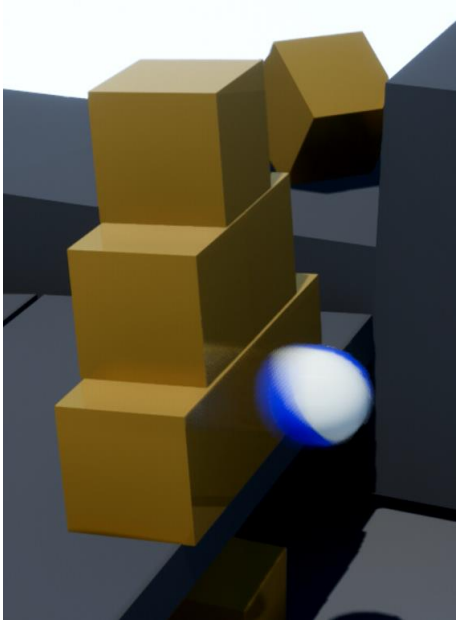
Marbler



Ronald Schartmüller (01426523)

"Marbler" is a physics platformer game with small puzzles. You control a little marble ball and interact with other objects by mainly crashing into them. The main goal is to reach the end of the set course. The goal can be altered using the built-in script language.

Features

Physics driven

Scripting Support

Deferred Lighting

GPU-Particle System (+ Transform Feedback)

Gameplay

The player must navigate platforms and solve small puzzles by controlling a marble. Interaction mainly works by crashing into other objects to move them according to their physical properties. The level consists of a range of platforms which can be placed in various ways. With Script -Support it's possible to load maps on the fly and change behaviour and properties of most objects. Addionally built in power ups can be used and created. Examples for upgrades are flying helmets, speed upgrades and weight upgrades. Thanks to scripting, the goal of each level can be decided independently of the game itself, for example by reaching a certain mark, or collecting certain objects.

Controls

The marble is controlled via the arrow keys or WASD-Keys. The marble’s properties decide at which pace the marble moves according to the pressed key. These properties can be manipulated via scripting.

Key	Effect
W,A,S,D, Arrow Keys	Control Marble
Spacebar	Jump/Fly Up

Strg	Fly down
ESC	Quit game
F5	Reload Scripts

Technical Details

All objects in the scene are textured and lightmapped. There are 4 different types of objects that will be rendered.

The marble – Static Mesh with physics

The platforms - Static meshes

Physical objects and power ups - Static meshes with physics (Block vs. Overlap)

Particles - 2D - Billboards

Camera

The marble has a camera following it. The angle at which the marble is looked at can be adjusted by moving the mouse.

Illumination

There will be one global lightsource which is the sun, implemented as a directional light. Additionally, it will be possible to add spotlights and point lights in the scene if needed.

Collision detection

Collision detection will be handled by PhysX to check for collisions and overlaps.

Effects

Lightmapping (+Texture):

Objects used in the game will be able to use Lightmap-Textures, which are loaded as an extra texture. This might not make too much sense though, as level creation is supposed to be dynamic with the scripting language. If there is enough time left, light map generation will be used.

Scripting Language:

Squirrel will be used as scripting language to change game behaviour at runtime and add levels. These can be loaded at runtime. The reason Squirrel is used is its more C-Style syntax and the support for more structured code with classes, in comparison to languages like Pawn or LUA. If there is any trouble embedding Squirrel, LUA will be used as fallback language.

Deferred Lighting: To support a lot of lights wherever wanted, deferred lighting will be implemented.

GPU Particle Effects: If needed and enough time is left this will be added. Used to show marble movement by showing dust behind its moving vector.

Sketches