

# Marbler

## Implementation

The Game is basically split up into two parts. The engine, which exposes a basic Game Class which is used to develop a game, and multiple Components, and the game, which uses the components to build a game world.

Nearly every object in the game is a SceneObject, which can be made to a physical actor. A camera can be appended to a SceneObject by using a CameraComponent. When a target is set, the Camera can be rotated around the target, which in this game is the marble. The marble is added to the SceneObject by using a MeshComponent which holds the Mesh and its Material. These components inherit from SceneComponent which basically allows positioning in the scene. Physical Shapes can be attached to mesh components if their parent SceneObject is a physical actor. That is currently how anything is moved inside the game. The InputComponent allows for applying torque to the marble, also appended to the SceneObject. Each MeshComponent has Material which contains the used textures. Any textures can be read which are supported by the Devil Library. Objects are imported with Assimp. There are currently two lights in the scene. A directional light and a point light. They are implemented using Deferred Lighting, therefore rendering speed is not an issue with a lot of lights. Brightness can be adjusted in the config file. Scripting is done with the Squirrel Script Language, scripts have to be put into the "Scripts" folder. All basic functionality can be seen inside Main.nut, further indepth functionality in the Engine.nut. Functionality will be extended for the game event, to create an actual playable level. Instead of a particle system, shadow mapping has been implemented for now. Shadow-Map is created before any rendering. Shadow is drawn in the Geometry Pass of deferred lighting. GUI can be built with TextWidgets and ImageWidgets and is blended on top of the last render. Frustum Culling is done in the MeshComponent, checking if the bounding box of the object is within the view of the camera.

## Features

- Deferred Lighting (Directional Light, Point Light)
- Object Loader (Assimp)
- Texture Loader (Devil)
- "Nice" Structure
- Lightmapping

- Physics – Based
- Shadow Mapping
- Squirrel – Scripting
- Small GUI System
- MSAA (I think?)

## Illumination

All objects are illuminated with a directional light and a point light if nearby. Lighting is implemented using Deferred Lighting. Specular Lighting is also supported. Every object in the game is textured within the Geometry Pass.

## Effects

Light Mapping + Separate Textures (In Geometry-Pass the texture is just multiplied on diffuse)

Heads-Up Display (partly <https://learnopengl.com/In-Practice/Text-Rendering>, Image rendering is selfmade, by just rendering a rectangle with a texture)

Physics Engine (PhysX)

Basically done with the documentation. I already had a similar scene structure to physx, so I basically gave physx initial data (position, actors, shapes of components) and after every calculation step I read the new data and draw my objects accordingly).

Deferred Lighting

Followed tutorial from <https://learnopengl.com/Advanced-Lighting/Deferred-Shading>. I added another render step after the final light render, to add MSAA, which is basically done by OpenGL itself.

Shadow Mapping

I switched around a lot of tutorials, because it took a while to get it working within my Deferred Lighting. So a mix of different tutorials and my own intuition. Shadow map is rendered before geometry pass and then used in the geometry pass to calculate where the shadow is.

Scripting Language Integration (Squirrel)

This was done with the documentation and tutorials given by the developer. It's separated from the rest of the engine as good as possible.

## Additional Libraries

- Assimp (<http://www.assimp.org/>)
- DevIL (<http://openil.sourceforge.net/download.php>)
- FreeType (<https://www.freetype.org/>)
- Glew (<http://glew.sourceforge.net/>)
- GLFW3 (<http://www.glfw.org/>)
- GLM (<https://glm.g-truc.net/0.9.8/index.html>)
- JSONCPP (<https://github.com/open-source-parsers/jsoncpp>)
- PhysX  
(<https://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/guide/Manual/Index.html>)
- Squirrel (<http://www.squirrel-lang.org/>)

## Controls

- W, A ,S ,D – Control the marble
- Space – Jump (You can jump infinite times for now, changes via scripting to come)
- F1 – Help
- F2 – FPS
- F3 – WireFrame (Not really useful atm, since shading is still on while in wireframe mode)
- F8 – Frustum Culling off/on

## Tools for Meshes

Only used blender.

## ToDo

- Add Gameplay elements (Jump count, Speed boost, ...)
- Draw only geometry in WireFrame Mode
- Lightmapping is in the game, but needs some gameplay elements to make sense (Thinking of building a wall around some platform and put light map onto the ground)