Group 27

Unity-Based Fighter

Project and Tools - Winter Term Retrospective

Contributors
Cameron Markwell
Brayden Tremper
Gauge Hartwell
Aaron Koffel

1 Introduction

This document seeks to establish the goals and requirements of our project, as well as outlining the tools and technical approaches for development of the project. The “Tools and Technical approaches” section covers this in detail. The “Project Management” section further explains how the project was handled as a whole and how each piece was integrated into the full product at each step. The final section, “Advice/Recommendations” gives some advice for future work on the project.

The project that this document will cover, the “Unity Fighting Game” (internally dubbed “Dance of Warriors”), was originally created by one of the project team members, Cameron Markwell. Three more team members were recruited for this project, and then the idea was brought to the project partner, Benjamin Brewster.

Dance of Warriors was intended to be a fighting game that explored the interactions between the player and the enemy AI. It is inspired by other games with fighting mechanics that could be considered to resemble a “dance” between the player and their enemy, in the way that the player and the enemy interact with each other in a rhythm. This interaction is explored within Dance of Warriors by way of background music that the enemy AI is synchronized with to provide a timing for the player to play to.

2 Tools and Technical approaches

The Unity 3D game engine was used as the main tool for creating our project. Unity has multiple usage plans for different use cases (Personal and Student versions for individual use, and Plus, Pro, and Enterprise versions for team use). The team-use plans for Unity require yearly paid subscriptions, but for our project, we did not need a team-use version of the software. Unity itself has many different versions, as the software as a whole is updated quarterly. For our project, we chose version 2019.4.9f1. Unity turned out to be a good decision on the basis for creating our game, as it is well-documented, it has many tutorials created by other users of the software, and it is beginner- and user-friendly in general. If we had to do the project over again, we would choose Unity 3D as our game engine.

Visual Studio Code (VS Code) was our main code editor while developing our project. It integrated well with Unity and Github, while also not being too large of a software suite. Visual Studio (a different software suite) uses a lot of disk space for add-ons and plugins, so VS Code was the better option for us as the developers. Visual Studio would have been a good alternative to VS Code, but the installation size made it less accessible to us due to varying availability of storage space on our computers. Any code editor would work for this part of the project's workflow, but VS Code turned out to be a solution that each team member had easy access to, so we chose that.

Github was used as our source control solution for the project. Github can be used with either the Github Desktop client, or with Git through a linux terminal. This allowed for versatility when updating or adding features to the project repository. Github's other features that proved useful were the branch functionality that allowed each team member to work on a different part of the project without interfering with each others' work, automatic tracking of changes within the project (which proved useful for rolling back some accidental changes), and a centralized location for the project code that was accessible from anywhere with an internet connection. Another added benefit of using Github was the fact that it could also serve as a backup of our project, should any team member lose their changes or the project entirely on their machine. Other version control solutions were considered, such as Unity's own version control, but Unity integrates well with any version control solution and Github was already easily accessed and used by each team member, so it was chosen as the version control for this project.

Trello was used as a central location to keep track of all of the user stories associated with the project. Each user story was placed on a card and put into the team Trello board, where the team members could agree on a list of functionalities for each card (what would it take for the feature to be considered "done?"), add new cards, archive user stories, and assign themselves to work on a specific user story. When a user story was complete, the card for that user story could be archived, and a new card chosen to be worked on by that team member. Archiving the cards was the method for keeping track of what has been completed on the project and what still needs to be implemented at any given point in the project duration. Another way for the project team to keep track of features to implement would have been a central document, accessible by each team member, that was updated as each user story/feature was implemented. When a feature was completed, that user story could be removed or crossed-out of the document. Trello proved to be a more accessible option due to the fact that it handles such things for the user.

Discord was the main mode of communication between the team members, as it was the most accessible option for each member of the team. Other options for team communication include: A Slack channel, email, and Zoom meetings. We chose Discord as our mode of communication because of the ease of use it brings to the user. Discord allows a user to create a "server" and invite other users to it, where any conversation that occurs is available to all participants in the server. This is also achievable with a group chat room (also through Discord), but a chat room would send notifications to every participant every time a message was sent to the room. This would be very distracting, so a server proved to be a better option. Servers also allow the users to separate conversations by their topic (called a “channel” in Discord), so that unrelated or off-topic conversations are not mixed in with project discussion.

3 Project Management

Work on the project was divided evenly among the team members for the duration of the project. At the beginning of each project sprint, the team members met with each other to decide who would work on a feature that was intended to be in the game. Features were kept track of with a Trello board (mentioned above), which kept track of user stories for each feature within the game. Each user story was on a “card” on the Trello board, and a team member could assign themselves to a card if they wanted to work on it. Sprint meetings were held regularly throughout the duration of the project development in order to keep everyone up-to-date on the current status of the project. During sprint meetings, team members could ask for help with working on their feature, or they could simply update the team on their progress. If any major changes were made to the game, the team made a decision together on how to handle the change.

Communication between team members was handled well. Any conflicts that arose due to a disagreement on implementation of a game feature or any other decision about the project were resolved quickly and efficiently. Communication was handled through a Discord server, where the team members could communicate with each other quickly and easily. The Discord server also allowed each team member to be easily accessible to the rest of the team, so communication was much faster and more frequent for the group.

Throughout development of the project, every team member was active as much as possible and made frequent updates to the code base. Group assignments were done quickly and efficiently and without any conflict among the group. The only issues faced through the duration of the project were with merging updates with the master branch on the project’s Github repository. These issues arose from Unity itself, as Unity modifies a large number of files whenever any changes are made to the game configuration. The difficulty came from keeping track of which files in the master branch could be overwritten by the new update being merged in, and which files were not needed. The large number of files made this process take time, which slowed development. In order to overcome this issue we created a workflow when merging updates with the master branch. This included creating a temporary branch based on the current version of master, named “[branch to be merged]-masterDummy”, and merging your current branch into it to allow any conflicts or issues from the result of the merge to be fixed. Upon fixing any issues a pull request would be made to merge the masterDummy branch into master. This request would then need to be reviewed by two team members who would open the Unity project and test the added or modified features to ensure optimal functionality.

4 Advice/Recommendations

For future groups who work on this project, we recommend consistent communication with team mates and a fair way of sharing the work done on the project. Having one person handle project documentation and communication with the project partner in exchange for a smaller workload in terms of the development of the game could be advantageous in allowing the other team members to focus more on their parts of the game. When it comes to assignments that require all of the group members to participate, it is recommended that the whole team work on the assignment at one time. This allows constant communication between the team members and a faster completion of the assignment, while also allowing the team to double check each other's work while it is in progress.

As far as communication goes, it is recommended that the group has one central location for team conversations (such as a Discord server or a Slack channel) to ensure that everyone has access to any information shared between the rest of the group. This also allows the whole team to participate in conversations and make decisions together, rather than in a vacuum. Frequent discussions on project features is highly encouraged.

We have no recommendations for the project partner. Our project partner, Ben Brewster, was very collaborative with our group and had no stringent requests for our project. All of his requests were challenging, yet achievable.