the dart has reached the balloon. It gives way to the reduction phase.
- dartstopped: Stops the dart's advance once it has impacted.
- scaleGlobe: Scale of the affected globe. It is reduced from 1.0f to 0.0f to simulate exploding.
- timeImpactDart: A stopwatch that moves forward from the impact and regulates how long the balloon remains missing.

PhotoZone File

Render and animate a photo zone with various interactive and moving characters.

Woodstock

Woodstock oscillates vertically mimicking a soft flutter. The wings also rotate alternately (both animated by sin(fliesWoodstock * 5.0f)).

- flyWoodstock: base angle to calculate vertical movement.

Yoshi

It moves from side to side with an undulating jump while shaking its legs, rotates 360º when changing direction and makes a special animation with arms when it changes direction.

Important variables for animation:

- angSaltoYoshi: angle to animate jump in the Y-axis.
- jumpYoshi: result of the sin that defines vertical elevation.
- frequencyJumpYoshi, amplitudeJumpYoshi: parameters of the jump.
- angPatasYoshi: breast-based leg animation.
- movYoshi: horizontal scrolling on the X-axis.
- Yoshi advances: direction of movement.
- movArm, timeArm, angArmYoshi, angSpinYoshi: Rotating arm animation control and full rotation when it reaches extremes.

NPCs File

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

This file only positions static non-playable characters (NPCs) in different areas of the ferium. These characters are not animated, they are only rendered with translation, rotation, and scaling transformations.

Parameters:

- Model: Base matrix of transformation.
- uniformModel: Identifier of the model uniform.
- charactersNPCs: list of 3D models of characters.

Earth File

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

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

Parameters:

- 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.

Important variables:

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

Window File

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

They are the builders of the class. They initialize size, keyboard and camera status.

- **int Window::Initialise()**

Initializes the window, GLEW, and the OpenGL context.

- **void Window::createCallbacks()**

Assign keyboard and mouse handling functions.

- **void Window::KeyboardHandle(GLFWwindow* window, int key, int code, int action, int mode)**

Handle keyboard events for camera and light control.

Important parameters:

- Key: A key pressed.
- Action: Type of event (press, release).

- **void Window::MouseHandle(GLFWwindow* window, double xPos, double yPos)**

Calculate the difference in mouse movement for the camera.

Ambientation File

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

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

Food File

- **void food(glm::mat4 model, GLuint uniformModel, std::vector<Model*> foodobjects)**

Render the food stalls.

- objectsFood[0]: popcorn

Functions given by Ing. 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.

- **CreateDice()**

Build and store the mesh of a cube with textures applied to each face, simulating a die. It is saved 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.

Conclusion

The development of a project of this caliber meant putting into practice all the knowledge and techniques learned throughout the semester in the subject of Computer Graphics and Human-Computer Interaction, both in the theory class and in the laboratory.

It was achieved complete virtual stage inspired by a fair themed with elements of the universes of *Mario Bros*, *Snoopy* and *Gumball*, from a skybox that changes when "the night comes"; a large number of textured models with which the interactive game stalls, food and setting places were assembled, taking into account that all the games were animated and the main characters are in motion all the time; an automatic lighting system that simulates a day and night cycle; up to a set of 3 cameras between which you can switch at any time.

Although all the requirements that this work should have contained were satisfactorily met, I think there may be some things that could be corrected or changed, which unfortunately due to lack of time, could not be attended to as I would like, such as the issue of animations.

In this project we were able to apply the mentioned before things following a creative approach based on teamwork. I also want to mention that it was a great opportunity to learn how to use a remote storage system like GitHub, and work in a way that could happen in the future in professional life when collaborating in a team.

Model Links:

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

House (game booths): https://free3d.com/3d-model/old-farm-house-91130.html

Woodstock: https://sketchfab.com/3d-models/wii-u-the-peanuts-movie-woodstock-d7a8747fc83f49e194dd9ccde0cc2d49

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

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

Piranha: https://www.cgtrader.com/search?keywords=mario+piraña

Donkey Kong: https://www.models-resource.com/wii/mariosupersluggers/model/58606/

Yoshi: https://www.models-resource.com/wii/mariosupersluggers/model/58610/

Wario: https://www.models-resource.com/wii/mariosupersluggers/model/58612/

Bowser: https://www.models-resource.com/wii/mariosupersluggers/model/58611/

Trees: https://free3d.com/es/modelo-3d/free-tree-pack-109238.html

Popcorn: https://www.turbosquid.com/es/3d-models/3d-popcorn-corn-pop-model-1645559

Balloon base: https://sketchfab.com/3d-models/ocf-easel-b124839ed52a41379ba4fe973eb05fde

Dart: https://sketchfab.com/3d-models/dart-f4eadd9979de4378aa523228c992d5eb#download

Globe: https://free3d.com/3d-model/balloon-darts-v1--421125.html

Mole table: https://sketchfab.com/3d-models/whack-em-allwhack-a-mole-machine-66e181857962436ab4b7a806618cab61

Source: https://www.turbosquid.com/3d-models/3d-model-water-fountain-1387533

Texture Links:

Water: https://www.shutterstock.com/es/image-photo/dark-blue-water-texture-close-2563388969

Gumball Frame: https://www.youtube.com/watch?v=Mp_4noPC7Qc

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

Mole table:
https://www.youtube.com/watch?v=OWn0XgBJuLI&ab_channel=Badabun

Popcorn: https://es.vecteezy.com/foto/5273536-palomitas-de-maiz-sobre-un-fondo-amarillo-como-imagen-de-fondo

Stone: https://unsplash.com/es/fotos/un-primer-plano-de-una-roca-con-pequenas-rocas-en-ella-wuoRhml_xNU