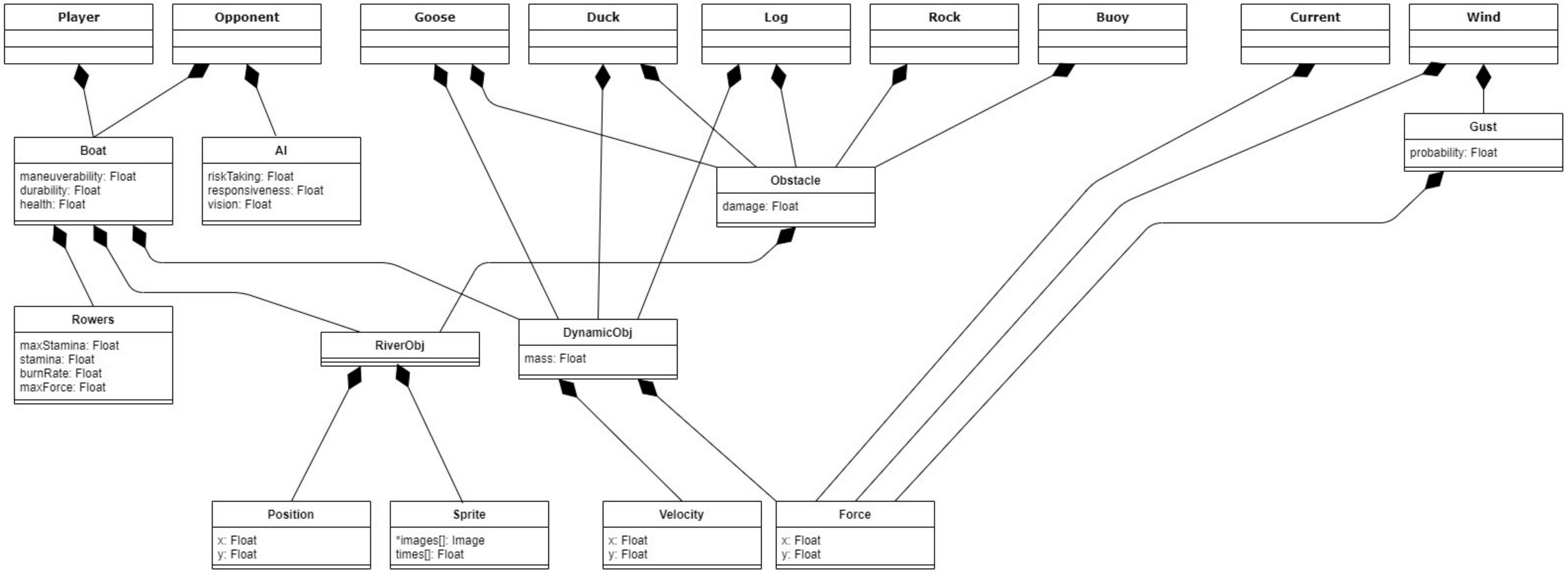
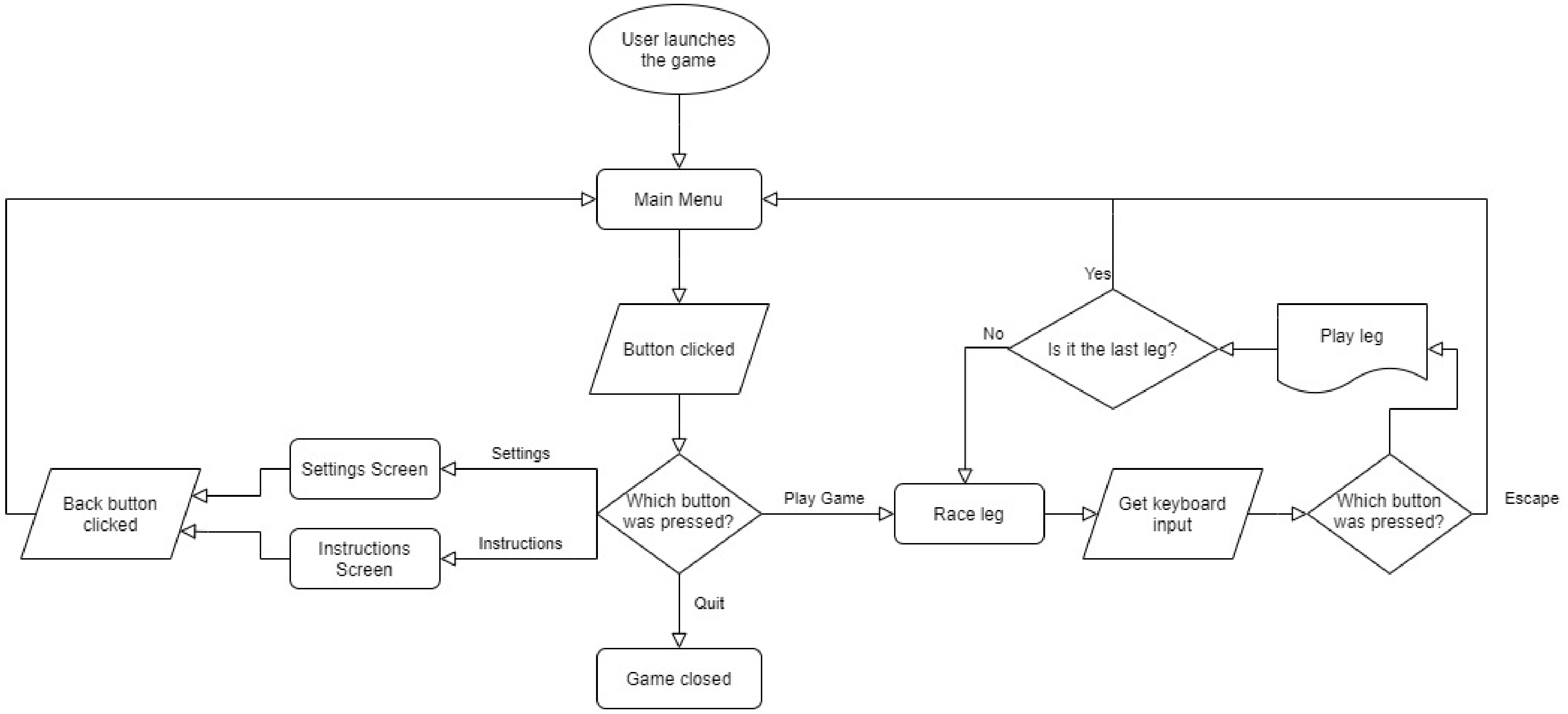
ARCHITECTURE

Quackthulu: Aaron Price, Alec Coates, Charlie Curedale-Rayner, Eleanor Griffin-Smith

To design our abstract architecture we used Diagrams.net (Draw.io) because it has an easy to use user interface with a multitude of predefined elements. It also has native Google Drive integration, which allows us to share changes quicker and more easily as a team.



We have modeled the abstract architecture for our game's objects using UML because it is a well recognised industry standard. This means that everyone who might need to read it should be able to understand it, which lowers the risk of confusion/miscommunication that could lead to future project delays and/or could cause the mis-implementation of systems.
Rather than use an entity hierarchy we are going to use an entity component system, which allows objects to use different elements of similar purposes, here this can be seen being used in the boat class, which has a RiverObj object and a DynamicObj object that have similar uses and would have a more complicated implementation in an entity hierarchy system.



We have modeled the abstract architecture for our game's logic using a flowchart because it is well suited for the task of logic flow, and is also widely used and understood by software engineers. This flow shows how the different game screens will interact with each other and what order they will be presented to the user in. Having a simple order of game screen logic is important to making sure the user does not get confused by the layout of the software and instead finds it simple and intuitive to navigate. Using screens to separate the game/menus makes code implementation much easier as it allows developers to work on separate parts of the code much easier and allows getting rid of old runtime objects in one go, rather than having to remove then re-add all the necessary objects.