Bunny-Mania Design Document

- ❖ Author: Ulla Herrmann, Matrikelnr. 263559
- ❖ Year, Season: summer semester 2024
- ❖ Course: PRIMA
- ❖ Professor: Prof. Jirka Dell'Oro-Friedl
- ❖ Fellow/Contributor: Lukas Scheuerle
- ❖ Executable application: https://lum7n.github.io/Prima/p4_BunnyMania/index.html
- ❖ Source code: https://github.com/Lum7n/Prima/tree/main/p4_BunnyMania
- ❖ Design document: https://github.com/Lum7n/Prima/tree/main/p4_BunnyMania/Bunny-Mania_DesignDocument.pdf

Table of contents

Description for the users on how to interact

Use WASD or the Arrow-Keys to move, while moving the Character "Bunny" through the Maze. The Goal of the game is to collect all the stars in the maze. After collecting all the stars, a key to get you out of the maze spawns. Collecting stars awards you points, which are visible in the VUI. Additionally, you can collect different Items, which have different effects. Lives give you back one heart, if you have been hit by an enemy. PowerUps increase your speed and make it possible to kill enemies. AdditionalTime increases the time left on a PowerUp. For balancing, Lives are exceptionally rare, followed by PowerUps, which are also quite rare. There are more AdditionalTime Items, since they are only usable with a Powerup.

Current status

★ Due to issues with the rigidbody and collisions, the enemies can just be killed on their spawn-spot.
★ The AdditionalTime Items don't add more time to the PowerUp, they just delete time from the previous playing time.
★ There could have been more sounds: like for killing a fox, or getting killed by them, or for the winning screen, or game over screen.

Criteria for the final assignment with explanation

1. Units and Positions

0 = the start of the maze in the top left corner, as that is where the game starts. Placing it in the middle of the maze or somewhere else would make it so that the items would have to be generated in both positive and negative x and z directions.
1 = the measurement of one block in width and height.

2. Hierarchy

The Main-Elements are the Maze, the Character and the Audio.
The Maze is keeping the Border, the Level-Cubes, the Ground, the Roof and the Items. The Character has its Geometry and the Camera Node. The Audio keeps the Sounds. The Hierarchy goes from the smaller Elements to the bigger ones, but actually it doesn't play that big of a role. The Building-Cubes are sorted from left to right and top to bottom. The Items are just sorted by type.

Blue = star nodes are added via code, the amount is random decided
Green = nodes are added via editor but disabled and then via code enabled, but not all of them, just a random decided percentage.

Game

- Maze
  - Border
    - Cube_left
    - Cube_right
    - Cube_bottom
    - Cube_top
  - Level 1
    - Building 1 - Part 1
    - Building 1 - Part 2
    - Building 2
    - Building 3 - Part 1
    - Building 3 - Part 2

      … more Building- Cubes
    - Building 23
    - Building 24
    - Building 25 - Part 1
    - Building 25 - Part 2
    - Building 26
  - Ground
  - Translucent-Roof
  - Items
    - Life1
      - Torus1
      - Torus2
      - Character
        - same as main Character
    - Life2
    - PowerUp1
      - Torus1
      - Torus2
      - Carrot
        - Body
        - Greenery
        - Greenery
        - Greenery
    - PowerUp2
    - PowerUp3
    - PowerUp4
    - AddTime1
      - Pyramid1
      - Pyramid2
    - AddTime2

      … more AddTime-Items

        - AddTime11
        - AddTime12
        - Key
            - Torus
            - Halm
            - Bart1
            - Bart2
            - Bart3
        - PowerMode
            - Torus1
            - Torus2
        - PowerMode
        - PowerMode
        - PowerMode
        - Star
        - Star
        - Star
        - Star
          …
    - Foes
        - Fox1
            - Body
            - EarRight
            - EarLeft
            - Tail
                - Part1
                - Part2
                - Part3
            - EyeRight
            - EyeLeft
            - Snout
            - Nose
        - Fox2
        - Fox3
        - Fox4
        - Fox5
- Character
    - Geometry
        - Body
        - EarRight
        - EarLeft
        - Tuft
        - EyeRight
        - EyeLeft

        - Snout
        - Nose
    - Camera
- Audio
    - Star
    - otherItem

3. Editor

Visual Editor: The Hierarchy with all Parent Nodes for all needed elements was created in the Editor, as the visual layout makes it very easy. Maze Elements and their components have been created in the editor, where they stay during the whole game and also stay the same way at any time. Plus, it was easier to create the maze while seeing it and being able to tell what needed to be adjusted. Other elements that don't need further adjustments, like the light and sound nodes, were easy to create in the editor as well. I created the camera in the editor as well, but that could have also been done in the code, I just preferred the editor as it was a bit faster to create it there and to immediately see, for example, how the joint between the camera and the character works.
I did, however, set the character camera inside the code as the viewport camera. The geometry for the Lives-, AdditionalTime- and PowerUp-Collectibles are created in the editor, then disabled and enabled via code. The amount is also randomised by code. The stars are completely created via code. The character model was created in the editor, but the setup was done in code, as there are many aspects that need to be adjusted.

4. ScriptComponents

I used a scriptcomponent to calculate and determine the positions of the obstacles based on how many collectables exist, so they get spread out evenly in the game. It didn't work out perfectly, so in general I'd say it wasn't that useful. It would probably be a useful part in other applications.

5. Extend

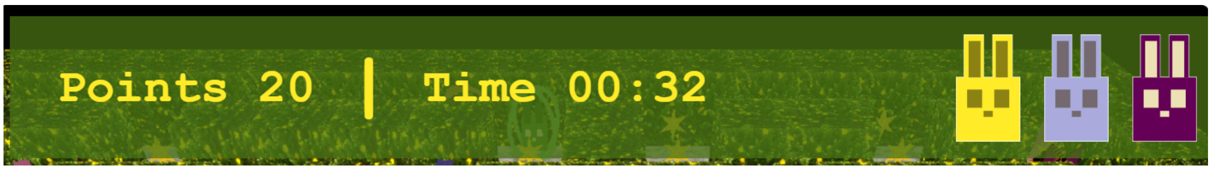
Other than the ScriptComponent, I derived classes for the Character as well as the stars and Items as ƒ.Nodes from Fudge Core. This was very useful for me, as I could use the classes to set the methods that are needed in the game.

6. Sound

I used sounds for the collection of stars, the collection of items, and at the end of the game. I got the sounds from freesound.org and cut and edited them to length. The Audio components are connected to the game graph.

7. VUI

I created a virtual user interface that shows the remaining lives and the amount of points collected. The VUI is placed at the top of the screen, the lives on the right and the score on the left, so that it can easily be seen and checked while playing. I also added a stopwatch for the playing time.



8. Eventsystem

Eventsystems were used to create a custom event called “collision” that checks for the collision of the character and an item. If those collide, the item disappears and activates its effect, as well as playing the eating sound. In addition, I made an event called “checkForFinalItem” that gets triggered in a case where all stars have been eaten. The use of the event system was useful, as it was an easy way to check for certain events.

9. External Data

In the external Data the maze is shown in 0 and 1 to determine if there is an empty space or a building placed. Since I couldn't determine or calculate the placement of the buildings via the code, I had to hardcode these, in order to be able to place items there.

A. Light

I chose an ambient light for Bunny-Mania, as there is no need for shadows in the Game. Additionally, most cubes are lit to make sure everything is visible all the time and that there are no darker spots throughout the maze.

B. Physics

The Bunny character has a dynamic rigidbody. All maze elements and items have static rigidbodies, so they can create collisions with the Bunny. The maze elements block him from moving out of the maze or skipping parts of the maze. The stars send an event to their state machine when a collection is detected, thus transiting into another job. Just the Foes, the Foxes have a kinematic body type, so that it moves within the animation, which sadly doesn’t work for me. (FudgeCore Wiki: Kinematic: The body is controlled by its node and moves with it, while it impacts the physical world e.g. by collisions)

C. Network

There is no network functionality used or implemented in Bunny-Mania.

D. State Machines

I used a Component State Machine for the Game Mode the Bunny is in. In the Normal Mode the Bunny loses a life if it gets hit by a Fox. In the Power Mode the Bunny is invulnerable and kills the Fox if they collide. Via the Component State Machine that was easily implemented.

E. Animation

I used the animation system to animate the rotation of the PowerUps-, Lives- and AdditionalTime-Items. Also for the ears of my character, its Power-Mode rings and the tail of the Foxes. Additionally the Foxes move along a specified path. The stars are not animated, as that would make for too much movement on screen.