

Pale Watcher

PTVC Submission 2

Group/Game Name: Pale Watcher

Students:

- Dominik Gramegna, 12224269
- Marcel Kozonits, 12122382

Genre: First-person Survival/Horror

Goal: Explore the dark forest, find the key, turn the power on return to the house to escape the mysterious creature hunting you down.

The implementation for Submission 2 includes advanced gameplay mechanics with complex object loading, with the “Watcher” which tries to hunt you down and kill you, while you, the player, are trying to find a key in the spooky woods to turn on the power again, return to your house and enter it to be safe again. Different end screens for surviving and for dying are added. The player has an inventory to keep track of picked up and util items (just the key for us). Some of the objects found in the game are interactable.

Gameplay Implementation (according to defined Features/Effects)

Feature	Description
3D Geometry	<p>Many different scene objects are loaded from a .obj file - these are some (partly low-poly) free resources from the internet. We imported all of these as Blender projects and exported the triangulated .obj with all needed parameters (Vertices, normals, uv, ...) we need.</p> <p>In the Geometry.cpp/.h files, we implemented a loadObj function, which parses the important data from the .obj file and puts the data in the Geometry object.</p> <p>In the Assets.cpp/.h files, all provided models are collected and loaded when needed for the scene.</p>
Playable	<p>The game is currently playable by controlling the main player (1st person view). The controls are listed below in the document. The player can walk around and jump in the scene, with collision and gravity enabled. There is a Debug Camera when activating F3 (Debug Mode), then pressing P.</p>
Min. 60 FPS and	<p>Currently, on the Nvidia RTX 2060 Super, the gameplay on 1920x1080 is</p>

Framerate Independence	always above the minimum requirement, different settings like view frustum culling and the fog improve this significantly.
Win/Lose Condition	<p>Win: The player needs to find the key in the forest, to be able to turn on the power again, return to the house and enter it. (The current task is shown in the lower left corner all the time)</p> <p>Lose: The watcher is always trying to kill the player, but is not allowed to move towards him if he is looked at. If the watcher gets too close to the player, he kills him.</p>
Intuitive Controls	Intuitive Controls are implemented, the player is moving with WASD, which is typical for computer games, jumps with Space-Bar (and sprints with shift when Debug Mode (F3) enabled). Interaction with the scene is possible with left mouse click, with which the player can pick up items or interact with them (key pickup, power turn on, enter house).
Intuitive Camera	For playing the game, the camera is always attached to the player. For Debug purposes, the default Orbital camera from the Framework is in place to freely move in the world (Press F3 for Debug Mode > then P)
Illumination Model	<p>The Scene has a directional light source from the sky (moon shining), and the player has a point light to simulate a flashlight. When the player interacts with items, they emit a short light impulse which is rendered in the game. Also, when the power is turned on, street lamps start to emit light (near the house and power plant)</p> <p>Each object has model properties assigned to modify reflection, and every model has correct normal vectors (Exported with Blender).</p>
Textures	<p>Textures are loaded with a .png loader (Can be found in Texture.cpp/.h) and are attached correctly to all game objects currently.</p> <p>There is also a cubemap texture for the sky and a terrain texture for the ground.</p> <p>Mipmapping and Trilinear Filtering is enabled for png-Textures.</p>
Moving Objects	The watcher is constantly moving towards the player, except when he is in the player's view frustum or when he has already been killed.
Documentation	This document summarizes the most important aspects of our game.
Adjustable Parameters	Screen resolution, FPS limit and Fullscreen can be changed in the window.ini file before starting the game (without recompiling)
<u>Optional:</u> Advanced Gameplay	Watcher is constantly moving the player, the player has to complete achievements to complete the game, like the key finding, turning power on again and returning to the house, while not being killed by the watcher. Music is also added to intensify the horror experience with distortion and jumpscares. Three different difficulties of the watcher are

	<p>given:</p> <ul style="list-style-type: none"> ● Normal: Watcher is hunting you, but only when you are in range and looking away. ● Hard: Watcher is faster, and he can also head to you to a certain extent when you look at him. ● Extreme: Same as hard & when being in range of Watcher and looking at him at least 1,5s, you get killed instantly. <p><i>* In Range means when your Screen is somehow distorted because Watcher is near.</i></p>
<u>Optional:</u> Collision Detection	PhysX is integrated into the game and enabled for the Terrain and Player. The scene objects also do have enabled physics, which makes it realistically to not be able to run through a house or tree for example.
<u>Optional:</u> Scripting Language Integration	<p>The logic of the watcher, which hunts down the player, is scripted separately with LUA / luaaa. There are three difficulties for the Watcher already included (see Optional: Advanced Gameplay), but you could modify these scripts at any time before starting the game.</p> <p>While playing the game, you can also switch the difficulty in F3 Debug Mode (by using F4-F6).</p>
<u>Optional:</u> View-Frustum Culling	<p>This can be enabled with F8 and is implemented with the files Frustum.cpp/.h and with appropriate methods in Geometry and checks in Main.cpp.</p> <p>In the Debug Info besides F8 Control, you see how many Objects are currently shown vs. how many objects exist in the scene.</p>
<u>Optional:</u> Heads-Up Display	<p>There is a HUD for displaying debug text and controls, as well as a crosshair in the middle of the screen.</p> <p>The Text is loaded by .fnt and .png files and the .fnt is parsed with all information about positioning, spacing etc. of each character. This information is used to render the correct UV coordinates in the HUD to display the correct letter (text.* shaders and HUD.cpp/.h files)</p> <p>A simple view for the next task of the player and the current inventory is also displayed.</p> <p>With F3, the HUD display and the “Debug-Mode” can be toggled, this enabled the function of some other debug controls, e.g. Frustum Culling or Wireframe mode.</p>
<u>Optional:</u> Camera Object Tracking	When the player gets killed by the watcher, the camera changes to the watcher, which still gets nearer to the player until the end screen is shown and til there, the watcher is being tracked by the camera

	It is rather hard to see -> it is best to try that one out with Difficulty Hard or Extreme as Watcher has movement logic in front of the player there.
Effects: Terrain: Tessellation from Height Map	<p>We implemented the terrain shader files (terrain.*) for a tessellated terrain, and parts of implementation can be found in HeightmapTerrain.cpp/.h and TerrainShader.cpp/.h.</p> <p>The provided heightmap is in format .png and was generated on a website, which allows to generate different images with perlin noise.</p> <p>https://learnopengl.com/Guest-Articles/2021/Tessellation/Tessellation</p>
Effects: Texturing: Procedural Texture	At the end of the map, is a brick wall, which is completely created at run time. This brick wall uses noise and fractal functions to calculate the coloring to make it look like old bricks.
Effects: Post-Processing Contours via Edge Detection	<p>This was implemented with FrameBuffer.cpp/FrameBuffer.h files and PostProcessor.cpp/.h files. Additionally, the texture and postprocess shaders exist for this.</p> <p>The outlines are just present for the interactable objects, therefore just the key (which is the only thing which can be put in the inventory), the power plant and the house have the outline, but only if it is really interactable (example: power plant just when key found and not turned on yet)</p> <p>Parts of https://learnopengl.com/Advanced-OpenGL/Framebuffer</p>

Additional Game Features (not defined in Requirements)

The following feature also exist in the game currently:

- Rendering a Coordinate-Grid or model normals: can be enabled, useful for debugging coordinated of objects
- Skybox with Cubemap-Texture for a Night sky (with a simulated moon glow)
- Switch Camera (Orbital simple camera or player)
- Show object wireframes, terrain wireframes, toggle backface culling
- Light lamps just turn on the light after the power is turned on.
- Sound effects like a constant background noise, different walking and running noises, random jumpscare and jumpscare when the player gets too close to the watcher.
- Scene gets distorted when the player is near the watcher.
- Different end screens for entering the house or getting killed by the watcher.
- Loading and start screen with difficulty selection

Short gameplay summary:

- Select a difficulty and click enter. You start at the house.
- Click F3 to enable the HUD displaying (shows you your position).
- Go to position x: -40, z: -60 and in the middle of the rock formation you can find the key hovering, left mouse click on it to pick it up.
- Go to position x: -40, z: 60 and click on the power plant to turn it on.
- Go back to the house and click on it to enter it.
- Try to not get killed by the watcher in the meantime.

All controls:

- **Player Controls:**
 - WASD: Move
 - Space: Jump
 - Mouse: Look around
 - R: Reset Position
 - F3: Toggle this HUD
- **Audio:**
 - Volume: ,/. to adjust
 - M: Toggle music
 - N: Toggle sound effects
- **Debug Controls (when F3 Debug is enabled):**
 - Shift: Run
 - P: Switch camera
 - F1: Global Wireframe
 - F2: Culling
 - F8: Frustum Culling
 - H: Terrain Wireframe
 - L: Grid
 - U: Normals
 - O: Outlines
 - I: Toggle Distortion
 - K: Test Jumpscare
 - J: Move Watcher
 - F: Toggle Fog
 - G / Shift+G: Fog Distance
 - B / Shift+B: Fog Start
 - V / Shift+V: Skybox Fog
 - F4: Load Normal Watcher Script
 - F5: Load Hard Watcher Script

F6: Load Extreme Watcher Script

F9: Reload Current Watcher Script

Additional Libraries used:

- We included stb_image.h and stb_image_write.h files to simplify loading .png textures (see https://github.com/nothings/stb/blob/master/stb_image.h).
- We also used physx for the gravity and collision management with other objects in the scene.
- OpenAL (<https://openal.org/downloads/>) was used for the audio system.