UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTY OF ENGINEERING

DIVISIÓN DE INGENIERÍA ELÉCTRICA

COMPUTER ENGINEERING

COMPUTER GRAPHICS AND HUMAN-COMPUTER INTERACTION

TECHNICAL MANUAL

"FAIR"

STUDENT:

319283064

LABORATORY GROUP: 03

THEORY GROUP: 05

SEMESTER 2025-2

DEADLINE: May 20, 2025

GRADE:_____

Content

Objective

Develop an interactive skill game fair inspired by the animated worlds of Mario Bros, Snoopy and The Amazing World of Gumball.

The environment will include games such as axe throwing, bowling, dice throwing, batting cage, darts, and "hit the mole."

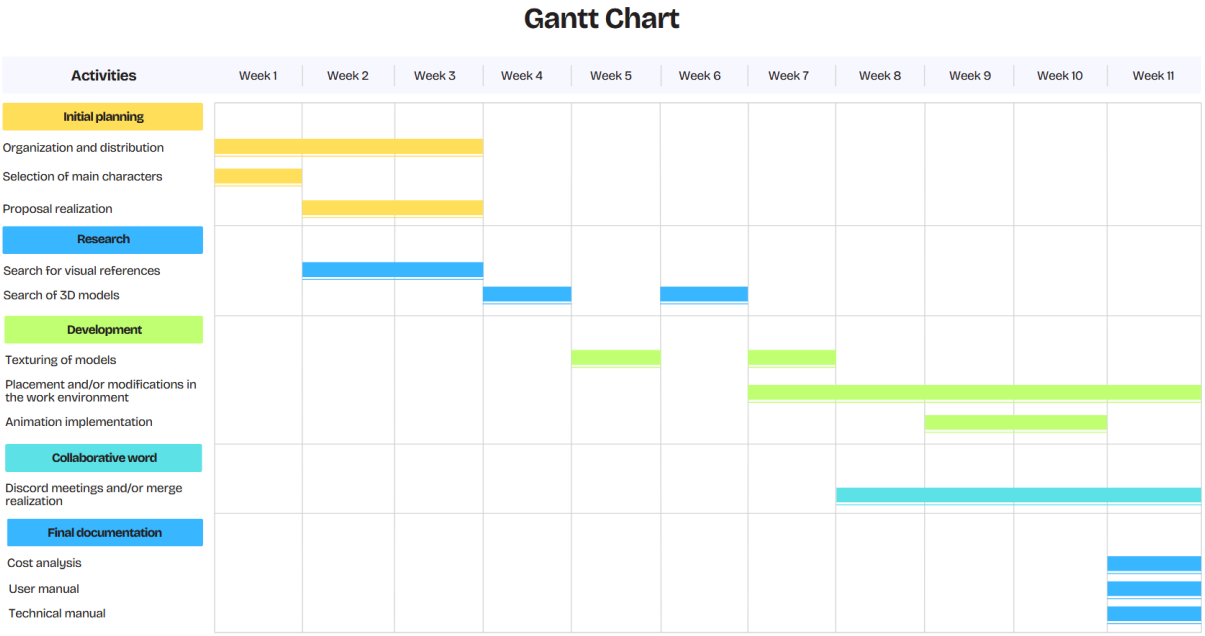
Typical decorative elements of a fair will be incorporated, both in the games and in the environment in general, in order to offer an experience that combines the use of 3D modeling for scenarios, objects and characters, along with texturing inspired by the different worlds. In addition, a system of lights will be integrated to improve the atmosphere, and cameras with different angles that will allow the user to go through and interact with the fair.

For the development of the project, tools such as OpenGL and C++ will be used for programming and implementation, 3ds Max for modeling and texturing, Lightroom and GIMP for editing and texture creation.

Scope

1. **Modeling**: Carry out the modeling of the facades of each game, maintaining a symmetrical structure that unifies the general style of the fair, decorate each game with elements from the universes of Mario Bros, Snoopy and The Amazing World of Gumball, and set the environment with models such as trees, benches, poles, characters, etc.
2. **Texturing**: Apply textures to each model using image editing tools such as GIMP or Lightroom by adjusting their dimensions or editing them as needed.
3. **Animation**: Implement interactive animations in each of the games, as well as custom animations for characters and decorative elements.
4. **Lights**: Develop a system of lights that allow illuminating the exterior of the fair, the interior of the stalls and simulating sunlight, incorporating ambient light. This system must include the switching on and off of automatic lights according to the day and night cycle, as well as the control of the lights inside the stalls by using the keypad.
5. **Cameras**: Develop three types of cameras to visualize the fair: an aerial one to have an overview, a third-person camera to interact with the fair and a front camera focused on games for a wide view.
6. **User interaction**: Allow the user to freely go thrugh the fair from a third-person view, as well as activate games and animations through the use of the keyboard.

Schedule



Cost

To facilitate the understanding of the costs associated with this project, a breakdown of the different types of expenses identified is presented. Although the project was developed by a team of three people, the costs presented here correspond to the work carried out by a single person.

- **Human resources cost**

The development of the project was carried out by a single person with an intermediate level of experience in the C++ programming language. The total workload was 100 hours.

Considering an average of $110.00 MXN per hour, the total cost for human resources is calculated as follows:

Human Resources: $11,000.00 MXN

- **Fixed costs**

Fixed costs correspond to those expenses that remain constant during the project, regardless of the amount of work performed. In this case, as it was a person working from home, the following were considered:

- **Electricity:** An Asus TUF Gaming A15 laptop was used, with an estimated monthly consumption of $430.00 MXN. For a project duration of 2.75 months, the total is: $1,182.50 MXN

- **Internet:** The Totalplay internet service was used, with an estimated monthly cost of $420.00 MXN. For 2.75 months, you have: $1,155.00 MXN

- **Equipment depreciation:** The equipment used has an approximate cost of $26,000.00 MXN and an estimated useful life of 60 months, which represents a monthly depreciation of $433.33 MXN. For 2.75 months: $1,191.67 MXN

Fixed costs: $3,529.17 MXN

- **Variable costs**

Variable costs refer to resources that depend on the execution of the project.

- **Software:** Free tools such as Visual Studio 2022, GitHub, 3ds Max, GIMP, and Lightroom were used.

Variable costs: $0.00 MXN

- **Net Cost**

Adding the three types of costs previously described, we obtain:

- Human Resources: $11,000.00
- Fixed Costs: $3,529.17
- Variable Costs: $0.00

Net Cost: $14,529.17 MXN

- **Cost with profit margin**

A profit margin of 30% was considered, resulting in:

- Estimated Profit: $4,358.75 MXN

Cost with profit margin: $18,887.92 MXN

- **Final Cost**

The corresponding taxes are added to the subtotal with profit.

- **IVA (16%):** $3,022.07 MXN

Final cost: $21,909.99 MXN

The final amount of $21,909.99 MXN per person represents a price that covers all operating costs, resources used, individual professional remuneration and corresponding tax obligations. This value ensures the economic viability of the project and fair compensation for the work contributed by each member.

Documentation

Main File

- **hacha(...)**

Renders all models corresponding to the axe throwing station. It is responsible for applying the necessary transformations and drawing objects such as the axe, the support and decorative elements from the immediate surroundings of the stall on the screen.

Parameters:

glm::mat4& model: model matrix used for transformations.

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

std::vector<Model*>& objetosHacha: list of pointers to the models of the post.

- **dados(...)**

Draws the elements of the dice throwing stand, including the table, dice, decorative mushrooms, and themed pictures. It also manages their transformation and placement within the main scene.

Parameters:

glm::mat4& model: transformation matrix.

GLuint uniformModel: location of the transformation uniform.

std::vector<Model*>& objetosDados: models related to the dice game.

- **ambientacion(...)**

Renders 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*>& objetosAmbientacion: static scenario models.

- **comida(...)**

Draws 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*>& objetosComida: models that represent food and stalls.

- **NPCs (...)**

Rendesr 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*>& personajesNPCs: list of characters to render.

- **zonaFotos(...)**

Renders 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*>&objetosZonaFotos: models that make up the scenario.

float deltaTime: time between frames, used for animations.

- **arco(...)**

Draws the entrance arch to the fair, including the doors and the animated sign. This feature applies texture offsets (UV offset) to animate the writing of the sign.

Parameters:

glm::mat4& model: transformation matrix.

GLuint uniformModel: uniform of the model.

GLuint uniformTextureOffset – uniform offset for UV animation.

std::vector<Model*>& objetosArco: structural models of the arch.

float deltaTime: delta time for animation.

Texture & letreroTexture: texture with sign characters.

std::vector<Mesh*>& meshList: list of meshes used here to render the characters in the sign.

- **actualizarHacha(float velocidad)**

Updates the rotation and displacement of the thrown axe model, simulating its flight path. A delta (velocity) value is used to achieve smooth animation over time.

Parameter:

velocidad: Scaled time (Delta Time) that regulates the speed of the animation.

- **actualizarDados(float velocidad)**

It manages the rotation and bounce of the dice after it is rolled. It allows a basic physical simulation of the movement of the dice on the game table.

Parameter:

velocidad: Value proportional to Delta Time, controls the animation time.

- **lanzarHacha()**

Activates the axe model's throw animation, starting its trajectory towards the target. It is usually activated with the T key.

- **lanzarDados()**

It simulates the throw of the die, activating its rotation and movement with bouncing. It usually responds to the U key.

Separate files

- **Dados.cpp**

Variables

glm::vec3 postDado1, postDado2: current positions of each dice.

glm::vec3 rotDado1, rotDado2: cumulative rotations of the dice.

float velY: vertical speed to simulate gravity.

Int rebotes: number of bounces remaining.

bool lanzandoDados: if the cast animation is active.

Functions

void dados(glm::mat4 model, GLuint uniformModel, std::vector<Model*> objetosDados): Renders the entire environment of the dice stand.

void renderDado(glm::mat4 model, GLuint uniformModel, Model& dado, glm::vec3 posicion, glm::vec3 rotacion): Draws a die with specific position and rotation.

void renderStand3(glm::mat4 model, GLuint uniformModel, Model& stand): Rendes the stall model.

void renderMesaDado(glm::mat4 model, GLuint uniformModel, Model& mesa, glm::vec3 posicion): Renders a table at the specified position.

void renderHongo(glm::mat4 model, GLuint uniformModel, Model& hongo, glm::vec3 posicion, int rotY): Renders a decorative mushroom with Y rotation.

void renderCuadro(glm::mat4 model, GLuint uniformModel, Model& cuadro, glm::vec3 posicion, float escala): Renders a decorative painting with scaling.

void actualizarDados(float deltaTime): Updates the animation of moving dice.

Void lanzarDados(): Starts the roll animation of the dice.

- **Hacha.cpp**

Variables

float avanceHacha: forward movement of the axe.

float rotacionHacha: rotation during flight.

Bool lanzandoHacha: If the axe is in animation.

Bool impactoHacha: if it has hit.

float tiempoEsperaHacha: time before restarting.

Functions

Void hacha(glm::mat4 model, GLuint uniformModel, std::vector<Model*> objetosHacha): Renders the entire axe post.

void renderStand(glm::mat4 model, GLuint uniformModel, Model& stand, Model& reja, Model& tabla, glm::vec3 posicion): Renders the stand and its decorations.

void renderHachas(glm::mat4 model, GLuint uniformModel, Model& axe, glm::vec3 posicion): Renders static and animated axes.

void renderObjetivos(glm::mat4 model, GLuint uniformModel, Model& objetivo, glm::vec3 posicion): Renders the game's targets.

void lanzarHacha(): Activates the animated axe throw.

void actualizarHacha(float deltaTime): Updates axe position and rotation in flight.

- **ZonaFotos.cpp**

Variables

float vuelaSnoopy, velocidadSnoopy, amplitud: Snoopy's circular flight control.

float snoopyY, snoopyZ, movSnoopy, movOffsetSnoopy: desplazamientos.

bool avanzaSnoopy: controls direction of movement.

Functions

void zonaFotos(glm::mat4 model, GLuint uniformModel, std::vector<Model*> objetosZonaFotos, float deltaTime): Renders and animates characters in the photo zone.

void renderSnoopyAviador(glm::mat4 model, GLuint uniformModel, std::vector<Model*> objetosZonaFotos): Renders and animates the character Snoopy by simulating that he is flying as an aviator. It moves in a circular path (on the Y-Z axes) while moving laterally on the X-axis, making a round trip. The animation combines smooth rotations and periodic scrolling.

- **NPCs.cpp**

void NPCs(glm::mat4 model, GLuint uniformModel, std::vector<Model*> personajesNPCs): Renders all NPCs in their positions.

Each one renders the corresponding character with position and rotation.

void renderSnoopy(glm::mat4 model, GLuint uniformModel, Model& personaje, glm::vec3 position, int grados)

void renderPeppermint(glm::mat4 model, GLuint uniformModel, Model& personaje, glm::vec3 position, int grados)

void renderCharlie(glm::mat4 model, GLuint uniformModel, Model& personaje, glm::vec3 position, int grados)

- **Tierra.cpp**

Functions

void tierra(glm::mat4 model, GLuint uniformModel, Texture& tierra, std::vector<Mesh*> meshList): Renderiza todo el terreno general con textura.

void renderTierra(glm::mat4 model, GLuint uniformModel, Texture& tierra, Mesh& floor, glm::vec3 posicion, glm::vec3 escala): Renders a block of textured terrain in given position and size.

- **Ambientación.cpp**

Functions

void ambientacion(glm::mat4 model, GLuint uniformModel, std::vector<Model*> objetosAmbientacion): Renders trees, benches, luminaires, and other decorative elements.

Each function positions and draws an object from the environment

void renderLuminaria1(glm::mat4 model, GLuint uniformModel, Model& modelo, glm::vec3 posicion)

void renderBote1(glm::mat4 model, GLuint uniformModel, Model& modelo, glm::vec3 posicion, int grados)

void renderBote2(glm::mat4 model, GLuint uniformModel, Model& modelo, glm::vec3 posicion, int grados)

- **Arco.cpp**

Variables

std::vector<glm::vec2> mensajeLetrero: UV coordinates of letters.

float tiempoLetrero: time control to display letters.

int posicionLetrero: current letter on the screen.

glm::vec2 toffset: Animated UV scrolling.

float anguloPuerta, velocidadPuerta: opening doors.

Bool abrirPuerta: opening status.

Functions

void arco(glm::mat4 model, GLuint uniformModel, GLuint uniformTextureOffset, std::vector<Model*> objetosArco, float deltaTime, Texture& letrero, std::vector<Mesh*> meshList): Renders the arc at the entrance with door animation and sign.

void inicializarMensajeLetrero(): Initializes the coordinates to animate text on the sign.

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.

- **CreateDado()**

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: intensidad difusa.

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: intensidad difusa.

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 y: 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.

Conclusions

This project represented an opportunity to apply the knowledge acquired in the field of Computer Graphics and Human Computer Interaction. It was possible to combine modeling, texturing, lighting, cameras and interaction in a complex environment such as a game fair, inspired by worlds such as Mario Bros, Snoopy and Gumball. Although the biggest challenge was the complexity of the project, especially due to the requirements and time, it was gratifying to see how everything took shape.

During the development, different aspects were worked on such as the design of the stalls, the setting of the environment with themed decorations, the implementation of an automatic lighting system with a day-night cycle, the use of cameras that offer different perspectives, the animations for each of the games and the main characters, among many other things. Keyboard interaction was also added, allowing the user to interact with the environment.

This project allowed us to apply what we learned in class in a creative and technical way at the same time. Although it was demanding in complexity and time, it was an enriching and entertaining experience, which also encouraged collaborative work and communication.

Model Links:

Casa de Snoopy: https://sketchfab.com/3d-models/snoopy-7ca97398a08c4e9fb9a216c6982564dc

Algodon de azucar: https://www.cgtrader.com/items/4617534/download-page

Tabla: 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

Garbage 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

Peppermint: https://sketchfab.com/3d-models/wii-u-the-peanuts-movie-snoopys-grand-adventur-dbe36b120be94397b0327c22efbcfe75

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

Charlie Brown: https://sketchfab.com/3d-models/wii-u-the-peanuts-movie-charlie-brown-9acef7efdb42429ab0f79be59c193d41

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

Dice table: https://free3d.com/es/modelo-3d/craps-table-v1--845885.html

Axe mesh: https://free3d.com/3d-model/fence-86772.html

Axe: https://free3d.com/3d-model/ax-92952.html

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

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

Axe Target: https://sketchfab.com/3d-models/archery-target-bfa271fc02a94640886a9dfc2cd6ad4a

Links Texturas:

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

Dice:

https://es.ssbwiki.com/wiki/Huevo_de_Yoshi

https://mario.fandom.com/es/wiki/Cheep_Cheep

https://cl.pinterest.com/pin/279645458107164172/

https://mario.fandom.com/es/wiki/Bill_Bala

https://mario.fandom.com/es/wiki/Flor_de_Fuego

https://mario.fandom.com/es/wiki/Champi%C3%B1%C3%B3n

https://www.pngwing.com/es/free-png-zhaxf

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

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