‘Morphic Rush’ Documentation

Submission 2

Implementation

In the following section we will briefly describe the main implementation work we made since submission 1.

GAMEPLAY
For our gameplay we added the functionality to our ‘flyer’ object which can be controller in the air. The same as ‘roller’ object, the ‘flyer’ has to overcome several obstacles. But as soon the ‘flyer’ touches the ground, the game is over. The longer you survive in the game, the more points you get at the end of the game.

EFFECTS
A huge task for submission 2 was implementing all necessary effects. The effects will be described in later sections below.

ANIMATION
Unfortunately, we did not make it to implement the hierarchical animation for now.

BLENDING
In the game it is not possible to turn on/off the blending. Since we had some problems with the effects working together. So finally we decided to have blending always turned on/off depending on the effect. For example for motion blur it’s necessary to have blending turned off, but for particle effects we need to have blending being activated.

Illumination

We have pointlight as the only lightsource. Its position will be updated according to the z-position of the game object.

Additional libraries

Collision Detection | PhysX 3.4 | https://github.com/NVIDIAGameWorks/PhysX-3.4
Model Loader | Assimp 3.3.1 | http://www.assimp.org/index.php/downloads
Text Rendering | Freetype | https://www.freetype.org/
Image Loader | DevIL | http://openil.sourceforge.net/docs/index.php

Effects

In this section we will describe the effects that we have implemented for our game. We will provide the according links and references we have used in order to implement those effects at the end of the documentation (see References).

NORMALMAPPING |0.5

For texturing we used normal mapping, so our scene gets a touch of bumpiness. All textures we used for the game, can be either found in in 'Total Textures Repository" or in https://3dtextures.me/. We calculate the TBN matrix, with the tangent information we got from assimp, in our vertex shader. All calculations are done in world/view space.

MOTION BLUR |1.0
For the implementation of motion blur you can retrieve all information form MotionBlur.h File. The effect is realized by storing the rendered Image as well as the motion vectors into textures of a framebuffer. Afterwards the textures are applied on a quad where the motion vectors are sampled to create the blurred effect.

OMNI-DIRECTIONAL SHADOWMAPPING (WITH PCF) |1.5
With shadows we want to add more realism to our scene Since we have pointlight as light source, it's necessary to have shadows in all directions.

PARTICLE EFFECTS |1.0
The implementation work for the particles are done separately in ParticleEffect.h. It mainly uses a compute shader to calculate the lifecycles of the particles. After calculating the current lifetime of a particle, it gets passed into a geometry shader to create a quad for billboarding the particle texture.

Models

All models used in our scene are created in Blender.

Key-Mapping

F3 - Wire Frame on/off
F4 - Shadow Mapping
F5 - Blending on / off → not working as above mentioned
F6 - Motion Blur on/off
F7 - Normal Mapping on/off
F8 - View-frustum Culling on/off
ENTER - Debug Mode

CONTROLS

KEY	EFFECT
A	Fade to the left
D	Fade to the right
W (Flyer Mode)	Fly upwards
Space Bar	Transform

References

https://learnopengl.com/Introduction
http://www.opengl-tutorial.org/
https://www.youtube.com/user/thebennybox/videos