Abgabedokument 2. Abgabe

Controls

F2 - FPS on/off
F3 - Wireframe on/off
F4 - Texture sampling quality: Nearest Neighbour / Bilinear
F5 - Mipmapping quality: Off / Nearest Neighbour / Linear
F7 - VSync on/off
F8 - Viewfrustum culling on/off
F9 - Transparency on/off
Space - Start Hulk / Perform Nosedive / Restart Level at End
A - Strafe left
D - Strafe right
Mouse - Move camera
P - Pause game (+ freely move camera with mouse and A,S,D,W)
Esc - Exit game

Settings

Some settings are controlled through editing the “settings.txt” file.
Fullscreen mode and screen resolution can be changed in the file.

Gameplay

The Hulk starts jumping off when the user presses the spacebar. The closer the pointer of the “Rage wheel” comes to the “Max-Rage” Symbol, the further the Hulk is going to jump. In the air, the Hulk can be controlled with the A and D key (move left, move right). The mouse is used to control the camera perspective to have a look around.

When the Hulk jumps off the ground, the length of the jump is displayed at the top. When he comes near the ground, an altitude warning starts to show up. When the Hulk hits the ground, he starts bumping off until he comes to a halt. After that the game can be restarted by pressing Space again.

On the top right of the window, in the “Rage-o-meter”, the remaining amount of rage is shown. Rage is necessary to do nosedives by pressing Space during flight.

Game Features

- Selft made sounds for the Hulk
- Terrain generated from height map
- Terrain moves with camera, height lookup is done for world coordinates -> Infinite terrain

- Skybox (moves with camera)
- Normal mapping on terrain.
- Movement of tanks along terrain according to tangents. Linear interpolation of position and angle according to heights and tangents of terrain.
- Motion blur at start and when doing nosedives
- GPU Vertex Skinning on Hulk
- Hierarchical animation of tanks (movement of turret)
- Fade out of tanks when hit
- Fade in of tanks when positioned on terrain
- A bit of transparency on “rage wheel” (beginning of the game)

Models

The models are loaded with the assimp import library. All important data structures such as vertices, normals and UV coordinates are loaded into VBOs and are sent to the gpu. The imported hierarchy is saved in a self-implemented class structure. Every imported object has an attached material, which handles textures and illumination constants for each model. The model can then be rendered recursively by traversing the node tree. Textures are loaded from separate image files with FreeImage.

The models were loaded from the internet. Autodesk Maya was used to export the models in Collada file format.

Illumination

One light source rotates around the current camera position on XZ plane. The height of the light source is calculated as a sine wave and reaches its highest point in front of Hulk and the lowest point behind Hulk. Symbols (a sun, a moon) in the lower left corner of the screen are indicating the points where the global light source is around the lowest or highest point.

The light source is moving in the scene also when “Pause game” (key P) is activated. This is to observe the different light effects more easily.

The terrain, the hulk and the tanks are Blinn-Phong shaded and are therefore depending on the position of the light source. In addition the terrain is normal mapped, which can be easily seen as the light moves through the scene.

Effects

Motion Blur

Motion Blur is adapted from:

http://john-chapman-graphics.blogspot.co.at/2013/01/what-is-motion-blur-motion-pictures-are.html
and:
http://http.developer.nvidia.com/GPUGems3/gpugems3_ch27.html

It is done by calculating, for each pixel, the change between current frame and previous frame. This creates a vector map where the movement of each pixel is recorded. Along each vector (called blur vector) multiple pixel samples are taken and averaged (which creates the blur).
We use 32 samples along each blur vector and the vectors are scaled according to the current frame rate to create similar results independent of the actual FPS.

Sequence of actions:

- The current scene and its depth buffer are rendered into textures by using FBOs.
- With these two textures, the ViewProjection (VP) matrix of the last frame and the ViewProjection-Inverse (VPI) of the current frame
  For each pixel:
  - the world position of the pixel is calculated (by using VPI and depth)
  - the pixel position in the last frame is calculated (by using VP)
  - the blur vector between these pixel positions is calculated
  - multiple samples are taken along the vector and averaged
- The result is displayed in a fullscreen quad (2 triangles)

Please note that motion blur is only influenced by camera movement and not for individual objects moving in space.

Normal Mapping

Literature:
http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-13-normal-mapping/
http://stackoverflow.com/questions/5281261/generating-a-normal-map-from-a-height-map

Normal mapping is applied onto the terrain. Which is a mesh with an applied height map. The heights are applied for each vertex in the vertex shader. Tangents and normals are calculated in the fragment shader. By calculating these in the fragment shader smoother results are obtained, but at the cost of performance.
In addition Blinn Phong shading is used. Shading calculations are done in tangent space.

Sequence of actions:
For each pixel in fragment shader:

- Surrounding heights are sampled from height map (left, right, top, bottom)
- Tangent and bitangent are calculated according to heights
- Normal is calculated from tangent and bitangent
- Tangents and normal is converted to eye space with the normal matrix
- Tangent matrix is created from tangent, bitangent and normal

- Inverse tangent matrix is calculated
- View, light and normal is converted to tangent space
- Blinn Phong shading is done in tangent space

GPU Vertex Skinning

Literature:
http://ogldev.atspace.co.uk/www/tutorial38/tutorial38.html
http://gamedev.stackexchange.com/questions/26382/i-cant-figure-out-how-to-animate-my-loaded-model-with-assimp

Skeletal Animation is applied to the Hulk, the main character of the game. The bones as well as per vertex bone weight and bone id information is extracted from the imported collada file. The weights and ids are sent to the gpu, the bones are stored in a map as well as in the node hierarchy. Animation data is also extracted from the file and stored in our own data structures.

Every update() function call starts a recursive Animation. The node tree is traversed by the function. If animation data is available for a bone node, transformation data is calculated and interpolated from corresponding keyframes. The resulting matrix is multiplied with a parentTransformationMatrix and a boneOffsetMatrix and is sent as a uniform matrix to the shader.

In the shader, the actual vertex skinning takes place by multiplying corresponding boneMatrices with perVertex bone weights. Adding these multiplications up yields the final matrix. Every vertex and normal is transformed by the final boneMatrix. A vertex can at most be influenced by 4 bones.

The model was rigged with Autodesk Maya.

Used Libraries / Tools

FreeImage http://freeimage.sourceforge.net/
Assimp http://assimp.sourceforge.net/
FMod http://www.fmod.org

Autodesk Maya 2013 Student Edition