Documentation 2nd Submission

Description of the implementation:

- Gameplay: You can steer the astronaut in all 3 dimensions. With the keys A,W,S,D it can be moved left, right, up and down and with K and M it moves faster and forward or slowly backwards. The goal is to reach the earth without hitting obstacles more than 5 times or without losing control because of the flat spin. Additionally to make it a little bit more difficult to fly around the obstacles, the character is making summersaults repeatedly.
- Complex Objects: We created all our models in Maya, exported them as Collada-files and imported them with the library “Assimp”. We also used 3 different models for the earth because of its complexity.
- Animated objects: The summersaults of the character are done with Vertex Skinning and is animated all the time. This is achieved by adding bones to the astronaut including weights which influence all our vertices.
- View-Frustum-Culling: We added Bounding-Boxes to all our objects. They were created by extracting the min/max values of the mesh for all three axes and with those values we span the Bounding-Box over the object.
- Transparency: We took the Alpha-Channel to get the transparency for the lens flare and the 2D-Text-Objects like the banner "Game over". Therefore we used additive blending with alpha as blend function.
- Experimenting with OpenGL: We used the key-mapping given on your homepage. So you can enable and disable the display of the frame time (F2), wire frame (F3), Texture Mapping (F4), Mip Mapping-Quality (F5), View-Frustum-Culling (F8) and Transparency (F9) and when one of this changes there is shown an information on the screen for a short time.
  We implemented Vertex-Buffer-Objects and Vertex-Array-Objects for all of the objects. For example for the astronaut we created and bind a VAO. Then we created a couple of VBO (facebuffer, positionbuffer, normalbuffer, UV-buffer, weightbuffer, indexbuffer) and bind those as well. After everything is done we unbind the VAO. Every time the draw-method is called we bind the VAO again until the whole object is drawn to the screen.

“Features” of the game:

- We load complex, self-created models instead of simple primitives.
- We used a Blinn-Phong to illuminate our scene.
- We used separate shaders for the animated character and the rest because it wasn't necessary to calculate a transformation matrix for those.
- By steering the astronaut, he is not only translated to another position, he also keeps rotating around his own axis and doing summersaults repeatedly. The speed of the rotation can be modified as well.
- With the key "P" you can pause the game
- When the game begins you can hear the last messages from Felix Baumgartner before he jumped out of the capsule.
- In the background you hear a typical deep space sound. When a collision happens the astronauts screams "Ouch".
- If you accelerate the character too much, it will end up in a flat spin which you should avoid or at least stop it by pressing "Space" 5 times quickly. Otherwise you'll lose the game.
- As soon as the screen turns reddish it is a first sign for the dangerous flat spin.

- You also have to keep an eye on the altimeter to start the landing process in time. To keep track of that you have to look on the little lamp on the altimeter which turns red when it's time. There is also a banner in the right upper corner of the screen that tells you when to land.
- You can look around in the scene by moving the mouse and also zoom in and out. With the key "R" you can reset the camera to its original position.

Objects illumination and texture:
All our objects are textured and illuminated. We have one global light source, the sun. The light has a fix position and consists of an ambient, diffuse and specular part.

Used Libraries:
- Assimp
- FreeImage
- Glew32
- GLFW
- FMOD
- OpenGLUT
- Bullet Physics

Useful tutorials:
- http://www.opengl-tutorial.org/beginners-tutorials/tutorial-6-keyboard-and-mouse/
- http://www.lighthouse3d.com/cg-topics/code-samples/importing-3d-models-with-assimp/
- http://www.lighthouse3d.com/tutorials/view-frustum-culling/geometric-approach-testing-boxes/
- http://www.informatik-forum.at/showthread.php?90399-Assimp-Bones-Transformation-Chain

Effects and their Implementation:
- Level of Details: We decide the current level of detail for the earth with the position on the z-axis and the distance between the earth and our astronaut. We have 3 different levels beginning with a sphere, then small bumps (ocean, mountain, valleys) and ending with seeing small cities and houses on the surface. The increasing number of vertices can be seen in the wireframe mode.
- CPU Particles systems: We have triangles for the particles with their own texture. We set a start point, a spawn and a life time. To make it more flexible the life time is random. We split the screen in 4 parts and every part has its own spawn point which emits 8000 particles toward the spawn direction. When their life time is over, the particles were deleted and immediately new ones were created.
- Lens flare: For the lens flare we used plane objects in different sizes and with different textures. Depending on the view direction of the camera those planes are moved. For a photo-realistic lens flare effects we used the blend function with the alpha channel. As soon as the earth is occluding the sun the lens flare isn't visible anymore.
- Vertex Skinning: In Maya we add bones to the astronaut including weights which influence all our vertices. We had some problems with exporting the Maya-file into a Collada-file (wrong number of bones and weights per vertex). With another import method from Assimp we could set new flags to solve these problems. We needed a new vertex shader to handle the transformation matrix and there is also

an own fragment shader which is pretty much the same as the other texture fragment shader. The Assimp animator creates a data structure which can be traversed recursive. It saves all the names of the bones and helps to create the model matrix of the astronaut.

Tools used for the creation of the models:

- Maya
- Photoshop

Complex interaction sequences:

- Start the game: "Enter"
- Pause: "P"
- Stop flat spin: 5x "Space"
- Landing: "Enter"
- Reset Camera: "R"

After the first submission we created work packages and split them between us for an efficient working environment in the following way:

- Thomas Trautner:
  - Level of Detail
  - Particle Systems
  - Lens Flare
  - Vertex Skinning
  - Sound
- Martina Kratky:
  - Collison Detection with Bullet
  - 2D Text Output on screen
  - key mapping
  - Advanced Gameplay