

Brief description of the implementation, in particular a short description of how the different aspects of the requirements (see above) were implemented.

The HUD was created using the library FreeType.

The lightmaps were baked using blender. Four pointlights are positioned in our room, shining light on 6 floor objects, 8 outer wall objects plus 3 inner walls and all of our objects. The point lights produce very hard shadows, which are more visible on our dark flooring than soft shadows would be. Since all objects of similar type share the same texture, we used separate lightmaps and combine them in the texture shader.

We chose Nvidia PhysX as our physics engine.

Complex 3D Models from Files

Models were created using Blender. Objects of the same type were assigned children and parent (e.g. the first roof object is the parent of all the other ones). This helped organise the objects in blender but also made it possible to create hierarchical animations.

Hierarchical Animation (manually)

A saw blade object consists of a body and 12 children (the “spikes”). The spikes share the same origin as the parent, so when the parent updates its rotation, it calls the updatefunction of all of his children with the same rotation matrix. This way every part of the saw blade moves as if being a single object.

Frustum Culling

View Frustum Culling was implemented like described in the tutorial by Lighthouse3D

Controls

The character is controlled using W, A, S, D for movement and SPACE for jumping.

Debug Options (Frame Rate, Wire Frame, View-Frustum Culling)

“Features” of the game.

- dangerously fast moving sawblades
- lasers that switch on and off by themselves
- rolling on the floor
- rolling on the ceiling
- automatic doors
- beautifully rendered graphics
- fast-paced action-filled gameplay
- fireworks

How and which objects were illuminated (description of light sources) or textured.

The room is illuminated by four pointlights located at the ceiling.

The room of our level consists of four components. 8 wall objects, 6 floors objects , 8 Roof 2 Separators (Walls between the two halves of the room). Similar objects share the same texture. Every object that’s not part of the room shares the same texture.

What additional libraries (e.g. for collision, object-loader, sound, ...) were used, including references (URL) (see restrictions)?

We use NVIDIA Physx for physics simulation, collisions etc.

<https://developer.nvidia.com/physx-sdk>

Assimp for loading the geometry

<http://www.assimp.org/>

ffmpeg to import videos

<https://www.ffmpeg.org/>

freetype for text rendering

<https://www.freetype.org/>

Which Effects are implemented

- Water Mesh: The lava in the lavapit is realized as a watermesh.
- Particles: We used Particles for the sparks coming out from under the sawblades and for the fireworks seen when finishing the game and walking through the door.
- Procedural Textures: The lava, the laserbeams and the character model are textured using procedurally generated textures.
- Video Textures: The TV-screen seen at the beginning of the level is realized using a video texture.

How you've implemented those Effects (Links/References to papers, books or other resources where the effect is described and a description of your extensions to it)

Water Mesh: The water mesh was implemented using sine curves as described in the tutorial slides.

Particles: Particles were rendered using transform feedback buffers, as described here:

<http://ogldev.atspace.co.uk/www/tutorial28/tutorial28.html>

Video Texture: Was implemented using this tutorial:

<http://dranger.com/ffmpeg/>

Plus the online documentation of ffmpeg.

Procedural Texture: Was implemented using the formula described here:

<http://www.upvector.com/?section=Tutorials&subsection=Intro%20to%20Procedural%20>

The noise function used was taken from here:

<https://gist.github.com/patriciogonzalezvivo/670c22f3966e662d2f83>

What Tools have you used to create the Models (Maya, 3DS MAX, ...).

Blender version 2.78c.

<https://www.blender.org/download/>

Textures were converted to the .dds-format using the "Nvidia Texture Tools for Adobe Photoshop".

<https://developer.nvidia.com/nvidia-texture-tools-adobe-photoshop>

For complex interaction sequences (which could already be something like opening a door in the game for example) please also include a step-by-step instruction on how to get through the game.

The only feature that really differs from mainstream platformer games is the inclusion of our magnetised rail. When the character jumps against the bottom of the rail, it will stick to it. The character can still fall off the rail if he gets too far on the edge.

There is a shortcut included in the level, it is possible to move through the right wall in the corner and reach the goal without needing to have fun actually playing the game.