**The Escape Room Postmortem**

**GAM-305**

**Michael Johnson**

**Southern New Hampshire University**

**Project Planning and Scope**

The first problem with our project management was figuring out our time zones and availability. We all come from different areas of the United States and once we were placed into our respective teams, we had to figure out what our availability was. This impacted the life cycle of the development of our project only minimally. As soon as we knew who was on our team, contact was made with everyone and invites to chat were sent and even a meeting was scheduled to give introductions. Constant contact kept the project going in the right direction.

The plan to meet twice a week was a great plan because first we confirm our assignments and when next we met, we see how that worked out for us and if we need further assistance. We all tackled a part of the game on a first come first serve basis and it all worked out with a few hiccups with some conflicts here and there but having one person oversee merging everything was a big help. This was to prevent overlapping work or conflicts in programming that may cause crashes in the game.

**Development**

The big positive aspect is how we all went with tasks that we are familiar with and were able to go on our way and return to bring it all back a week later without conflict. The key decision in having one person control the merging was a big contributor to our success, because if you were to have everyone have the same access, you run the risk of having work get deleted, overwritten, or modified in a way the originator did not intend and may lead to internal arguments.

I think one of the major problems was the inevitable space between all of us. Time and distance. For the most part we were all just words on a screen, but at times we could catch each other’s voices in chat and one thing that I think is important in a work team environment is a personal connection with those that you work with. It may not seem as important when all one sees is digital progress, but workers will go the extra mile for those they work for when they are appreciated and seen as an asset. In our case, the best we could do was constant communication through Discord.

In general, the assignments were Team Lead, Artist, Programmer, UI/UX Programmer/Designer, and Level Designer. Ideas and level design were immediately discussed, and work began rather quickly. The later stages of development were taking care of glitches as well as cosmetics, like enemy skins and animations, along with any title or splash screens.

The design process and provided documents helped us to be successful in this project. Specifically using the Bitbucket to properly log our progress and keep track of our tasks and completion rate. Our weekly meetings on Sunday were the key to our success because it falls on the last day of the week to turn in assignments on time and provided us with a chance to synchronize our work and turn it in as a group.

**Quality Assurance and Testing**

All team members were vigilant in looking for bugs while testing the game. After each new feature was added, we wanted to test the game around that feature to make sure it seamlessly meshed with the rest of the game. This process worked as it related to that feature, but when adding a feature that rippled to other areas it wasn't as clear. For example, When implementing the character models for the player and the enemies, we were focused on how they looked while playing the game and shooting the enemies, but we hadn't considered the attacks of the enemy and the implementation of that. If the melee enemies got too close to the player, they would just stand there and wouldn't attack. This wasn't a bug we found early in the process, because while playing the game, it isn't the intention of the player to allow themselves to be overrun by the melee enemies. So, it went well in that we could find the immediate bugs from implementing a feature, but it kept our blinders on from finding bugs that weren't as obvious.

To mitigate the number of bugs present, we used a combination of testing as we were merging as well as at the end of every week when we had the next stage completed. It helped us to merge incrementally and test as we go throughout the week as opposed to waiting until the last day to merge everything and work through the conflicts. Before implementing a feature, we tried to think about which parts of the game it would affect so that we would know where a bug may show up, thus being able to fix it before it became an issue. We had team members constantly double checking the game design document to make sure we were still on track with our requirements, and that we were still including everything necessary. When we had our twice-weekly meetings, we would bring up the elements that we were still working on so that we could continue to remember what would need to be done.

Next time, utilizing more of the non-functional testing methodologies like performance testing and usability testing would provide a more well-rounded testing experience. As a group we put a lot of focus on making sure everything was working as intended, but we didn't have as much time to make sure that the controls were user-friendly, or that the UI was optimized for in-game clarity. With more overall time, I believe that this testing would have been more feasible for us to accomplish. Knowing what we know now, having that extra time of testing would have been valuable to our overall player experience as well as an even more compelling game.

**Tools and Practice**

The tools and techniques that were key to our team’s success in this project were Git, Bitbucket, Discord, and Unreal Engine itself. All of them contributed greatly to our success as Git and Bitbucket sort of work together in getting our teams work together and saved for us all to be able to view it and use it. Discord on the other hand was our communication hub this was probably the biggest part of the project and contributed greatly to our success as without it I don’t think our team would have communicated as well at all! Lastly, Unreal Engine itself obviously was a tool we needed as we needed it to develop our game with all the built-in tools and things necessary to create and run our game.

Tools or techniques that were not successfully used in our project were SourceTree itself and we ended up failing in making use of the beginner amount of stored space in Bitbucket. Our team did not use SourceTree as I personally do not like using SourceTree and I think the use of Git is way more useful not only in the field later in our development careers but also currently for the project it skips the step of having to use a program to make good use of the Git program. We failed in making the use of space in our beginner Bitbucket account due to not paying enough attention to our resources on Bitbucket and not paying enough attention to our git large file sourcing, doing so we failed pushing our beta up for release towards its deadline and our Lead Programmer Michael ended up purchasing extra space for our project in time for our release. In the future we plan on being more aware of issues such as these, so we catch them before a crucial time error on our part occurs.

The initial analysis of our game design document contributed successfully to our decisions to use our specific tools and techniques by giving us a sort of path to lead on in the beginning, for example my use of Discord was little to none before using the course now I am all over the platform loving the use. The analysis also helped pave a way for our design of the game and the way we created it, the document gave us rules to follow or requirements to live by which successfully helped us in picking the correct assets and programming them accordingly based on what we wanted for our game.

**Communication & Collaboration Strong Points**

The team utilized many strengths to ensure a successful project. One of the strengths was the ability to effectively communicate ideas between the team members. Each team member contributed many ideas throughout the development of the game project. In discussing these ideas, everyone did an excellent job of thoroughly explaining their ideas in a way that each member had a solid understanding of what was needed from each member to make the idea a reality.

The team also did an excellent job of communicating any issues between the various roles to help solve any bugs or other problems in the game. This consisted of feedback between team members and from alpha and beta tests as well. Being able to communicate issues and give constructive feedback and act upon that feedback played a critical part in the project’s success. At various times, team members would work together and communicate resources or other tools that could help in resolving issues and bugs or to properly test functions and other logic within the game.

**Communication & Collaboration Problem Areas**

There were not too many issues or problem areas when it came to communication. While there was a document created by the repository lead, there were some issues with using git, which was understandable considering the team was new to the tool. This required some meetings between team members and some further communication on using git. While the document was planned out well, it seems the communication of the document varied between users.

Aside from the git tool, the only other problem area we ran into was being able to join in voice chat for meetings. There were times where certain members of the team were able to join in voice while others were only able to join the meeting via text discussion. This made it slightly difficult as information often had to be relayed in both voice and text channels to make sure everyone received and understood the information.

**Communication & Collaboration Strategies**

Many different ideas came up in the discussion on how to best ensure communication stayed strong and consistent between the team members. In the end, we devised a plan for how the communication would work between the team members.

To help keep every team member up to date, we decided on bi-weekly meetings spaced strategically throughout the week. The first meeting was set up to discuss the work that needed to be performed during the week and to assign the work to a specific member/role. After this was done, any bugs or issues from the previous week were also discussed and assigned as needed. The follow-up meeting, placed later in the week, served to gauge where everyone was at, ensure everyone was on track, and to provide help if someone was having problems with their assigned work.

To help assign and keep track of issues, we decided it was best to utilize the issue tracker built into bitbucket. This served as an excellent way to document and communicate issues as it allowed the issues to be assigned and followed up on in an easy manner. This was used in conjunction with the Discord chat, where bugs/issues were initially posted. This allowed the team to be aware of the bug/issue and then track them, adding comments as work on the issue was performed.

The largest strategy that helped mitigate problems was the use of Discord itself. While we were able to communicate through email if necessary, Discord provided a means to communicate quickly between members regardless of where we were. We could use the phone application to check messages and receive notifications at any time and, even if we were not able to access the game project, we could still provide important and useful feedback.

**Conclusion**

Suggestions I have to help future game development teams be successful in project scope and development are to communicate early, communicate often, and do your best.

Communicating early has to be the top suggestion in my book just because if everyone gets to the meeting early and meets and greets everyone else they are working with it sets up a comfortable flow with the team and if everybody has an assignment already then there is no lacking of work, basically as long as you set everything and everyone up nicely in the beginning a lot of the rest of the work for the rest of the time is going to be a lot easier. Not only the teams comfortability with the rest of the team but also getting the right tools and learning and understanding the tools the rest of the team are using, for example we decided to not use SourceTree in our project and in doing so we learned the lessons needed in Git to complete the project more efficiently, if this was set up fully and everyone understood it fully in the beginning it could have saved us some time for future work in the project.

Communicating often holds a high place as well, speaking up when there are problems as soon as there are problems could save enough time for the work to be completed by the deadline if the team is comfortable with each other and everyone speaks up when they need too then problems can be fixed a lot easier, lessons can be learned a lot better, and time can be saved that is much needed for the rest of the projects work! Lastly, doing your best work will get the best results, making sure to do your research where it is needed and being positive to keep your code clean and tested can help get the best product while developing software no matter the area of study!

It is also important to ask for help when there are issues and other team members might be able to help. While working on the game, I had a few times when I had trouble using Git due to me never using it before. However, asking the members of the group for help allowed me to learn what to do, which led to me getting better at using Git. Also, it is important to figure out the scope of the game before anything is created. It will allow everyone to avoid having too much work to do due to adding several things to the scope.