Requirements for PinBallsim

For the course DFSS3101-1 21H Utvikling av smarte systemer

I set out to create a simulation of the pinball machine, using the game engine Unity.

To begin with I created a set of requirements to work form. And during the course these requirements have been modified and extended upon. I will here list all of them and evaluate if they have been fulfilled.

A requirements

A 1

Have a setup that mimics the final design of the physical pinball machine.

Done

The board looks like the final machine, with correct dimensions. However not all the components of the physical machine are implemented. This is because some of the more complex shapes are difficult to implement in Unity. And therefore needs 3 party software to model. Software that I have no experience with. Also, the “ball trench” is not curved on the top. The CAD files that were made for the physical machine can’t be used for this purpose, because there is around 100 parts with several hundred individual surfaces. Wich is something Unity wouldn’t like that much.

A 2

The coded functionality should mimic the actual physical system.

Done

The flippers, matrixes, goals and to a less extent the pull spring. All work like the physical system. However, some functions are not fully implemented yet. For example, the matrixes don’t calculate exactly where and when on the flipper the ball will hit. This is because the code pretends that the flipper is horizontal. Whilst it is slightly sloped. Also, the pull spring was designed before the physical one. So, I just had to make something that works. However, since the physical one was made, I have stopped using the pull spring for testing purposes. And kind of forgot about it even existing.

A 3

Performance should be good enough.

Done

The simulation can run consistently at 300+ fps while the sensors is running. However, there is something with my laptop that limits Unity to run at just 30 fps. This started midways in the course.

B requirements

B 1

Automatic control of the flippers.

Done

The matrixes read in the data from the sensors and will calculate where the ball will hit the flippers, based on this data. It calculates the distance to the flipper and the speed the ball travels at. It will then tell the flipper to activate based on the distance from the last sensor reading to the flipper and the speed of the ball. With this it can play the game on its own. However, the calculations are not entirely complete as it doesn’t consider that the flippers are sloped. This was not done because focus was shifted towards getting the machine learning up and running instead. I also had trouble figuring out how to calculate this, as it needs to be done dynamically and whilst a Raycast from the ball would work. We don’t have access to this in the physical machine. Therefore, I didn’t want to add a Raycast to it.

C requirements

C 1

Machine learning.

Done

I have added Unity.ML to the simulation that is a third-party machine learning library that works with Unity. The simulation has fully implemented this library and have been tested and trained. There are some small issues with it, however, it does get smarter the longer it trains, and everything seems to work fine.

C 2

Upload the Unity simulation to the raspberry pi and connect it to the serial ports. To control the flipper with machine learning.

Scratched due to time constraints.

I started to read upon how to do something like this, but never had the time to implement this.

C 3

Display the physical board state to an external monitor connected to the raspberry pi, based on the Unity simulation

Working in theory.

To extend the last requirement, I wanted to connect a monitor to the raspberry pi to show the state of the board and what the Unity.ML Agent was thinking. This works in theory as the Unity simulation is set up to do that. However, with the previous requirement scratched. There was no point in pursuing this. Unity.ML comes with a build in feature that displays what the Agent is thinking, to a web page. The idea was to embed this page to a custom web page, hosted on the raspberry.