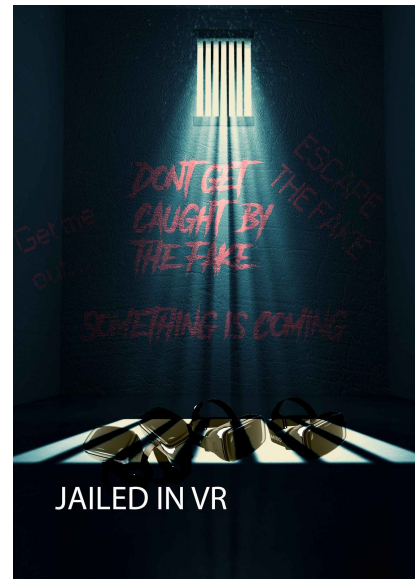


JAILED IN VR

Fabio Francescato (01526912)

Munteanu-Calen Alin (01528098)

Jailed in Vr is a horror puzzle game set in a first person perspective. The point of the game is to solve puzzles, dodge monsters and reach the end of each level to beat it. The main mechanic was supposed to be the double sight mode, the normal and the VR one, which hid and revealed specific objects of the level. Lastly, we reverted to keeping only one mode due to the underestimated difficulty, time consumption faced implementing the game, it's failed physics and the effects.



Implemented features

Playable game and win condition

For the first submission the document upload had failed so this one will include a resumé of both submission's documents.

For the first submission we have implemented a simple gameplay, in the absence of proper physics a simple puzzle, as in collecting of letters, as well as animations, textures, objects, level, camera, win condition and other parameters.

For the third submission we have implemented necessary effects and corrected the gameplay a bit as well as corrected some effect issues that were pointed out, as in made it more stable and changed the outlook.

The point in the first submission is to have a taste of what the skeleton of the game looks like and to find all the letters in the level. After having found them, return to the table. The monsters are roaming around, though no damage will be inflicted.

In the second submission, to the monsters has been added a cel shading look and swinging fireballs, GPU particle snow is storming and the enter/exit portals of the level have been included using video texturing. We did not manage to successful implement physix, so the gameplay has not changed much otherwise.

For the third submission we have added a basic collision system for the walls and the enemies.

To move around use the WASD to move and the mouse to change the direction of it and of the view.

The puzzle table is to be found at the start of the level in the first room. The letters are split among all the other rooms in the level.

To collect the letters, simply walk into them (or around them in some problematic cases) and they will automatically form an encrypted symbol at the table.

Once all the letters have been found, the message will be displayed at the table.

To finish the level, go back in front of it and the victory message will be shown.

To quit/terminate the exe at any given time, press on ESC.

3D Geometry and Textures

The objects have been modelled in Blender and imported into it using the Assimp library. The code for the Assimp importer can be found in the `namelike` method under `Geometry.cpp`. The textures were saved as `.dds` files and imported using the given functions. The entire model loading and texturing is done in the `main.cpp`, right before the render while loop.

Moving Objects

The objects are drawn inside the render loop, where the animation also takes place, based on the framerate. Using `translate`, `rotate` and `scale` and with the help of variables and conditions, the animations, movement and win conditions are implemented and run in this section.

For the third submission we have made the movement of the objects framerate independent.

Controls and Camera

We have implemented a camera that can look and change the direction of movement in around the horizontal line of the level. While the current movement is assigned to the usual movement buttons assigned in many video games (W,A,S,D). The implemented camera and movement can be found in the **`FPSCamera.cpp`** file. For the second deadline we have made it possible to look in a 360 rotation.

KEY	Effect
W	Move forward
S	Move backwards
A	Move left
D	Move right
ESC	Exit Game

Adjustable parameters

We have added a full screen mode and brightness multiplier into the existing config file and implemented those parameters into the game. All of them can be found in the **`main.cpp`** file.

EFFECTS

Hierarchical Animation

We have implemented a fire sphere rotating around the monster in the main room. The fireball rotates first around it's own x axis and then around the monster in an y axes, around his center point, making use of translation and of his actual shifting coordinates to achieve this effect. The second fireball included is dependent on the first one and both of them give the effect of an axe swing/sling shot/ disc throwing/launch. The objects have been initialised in **main.cpp**, the animation code can be found in the while loop.

Cel Shading

For the cel shading we have implemented a new shader object (celShader in main.cpp) and made it render in specific intensity intervals. The enemy models(monsters) are rendered using the cel shader. Implementation and shader code in **main.cpp**, **celShader.vert** and **celShader.frag**.

For the last submission we added a phong lighting method to the celShader, because it wasn't enough to be only flat-shaded

Video Textures

Implemented Video Textures using a vector of already existing "dds" files and made it iterate through that vector, the textures change in regards to dT and a multiplier. Implementation is in file **VideoTextures.cpp** and header implementation of Video Texture is an underclass of **Material.h**. There are two objects using the video textures in the game, namely the entrance and exit 'portals'.

For the last submission we have added more textures (21 textures) and now they act as block paths.

GPU Particle System using Compute Shader

We have implemented the GPU Particle System using Compute Shader with two separate shaders, one compute shader and one vertex-geometry-fragment shader. We had to insert a new constructor into the **Shader.h** file for both shaders, because the framework model doesn't support those kinds of shaders. The general implementation as well as the draw call is in **GPUParticleSystem.cpp** and **GPUParticleSystem.h** file. The glsl code is in **particleSystem.vert**, **particleSystem.geom**, **particleSystem.frag**, **particleSystem.comp**. In the game the particles are represented by the snow particles falling down from the sky. For the third submission we have made the GPU particle system framerate independent.