Honey Hero

Implementation

The game features a 3D Environment, with a movable bee character in third person view, based on our implementation for the ECG course. It is the objective to reach flowers and pollinate them in order to get points. This requires the player to get near the blossom and interact with it using the on-screen directions. Physics simulation, as well as all movement on the map are framerate independent and use frame times to compute advances. The goal of the game is to achieve as high a score as possible in the given time. The controls are standard video game controls, that allow for movement in all directions. Holding LMB allows the player to rotate the camera around the bee. Movement is always relative to the current viewing direction and the bee also always looks in the same direction as the camera. The Scene is lit by a single directional light representing the sun and illumination is done using the object's normal vectors. All objects in the scene are textured using image textures with the exception being the tree stump, that uses a procedural texture. Besides the bee controlled by the player, there is a power up sphere floating up and down. The adjustable parameters in the settings file include refresh-rate, full screen toggle, brightness setting, LOD settings, screen resolution, game time constraint and the given ECG-Framework settings.

Controls

CONTROL	FUNCTION
W	Move forward
A	Move left
S	Move back
D	Move right
SHIFT	Move down
SPACE	Move up
U	Toggle HUD
E	Interact with flower
ESC	Exit the Game
Mouse wheel	Zoom
LMB	Hold to move camera
L	Toggle LOD

Libraries

- Assimp (https://github.com/assimp/assimp): for importing 3D assets
- NVIDIA Physx (https://github.com/NVIDIAGameWorks/PhysX): for collision simulation/detection
- Freetype (https://freetype.org/index.html): for on-screen text rendering

Effects

- Procedural Texture: We implemented a procedural wood texture for the tree stump, which was placed once in the middle of the map. The shader uses a noise function, and its parameters, such as the two colours, frequency, noise scale, ring scale and contrast are configurable.
- Cel Shading: Our plan was to implement cel shading together with contours for the characteristic toon look, but unfortunately, we did not manage to do both. Nevertheless, the cel shader works and is currently set to map the lighting values to 4 discrete shades. This is the default shader.
- LOD: using an octree All objects on the map are saved in an octree. To accelerate the performance, every object has three different models: highest definition, lower definition, and a simple sphere. For the lower definition we simply reduced the polygon count of the models in Blender. To turn LOD on/off, press L.
- Bloom: We implemented a coloured Bloom with a gaussian blur to make the bright regions bleed around the light source.
- CPU Particle System: A maximum number of particles can be defined in the settings file. All particles start out from the same position. Each particle has a set lifespan and a random deviation from a set movement vector, as well as a random size. The used approach employs instancing, where a single buffer exists that holds the position data (and size) of all particles which is updated in every loop.

Models

Baum: Low poly tree 3D-Modell by Simon Telezhkin on hum3d.com

Baumstumpf: Low-Poly Tree Stump by jwilson32 on sketchfab.com

Blume: Low Poly Flower by JayDesigns3D on sketchfab.com