PinguRun Design Document

- Author: Sarah Weidenhiller, Matrikelnr. 263128
- Year and season: Wintersemester 2022/2023
- Course: PRIMA
- Docent: Prof. Jirka Dell'Oro-Friedl
- Executable application: https://s-wdh.github.io/Prima/PinguRun/index.html
- Source code: https://github.com/s-wdh/Prima/tree/master/PinguRun
- Design document: https://s-wdh.github.io/Prima/PinguRun/Documents/PinguRun.pdf

Table of contents

Description for the users on how to interact

Use A and D or left- and right-arrow keys to walk and the space key to jump, while moving the Character "Pingu" through the Track.
The Goal of the game is to bring Pingu back home safely before the time runs out.
On the way you need to collect three stars, so Pingu is able to get home. After collecting a star, it flies up to the night sky, where you always can check how many stars you've already collected.
Additionally, you can collect coins for Pingu. These don't give you any advantage, but Pingu is thankful about anything he can get.

Checklist for the final assignment

1. Units and Positions

0 = start of the track on top of the bottom track block. This makes it easy to check, if Pingu falls down of the blocks, as his position then has a negative y-value. The x-value is at the beginning of the track, as that is where the game starts and placing it in the middle of the track or somewhere else wouldn't make sense.
1 = measurement of one block in width and height, as well as the size of the Pingu-Character and the collectables.

2. Hierarchy

The Track Elements are all collected together, so the Hierarchy in the editor isn't overloaded by the three different kinds of Track Elements. The collectables all get the same parent Node, as they are all there to be collected. For the child nodes of it there is no need to separate them, as they belong to two different classes based on their different aspects. The Camera is connected to the Character node as it needs to be moved with the character together through the course. The different Sound nodes are all bundled under the Sound parent.

Blue = nodes (/ components) added via code

Game

- Track
  - Bottom Track
    - Track Block with Track Elements
  - Top Track
    - Track Block with Track Elements
  - Bridge
    - Handle (joint component via code)
    - Swing
- Collectables
  - Star
  - Coin
  - …
- Character
  - Pingu (components via code)
  - Camera
- Light
- Background
- Sound
  - Nodes for all needed Sounds with attached ComponentAudio
- Igloo

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. Track Elements and their components have been created in the editor, as they stay during the whole game and also stay the same way at any time. Plus, it was easier to create the track while seeing it and easily being able to tell what needs to be adjusted. Other elements that don't need further adjustments like the

background, the light and the sound nodes where easy to create in the editor as well. I created the camera in the editor, but that could have been made in the code as well, I just preferred the editor as it was a bit faster to create it there and then only adjust the necessary parts in the code.
The collectables that need to be created and deleted after collection are easier to create in the code, as well as the character 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 brought me the advantage of needing those code part only once in the scriptcomponent and not twice in each collectable class. So, in general I would say it wasn't as useful as it could be in other applications, but it also wasn't unuseful or additional work for me.

5. Extend

Other than the scriptcomponent I derived classes for the Pingu-Character as well as the Coins and the Stars as f.Nodes from Fudge Core. This was very useful for me as I could use the classes to set the methods which are needed in the game.

6. Sound

I used sounds for the collection of stars, collection of coins, when Pingu is jumping as well as at the end of the game in case of loosing and winning. I got my sounds from pixabay.com and mixkit.co. The Audio components are connected to the game graph. I choose to not use a background music as it is unnecessary for a game like this in my opinion.

7. VUI

I created a virtual user interface that shows the remaining time of the game, the amount of collected stars and the amount of collected coins. In front of every number is the description what exactly it is. The VUI is placed at the bottom left corner of the window, so it can be easily found and checked while playing.

8. Eventsystem

I used the Eventsystem to create a custom event "checkGameEnd" that gets triggered in a case where the game ends. Those cases are: timer has reached 0, Pingu falls down and Pingu has reached the igloo after collecting three stars. The detail of the event contains the reasons why the game ended. Loose means Pingu fell down, win that he reached the igloo with the three stars and timeOut, that the time of the game has ran out. For each of the cases there is a different alert letting the player know why the game has ended. Additionally, the winning and loosing sounds are played and the event listener of the Loop frame gets removed, so the player can't go on with playing. The use of the event system was useful, as it was an easy way to be used in all kinds of game end cases.

9. External Data

I used the external data to load the amount of stars, amount of coins and the game duration into the application. This is useful as they can be easily adjusted this way to make the game harder or easier. It also would be possible to add different levels of difficulty that way and then adjust the content based on the chosen difficulty.

A. Light

I choose an ambient light for Pingu Run, as there is no need for shadows in a 2D Game. I had the idea of making the scene darker over time and the collected stars lighting it up so the player can see more. I decided against this idea as I wanted to rather work with more physics components in the game to practice using those.

B. Physics

The Pingu character has a dynamic rigidbody. All track elements and the stars have static rigidbodies so they can create collisions with Pingu. The track elements block him from moving into that direction when he hits them. The stars send an event to their state machine when a collection is detected and thus transit into another job.

For the jumping movement of Pingu I used the applyLinearImpulse method of its rigidbody.

The bridge track element is a Joint Revolute which swings down when it is hit with enough force by the character. In order to do so you normally have to jump up and hit it at this height.

C. Network

No network functionality is used in PinguRun.

D. State Machines

I used a Component State Machine for the stars as it was easy to distinguish their three stages of live: Idle (being on the track, waiting to get collected), fly (flying up in the sky after getting collected), shine (being placed above of the track)

E. Animation

I used the animation system to animate the rotation of the coins as well as to animate the flying up of the stars after their collection.

Pingu has a Sprite Animation for his sprites so it looks like he is actually walking and not sliding along the track.

Sprites and Textures

I designed and drew all Images in the program “Pixelorama” in the Pixel Art art style. For some of them I used some pictures as inspiration.