Software Specification

Design Document

Group 10: Adam Teehan, Bobby McGonigle, Cathal Kinsella, Owen Auch, Lauren Duffy, Svetlana Cvetic, Dan Ormsby

1. Introduction

1.1 Purpose of System

Our system's task is to have a child friendly game consisting of a 2D plane with a path. The user will be able to run around the plane as a circular race. There are several objects that the user will try and avoid as part of the game. The purpose of our game will contribute to the clients study of engaging young children with computer science and programming. The game aspect will engage the students whilst the logic of the game will improve their problem solving and deductive reasoning skills. There are a myriad of impacts the development of problem solving skills at a young age can bring and help them in their current and future education.

1.2 Scope

In Scope

- Start screen with buttons for "Start", "Options", and "Quit"
- Options screen with the ability to change input types between keyboard and Kinect
- Simple level select screen
- Easy to use and cohesive UI
- A set amount of tracks that increase in difficulty
- Generic obstacles and track code to be used in other projects
- Basic character

Out of Scope

- UI aesthetic - making the UI colorful or overly artistic and beautiful
- Alternative options - options other than changing input type
- Complex level select screen
- Custom 3D unique character
- Randomly generated tracks

1.3 Definitions, abbreviations

There are no relevant definitions or abbreviations needed for our project.

1.4 References

https://docs.unity3d.com/Manual/index.html - Unity User Manual
https://unity3d.com/learn/tutorials/s/roll-ball-tutorial - Roll a Ball Tutorial
https://stackoverflow.com/questions/37937984/git-refusing-to-merge-unrelated-histories
- Helped Solve Git & Unity Problems
https://unity3d.com/learn/tutorials/topics/user-interface-ui/ui-canvas - UI Canvas guide
https://docs.unity3d.com/ScriptReference/UI.Button.html - Button Class

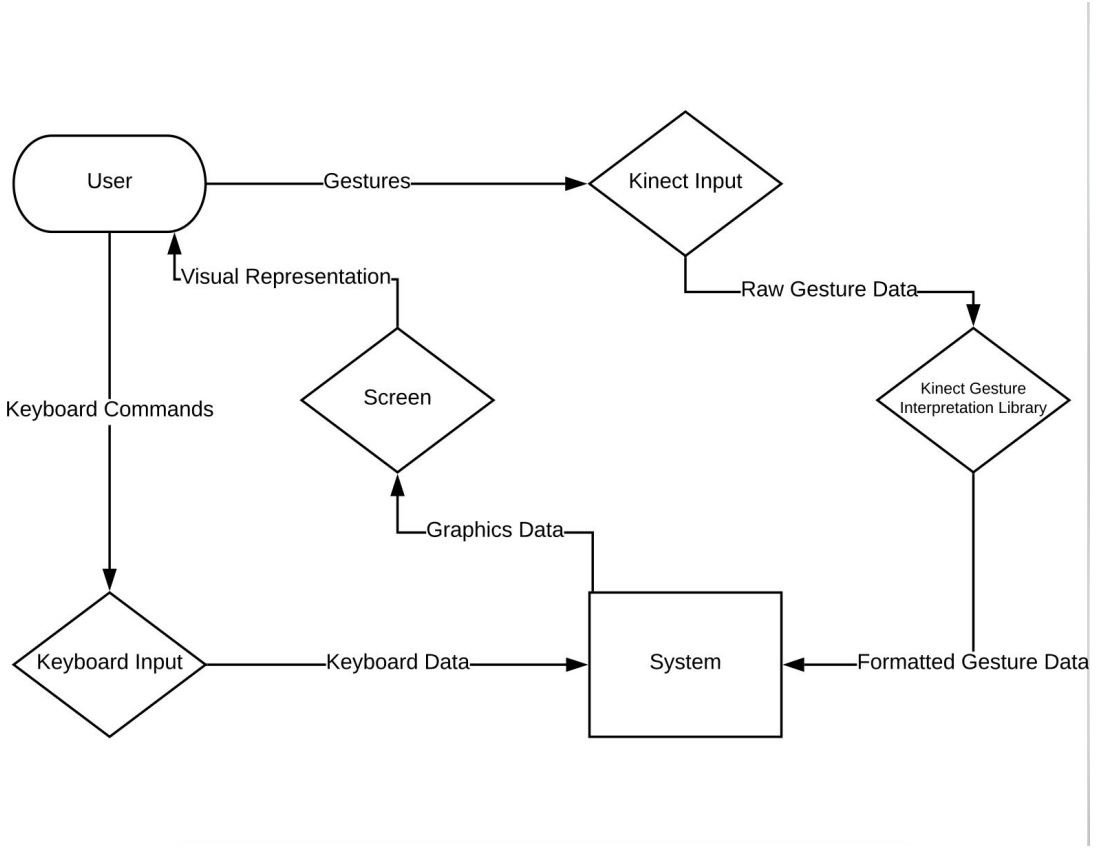
2. System Design

2.1 Design Overview - Tools, Frameworks and Languages

The game itself will be developed in C#, one of the main languages supported by Unity; The game engine we will also be using. Unity is a multipurpose game engine that supports 2D and 3D graphics, drag-and-drop functionality and scripting using C#. The game itself will be built from the ground up by our team, with the exception of default Unity assets and some imported ones. Based on our clients requirements we will be writing our code as generic as possible, so that our game can be developed on and implemented with other projects in the future. For example with Kinect controllers.

2.2 System Design Models

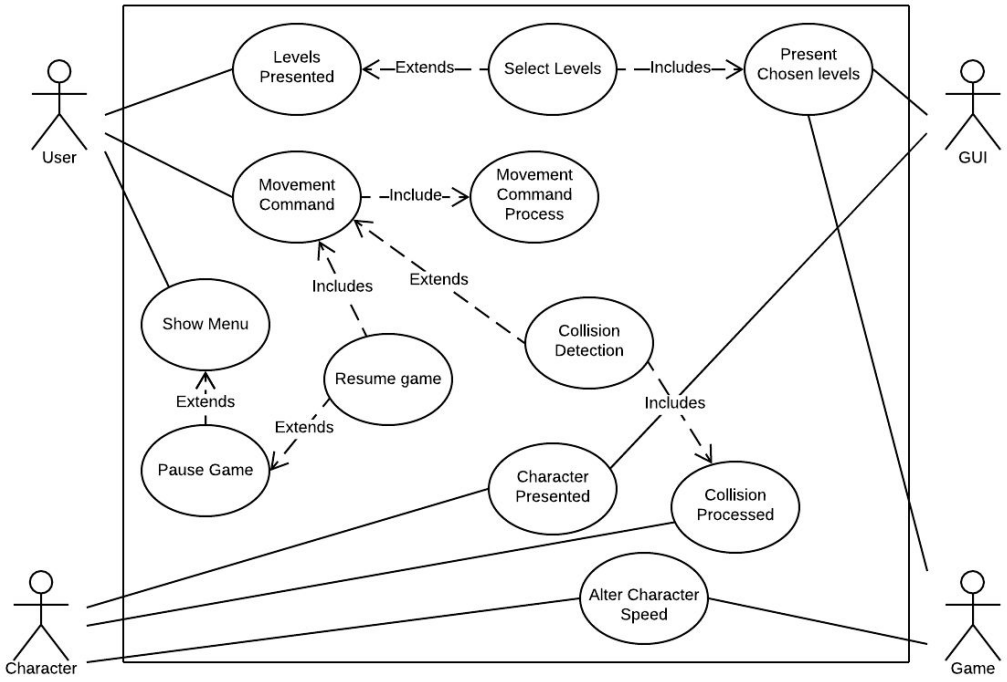
2.2.1 System Context



Explanation of System Context:

Although we are not responsible for implementing Kinect input in the game, it is the ultimate goal of the client's extended project so it feels suitable to include it in the system context. Our system will respond to user's input via keyboard commands and represent their input on the their screen.

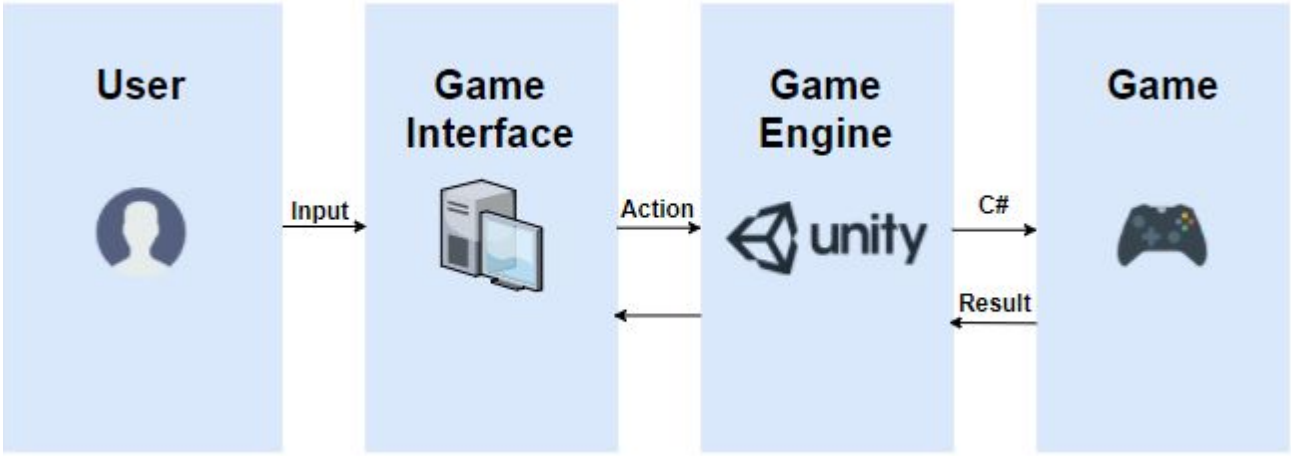
2.2.2 Use cases (from Requirements)



Explanation of Use Case:

The User interacts with the system in three ways, which are controlling the characters movements, selecting a level and pausing the game. For the purpose of readability, the movement command process is the combination of dodge, jump, duck and speed control for the game. The GUI will display the level and the character traversing it. Collisions with obstacles will trigger end the current try for the User.

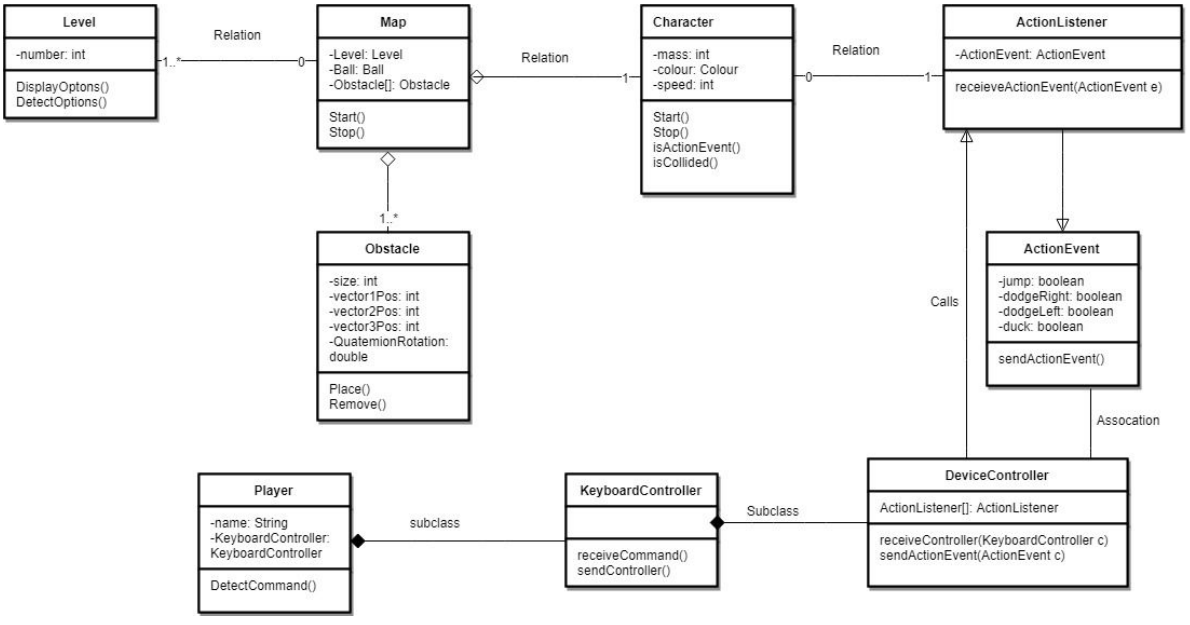
2.2.3 System Architecture/Components



Explanation Of Architecture:

The user will will input the gesture/movement they want to make into the game interface (this code will be generic so ways of input can be changed). Unity, our game engine will then take this action, perform functions in our C# code and the game will return the result to the game engine and subsequently display it on the game interface to the user.

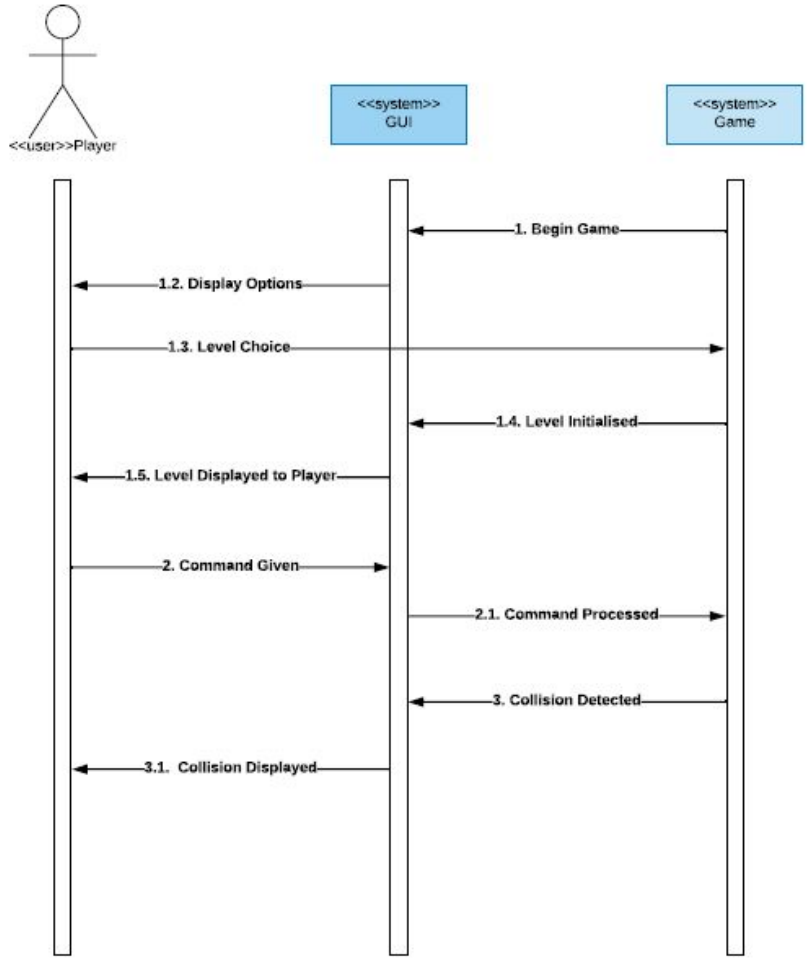
2.2.4 Class Diagram



Explanation Of Class Diagram:

The selected map controls the obstacles that are generated. A more difficult map will have more obstacles spawning. The device controller must easily portable as the client plans to use the Kinect as a controller and not a keyboard.

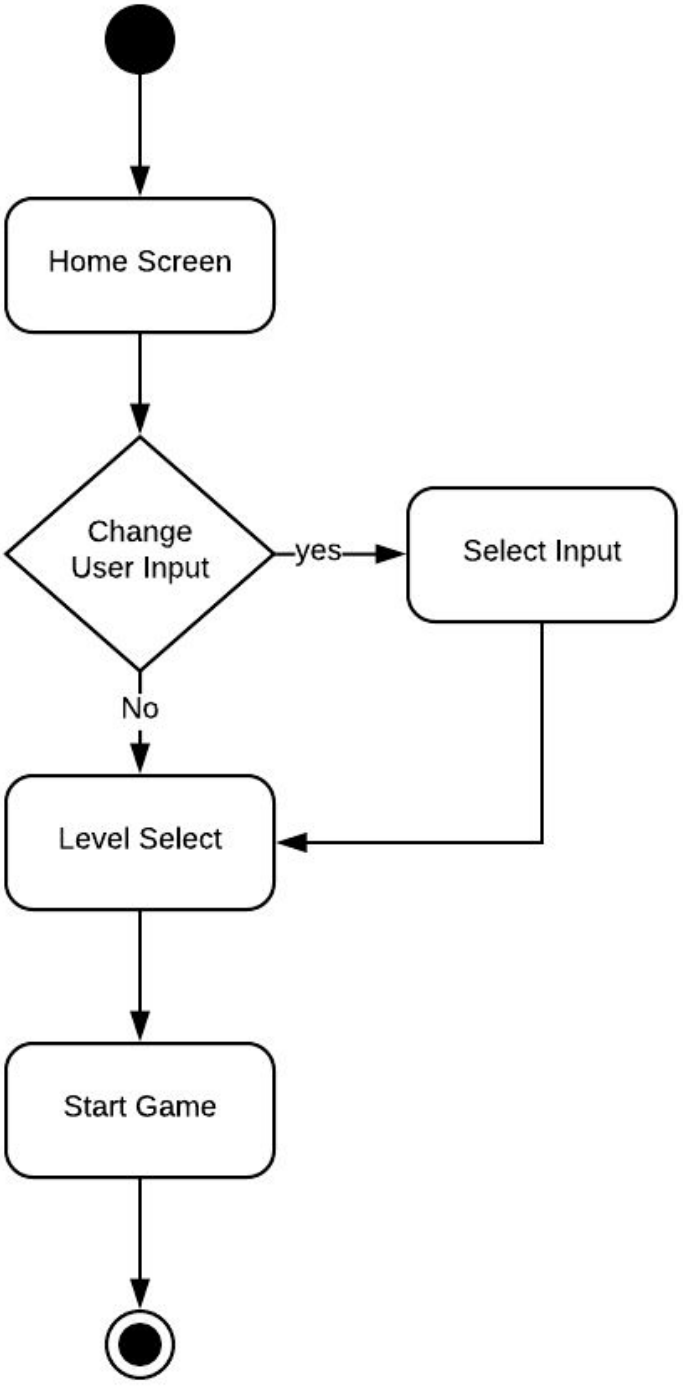
2.2.5 Sequence Diagram



Explanation Of Sequence Diagram:

When the game begins, the GUI will display a set of options for levels, and the user will provide input to choose a level. Then the game will initialize the level, and the GUI will display the level to the player. Then, while the game is running, a player will input a command to the GUI, and the command will be processed by the game. Finally, when a collision is detected by the game, it will display this to the user via the GUI.

2.2.1 State Diagram

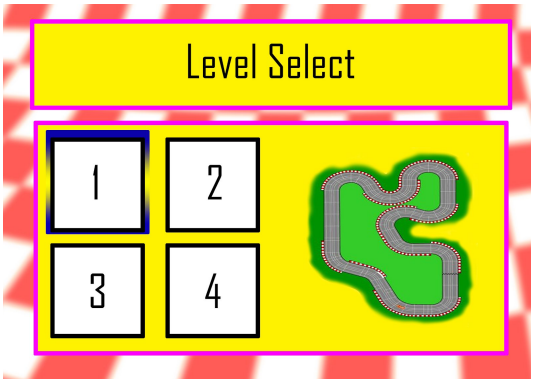


Explanation Of State Diagram:

This is how the user will start the game and select their preferences prior to starting a level.

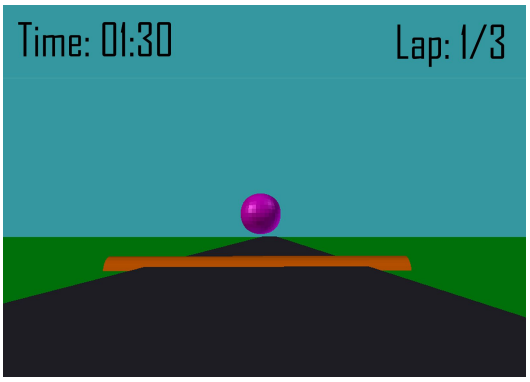
Design Mockups

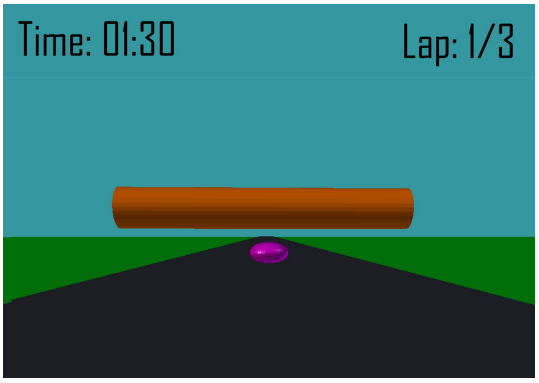
Title Screen

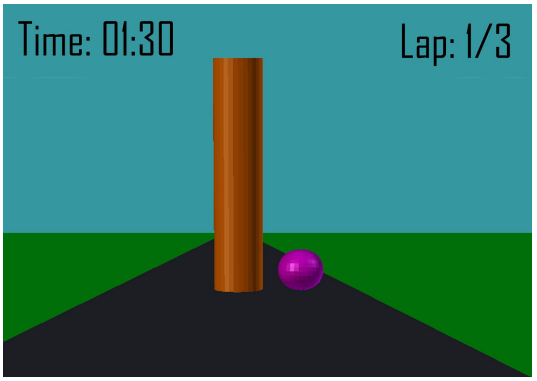


Level Select Screen

Gameplay Design concepts (Jump, Duck & Dodge movements)







(Initial mockup, the character will no longer be a ball but an avatar)