“Back To Blue” Technical Risk Assessment

Patrick Gregg

Team Orange

2-25-18

Game Production 1

“Back To Blue” is a sea-based educational game intended to raise awareness of oceanic pollution among younger students, in development for Unity by Team Orange for Windows and iOS. All risks associated with platform are purely control-based, since the Unity engine is capable of easily building for both platforms and there is very little risk in graphical capability. Windows-based controls are WASD based, using only a few buttons other than the movement keys. For iOS, this will have to be transformed to allow for the player character to move towards the touched part of the screen and have a seperate button to control the rope.

From a gameplay standpoint, there is very little risk in Back To Blue’s fairly simple mechanics. The game’s main mechanic, using line to drag objects, is a simple enough task that requires some basic physics for coding, however, Unity already provides many of the needed tools for such an idea. Simple physics are all that is needed, to allow the movement of creatures, objects, and the player around the level. All systems should be able to handle Unity’s physics engine, and thus the game seems unrestricted in terms of platform limitation. Since the game is intended to be played on school computers, this can prove incredibly helpful considering the broad range of possible public school computers.

The most risky aspect of the project is likely going to be the artificial intelligence, which is required for the various sea creatures that are planned to make appearances in the game. The AI has to be predictable enough for the target audience and complex enough to make for interesting interactions in-game, all while representing the real-life actions of the creatures. Thus, the currently planned AI will be simple, mostly limited to interaction with the player and sometimes various objects in the environment, such as trash and pollution, in order to keep in-scope with the project. However, with multiple levels, new AI will likely be needed, and thus it is expected for AI to be created or re-used. The more levels planned, the riskier it becomes.

While AI can be a bit of a risk for larger amounts of levels and will no-doubt require playtesting, the project is simple enough and certainly in scope with the current tasks and planned features. The game is also open enough to allow for additional features to be added in the case that something were to change, making the game rather technically flexible.

The art pipeline is strictly 2D, with some parallax scrolling mechanics and layering to consider when implementing new code. Perhaps the most difficult aspect to it will be linking the art to gameplay feel, i.e making colliders that feel consistent with the art and a story that compliments it. Unity is, again, key to the art pipeline and the ease of use to implement art.

GitTortise is used as version control for this project, and it is crucial for design implementation. Unfortunately, the lack of familiarity with git and various issues with the repo have occasionally caused issues with the build over the milestone, including accidental overwrites and inability to access the build via normal channels. There seems to be some risk with version control, although it is likely going to become safer as it becomes more natural for the team to use.

For milestone 3, technical risk has still been dominated by Git. At times, files are accidentally lost or issues with pulling cause delays in progress. It seems like there was not enough time for the team (including myself) to get used to all of git's features and uses, which has had clearly negative effects on our ability to work. Despite this, it remains a fairly minor nuisance.

Besides version control issues, the major risks for this part of the project mostly fall to Unity's object system and making each level simultaneously self-contained and a part of a larger project that uses the same sets of tools that we've been creating and refining from the past milestones. Making sure that each tool is compatible with every scene has the potential to be the most time consuming and co-ordinating task of the milestone if things were to go wrong. Especially as the group works to finish up the game, setbacks and unexpected issues can easily make this project much more difficult. This avoidance of this problem via communication and coordination when it comes to tool implementation is critical.