

# Implementation Overview

## Resource Loading & Object Construction

- **GLTF Loader**
  - Utilizes the tinyglTF library to load “.gltf” and “.glb” files at startup.
  - Extracts mesh geometry (positions, normals, UVs), material definitions, and node transforms.
- **RenderObject Management**
  - All RenderObject instances are stored in a central std::vector and go through the render pass.
  - Each frame, the game loop iterates through visible objects and issues draw calls using their assigned shader and world-space transform.

## Physics & Collision

- **PhysX Integration**
  - Integrated NVIDIA PhysX (v5.0) for rigid-body dynamics and character controller.
  - The player is a PxCapsuleController; collisions with static and dynamic meshes are handled via PhysX’s querying/filtering.
  - A custom FilterShader determines collision responses between different object types (player vs. loot, player vs. world geometry, remote vs. pressure plate, etc.).
  - Gravity, jumping forces, and simple rigid-body interactions use PhysX.

## Worlds & Shaders

- **Dual-World System**
  - Implemented two “parallel” worlds:
    1. **Bloomy World**
    2. **Dither World**
  - Pressing **Right Mouse Button (LMB)** triggers a world switch:
    - Player movement is temporarily disabled.
    - A full-screen quad fade animation interpolates between the two worlds.

- Internally, an enum flags which objects should be rendered.
- Some objects exist in both worlds, others only in one.

- **Shader Pipeline**

- On startup, all GLSL shaders (vertex + fragment) are compiled and linked.
- For each object, the engine selects:
  - **World-specific material** (e.g., bloom post-process for Bloomy world; dither post-process for Dither world).
  - **Lighting model** (Phong/Blinn-Phong with shadow map integration).
- Render passes:
  - **Shadow pass** (generate shadow map from sun/spot light)
  - **Base Pass** (render scene with lighting etc)

## Misc

- **Fog**

- **Linear Depth Fog** (in Bloomy world): computed in fragment shader

- **HUD & GUI (ImGui)**

- Integrated for all in-game menus & HUD.
- **Main Menu**: initial screen on startup, with “Start Game” & “Quit.”
- **Pause Menu**: toggled via **Esc**, lets player Resume / Restart / Quit.
- **Game Over Menu**: shown if player health  $\leq 0$ .
- **Win Screen**: displayed once the remote is thrown into the wormhole/pit.
- **Controls Guide**: toggled via **Q**, displays keybindings.
- **HUD Bars**:
  - **Health Bar** (current HP / max HP)
  - **Stamina Bar** (depletes on sprint, refills over time)

- **Remote Charge Bar** (only shown after picking up remote; depletes in Dither world, recharges at puddles)

## Feature List

GLTF Loader
Linear Depth Fog
Procedural Sky Noise
Gradient Skybox
Menus
HUD
Vertex Shader Animation
Loot Pickup Animation
Procedural Textures
Physics System
Dither Shader, Texture Shader, etc.
Shadow Mapping with PCF
Player State
First-Person Camera with Head-Bob
Collision
World Switching
Object Picking
Hazards
Remote Throwing
Pressure-Plate Trigger
Normal Mapping
Bloom/Glow Post-Process
Ambient & SFX Audio

## Effects Implemented

- **Shadow Map + PCF:** Soft shadows with stylized dither edge.
- **Vertex Shader Animation:** noise ripples for puddles; GPU-only floating/rotation for loot.
- **Procedural Texture:** Animated noise for puddle liquid; noise for rough floors.

- **Simple Normal Mapping:** Standard TBN-based normal mapping for detailed surface lighting.
- **Bloom/Glow:** MRT (HDR + brightness), blur bright areas via ping-pong FBO, composite blurred brightness with HDR (Dither world: puddles, Bloomy world: key objects, dithered water)
- **Dither Shader:** Ordered Bayer-matrix dithering for the Dither world (and underwater transitions).

## Gameplay Description

The main gameplay sequence starts with finding and picking up the remote. Then the remote indication appears, and it's possible to switch to the dither world with LMB. In the dither world, the remote loses charge and the player consistently takes damage. They need to run to a puddle, where the remote recharges, the damage stops, and the player receives a small healing boost. When switching to the bloomy world, health regenerates. The dither world is taking over the bloomy world in the form of a dither floor moving towards the player, causing significant damage. Scattered over both worlds, various platforms are in the sky, on closer inspection it is clear that the player has to jump on the platforms and switch worlds in between to advance. On top is a note which explains how to win the game.

The player needs to run from puddle to puddle in the dither world until they find the final "wormhole." There they have to throw the remote into the hole - otherwise, they will be stuck in the dither world forever and slowly die.

## Additional Libraries

- Tiny glTF - <https://github.com/syoyo/tinygltf>
- PhysX - <https://developer.nvidia.com/physx-sdk>
- Dear ImGui - <https://github.com/ocornut/imgui>
- SFML - <https://www.sfml-dev.org/>