***Henry Crofts – Postmortem – S183848***

|  |  |
| --- | --- |
| **STUDENT NAME** | Henry Crofts |
| **PROJECT NAME** | Ship Happens |
| What do you think went well on the project? | I believe that the project started strongly, as a team of two programmers we did not have a designer to help with the design aspect, or in the creation of assets. However, Tom and I sat down together and worked out both of our strengths and weaknesses, most of which were relatively similar, although Tom explained he had some experience with 3D modelling but to a very basic level.  After discussing our strengths and weaknesses we proceeded to create a risk assessment form to help us prioritize tasks and investigate ways to mitigate any issues that might arise from areas we might not be very strong, for example needing extra time to model any assets, and areas which wouldn’t be such an issue such as programming most of the system. The risk assessment form was updated for most of the project. Alongside the risk assessment form, we also created an availability spreadsheet to help setting up meetings during times when we would both have the most availability. This helped the team keep on top of regular meetings and studio jams.  As part of the mitigation decisions the team were able to find an asset pack made by Kenney (Kenney.nl, n.d.) which helped with the theme of the game, while most assets were usable the main ship model was not and was not easily editable as it used textures rather than materials (as stated on the page). This led to Tom having to model a ship removing him from programming tasks for most of a sprint.  At the beginning of development after creating a few prototypes Tom and I had to spend a lot of time researching into managerial type of games such as Dinner Dash (En.wikipedia.org, b, 2019) and Cake Mania (En.wikipedia.org, a, 2019). We also used this time during the design phase to meet with Rob Kurta and Dave Pimm whenever a new development was made to receive feedback and guidance. We were directed to some reading material that I believe helped the team learn most of the design techniques we would need to progress through the development of the project.   * **Casual Game Design – by Gregory Trefery** * **Hooked – by Nir Eyal**   As our design skills are not as strong as our programming skills, as stated in the risk assessment form we elected to focus solely on the design knowing we could cover lost time once we hit the development stage, rather than code systems that would not be implemented wasting further time.  During the development of the project Tom and I would perform regular assessments of the product and our goals to ensure we were keeping the scope at a realistic level reevaluating our risk assessment form. As a team of two communication was critical to the success of the project, as a team we were able to keep up a high level of communication with each other, and later Amy Potter once she had joined the project kept up the high standard. Due to the high standard of communication any issues that needed to be dealt with as a team were able to be resolved quickly allowing development to continue. The team mainly worked across multiple studio-jams in a sprint which we found to be the most efficient method for working as everyone knew a time where the whole team would be working, and any issues could be dealt with quickly and efficiently. Whenever a studio-jam could not be conducted the team were  able to host a remote jam using Discord Voice Chat.  Because the team held multiple studio-jams and meetings we were able to audit the project to a very high standard and would be able to look back to any point in the project and be able to resolve any issues and / or tasks than needed to be dealt with.  The team were able to get a lot of feedback on the project receiving code reviews with Chris Janes which helped with the optimization of a lot of the code alongside teaching the team better ways to approach problems further along in the project and after. The team also met with Rob Kurta and Dave Pimm for feedback on the design of the game and guidance on iterations based on feedback received either from the tutors or testers.  Part way through the project during a session with Heidi Love a Unity3D Master Builder came in to speak to everyone, once he had finished his talk he was able to spend some time with Tom and myself to look through the performance issues with our Unity project explaining the use of some tools such as batching (Technologies, 2019) to help smooth the performance of the game.  Owing to the efficiency of the team’s communication and studio-jams we were able to begin playtesting with 8 weeks remaining of the project to continue iterations, development and polish from player feedback. Due to entering the playtesting feedback as an early phase the team were able to get valuable feedback that directed the development of the game for example removing mechanics such as the firepit and changing how mechanics behaved slightly such as the seagull event. |
| What do you think needed improvement on the project? | Although previously stated as a positive, the asset pack that we had initially intended to use to mitigate the need for 3D models also proved to be an issue as the models were not as easy to edit as first thought which meant we had to lose some time that should have been spent on development had to be spent on creating some 3D models inhouse. As a team of 2 programmers we had a slight disadvantage over the other groups as we had no designer to help with models, or the design of the game. However, I do believe we handled this well, although I do believe the team would have benefited by having a designer during the earlier stages as team had to spend valuable time researching design principles.  During the early stages the team were still learning how much time to estimate some tasks and were estimated to take longer than they did while others were estimated too short, to start combatting this team would discuss some lower priority tasks that should be completed should all other tasks be completed within time, we also started to allocate contingency tasks just incase while we were still honing in on the estimations. By the end of the project the estimations were only slightly out +/- with few tasks being drastically different to the estimation.  While estimations were an issue with the JIRA board the team also came upon some tasks that suddenly took a high priority during the middle of a sprint which required some lower priority tasks be pushed into the backlog if not completed. We also started out with a slightly over-scoped project that after a couple of meetings with Rob Kurta and Dave Pimm the team were able to scale the project back and produce a deliverable product.  Although the team audited the progress of the minutes in a very effective and efficient manner sometimes the JIRA board was not updated in real-time as tasks were completed, while in house this was not as much of a problem as most work was completed in a studio-jam capacity however for the benefit of ‘investors’ as a team we are aware that we should have made sure to update JIRA at regularly as and when work is completed so everyone knows where the project is at. There was an issue with JIRA for a short while where the team could not update JIRA or create a new sprint, when the team spoke to Chris Janes about this problem we were advised that it is a known and logged issue that most teams are experiencing. Chris was able to resolve the issue quickly, while JIRA was down, all tasks could be found within the meeting minutes (as they are during every sprint).  Another set back that could have been avoided was as a team there were roughly two stages throughout the development where the team needed to reevaluate the code that had been written and rewrite some of the scripts to make the code more efficient and less memory intensive. While also having to revisit finished scripts as bugs that weren’t apparent when created start to become visible as the project develops and some safety checks are no longer valid etc. We also needed to add in new systems such as *object pooling* “where you pre-instantiate all the objects you'll need at any specific moment before gameplay — for instance, during a loading screen.” (Placzek, 2016) to remove the unnecessary instantiate and destroy commands removing the need for the garbage collector where “Repeated[*sic*] calls to Destroy() frequently trigger this task, and it has a knack for slowing down CPUs and introducing pauses to gameplay.” (Placzek, 2016) to release memory back to the system.  The team also ran into some problems with version control throughout the project as to begin with we were using both Unity Collaboration and GitHub simultaneously for a portion of the project. Eventually we decided to scrap Unity Collab and use GitHub as it would keep all files in a centralized location such as Unity files, meeting minutes, assets etc. The team ran into a merge conflict that broke the player’s input commands and resulted in the team having to spend a couple of hours rewriting a new input system as reverting the project would not work and resulted in more and more work being lost.  Despite all that the two main points that could have been improved are the colliders and the tutorial. There is sometimes an issue with colliders where the player would not be able to interact with some of the objects. This is because we check the collisions using Unity’s tag system however, later into the project we learnt about layer-based collisions that Unity handles in a much more efficient manner. As a team we discussed the new system and decided that it was too late into production to start adjusting the collision system, but had we had more time as a stretch goal we would have switched.  The tutorial received positive feedback however where we struggled most was making the experience fun for two players like the main game, but it ended up with most stages requiring input from one player and player two standing around watching. Would the deadline have been further away we would have continued iterating to rectify this issue. |
| What do you think of your own contribution to the project? | I believe that I had a solid contribution to the overall completion of Ship Happens. I have been able to learn a lot more about the Unity3D Engine going through multiple rapid prototype iterations, then developing the game through to a more polished state than any other project I have worked on during my time at UoS. I have also been able to learn a lot more about the C# language itself and able to apply this to the project. I was able to expand upon my ability to use object-oriented programming (OOP), inheritance and optional parameters.  I was able to use and expand upon my knowledge of inheritance when creating the interactable objects such as the plank of wood, cannonball, bucket etc. This allowed me to have a base class that all children classes would implement, and the player could call that same function, but each child will call their own implementation. This sped up production time and make it a lot easier to add and remove some objects should they not be needed anymore. After meeting with Chris Janes for a couple of code reviews I have learnt that while using inheritance is fine, I ended up with objects using functions that didn’t quite add up, as in C# a class can only inherit from one class this could become problematic. Chris suggested that I should have used an interface which is like inheritance but uses abstract classes and a C# class can use more than one interface which makes them a bit more flexible.  As our group (for the most part) consisted of just two members I had to be involved within all aspects of the games production, designing prototypes and implementing them. Once we had decided on a game I had to research into various game design techniques to make up for the lack of designers in the group making sure everything was iterated and documented correctly so as a team we could follow a clear and concise path to the final build.  As well as working on a lot of the design side, I built half of the systems implemented within the game, working with Tom at times to make sure that everything came together. My communication with Tom was to a very high standard making sure that he knew what scripts I was working on and what I was planning to implement. This allowed us to plan the project around each other’s systems making sure they were built in a Tetris fashion building block style. I was always in contact with the group attending most of the meetings and studio-jams, any that I was unable to attend I gave prior notice to the team, so they were aware and made sure to catch up with them at the next available convenience.  As a group member I feel I helped keep the project moving forward and completing my tasks on time, only a few times was I unable to complete my tasks, although sometimes I didn’t update the JIRA board in real-time I made sure the team knew where I was with my tasks. I was also involved in taking feedback on board from playtesters and shifting through the data to help us pull out and group up individual components. This feedback proved to be vital to the progress of Ship Happens, with player’s giving us good indications on how mechanics should be adjusted/removed. I was also able to take this player feedback and record any strategies used, how many won/lost and use this data to adjust the values of the game to balance the main game level ready for minimum viable product (MVP). During the code sessions with Chris I was able to get engaged with and talking to Chris through the game and code base, discussing ways Tom and I could have implemented certain systems betters (such as interface rather than inheritance) and how these changes would have a beneficial change. Each code review with Chris gave us a lot of feedback and gave me a good idea of how to reimplement the scripts and how in the future I can think about and plan a course of action producing more robust code and in a more time efficient manner.  As a manager for the group I believe directed the project in the correct direction, assigning tasks in a fair manner, holding regular meetings and keeping an easily auditable trail. I also partook in all presentations, presentation prep work, video recording and voiceover walkthroughs for the game. |
| **OVERVIEW** |  |
| **Thinking about the project you have worked on this year, what are the important lessons that you will take away from the experience for your next group project?** | Communication is crucial for the success of a project, I have stated the same in every post-mortem of a project. I feel our team made sure that all members were kept in the loop about every change, iteration, design choice and general discussions, either by holding many meetings, but if people are unable to attend these meetings making sure that the meeting is documented and has clear minutes that any absent member can view is another vital part of a project’s success. Without these two core components a project would only be able to move forwards slowly if not at all.  Alongside these points having a good project lead/s is also very important, having a clear direction to take the project and keep iterating towards that goal through constructive criticism either inhouse or from playtesting and presentations.  Team dynamics, while not essential are something that should be considered when forming a team. While Tom and I were capable and managed to produce a deliverable product, had we started with a designer we could have gotten into development quicker, which could have meant more time towards the end of the project for polishing and iterating from player feedback.  Version control is essential, especially with a project of this size. However, it would be beneficial for me to learn more about branches as this would mitigate a lot of the merge conflicts that have happened in multiple projects as any work I produce would have to be merged into the main branch only after I had finished implementing the segment of the project I had been working on at the time.  Planning is critical, as a team I believed we saved a lot of time focusing on all design aspects of the game and getting feedback in a verbal capacity before we wasted time developing a prototype that would inevitably get scrapped, however I believe we could have planned our code slightly ahead of time as we did end up having to rewrite a few scripts.  Playtesting is what makes or breaks a project. Without real world people playing the product, so many issues can go unnoticed, each time we went through a round of playtesting at least one tester was able to find some bug or glitch that no other tester had found. These issues would not be spotted had we only tested the game ourselves and shows how different each individual plays a game.  Studio-jams were so important during the development of Ship Happens, being able to sit down with your group and worth through tasks together motivates the entire team and encourages them to keep going, rather than working from home where members can easily get distracted.  Project scope falls under planning but nearly every group starts to fall down this rabbit hole at this level and it is crucial that this doesn’t happen, especially during a real investor funded project. |

# Bibliography

En.wikipedia.org. a (2019). *Cake Mania (series)*. [online] Available at: <https://en.wikipedia.org/wiki/Cake_Mania_(series)>

En.wikipedia.org. b (2019). *Diner Dash*. [online] Available at: https://en.wikipedia.org/wiki/Diner\_Dash

Kenney.nl. (n.d.). *Kenney Pirate Kit.* [online] Available at: <https://www.kenney.nl/assets/pirate-kit>

Placzek, M. (2016). *Object Pooling in Unity*. [online] raywenderlich.com. Available at: https://www.raywenderlich.com/847-object-pooling-in-unity

Technologies, U. (2019). *Unity - Manual: Draw call batching*. [online] Docs.unity3d.com. Available at: https://docs.unity3d.com/Manual/DrawCallBatching.html