S.C.A.R.

Controls

Key	Effect
W	Move forward
A	Move left
S	Move back
D	Move right
Space	Jump
Q	Fly up (cheating, at least try without)
E	Fly down (cheating, at least try without)
Mouse	Rotate First Person Camera
LMB	Recording Start/Stop (only on Timefield)
RMB	Play back Start/Stop
ESC	Exit Game
F1	Helpscreen
F2	Show FPS
F3	Wireframe On/Off
F4	Texture Sampling Quality
F5	Mipmap Qality
F8	Viewfrustrum Cilling On/off
F9	Tranparency On/Off
F11	Music On/Off
F12	Reset Current Level

Gameplay

S.C.A.R is a game about timing and planning. The goal in each level is to exit through the door to paradise. It isn't as simple as it sounds though. All door-buttons have to be activated at the same time for the door to open and you can not stand on a button and go through the door at the same time. This is where the recording fields come into play. They allow you to record your actions within a time limit. After that you will be teleported back to the recording field. You can later recall those actions and thus help youself to reach paradise. Each recording field can store one recording. Somtimes more than one recording may be necessary to finish a level. To make your task more difficult there are pits of lava and an enemy robot that will kill you in an instant. Furthermore you will find different buttons that can activate elevators or bridges.

3-D Models

All assets used in this game have been modelled by us using a software called Blender (https://www.blender.org/) including uv unwrapping. We have also experimented with texturepainting an baking normal maps. The results may not used in the current game, but can be found in the texture folder.

The models are exported to the .OBJ format and loaded using assimp.

Both the enemy robot and the ghost have animations hard coded using tranformation matricies.

Effects

We have implemented:

Shadowmaps (directional light, PCF)

CPU Particle System (Instanced rendering, depth ordering, transparent textures)

Bloom (we render the scene to an FBO, blurr the bright regions using 2x1D gauss filter and add the images together)

Procedural Animated Noise Texture

Normal Mapping(tangents are precomputed, bitangents calculated in the shader)

Infinite Zoom Mandelbrotshader (just for fun, Mandelbrotset rendered in real time)

Lighting

We have a Pointlight in every level. It is circling around the position specified in the level file and works especially good with the normal mapping. Then we have a directional light that can also cast shadows.

The lava shader only displays a texture.

The diffuse shader uses the point light to calculate the diffuse reflection.

The specular shader calculates diffuse and specular reflections using the point light.

The shadow shader draws pixels in the shadow darker(shadowmap)

The shader may combine several of these tecniques as (pretty consistantly) indicated by their name.

Other

To improve performance we have implemented view frustrum culling. We calculate the bounding box when we load the meshes. Before drawing a mesh we check if any part of it can be visible using the corners of the bounding box.

We use VBO's and VAO's for all our meshes. We also use FBO's for post process filtering (bloom).

For the recordings the position and rotation of the player is saved each frame into a vector, along with a timestamp. To play back the actions the entry with the closest timestamp to the current time is selected and a ghost will appear at that location. This is also used for collision detection with buttons etc.

We use a custom level loader to read in shaders, textures meshes and initialize all objects.

Librarys

We use GLEW, GLFW for the window, freeimage to load the images, assimp to load the models, bulletphysics for the collision detection, glm for vector and matrix math, SFML for the audio, Freetype for text rendering.

Walkthrough

<!---SPOILERS---!>

Please, at least try first.

Level 1

This is a simple level, step on the yellow timefield right in front of you, start recording and try to jump on the platform and reach the end of the tunnel, make a left turn and step on the door opener as fast as possible. The time is rather short so you should hurry. Afterwards complete the level by starting the playback of the ghost and wait at the door for the ghost to open it.

Level 2

First step on the red timefield, start recording, run downstairs and activate the elevator button for some time. Afterwards jump down the hole on the left of the elevator button and activate the first door button. After completing the recording step on the blue timefield, run to the elevator in the same room, wait for it to lift. Afterwards step on the bridge button in the 1st floor for some time and in the last few seconds step on the door opener. After finishing the second record you have to start the playback of the ghosts (RMB), go with the same elevator in the 1st floor, step on the bridge (attention you have to walk while the bridge is moving) and go through the door.

(Hint) Level three has no exit so far, but try hitting F10 to see some of our experimental shader/failed attempts

!!! If you have epilepsy or are sentitive to flashing lights you should probaply not try this !!!

References/Links

http://wiki.delphigl.com/index.php/Hauptseite

http://www.opengl-tutorial.org/

http://www.tomdalling.com/blog/category/modern-opengl/

http://developer.download.nvidia.com/SDK/10.5/direct3d/samples.html

http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-13-normal-mapping/

http://bulletphysics.org/mediawiki-1.5.8/index.php/Tutorial_Articles

http://www.learnopengl.com/#!In-Practice/Text-Rendering

http://www.learnopengl.com/#!Model-Loading/Model

http://soundbible.com/

http://youtube.com/

http://opengameart.org/content/50-free-textures-4-normalmaps

and our best friends Google and Wikipedia.

Awesome Music: The Battle of Lil' Slugger

Original Game: Super Meat Boy

Composer: Danny Baranowsky

Probably Copyright Infringement, please do NOT distribute.