Sonic vs Darkness Design Document

- Title: Sonic vs Darkness
- Author: Simon Eugen Josef Daiber / 265607
- Year and season: Wintersemester 2023/2024
- Curriculum and semester: MKB 7
- Course: PRIMA
- Docent: Prof. Dipl.-Ing. Jirka R. Dell'Oro-Friedl

Executable Application
https://seppel2014.github.io/Prima/Sonic/index.html

Source Code
https://github.com/Seppel2014/Prima/tree/main/Sonic

Design Document
https://github.com/Seppel2014/Prima/blob/main/SonicVsDarknessDocumentation.pdf

How to interact
The Controls are:

- A = Move Left
- D = Move Right
- Space = Jump

Checklist for the Final Assignment

1. Units and Positions

Where is 0? Sonic spawns at the Position (2,0.4,0.02). 0 is therefore slightly below him and 2 units to the left. He is slightly moved on z in order to prevent overlapping of other objects. Since the Sonic Object has an individual spawn point it doesnt matter too much where 0 is on the x axis. On Z it should not be move to far since rigidbodies will at some point no longer interact. Y should not be move too much since it is used to calculate how far he can fall before he dies.
What is 1? The Meshes of the Floor Objects have a width and height of 1 Unit. This makes it easy to seamlessly add new Objects to the Level.
The movement on the y axis is also used to measure if he fell down and therefore loses a life. If he moves below -5 on the y axis he loses a life and is reset to the startpoint.

2. Hierarchy

- **Game**
  - Terrain
    - Floor1
      - CmpMesh
      - CmpMaterial
      - CmpTransform
    - Copies of Floor1
  - Sonic
    - SonicAnimation
      - CmpTransform
      - CmpAnimator
      - CmpMesh
      - CmMaterial
    - SonicAudio
      - CmpAudio
      - CmpAudioListener
  - TrapsPlaceholders
    - Trap1
      - CmpMesh
      - CmpMaterial
      - GetTraps (CustomComponentScript)
    - Copies of Trap1
  - TrapsBlocks
  - Golds
    - gold1
      - cmpMesh
      - cmpMaterial
      - cmpAnimator
      - cmpTransform
    - Copies of gold1
  - StateMachines
    - Machine1
      - cmpMesh
      - cmpMaterial
      - cmpTransform
      - cmpRigidbody
    - Copies of Machine1
  - AmbientLight
    - cmpLight (Ambient)
  - Background
    - cmpMesh
    - cmpMaterial
  - EndPost
    - cmpMesh
    - cmpMaterial
    - cmpTransform
    - cmpRigidbody

Terrain holds all of the Floor Objects which each consist of a mesh, textured and flat Material, as well as a component Transform for movement.
TrapPlaceholders holds all the Objects which are later used to generate the physics based Traps. Each Placeholder has a mesh, black material and a ScriptComponent. The Script Component disables the material in the game.
Golds holds the Gold Objects which are animated using the component animator. The Parent is later used to check whether Sonic touches (and collects) a gold.
StateMachines holds the StateMachines. Each of them has mesh and material and a rigidbody which is later used for the interaction with Sonic.
Sonic itself contains two separate Nodes as well as several Components. CmpTransform is used to move Sonic. CmpCamera is the main Camera of the Game which always follows the node. CmpRigidbody is used for interaction with other Rigidbodies.
-The node SonicAnimation contains the Mesh, Material and Animator, used for Sonic.
-The node Sonic Audio contains the AudioListener as well as the currently played Audio (which is later changed using code).
Background is simply a large white meshplane to give the game a background
EndPost uses a rigidbody to detect when sonic has reached the end of the track
AmbientLight is the light used in the Scene. It is later influenced by code
TrapsBlocks is empty and is later filled with the actual traps used in the game. This basically copies the TrapsPlaceholders Node and give it material and rigidbodies.

3. Editor

Creating the basic hierarchy is easily done in the VisualEditor. The basic Level Objects (floors) were also created and placed in the editor using Copy and Paste. All Animations were done in the editor since the visual interface makes it easy to set them up correctly and well timed. Adding Rigidbodies was also mostly done in the editor. Influencing Rigidbodie was mainly done using Code since it offered more in depth influence on the behaviour.
Another advantage of the visual editor was creating placeholders and later replacing them with code. Using this method i had a list of positions and just needed to copy them to the actual traps.

4. Scriptcomponents

Hiding the Placeholders for the Traps was done using a Script Component. It simply deactivates the visibility of the Placeholder Materials upon starting the Game. Using this method there are no calls necessary for hiding these objects.
This can be used for all Objects and could therefore be even more helpful with more individual Placeholder Objects.

5. Extend

Both Sonic and the Trap Objects were derived from f.Node. It was helpful since this allowed me to outsource some methods to theses classes and cleanup the Main File.

6. Sound

There is a main theme running in the background to give the classical Sonic Feeling. Other Sounds represent him interacting with the surroundings. Examples for this are the jumpsound, error sound (when he falls or loses the game), a sound for gold collecting and a sound for when he reaches the Endblock and wins the game.

7. VUI

The Interface is placed in the top left and shows the remaining time (counting down), lives left, coins connected and how many traps have been triggered. The corresponding name is placed above each number.

8. Event-System

I used the Event System to create an eventListener that catches when Sonics Rigidbody collides with another rigidbody. This is used to check things such as the collision with the Endblock (and therefore triggering the Endscreen) or updating the interface on how many traps have been triggered by Sonic.

9. External Data

External Data is used to change the difficulty of the game through the changing of values such as the opacity of the traps (the lower the opacity, the harder it is to see traps). Maximum Darkness, Maximum Time for the game, Maximum lives of Sonic and the Ammount of Light that you get back when hitting the yellow boosters.

A Light

I use a single Ambient Light and most Objects use a Flat Material which then uses this Ambient Light. The Scene gets darker over time and the three Lightboosters (yellow blocks) restore some of the light upon collection. The maximum amount of darkness and how much the boosters restore depends on the config.

B Physics

I mainly used Physics for the Traps. Sonic has a rigidbody which is then used to trigger the traps. When Sonics rigidbody applies force to them their joints break and they therefore start to fall. Another use case are the lightboosters which are simple yellow block with a rigidbody. Upon touching them their rigidbodies turn dynamic and they start to fall.

C Net

I did not use the network functionality

D State Machines

I used a State Machine for the light boosters. I originally wanted them to move on the x axis in the patrol state but wasn't able to get it to work. Therefore it only has the idle state in which it simply exists, and the die state in which its rigidbody changes to dynamic and it continues to fall according to gravity.

E Animation

I used Sprite Animations for Sonic and the gold coins. The goldcoins simply repeat their animation while sonic changes the animation according to user input. Sonics animations include idle, jumping, walking right and walking left.