This document will outline the main risks we will face while developing *Ship Happens* as a team of two programmers. As a team we have limited knowledge of design theory and will be spending a couple of weeks at the start of the project researching into the primary design theories to help support and guide our choices about any aspect of the game’s design.

# Gameplay Experience:

The game will require players to recognise what obstacles are, whether they are *Hazards* or *Interactions* and how to manage these as a team each time they appear within the game world.

## Telegraphing

We will try to make it clear to the player what obstacles currently require the players attention we aim to do this by;

* Implementing speech bubbles shouted out by an NPC sitting in the crow’s nest. Information contained within these speech bubbles will alert players to
  + An enemy location.
  + How long the level has left.
  + Player ship’s remaining health.
* Placing a hit marker for enemy cannon fire on player’s ship so players will know where a hazard is about to appear and where to avoid.
* Audio to help alert players to some information about the game, this will work alongside the speech bubbles. Some of the audio we will include will be;
  + Crow’s Nest callouts.
  + Sea Shanties.
  + Cannon Fire.
  + Damage to the player or enemies ship.
  + Seagull cries.
    - Splat noise of seagull excrement.
* Type of Enemy Flag to show the player what enemy is coming, this allows them to prepare for the enemy having increased damage, or increased speed and any other changes across the different enemy types.
* Flag Moving Down Screen for Enemy Ship Location to show the player where the enemy ship currently is in relation to their ship and cannon location.
* Cannon throbs or animates to show when the enemy ship is within the *Hit Zone* and can be fired upon successfully.

While all the ways we are planning on telegraphing interactions to the player will be easy to implement, they will require rigorous playtesting to ensure that players are aware of what they all mean and are able to react accordingly. To mitigate this issue as much as we can we will spend time designing paper prototypes and using whiteboards to ask potential play testers if they would understand before we use development resources to develop them into the game and find out they do not work.

## Game Balance

Balance of the game is crucial to the success of the game, if the game is too easy then players are likely to get bored very quickly and stop playing, however if the game is too hard then players will become too anxious and again, stop playing. As balancing of the game requires a lot of design work to mitigate this issue to find the correct balance between Boredom and Anxiety we will have to spend a lot of time playtesting through each iteration of the game ensuring the correct balance is found.

## Level Design

The design of each level will require a lot of attention as players could create a winning strategy that is not intended by the game by finding an exploit. The design of the level can also help to influence players choices and guide players down the correct path. To mitigate this issue the team have decided to use a similar game space across levels, vastly reducing the amount of time needed to design large levels and work in a highly play tested area. Playtesting again will be crucial to getting this right, especially since players will be in principally similar areas we will need to ensure players do not get bored either. The team must account for and allocate enough time to get playtesting complete without dampening too much time for the rest of the project.

# Visual Assets

## 3D Models

As a team we have very limited experience with creating 3D models, both members have spent some time using [Blender](https://www.blender.org/) in the past any models that we would be required to make would be of a very low quality and very time consuming to make.

* + To mitigate this issue, the team will be looking around on the internet at potential *paid* assets that we could use from places such as the [Unity3D Asset Store](https://assetstore.unity.com/).
  + Another solution to mitigate the problem would be to outsource the work to an individual or team of modelers who would be able to produce the work for us, however this produces potential reliability / availability issues where the outsourced member may just disappear or might not be able to produce the quality of work needed within the schedule of the game’s development.
  + A solution that the team spoke about and will potentially use to mitigate the issue would be to source *free* assets from places such as [Kenney's Game Assets](https://www.kenney.nl/assets) or [Dev Assets](http://devassets.com/) however, by using premade assets there could become an issue where the assets will not follow the same design throughout the project causing a break in the immersion.

## Animations

Animations are a way that as a team we can telegraph the hazards and obstacles to the player.

* Henry has experience with the animation system within the Unity3D Engine which could prove to be useful, however, that doesn’t extend much further than past the basics and would still prove to be time-consuming.
* Another solution to using animation would be to use the particle system which both member of the team has experience in, and although this will also be time-consuming for the team the tasks can be split between both members, ideally, we will have a mixture of animations and particle effects.

# Programming

The team comprises of two programming, so although some basic design principles can become a huge risk, the code side of the project should be relatively low risk. We have never worked as a team of two comprising only of programmers, however we have worked together before on other projects which mitigates the risk of a new team having to learn each other’s abilities and skill sets.

We are both aware that over scoping is a potential problem and as a result we will be constantly checking our own timelines and design documents to ensure we are able to produce a Minimum Viable Product (MVP) within the allotted time. Additionally, we will constantly keep in contact with our design tutors (Rob Kurta and Dave Pimm) to ensure we are not going too far away from an MVP we could realistically create in the timeframe.

As programmers who have worked with other programmers we are both aware how messy and unorganised scripts can potentially get, causing a loss of time while deciphering the code. To mitigate this issue, as a team we have decided to have at least weekly code reviews to provide a run through of any scripts produced to ensure both members of the team understand them.

In addition to code reviews we will also be selecting a naming convention, and preplanning scripts together to ensure the most efficiency. As a result, we will be spending a lot of time in the labs as a studio setting working together to increase production time.

Saving the players profile could be a potential problem since neither team members has worked on this before, however after searching the internet and speaking to our programming tutor (Chris Janes) we have found that this shouldn’t be too difficult a task with the help of Unity’s built-in “Player Prefs” and other helper plug-ins from the “Unity Asset Store”.

Both members of the team have worked within Unity3D before and therefore expect there to be no risks with using this software to produce the intended game. Both members are also familiar with developing and producing a PC game with experience making the games compatible with an Xbox One controller. However, we believe porting the game to console may well be out of the scope intended for this project.