13 June 2023

Objective:

* Reflection on meetings with James on 11th June and James & Kemi on 12th June
* Preparing a scrum spreadsheet for the development of the dissertation project

Evaluation:

The initial meeting with James on 11th was interesting as James was unsure of how to get started on the project with a wide scope. I informed James that it should start off fairly simple with a blank canvas, a player, an enemy and the ability to fire a gun, destroying the enemy. However, after several rounds of back and forth regarding the specifications of our whiteboxing, he was in a way still unsure about how to get started. Additionally, he was also unsure of whether every small decision we make should be researched on. A github repository was set up with James as owner and Sheng as a collaborator.

The meeting with Kemi on 12th June helped clear up some of the uncertainties in regards to the project as she explained the difference between dissertation report and the corpus. Kemi also showed some sample structure of various projects where we learned that most projects had adopted a specific methodology in their development cycle. We stayed back for about 15 minutes after Kemi left to discuss further on the specifications of our basic stage of whiteboxing. We managed to come to some sort of conclusion in the end on extremely rough and basic details of what to expect on our first whiteboxing stage. We also agreed to adopt the scrum methodology.

The next day, after doing some preliminary online research online in regards to scrum methodology, I created a scrum spreadsheet where I detailed the product vision, product backlog and sprint planning. I outlined the various user stories and tried to incorporate easy high level separable chunks for us to address. In the sprint planning, I referenced the user stories while focusing on our first sprint cycle.

Conclusion:

We gained a better understanding of our dissertation projects and our deliverables. We have also decided to incorporate the scrum methodology to assist the development of our dissertation project. I have shared the scrum spreadsheet with James and will be awaiting his input which includes only the first stage of our sprint cycle.

Action items:

* Refine first stage of sprint cycle
* Further breakdown of complex product backlogs, if needed
* Addition of other product backlogs, if any
* Start planning the second stage of the sprint cycle

20th June 2023

Objectives:

* Reflecting on meeting with James on afternoon of 20th June
* Further refining of the scrum spreadsheet
* Further reading of the scrum development

Evaluation:

In the meeting with James on 20th in the library, James first showed the progress he has made in regards to the development of the Unity game. He has showed the various features developed:

* 1. Enemies walking around and chasing the player once seen by the enemy
  2. Animations of enemy walking around, being shot at (with blood effects) and dying animation
  3. The model of the gun and bullets animation
  4. Ability of player to walk around

Unfortunately, James has not committed the project or followed the scrum spreadsheet. We engaged in a lengthy discussion to adhere to the scrum spreadsheet from this moment on. James also raised the question of whether to continue with what he has now or start the project from scratch. I replied that since there was quite a bit of work done by him already, we should just continue from what he has.

After finding out that there was no initial commit, we worked together to make the first commit and ran into issues with model files being too large (around 100MB). There were about 5 model files that included the animations of the bullets, enemy movements and bullet spray animation. We had to use Git LFS (Large File Storage) to commit the files onto Github and learnt that there was a set amount of size of 2GB.

To ensure that we both were able to properly apply the scrum, we also agreed to do some reading on scrum itself. What I have learnt from my scrum reading is that scrum is not a prescribed methodology and is instead a framework. The various items such as product backlog is directly related to a specific sprint run which consist of determining a sprint goal, planning of the sprint and refining the sprint backlog. Along the way, there are daily scrums to inspect progress towards the sprint goal, sprint reviews to constantly evaluate the outcome and changes of the sprint and sprint retrospective which consists of increasing the quality and effectiveness of the sprint to improve on the next few sprints moving forward.

We agreed to go over the sprint document tomorrow (21st June) and to make appropriate changes according to the level of work done by James.

Conclusion

We made our first commit of our project. I gained a better understanding of the scrum framework and have agreed to follow the scrum framework religiously.

Action items: Refining of the scrum spreadsheet

26 June 2023

Objectives:

* Reflecting on work progress so far

Evaluation:

Over the weekend, James has started working on the main scene which includes two stages. He has specifically worked on the second stage and has managed to complete the level design and atmosphere. However, during the process, 3 of my implemented sprint backlog features were broken, namely:

* When the health bar reaches 0, the game ends
* A restart button appears on the screen after death allowing the restart of a game
* Health pickups regenerate a portion of the player’s health bar

I have already fixed the first two broken features, however, I have informed James that I hope that this will be the last major breaking change. This is due to the fact that the time was spent fixing bugs instead of developing newer features. There has also been a problem with certain aspects of the game not running properly after James pulls down the latest changes from github.

When fixing the specific bugs, I was having trouble locating specific functionality in scripts. Since we are on our 2nd sprint and only focus on implementing features, I predict that by sprint 3 or 4 that we should start looking at implementing specific software/game design principles as this will allow us to easily debug problems and allow us to implement new features.

Conclusion

With a refined scrum spreadsheet, I have worked on several features but due to the nature of James’ work it has broken quite a few of my implemented features. Fixing these features has allowed me to start considering refactoring our scripts to something more manageable so that it takes less effort to debug and allows us to quickly implement new features.

Action items:

* Work on backlog features of Sprint 2
* Read up on software/game design principles
* Considering development of a documentation

28 June 2023

Objectives:

* Evaluating progress of Unity’s unit testing
* Reflect on completed task of pickups in upgrading player damage

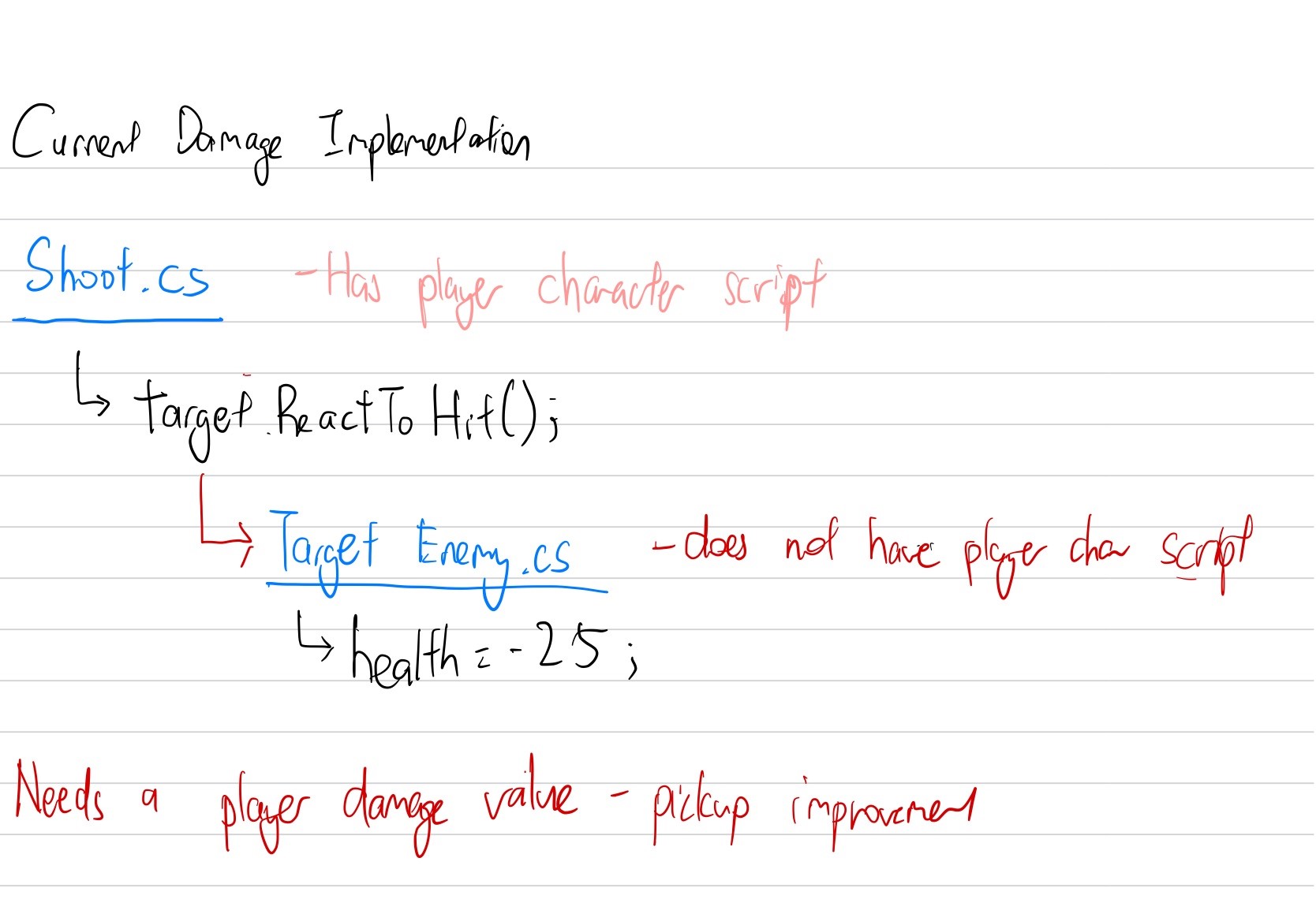
Evaluation:

Writing unit tests for health collectible objects was extremely challenging. I learnt that there are two types of tests in Unity, edit mode tests and play mode tests. Edit mode tests are similar to unit tests in the sense that they test isolated parts of the system. Play mode tests are similar to system tests as they simulate actual user interaction of the system.

I first attempted to test the interaction of the health collectible item when colliding with a player object. However, the functionality of increasing the player’s health keeps failing to no avail. After playing around with the multiple components of both the player character and collectible item objects, such as adding rigid body components, setting specific attributes of the components and adding collider components, I was still not able to get the tests to pass.

I have decided to put the testing on hold for now as it has taken significantly longer than expected and will come back to it soon enough.

The next feature I implemented was upgrading the player damage. The power up asset was imported from opengameart.com and adding it into the game was simple enough. The implementation required mapping the current logic flow of the game as shown below:



Additionally, the player damage pickup interacts with the PlayerController script which contains the fixed damage of the player where the damage is updated with every subsequent pick up.

After testing the feature out, it allowed me to thoroughly test out the game for a long period as I was able to more easily survive. One of the interesting observations was that when I was strong enough to kill enemies quickly, I was actually struggling to find new enemies to fight. The interaction with the doors also meant that some enemies had to be ‘found’ by opening the right doors.

Conclusion

Writing unit tests is still an ongoing process and requires more effort. The damage feature is implemented but the challenge and fun aspects of the game dropped considerably as I had to ‘find’ the enemies due to the challenging interaction of the switch, doors and layout.

Action Items:

* Figure out ways to keep the game challenging as time progresses

29 June 2023

Objectives:

* Implementing state machines for Enemy AI

Evaluation:

Today, I attempted to refactor the main two enemy AI scripts (chasing enemy and target enemy) into a finite state machine. I ran into some problems due to how classes and interfaces work. My initial implementation was to use an Enemy AI script and a base Enemy state class with multiple Enemy state subclasses (e.g. ChasingState). However, this did not work as intended due to Chasing State (subclass) requiring the base Enemy state class to inherit from MonoBehaviour.

So it seems that the state machine base class needs to have a ‘MonoBehaviour’- derived controller. This means that each state can be a simple basic class (not derived from MonoBehaviour) that holds logic for that state, and the controller can call methods on the current state object in its ‘Update’, ‘FixedUpdate’ etc. methods.

This would mean that the simple classes implement an interface of some kind. This interface could include methods for entering the state, exiting the state, and any periodic operations that should be run during the state (which might previously have been handled by a coroutine).

Then, the state machine (which is a MonoBehaviour) manages the transition between states and the scheduling of these periodic operations.

The potentially new layout could be:

* interface IState
* Chasing State inherits from IState
  + Contains actual implementation of Enter(), Execute() and Exit()
* The State Machine
  + Contains current state and current routine
  + Handles changing of states

Conclusion:

I have attempted to start implementing a Finite State Machine (FSM) for my enemy AI. However, I ran into some issues and have attempted to brainstorm and reorganize the layout of my Finite State Machine in order for this new solution to work.

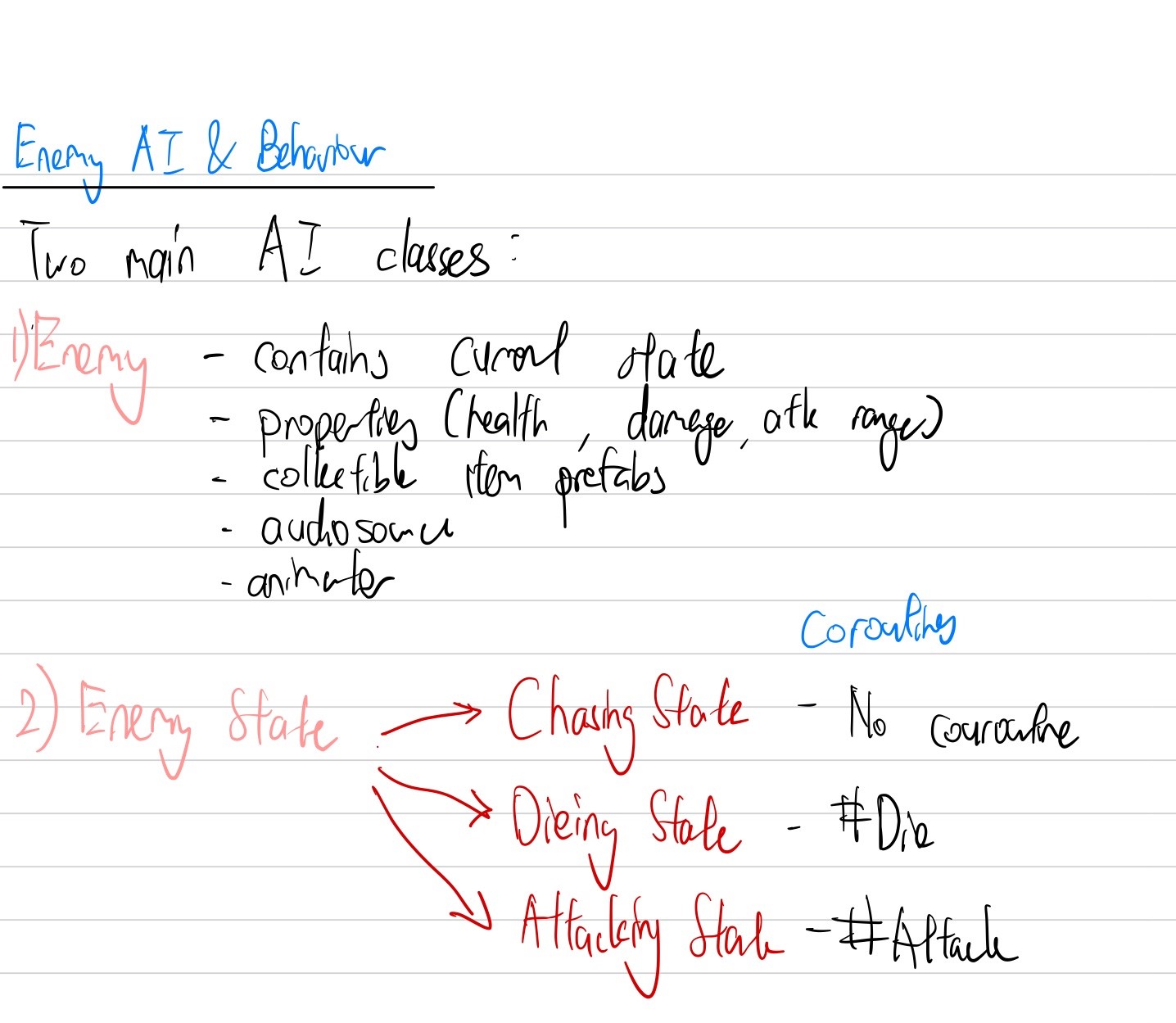
3 July 2023

Objectives:

* Finished implementation of FSM
* A bug with integer overflow due to calling of game end multiple times
* Improve damage frame of attacking animation
* Attempt at fixing health collectible item tests failing

Evaluation:

I have finished the implementation of FSM. Below was a diagram I sketched out that outlines the general gist of the implementation.



One possible interesting bug that I found was when the player was always in range of the enemy, the enemy always consecutively attacked but only showed the animation for the first attack. This was due to the fact that the animation for attack couldn’t transition back to itself. This was easily fixed by adding a self transition to the attacking state.

Another bug encountered was integer overflow error popping up at the end game UI. James was informed of this and he found out that he actually called the game end multiple times which surfaced this specific bug.

I have also fixed the damage frames of the enemy that seem to be janky. The way I went about it was looking through the frames of the animation itself and determining which specific frame windows should be used for actually registering damage hit on the players. In the attack coroutine, we only implement damage to the player for specific frames by waiting for specific animation frames.

There has been another attempt at fixing health collectible test items but to no avail as the OnTriggerEnter is still not being triggered.

Conclusion:

I have successfully refactored the enemy class. I have also improved the attack frame animation of the enemies so that their attacks that damage the player seem more consistent and plausible.

Action Items:

* Determine all approaches in regards to the next Sprint which focuses on DDA

5th July 2023

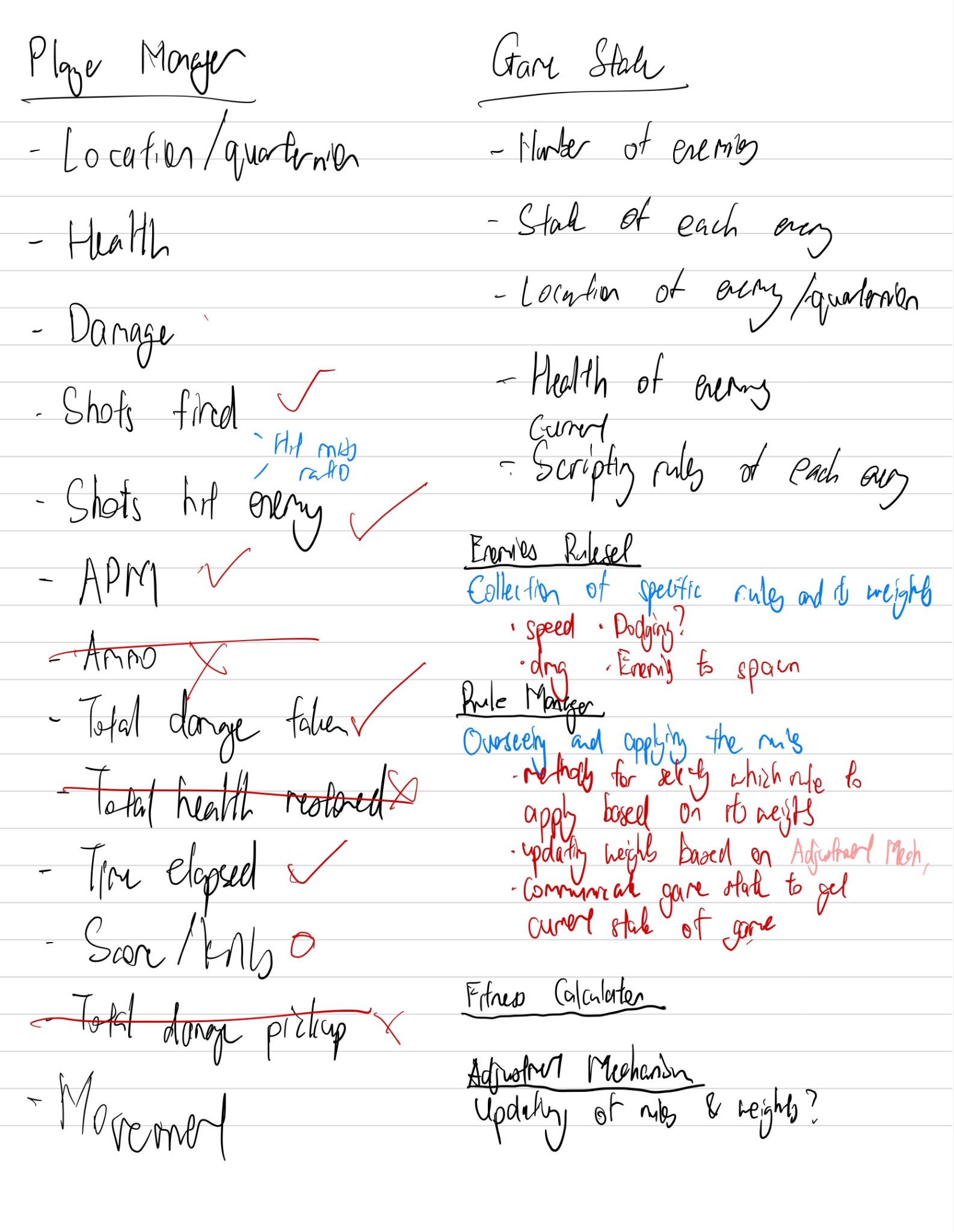
Objectives:

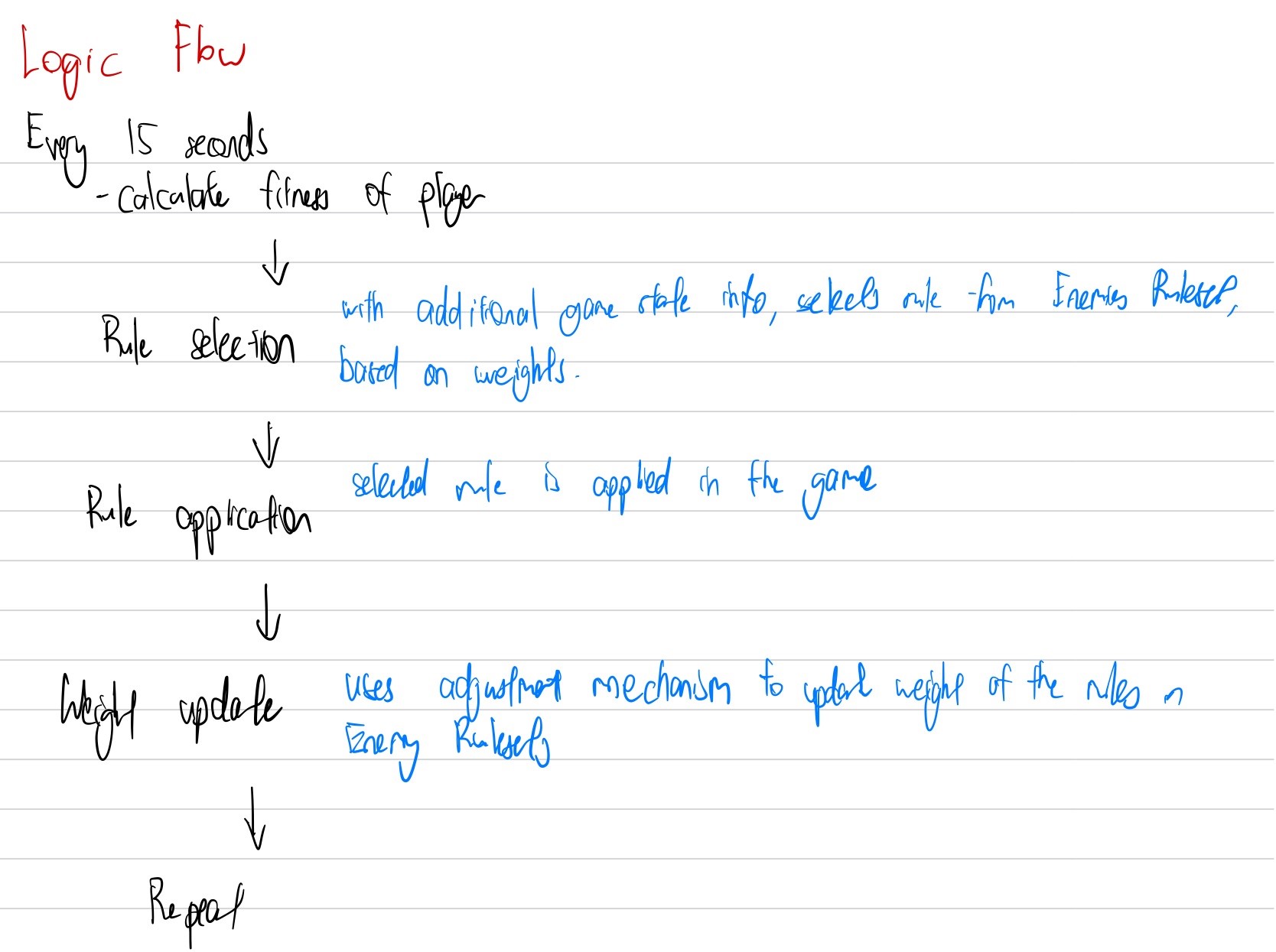
* Determine list of approaches for DDA
* Evaluate Implementation of Dynamic Scripting as a baseline for DDA

Evaluation:

After reading up on general approaches to DDA, there were 7 main applicable DDA approaches. The general overviews and descriptions are outlined in the DDA Research Document. In the end, Dynamic Scripting was chosen as the first base implementation of DDA and one additional DDA would be chosen and implemented if there was additional time.

Additionally, time was also spent in determining the project structure and script files necessary to implement DDA which would make things more modular in the event a different DDA approach was to be implemented. Below is the general overview of the architecture involved.





Conclusion:

The high and medium level structure of the code and logic flow was planned out. The next step would be the implementation.

Action items:

Implementation of fitness function and dynamic scripting

6th July 2023

Objectives:

* Reflect on Git Conflicts & Pair Programming with James
* Change in the way we merge our works

Evaluation:

In the evening, it seems that James ran into some git conflicts as we were working on the same file. He was extremely distressed as he thought that he lost all his work. Due to James’ being new to Git, I suggested that we engage in a video call in which he shared his screen so that I am able to easily resolve his problem.

We first looked at Github’s GUI interface and explained to him what to pay attention to, namely how to spot which files have git conflicts. He proceeded to “fix” the merge conflicts extremely quickly but to his own detriment as he did not properly review each line of the conflict and only glanced through the lines and made very general assumptions. Along the way, it was good to get a sense of James’s personality as he is very strong headed and it was important to not backseat him too much as he might not sit well with it. To be honest, I barely said much and let him do what he thought was right.

Once he ‘fixed’ the various merge conflicts, however, there were still errors when compiling the Unity code. This was as expected as he did not properly review the changes and made a few mistakes here and there. James seems averse to the details of Git and I did not want to push him too much when he was ‘fixing’ the conflicts. In the end, we decided to send him a manual copy of the files instead to fix the issue.

At the end, I informed James that we will be implementing a better workflow. I enforced the fact that we would need to work on our separate branches and when merging to master, we would need to create a pull request. Additionally, we would need to review each other’s pull request.

Conclusion:

Working with James provides a very interesting experience due to his strong headed personality and patience level. We have implemented a better workflow when merging our work together

12th July 2023

Objectives:

* Reflect on progress of DDA
* Reflect on completed Script Generation (part of DDA)
* Reflect on upcoming work of Weight Adjustment and Enemies’ fitness functions

Evaluation:

There are currently three major implementation parts of DDA for dynamic scripting, namely the script generation, weight adjustment and the fitness functions. I have completed the script generation aspect. It was a challenge to carefully think through the flow of the different classes for the script generations. There are two main aspects that are being adjusted, namely specific enemy attributes and the game environment.

The two sets of different classes, enemy rulesets and game rulesets have similar code logic and could potentially be refactored in some way. Additionally, it is possible that classes of the rule and rule set are possibly over-engineered. However, currently we will leave things as is as the next parts of the DDA will be sufficiently challenging and might take a longer time than expected.

The upcoming work of weight adjustment will be a big part in DDA as the rules will be dynamically adjusted based on the enemies’ fitness functions. The enemies’ fitness function was not thought about until now and the challenge will be to currently evaluate the enemies’ fitness functions. One of the challenges here is whether to evaluate the enemies’ fitness function as a whole or to have individual enemies’ fitness function be evaluated. The other question is also in which class/file should this fitness function reside in.

Additionally, there might be various implementation aspects that need to be tweaked in regards to dynamic scripting as there are notable differences in the mechanics of the game for this game development and the research paper’s (Dynamic Scripting) game development.

Conclusion:

The first part of Dynamic Scripting, script generation was successfully completed. The next chunk of work will require more thinking and planning (weight adjustment and enemies’ fitness function).

Action items:

Planning for the implementation of Weight Adjustment and Enemies’ Fitness Function

20th July 2023

Objectives:

* Reflect on progress of DDA so far
* Evaluation of Dynamic Scripting

Evaluation:

Previously, the implementation followed dynamic scripting extremely closely, with various rules such as increase enemy speed, decrease enemy speed in conjunction with a weight value. The weight values would be tweaked according to an adjustment calculation. However, I have since scrapped that implementation as we do not have many “rules” in our games, instead we have just various parameters (e.g. speed, damage, etc.).

Currently, the implementation works by calculating the adjustment which calculates the fitness of the player and the enemy. The difference between these values would be used to adjust the parameters of each specific enemy (e.g. damage and speed) and number of enemies. The specific adjustment calculation was incorporated from dynamic scripting and tweaked appropriately. Currently, we are able to tweak the values of various damage and speed but it needs further work to ensure the values grow slowly.

Conclusion:

The exact implementation of dynamic scripting has been scrapped and modified to fit the structure of the game. Additionally, the adjustment weights according to the fitness needs more fine-tuning.

Evaluation:

Further fine-tuning of the weight adjustment calculation

20th July 2023

Objectives:

* Reflect on recent work done - Merging DDA into main branch on Github
* Continuous Integration & Deployment
* Backend API

Evaluation:

Having completed the main chunk of my branch, I opened a new Pull Request that merges my ‘sheng’ branch into the main branch of Github. I have informed James Craigie to review my work and he merged it with no further comments. To ensure that my ‘sheng’ branch followed the same git history of the main branch, I had to rebase my branch on the main branch of the repository. There were about 2 minor conflicts in total.

I have also attempted to improve our continuous integration workflow by rewriting the circle ci config file. However, currently there is a problem with checking out the code in the circle ci image. This is due to Git LFS as the output states “This repository is over its data quota. Account responsible for LFS bandwidth should purchase more data packs to restore access”. The error seems to be related to the bandwidth requirement of Git LFS on Github. However, as I am not the owner of the repository, it is difficult for me to debug the issue as it seems that the repository belongs under James and the data quota credit affects his account only. I have suggested to James to move the project into an organization which will give us both equal rights to the repository and allow me easier access to debug errors relating to our CI/CD workflow.

Additionally, I have also started developing the backend API needed for the game. The backend API of the game will be responsible for storing necessary game statistics such as performance of player, statistics of the enemies and a global leaderboard. The backend API was developed using Ruby on Rails with a Postgresql database attached to it. The application will tentatively be deployed on Digital Ocean with Dokku sitting on top of it, which helps us easily manage the lifecycle of various applications in the Digital Ocean Droplet.

Conclusion:

James has merged my Dynamic Difficulty Adjustment mechanism chunk of work in the main branch of Github with no further comments. I have attempted to develop the CI/CD workflow of the main repository but have run into some Git LFS bandwidth issues that need to be resolved by James. I have also started developing the backend API for the Unity Game.

Evaluation:

Move the current repository from personal ownership to organization ownership.