HOMEWORK 2

By Anton Volkov

1849922

Structure

The project is built with an Object-Oriented paradigm, with any data structure added to the project defined as a class, this to simplify scene construction since the presence of multiple entities to render.

There is a general Entity class where common operation and data are defined, and from which specializations branch in case of need of a particular behavior, case in point the Camera where the render function needs to act differently from the general object entity.

Entities can have parent and children entities. Children entities have their transform relative to the parent transform, that is to simplify animations, so that only a parent entity needs to be transformed and the children maintain the relative position and rotation.

Mesh data is stored in an entity variable to detach it from the entity transform if needed.

Scene

The scene is built from a world entity, of which the hill, camera and kangaroo are children, so that when the rendering is called only a single call is made and the rest are called recursively over each entity children.

Lighting is provided by a directional light defined in the shader programs.

Kangaroo

The kangaroo is built by a sequence of scale and rotated cube, with the torso as centre and the legs, arms, head and tail children to it. Each mesh placement in respect of the entity it's associated with is such that it mimics joints positions.

Fur is made with the application of a fur image texture onto the cubes, with difference for the head where there are a pair of eyes in the front. On top of that a normal texture is applied to give the fur more detail.

Hill

The hill is constructed by a grid of 128*128 vertices, with the height calculated through the gaussian function, so that to obtain the shape of a more natural looking hill.

For the grass a simple grass texture is used.

Camera

To provide an easier implementation of the camera controls, a camera class has been implemented. It sends to GPU the view projection matrix.

Controls are implemented with mouse and keyboard controls. While on the canvas if the left or right mouse button are clicked, the camera will yaw and pitch accordingly to the mouse movement. With w,a,s,d it will move front, left, back and right respectively and with spacebar and shift will go up and down respectively.

Techniques

Multiple objects rendering

To render multiple objects with graphical glitches every mesh keeps a pointer to its buffers, that are loaded to the frame buffer at start time and activated before rendering and deactivated after rendering. This makes so that all data is passed at the start and only the needed data from each object is used when rendering it, improving performance with respect to loading a mesh data to the frame buffer every frame.

Normal mapping

To implement normal mapping each mesh calculates its own normals and tangents, then in the vertex shader the bitangent is calculated and a tangent space matrix is calculated. The matrix is then passed to the fragment shader and used to transform the extracted normals from the normal texture from tangent space to global space, then the result used to calculate lighting for the surface.