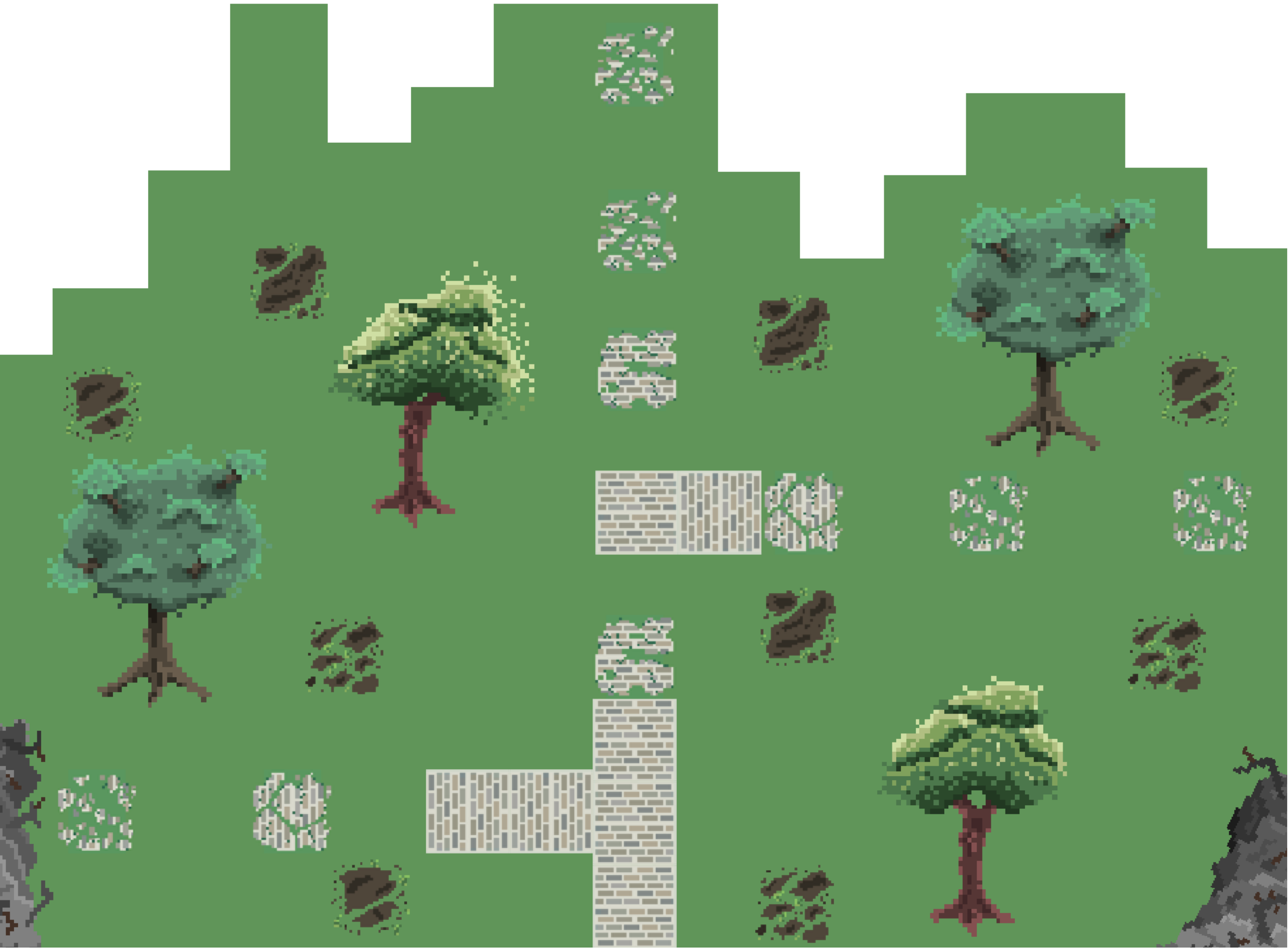
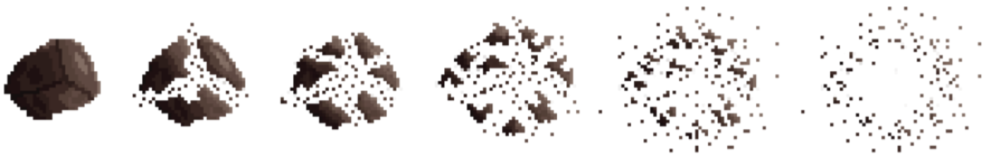
Individual Portfolio

DECO2800

Lucas Bouwman
4583074





Introduction

Throughout this project, the majority of both my individual work and my team has been centred around player-orientated aspects, with us branching off that towards the end of the project. My main work was focused around designing the player's health, as well as overhauling the player's melee attack, adding new weapons and a lighting component for the player's torch.

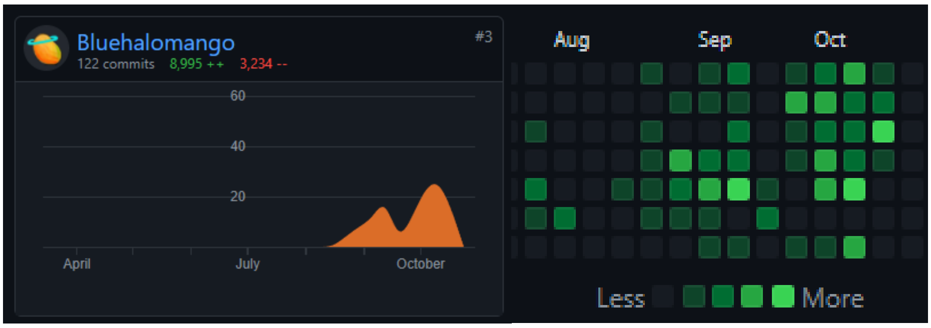


Figure 1 Commits and stats over semester

Individual Work

Player Health

The first aspect was mainly laying out the foundation for what the player would be built upon and involved several encompassing design choices that would affect the overall direction. The task was mainly composed of designing the player's health system and how this would interact with other elements.

In order to create a more interactive system, I decided to explore alternatives instead of the traditional numerical health bar as seen in most games. My main inspiration here was based around a wound system, where the player's health could instead be represented by levels of wounds (i.e., 'Scratched', 'Wounded', etc.). The player's wound state was also integrated into other player stats, creating dynamic interactions through reactive gameplay. While I did not implement that feature, I did influence it both with devising the wound states and advising the changes.

Unfortunately, it proved difficult to find an ideal health balance using just wound states, so an additional numerical health bar was added. I had to make this compromise as having too many wound states overcomplicated the player's stats and made balancing the enemy attack damage difficult. The result, as seen below, split the health into 3 wound states with 3, 4 and 5 health points respectively.

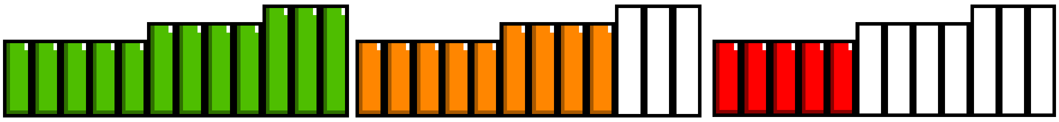
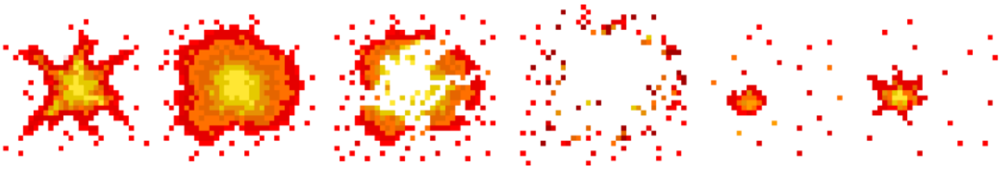


Figure 2 Final Health Bars, From Healthy, Wounded and On Death's Door (Left to right)



Player Melee Attack

In the following Sprint, I then overhauled the player’s attacks so that they were directional instead of in a radius around the character. This proved difficult as the attack involved the multiple components, and I was using a codebase that other team members had written in the previous Sprint but were unable to get working. In hindsight, it would have been better to have had a single member work on the feature or better documented the code that was originally written so that I would have had a better understanding to start with.

I also had to ensure that the code I designed was flexible so that the weapon properties could be modified when adding new weapons later in the project. The structure and complexity of attacking an enemy is evident in the sequence diagram below.

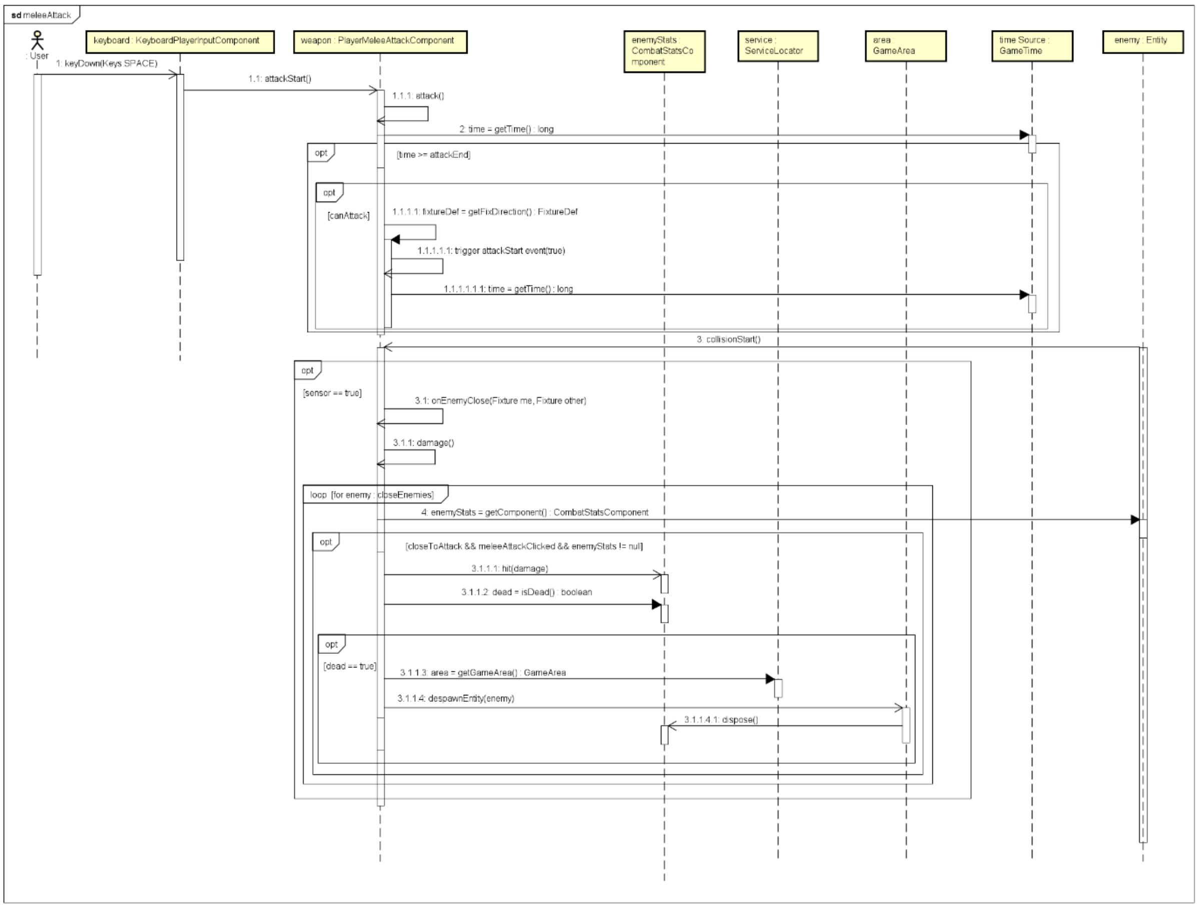
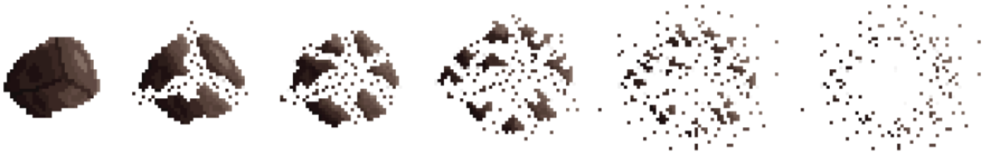

sd meleeAttack
: User
keyboard : KeyboardPlayerInputComponent
weapon : PlayerMeleeAttackComponent
enemyStats : CombatStatsComponent
service : ServiceLocator
area : GameArea
time Source : GameTime
enemy : Entity
1: keyDown(Keys.SPACE)
1.1: attackStart()
1.1.1: attack()
2: time = getTime() : long
opt
[time >= attackEnd]
opt
[canAttack]
1.1.1.1: fixtureDef = getFixDirection() : FixtureDef
1.1.1.1.1: trigger attackStart event(true)
1.1.1.1.1.1: time = getTime() : long
3: collisionStart()
opt
[sensor == true]
3.1: onEnemyClose(Fixture me, Fixture other)
3.1.1: damage()
loop [for enemy : closeEnemies]
4: enemyStats = getComponent() : CombatStatsComponent
opt
[closeToAttack && meleeAttackClicked && enemyStats != null]
3.1.1.1: hit(damage)
3.1.1.2: dead = isDead() : boolean
opt
[dead == true]
3.1.1.3: area = getGameArea() : GameArea
3.1.1.4: despawnEntity(enemy)
3.1.1.4.1: dispose()

Figure 3 Sequence Diagram for melee attack

New Weapons

Adding the first 2 weapons was seamless, as they were designed to create and fulfil new roles and I had previously designed the code implementation with new weapons in mind. However, adding the next 3 weapons proved difficult as I had to design them to be ‘upgrades’ to the base 3 weapons.

For designing the weapons, I started trying an existing program I was familiar with, Krita, but after finding that it didn’t work well with pixel art, I tried several free alternatives (Piskel and Pixilart)



before settling on Photoshop after finding we had a student licence for it. Additionally, I was unfamiliar with drawing pixel art, but I quickly familiarised myself with it as an art style after starting.

When first devising the designs for the new weapons, I focused on showing the weapons as upgrades and designed them to feel powerful. For the 3 ranges (short, medium, and long) I did research into popular weapons of those types, using the 'rule of cool' to justify the choices. The designs, evident below, feature a Katana as the medium-ranged upgrade, a Bloody Axe for the long - range and Dual Daggers for the short-range.

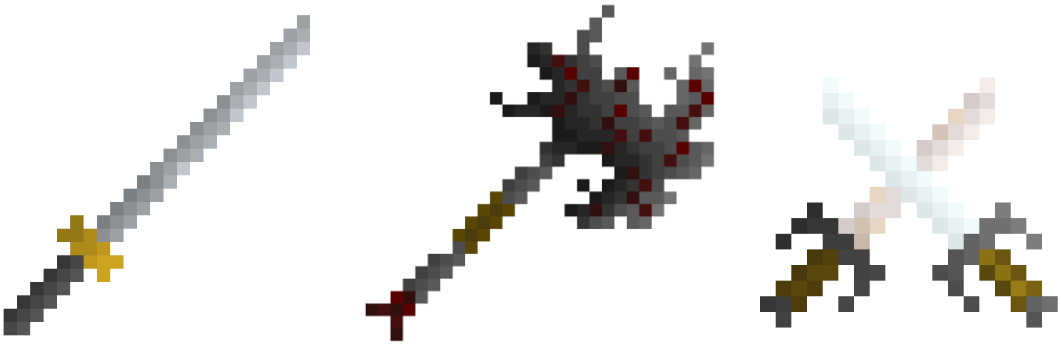


Figure 4 First design of new weapons. Katana, Bloody Axe and Dual Daggers (Left to right)

While users did consider the weapons awesome and powerful, they found they didn't fit into the established setting of the game (this is explored in user testing). This made me realise that justifying something purely off the 'rule of cool' is not enough and it is more important to ensure integration beforehand.

Fortunately, by out early in user testing, I was able to redesign the weapons as well as one of the original weapons which the users found no longer fitted into the game. This time, I focused on weapons for the setting, that being post-apocalyptic, instead of weapons that looked 'cool'. This led to the designs below, with the Crowbar replacing the Katana, Baseball Bat replacing the original Sword, the Sledgehammer replacing the Bloody Axe and the Machete replacing the Dual Daggers. A more detailed explanation can be found here.

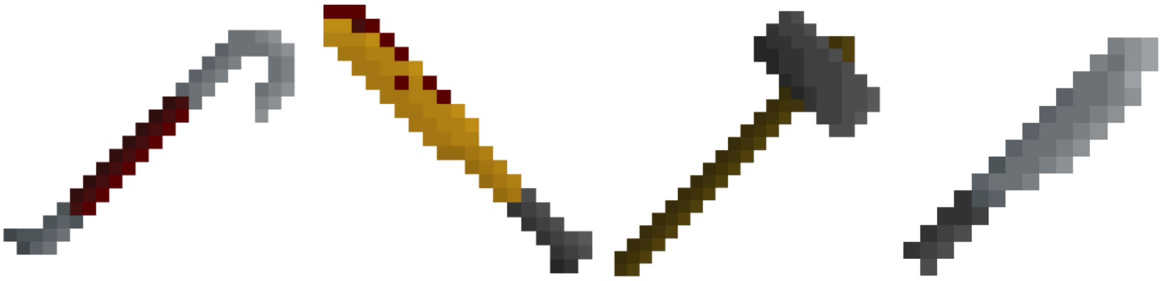
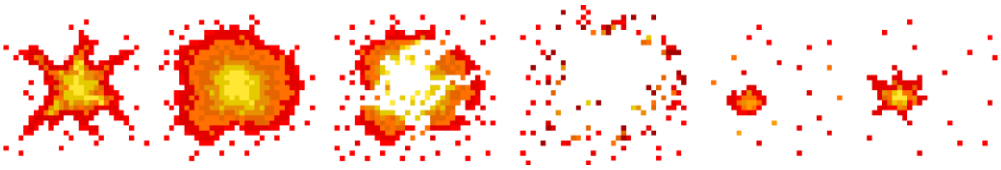


Figure 5 Final Design of new weapons. Crowbar, Baseball Bat, Sledgehammer and Machete (Left to right)

I also made animations for the new weapons and other assets throughout the project. I used GDXTexturePacker to create the atlas animation files necessary for engine. As I was the first person to create the atlas files, I ended up making all the atlas files for our team throughout the project.



Torch FlickerLightComponent

I also made a lighting component, the FlickerLightComponent, for the player after being inspired by a fellow team member. I spent some time researching similar ideas and used the findings from several articles as inspiration for my design.

I used several varied lights together to mock the different light intensities. Additionally, I introduced randomly changing values to my algorithms so the light would oscillate randomly to mock the flickering. The overall effect of the FlickerLightComponent is a realistic and juicy element that helps add a great degree of polish and flair to an otherwise static feature.

Individual Reflection

All in all, I'd say I am more than happy with the work that I have contributed to this project. The player related features were essential to the functionality of the game and did well to create a complete experience. On the other hand, the polishing I did to environmental assets and render elements, such as the FlickerLightComponent, significantly increased the polish in the project and did well to add additional flair and juiciness to the game.

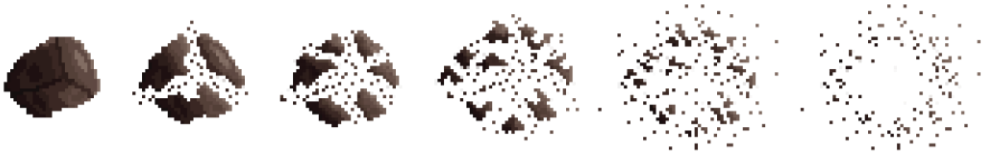
Admittedly, one of my main issues in my work methods was a lack of documentation. As my standards for my work are very high, I would typically work through a task and focus more on perfecting and polishing it than documenting it. Coupled with the fact that I prefer to start and finish a new task than complete the documentation for a previous task, this saw me jumping from task to task, completing the code and design components of each but leaving the documentation for later.

Obviously, this is a bad approach as it leaves a large amount of documentation to be completed at the very end of a Sprint while also making it significantly more difficult for other people to go through and understand what I had done. In hindsight, the best solution I can see to this problem would be to either reduce my workload to compensate for the additional time spent on documentation or to enforce strict practices on myself to ensure that necessary documentation is complete before moving on from a task.

Teamwork Description

Throughout the project, our team's focus was mainly on the player related side covering stats, attacks, items, and abilities, splitting off in the latter Sprints to also work on the Safehouse, level layouts and environmental assets. The overall balance of our team's expertise meant that we focused a lot more heavily on the coding side of things, which is why we choose to cover the player feature. That way, we left the player design to our main designer, ensuring that all the player assets integrated well into each other and that everyone else on the team could focus on the heavily code-based implementation of the player.

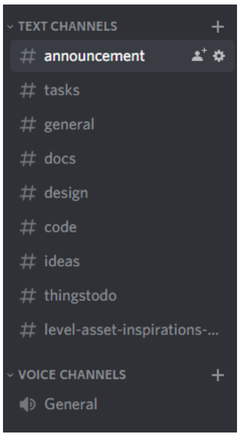
During Sprint 1, we encountered an issue with task allocation where members were uncertain of what tasks they were required to complete and what that entailed. This got worse as several tasks



collided, members ended up completing tasks outside of their allocated areas and multiple people were unknowingly trying to implement the same featuring, wasting time and effort.

To combat this, it was agreed that I would take the role of official team leader to better coordinate and manage ourselves. This was beneficial as I had previous experience as team lead for similar DECO group projects, allowing me to properly manage and allocate tasks for each member based on their interests, specialities, and preferences. This helped greatly throughout the project as there was never a case of a team member working on something that they disliked or were bad at. Additionally, I have quite a thorough understanding of game design, allowing me to make informed design decisions and help to positively influence the gameplay.

We also worked on maintaining a cooperative and democratic process for our group, which made resolving issues streamlined and efficient. Due to my familiarity with both leading DECO group projects and Discord as a platform, I also decided to create an independent Discord server specifically for our team. This was a major benefit and greatly helped to increase the efficiency of our team, as it meant we could have multiple channels and discussions going on without losing any information (evident on the left).



TEXT CHANNELS
# announcement
# tasks
# general
# docs
# design
# code
# ideas
# thingstodo
# level-asset-inspirations-...
VOICE CHANNELS
General

Figure 6 Screenshot of channels from our private Discord Server

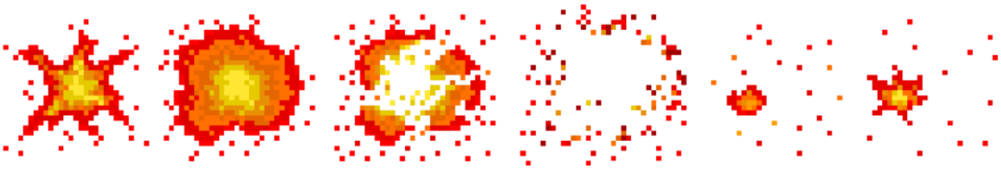
Additionally, we would start each Sprint by having a meeting to collaboratively work on a Sprint overview document, allowing everyone to properly voice their ideas, concerns and thoughts while ensuring that everyone understood the feature and task allocations. Towards the end of the sprint, we would do a similar activity with the Sprint Achievement form where we would get together in a group meeting to work through the form. This ensured that everyone's work was included and that everything was not left to a single person.

Teamwork Reflection

Following the processes outlined above, our team was able to maintain a very high work standard throughout the project and I am very proud to have led them on account of that. The only real issue our team ran into was that we were overly ambitious. This led to disparate workload commitments, as some members of the team were more than happy to follow our ambitious goals while others were only willing to do the minimum amount of work.

As such, several members of the team, myself included, tended to contribute significantly more than expected to the project, sometimes going to the extent of working on and completing tasks of other team members and, in the case of the last sprint, another team. While this wasn't a major issue, my recommendation is that it should be something addressed in the charter to ensure that every member is on the same level in terms of expected contributions.

Instead, I just changed how I allocated tasks so that I would pre-emptively give more tasks to those willing to do more work. Admittedly, this ambition may have stemmed partly from my influence, as I was both very enthusiastic and invested in the game's development, seeing it more as an actual project than just an assignment. In hindsight, I probably should have been either less ambitious personally or been less influential on the work ethics of my team members.



Appendix and Notes

All images that appear in this document, including the cover and headers, were created by me for the project.

Full hyperlinks in order of appearance.

Player Health:

https://github.com/UQdeco2800/2021-studio-6/wiki/Player-Health

Player Attacks:

https://github.com/UQdeco2800/2021-studio-6/wiki/Player-Attacks

Initial Weapon User Testing:

https://github.com/UQdeco2800/2021-studio-6/wiki/Sprint-3---User-Testing-Player:-First-Iteration-Additional-Weapon-Designs

Final Weapon User Testing:

https://github.com/UQdeco2800/2021-studio-6/wiki/Sprint-3:-User-Testing-Player:-Final-Weapon-Designs

Upgraded Weapon Designs:

https://github.com/UQdeco2800/2021-studio-6/wiki/Sprint-3---Upgraded-Weapon-Designs

FlickerLightComponent Wiki:

https://github.com/UQdeco2800/2021-studio-6/wiki/Sprint-4---FlickerLightComponent