

“They’re Watching”

GP Project – Reti Antonio - 30232

1. Theme Presentation

The project is a *survival horror experience* heavily inspired by the *"Five Nights at Freddy's" franchise's* design and universe. The player, a child, is trapped in a dark bathroom, only lit by a few candles, and must survive the attacks of two entities: *Nightmare Foxy* and *Nightmare Bonnie*. The gameplay revolves around time management and situational awareness, using *light to repel the monsters* before they get to you.

2. Scenario

2.1. Description of the Scene and Objects

- *The environment*: a claustrophobic bathroom featuring high-resolution textures, torn apart wall covering, paintings and two windows, one of them used as an entry point by one of the entities.
- *The monsters (animatronics)*: High-detail 3D models of *Nightmare Foxy* (at the door) and *Nightmare Bonnie* (at the window), designed by Scott Cawthon.
- *Lighting Elements*: *Three static candles* on a vanity, *a flashlight* that our protagonist uses to repel his nightmares, and *two thematic point lights* used to enhance each monster's theme (red and blue)
- *Creaky Door*: A dynamic object that opens when *Foxy* is present, making a loud noise to signal its presence.

2.2. Functionalities

- *Dynamic Lighting System*: Includes a flashlight with flickering effects that intensifies based on the animatronic's irritation.
- *AI Logic*: Animatronics appear at random intervals and require the player to shine a light on them for a specific duration to force a retreat (*foxy is getting irritated more easily, whereas bonnie is very stubborn*).
- *Camera Transition System*: Smooth interpolation between three main camera states (Spawn, Door, and Window) to simulate player movement.

- *Sound Cues*: loud noises to alert the player that an enemy is being irritated, is leaving or is arriving at the entry point.

3. Implementation Details

3.1. Functions and Algorithms

3.1.1. Shadow Mapping with Texture Arrays

Instead of using standard 2D textures, the program uses a *GL_TEXTURE_2D_ARRAY* to store the depth information for five different light sources in a single *4096x4096* texture unit.

- *The Layer Logic*: Layer 0 is reserved for the directional moonlight (*directional light*), Layer 1 for the flashlight (*spot light*), and Layers 2-4 for the individual candles (*point lights*). Each candle is orientated in another direction to get a *cubemap type of look* while still *maintaining high performance*.
- *Depth Pass*: During the shadow pass, the scene is rendered five times. For each pass, we bind a different layer of the array using the *glFramebufferTextureLayer()* function.
- *Shader Sampling*: In the fragment shader, the program uses `texture(shadowMapArray, vec3(projCoords.xy, layerIndex))` to retrieve the depth value specific to the light being calculated.

3.1.2. Smooth Camera Transitions

To avoid jumps between positions, the program uses a *linear interpolation* algorithm to move the camera.

- *The Algorithm*: We use `glm::mix(start, end, t)`, where *t* is the normalized transition time.
- *Easing*: To make the movement feel more natural, the transition uses a "*Smoothstep*" function to the *t* variable: $t = t * t * (3.0f - 2.0f * t)$.

3.2. Graphical Model

The core of the visual system is the *Blinn-Phong* model, which calculates the light intensity at each fragment by summing three distinct components: *Ambient*, *Diffuse*, and *Specular*.

The *total color of a fragment* is calculated using the following:

- *Ambient*: A low-intensity constant light that ensures unlit areas are not pitch black, though kept minimal (approx. 0.01 in game) to maintain a horror aesthetic.
- *Diffuse*: Calculated using the dot product between the surface normal and the light direction, simulating how light scatters off matte surfaces like the bathroom wallpaper.
- *Specular*: Uses the *halfway vector (H)* between the light source and the view direction to create sharp highlights.

The graphical model supports three distinct types of light sources simultaneously:

- *Directional Light (Moonlight)*: Simulates a distant light source where rays are parallel, used to provide a faint blue tint to the entire scene.
- *Point Lights (Candles & Rim Lights)*: These lights use *attenuation* based on distance (constant, linear, quadratic) to create localized pools of light.
- *Spotlight (Flashlight)*: A specialized light source attached to the camera that uses *a dual-cone system (inner and outer cutoff)* to create soft-edged illumination.

To achieve realistic depth and occlusion, the model uses *Shadow Mapping (Texture Array)*.

The graphical model follows the standard *MVP (Model-View-Projection)* pipeline:

- *Model Matrix*: Handles individual object positions, such as moving the animatronics and the door.
- *View Matrix*: Derived from the `gps::Camera` class, transforming world coordinates into camera-space.
- *Projection Matrix*: Uses a *Perspective Projection* to create a sense of depth and 3D space.

3.3. Data Structures

The *AI logic* is stored in a central structure that handles the *light irritation mechanic* and *its parameters*.

- *lightIrritation*: A float that increases when the flashlight is active and correctly aligned (alignment > 0.9) with the enemy.
- *irritationThreshold*: The limit after which the animatronic is forced to retreat and a sound cue is triggered.

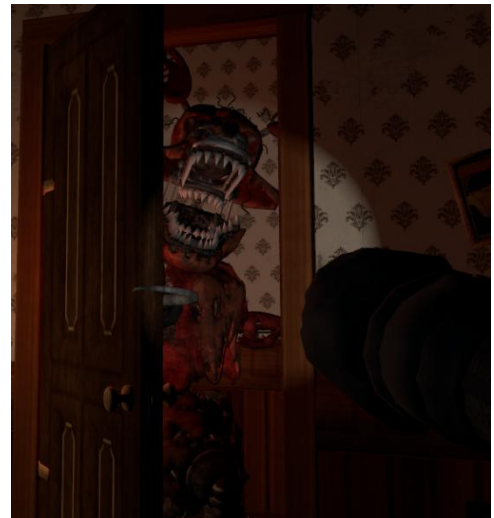
3.4. Class Hierarchy

- *gps::Window*: A wrapper for GLFW and GLEW. It initializes the graphical window, manages the OpenGL context, and polls for hardware input (keyboard/mouse).
- *gps::Shader*: Manages the lifecycle of GLSL programs. It loads, compiles, and links the .vert and .frag files and provides methods to send uniform data (like matrices and light colors) to the GPU.
- *gps::Model3D*: The high-level container for 3D assets. It uses tiny_obj_loader to parse .obj files and stores a collection of Mesh objects.
- *gps::Mesh*: The low-level geometry class. It handles the generation of *VAOs (Vertex Array Objects)*, *VBOs (Vertex Buffer Objects)*, and *EBOs (Element Buffer Objects)* to send vertex data to the GPU.
- *gps::Camera*: Encapsulates the mathematical logic for the *View Matrix*. It calculates the camera's orientation vectors (Front, Right, Up) based on Euler angles (Pitch and Yaw).
- *tiny_obj_loader*: A header-only library used within Model3D to parse geometric data from .obj and .mtl files.
- *stb_image*: Used for decoding image files (like .jpg or .png) into raw pixel data for OpenGL textures.
- *basic.vert / basic.frag*: The main rendering pipeline. It implements the *Blinn-Phong* lighting model and performs shadow sampling from the texture array.
- *depthMap.vert / depthMap.frag*: A specialized, lightweight pipeline used during the shadow pass to record the scene's depth from the perspective of each light source.
- *main*: orchestrates the *game mechanics, AI behavior and controls the rendering core*.

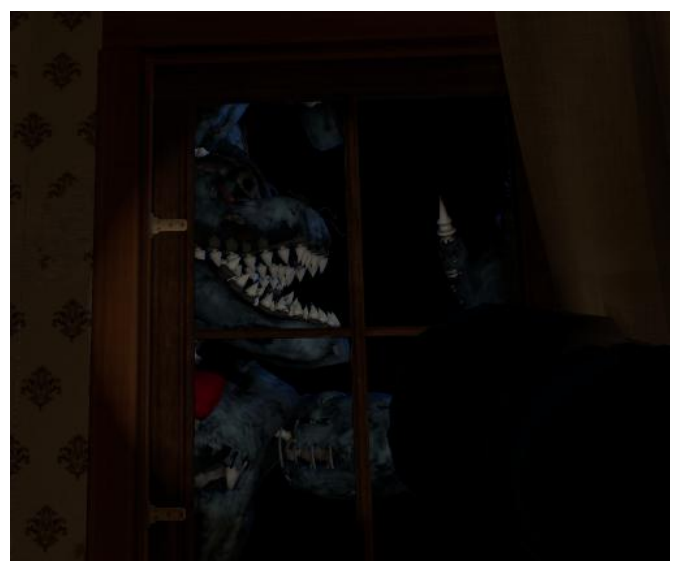
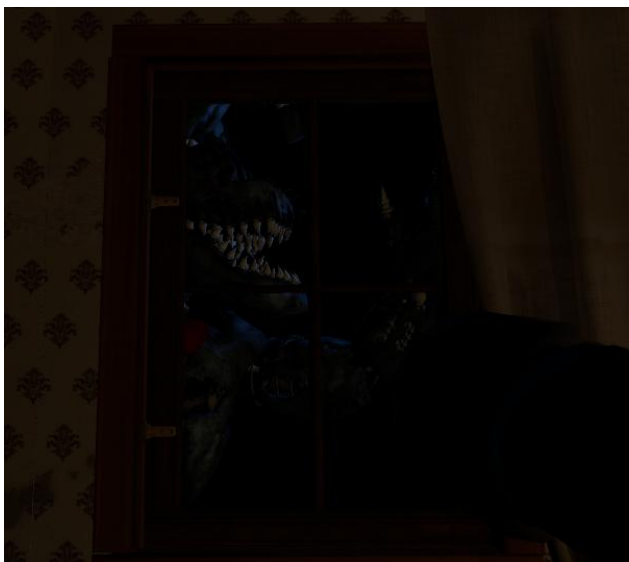
4. Control Scheme

This section serves as a *functional guide* for interacting with the application:

Foxy (The Door): When you hear a *creaky door* sound, Foxy has appeared. Use the '**2**' key to look at the door and keep your flashlight on him until the door slams shut.



Bonnie (The Window): Listen for a *scraping sound* against the glass. Use the '**3**' key to check the window. Shining the light directly on Bonnie will eventually force him to retreat.



To view the underlying geometry, the following keys toggle different OpenGL polygon modes:

- *J Key (Solid/Smooth)*: The default rendering mode. Surfaces are fully shaded using the *Blinn-Phong model*.
- *K Key (Wireframe)*: Renders only the edges of the triangles.
- *L Key (Polygonal/Point)*: Renders only the vertices of the 3D models as points.

5. Conclusions and Future Developments

The "*They're Watching*" project successfully demonstrates the integration of *OpenGL techniques* to create a survival horror experience similar to "*Five Nights at Freddy's*" franchise. By combining dynamic lighting, spatial audio, and randomized AI behavior, the application achieves a high degree of tension and player engagement.

5.1. Future Developments

In this section, I would like to say that working on this project was a *breath of fresh air*, because I could choose my *own theme* and do something creative built around it. For future developments, I would *gamefy the project completely*, from jumpscare and different menus to battery life and more enemies with many more unique nights (levels).

6. References

- *SFML* Documentation - <https://www.sfml-dev.org/documentation/> - 2D and 3D positional audio and sound cues
- *Laboratories @ TUCN*
- Learn OpenGL - <https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping> - *Shadow Mapping*