Xorean's Rise

Controls

Key	Action
W,A,S,D,Q,E	Rotate Planet
Right Mouse (Drag)	Rotate Planet
Left Mouse	Select/Build/Attack
ESC	Close Game
F2	Show/Hide Debug information
F3	Wire Fram on/off
F4	Texture-Sampling Quality: Nearest Neighbor/Bilinear
F5	Mip Mapping-Quality: Off/Nearest Neighbor/Linear
F8	Viewfrustum-Culling on/off
F9	Transparency on/off

Features

Lighting

The Planet and Towers are illuminated by a directional light (sun), which slowly rotates around the planet. The normals and specular reflection amount are stored in a single texture (normals = rgb, specular = a).

Scene Management

The Scene-Management is done by a class, which holds a reference to all Nodes that are in the current scene. For rendering, the scene is then split up into sub-scenes (background, main, foreground, ui), depending on the entity, which is attached to the scene-nodes, and rendered one after the other. The Background contains for instance the Skybox, which shows the Galactic Nebula.

Planet

The planet is created completely on the fly, which means, the number of subdivisions (the more subdivisions, the smaller the tiles will get) can be specified in the code. Currently the distribution of tile-types is randomly, but we are planning on using some kind of noise function to group the tile-types better. For selecting the right Tile upon clicks on the Planet, we are using some kind of raycasting algorithm.

Resource Management

Any Resource used by the Program is loaded by a Loader-Class which then stores the Data inside a corresponding Resource class. The benefit of this method is, that for each Resource Type, more then one ResourceLoader can be used. (For instance, for Textures a TextureLoader and a Loader for CubeMap

textures are used.) The Loaders also store a reference to all resources they have already loaded, to avoid loading the same resource more then once. All ResourceLoaders are then managed by the ResourceManager, which holds a reference to all registered ResourceLoaders with a given extension for each loader (TextureLoader can load ".jpg",".png",".tga", and some more Types).

AI

The "Enemy" is controlled by a simple AI, which builds new Buildings and attacks the player's buildings, depending on a simple algorithm and random values which are calculated every second. Which tower is attacked depends on the type of the tower, the difference of current units and the distance between the towers.

Battle-System

For the Battle-System a random number is generated for each defending and attacking unit. The unit which rolls the higher Number wins the fight. This is repeated until eather all attacking units died, or no defending units are left anymore. The range of the rolled random number depends on the type of the tile.

	Desert	Forest	Grass	Mountain
Desert	100	150	100	50
Forest	50	100	150	100
Grass	100	50	100	150
Mountain	150	100	50	100

This table shows the maximum values which are rolled upon a fight

Flocking

Upon attacking or building a bunch of "particles" are sent to the target location. Those particles travel towards the target direction and also try to avoid collissions by adding a force away of each other particle.

Libraries

- GLFW http://www.glfw.org
- GLM http://glm.g-truc.net/
- Assimp http://assimp.sourceforge.net/
- FreeImage http://freeimage.sourceforge.net/
- GLEW http://glew.sourceforge.net/

Effects

- Normal Mapping (1 point)
- Lens Flare (0.5 points)
- Shadow Maps (1.5 points)
- Billboards (and instanced rendering for the swarms)

Gameplay

You start with a single Tower, which is already full of units. (The NPC starts on the other side of the planet). The Start-Location is always the Sand-Type Terrain.

When a Tower is full of units (shown by the small star on top of the tower) it can build a new tower by first clicking the full tower and then clicking a free field owned by myself (the cursor displays if the hovered field is valid by showing a small hammer next to it). When building, 80% of the units are sent to the building spot.

The towers are always able to attack a visible enemy tower by first selecting the own tower which should attack and the clicking the enemy tower. (The cursor shows the sword on hovering the enemy tower). When attacking, 50% of the current units are sent to attack. By clicking multiple times on the enemy building the tower attacks again with 50% of the remaining units.

The towers regain new units automatically until they are full (displayed by the star).

Tools

- 3ds max
- Photoshop
- Self-written program for combining the tiles to a tile-map