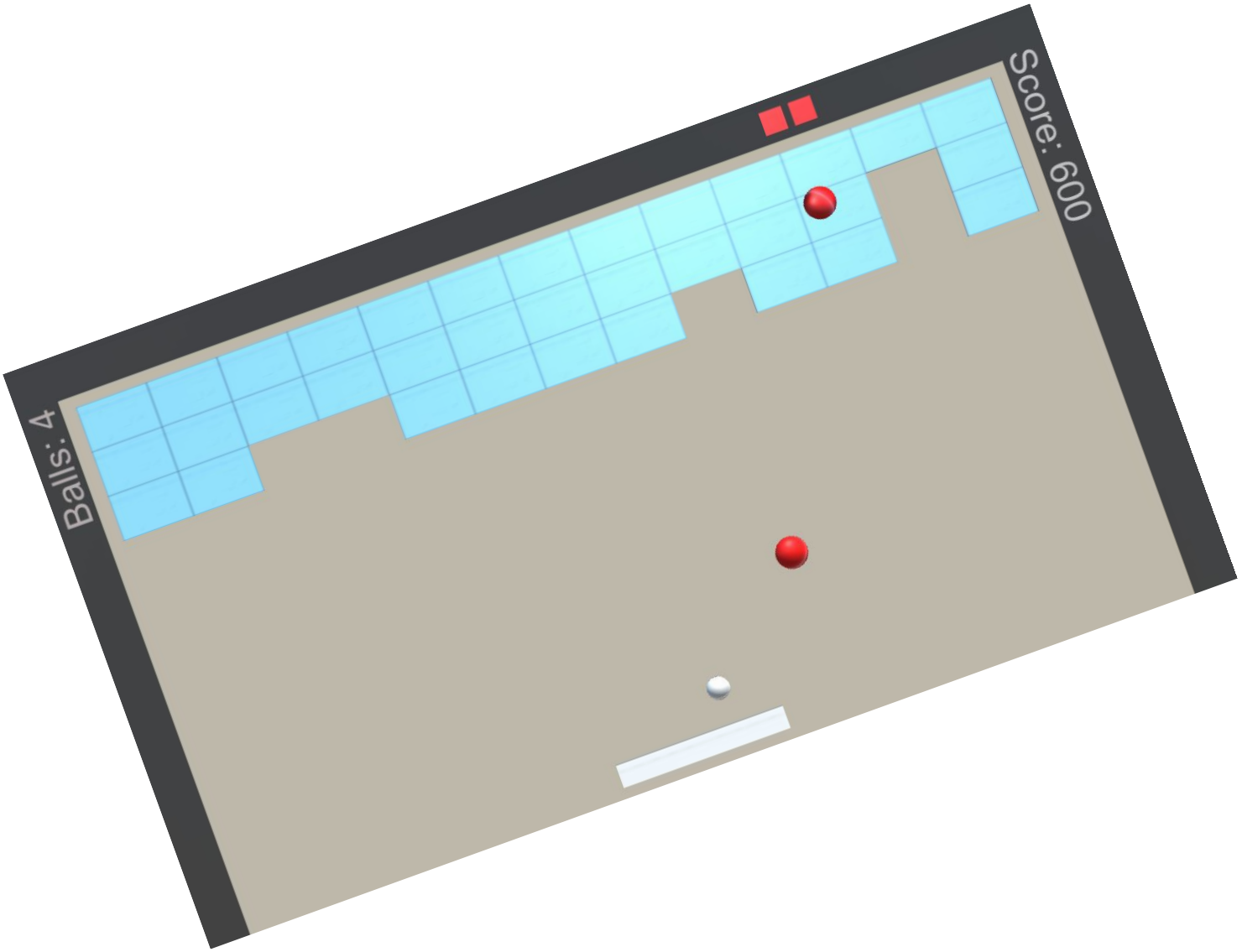
Dodgeball Breakout

Kern module 1: Retro game with a twist



Chantal van der Voort

3031042

19 September 2019

Index

The Game

Retro game

The retro game is Breakout. The player has to break the stones at the top of the game with a ball and has to prevent the ball from dropping out the gameplay field. The player does this by controlling the plank at the bottom of the game. When the ball falls down, the player moves the plank to the ball and hits it back up upon colliding.

Twist

The twist to my game is to add Dodgeball, or more exactly dodge rocks. In my game there will be enemies outside the gameplay field at the top and the sides. These enemies will throw rocks towards the player. The player has to dodge these rocks, a part of the plank will be destroyed when the player is hit.

Gameover

It is gameover once the player has dropped too many balls or if the whole plank is destroyed.

Inspiration

When I was younger I often played breakout. It’s a simple game and doesn’t have many rules, anyone can play it. I was wondering how it would be if I added a challenge to it. What if you have to dodge objects while still trying to avoid to lose the ball.

Code Structure

I tried my best to keep scripts from talking to each other, using mainly the manager to talk to the other scripts. During this project I also tried to use as many new things we learnt, but made sure that it made sense to use them.

First times

- SerializeField/ Private
    - I already knew about SerializeField but I never really used it before, I mainly just always put everything to Public so I could edit it in the inspector. This time I decide to think more carefully about making things public and tried to keep as many things private as possible, if they weren't supposed to be used by other scripts. I also tried to keep many variables private to search them up, instead of being depending on the inspector.
- Properties
    - Another thing I was aware off, but I never used it as I didn't understand how they work and why you would use them. Now that I understand them, I think its cleaner to use this method instead of double variables all the time.
- Dictionary
    - I've heard about this when I was working on my final exam project during my previous study, but didn't have time to figure out how it works. Afterwards I had forgotten about it and was remembered to it during these classes.
- Script based Object Pool with Queue
    - Normally I would always just throw a lot game objects in the game, from the inspector. Then get the parent holder and grab the children. This time I only took the prefab and parent holder, and created the objects in the script. This way I have more control on my objects and is it easier to edit the amount of poolable objects I want.
    - In my old way I would often just use an array or a list, this time I used Queue so I can quickly grab the first object in the pool.
- Generics
    - This is another thing I never touched cause I didn't understand them. But now that I finally understand how this works, it's really easy and useful.
- Singleton
    - I heard the word Singleton before but didn't know what it was and never looked it up. Normally I would just make a variable to get access to other classes instead of using singletons. But from now on I will probably use this method far more often on important classes.
- Interface
    - This was totally new to me, I hadn't even heard of it before. Everytime I heard interface I got confused, thinking it had something to do with the UI. Even though I understand what these are now and how you use them, I still find it hard to properly use them for now.

Unfamiliarity

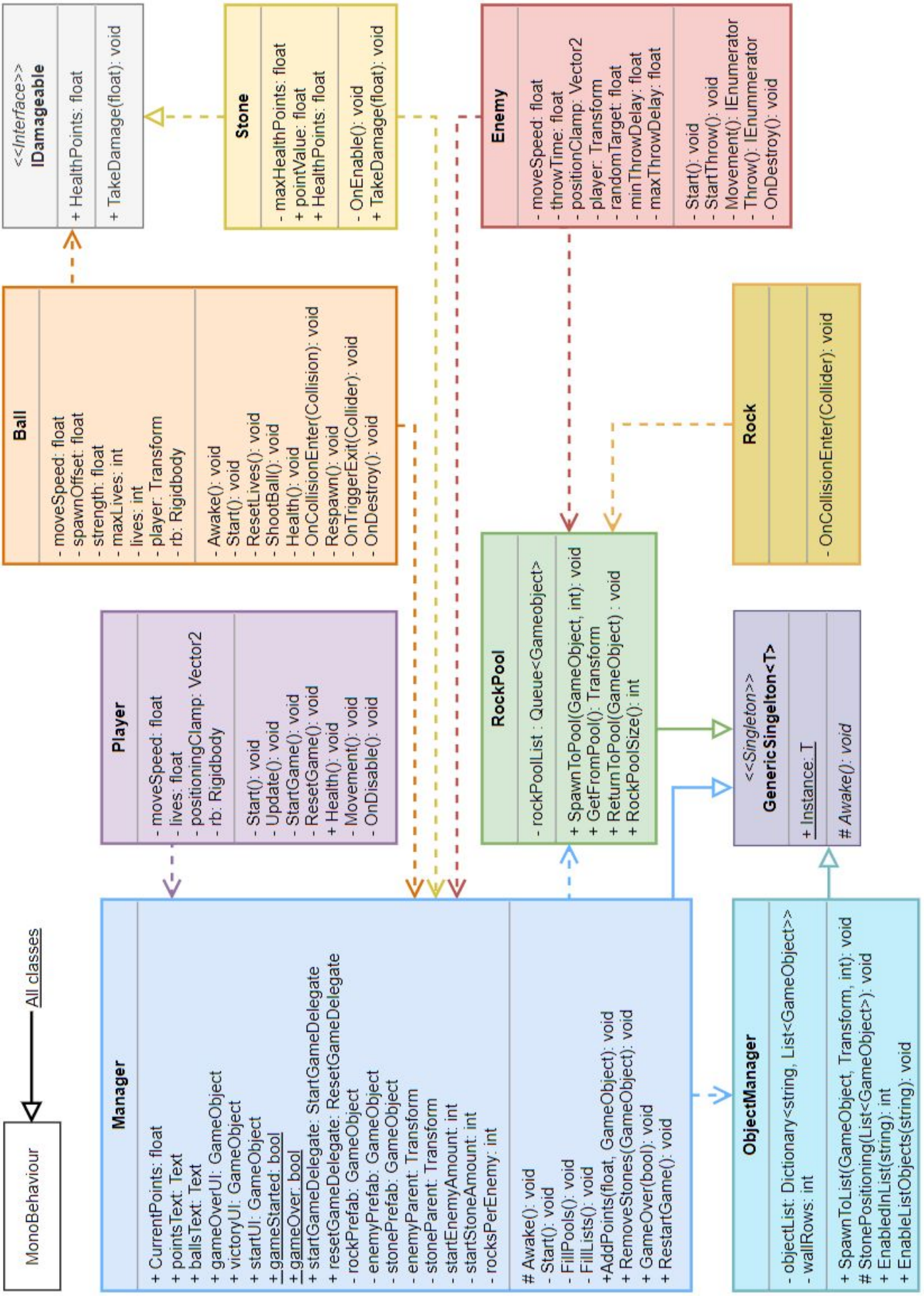
- Inheritance
    - I learnt these at my previous school in the second year, but up till half a year ago I never really understood how to use them properly. I finally started to try using them but still often try to avoid them.
    - In this project I used it for the generic singleton multiple classes were using instead of creating it on each of these classes.
- Delegate
    - I used this one time during the third year of my previous study but totally forgot how it was called. A while ago I needed to use a delegate but didn't remember how it was called, with my search keys I came to SendMessage which helped me back then but now I can use this way again.
- UML
    - In my previous study I learnt what a UML is but they didn't explain how to make it. I struggled a lot with making this, mainly with which arrows I have to use and which direction they have to face.

Problems I came across

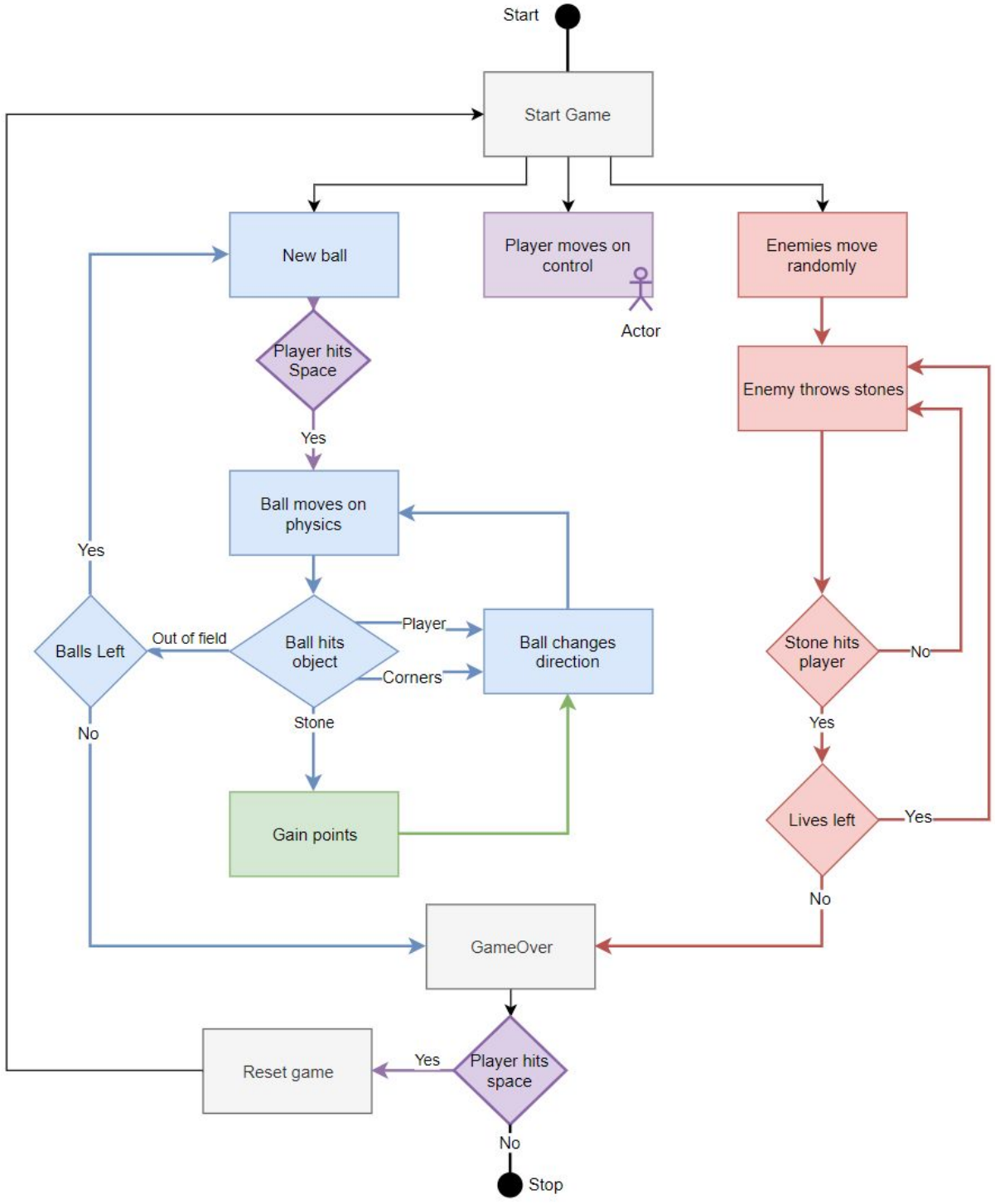
- First week
    - The first week I had a bad week, progress went very slow no matter how simple the part was. This caused a lot of stress as you immediately lose ⅓ of your time in the project. Lucky I was able to pick myself up again in the second week and still managed to finish in time.
- Execution Order
    - I tried to play with inheritance in combination of a generic singleton and ran into confusing problems where it would one time not work and another time did work. In the end it seemed to be an execution order problem. This problem solved when I moved more things from the Awake() to the Start().
- Delegate
    - I wanted to try using a delegate but just couldn't find a logical place where to use it. Since I couldn't find a purpose, I decide not to use it. But in the end I found a way to use it for starting and resetting the game. Sometimes It helps to not overthink things and just let them go first.
- Dictionary List versus Queue
    - I considered making the dictionary a queue and let the pool also use the dictionary, but I had to access the objects by object so I had to use a List. I also considered to let the pool use a List like I usually do but I wanted to let the pool use a queue to easily access the first in the list.

UML

UML Class Diagram



UML Activity Diagram



Credits:

Helped with my Uml

Helped with my Singleton