DECO2800 - Individual Portfolio

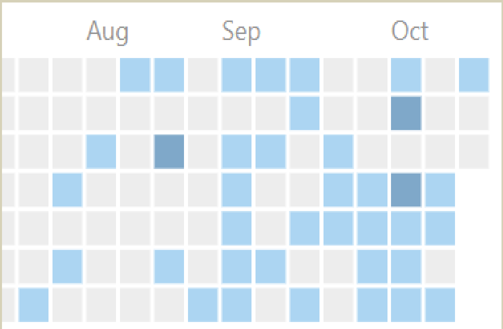
By *Tsz Yan Nicole Fung*

INTRODUCTION

PURPOSE : This portfolio serves as a reflective overview of my individual contribution to Studio 5's Skyfall game development in the entire semester.

SUMMARY : Throughout these 5 sprints,

1. **Role in team:** Designer (sprint 1-2, column 5-8)
   Programmer (sprint 3-5, column 9-14)
2. **Contributions scope:**
   - Combat process design for player and enemy
   - GitLab wiki page management
   - Animation integration
3. I have learnt through group collaborations that **my strength in a team** is to:
   - optimize and visualize complicated structures
   - ideate ways to manage team process



Days I contributed on GitLab, columns counted from the left (starting from column 5)

DELIVERABLES / TABLE OF CONTENTS

1. What is SkyFall?

SkyFall is a Cyber-punk survival game where the player has to walk around and kill robots in the post-apocalyptic world to gather resources. Inspired by the game Minecraft and Don't Starve, the player can craft different items like buildings and weapons.

How does my work fit into this game? / What do you need to know for this portfolio?

1. Combat team and Spells team:
   The Combat team has first created processes and classes related to combat system in Sprint 1, and worked on the hurt, attack, die methods and weapon-related classes in Sprint 2-3.

   From Sprint 3, we changed to Spells team to implement spells weapons. Spells weapons involve Shield (defend from enemy), Tornado (stun enemy) and Flame wall (attack enemy).

   Related classes in the game code:
   MainCharacter.java, Shield.java, Tornado.java and FlameWall.java

2. New Enemy team:
   There are 7 biomes with different themes (e.g. forest, volcano), which different level of difficulty of enemies are spawned. There is also 4 types of enemies Scout (easy level), Medium (medium level), Heavy (Hard level) and Abductor (signals nearby enemies to chase the player when player is detected). My work is to implement the new enemies above with basic combat methods (attack, hurt and die) and import animations sprites designed by the designer in my team.

   Related classes in the game code:
   Enemy.java, Scout.java, Medium.java, Heavy.java and Abductor.java

For details, Check out our wiki page!

2. INDIVIDUAL WORK

2.1 Combat Storyboard & Class diagram

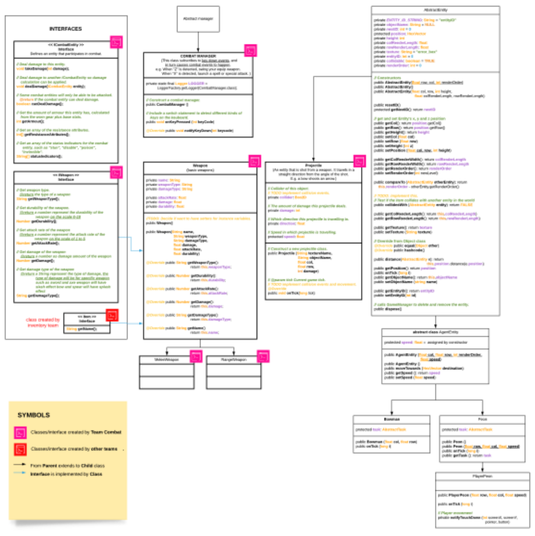


Figure 3 Combat class diagram before critique (link)

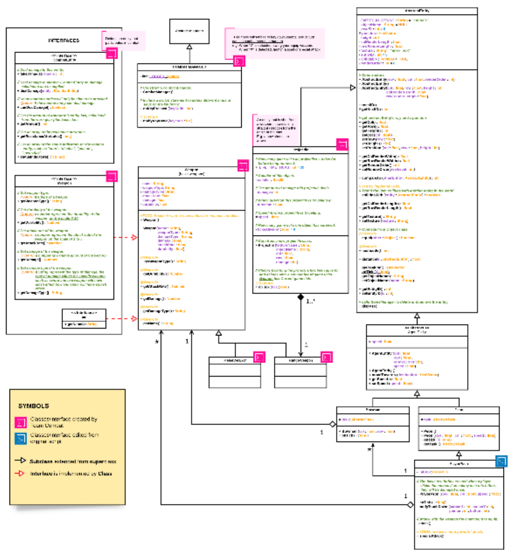
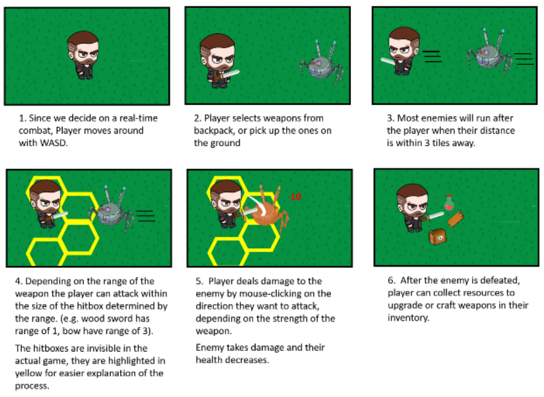
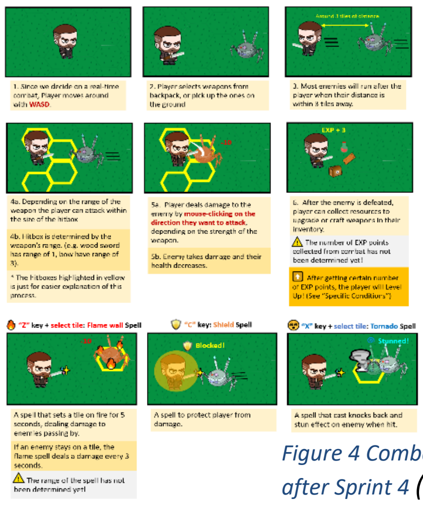


Figure 2 Combat class diagram after critique section (link)

In ticket [#42](#42), in order to **let other teams to understand the class structure of the combat system**, I <u>used LucidChart to make a class diagram</u> and post it on GitLab wiki page *Combat/Class Documentation*. During the studio's 1st critique session, I showed my work to the other teams. They complimented that <u>it includes Javadoc-like comments to explain each method but should also add some labels to explain classes</u>. In general, I think I have done well in making use of critique sessions to evaluate the weakness of class structures. **However, if I can further improve it, I will allow more spaces between classes and label to make it more readable.**


1. Since we decide on a real-time combat, Player moves around with WASD.
2. Player selects weapons from backpack, or pick up the ones on the ground
3. Most enemies will run after the player when their distance is within 3 tiles away.
4. Depending on the range of the weapon the player can attack within the size of the hitbox determined by the range. (e.g. wood sword has range of 1, bow have range of 3).
The hitboxes are invisible in the actual game, they are highlighted in yellow for easier explanation of the process.
5. Player deals damage to the enemy by mouse-clicking on the direction they want to attack, depending on the strength of the weapon.
Enemy takes damage and their health decreases.
6. After the enemy is defeated, player can collect resources to upgrade or craft weapons in their inventory.
Figure 5 Combat storyboard at the sprint's beginning (link)

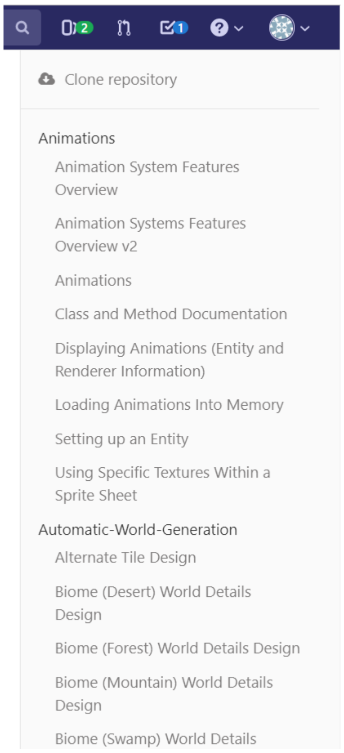

Around 3 tiles of distance
1. Since we decide on a real-time combat, Player moves around with WASD.
2. Player selects weapons from backpack, or pick up the ones on the ground
3. Most enemies will run after the player when their distance is within 3 tiles away.
-10
EXP + 3
4a. Depending on the range of the weapon the player can attack within the size of the hitbox
4b. Hitbox is determined by the weapon's range. (e.g. wood sword has range of 1, bow have range of 3).
* The hitboxes highlighted in yellow is just for easier explanation of this process.
5a. Player deals damage to the enemy by mouse-clicking on the direction they want to attack, depending on the strength of the weapon.
5b. Enemy takes damage and their health decreases.
6. After the enemy is defeated, player can collect resources to upgrade or craft weapons in their inventory.
The number of EXP points collected from combat has not been determined yet!
After getting certain number of EXP points, the player will Level Up! (See "Specific Conditions")
"Z" key + select tile: Flame wall Spell
"C" key: Shield Spell
"X" key + select tile: Tornado Spell
Blocked!
Stunned!
A spell that sets a tile on fire for 5 seconds, dealing damage to enemies passing by.
If an enemy stays on a tile, the flame spell deals a damage every 3 seconds.
The range of the spell has not been determined yet!
A spell to protect player from damage.
A spell that cast knocks back and stun effect on enemy when hit.
Figure 4 Combat storyboard after Sprint 4 (link)

In ticket [#72](#72), I observed in the studio's 1st critique session that **other teams are confused about the processes of combat, especially the key controls**. **My teammate who did the demoing of also had difficulty verbally describing a system to the studio**. Therefore, I <u>decided to design a storyboard to explain the process with visuals</u> and posted in on GitLab wiki page *Combat/Combat Process Flow*.

Throughout Sprint 1-3, I consistently [#256](#256) updated the storyboard to include more necessary details (e.g. adding orange boxes in the texts to help explaining which part of the process has/hasn't been implemented yet). **This aids teams like Animation, UX and Main Character team to know what our combat team is implementing.**

If I can redo it, I would add a section to include my Slack contact details so that if other teams have any questions regarding to the process they could communicate with our team.

2.2 Collapsible Customized wiki sidebar

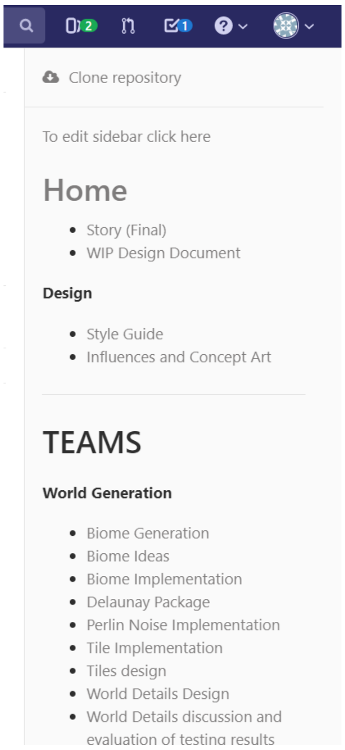


Clone repository

Animations
- Animation System Features Overview
- Animation Systems Features Overview v2
- Animations
- Class and Method Documentation
- Displaying Animations (Entity and Renderer Information)
- Loading Animations Into Memory
- Setting up an Entity
- Using Specific Textures Within a Sprite Sheet

Automatic-World-Generation
- Alternate Tile Design
- Biome (Desert) World Details Design
- Biome (Forest) World Details Design
- Biome (Mountain) World Details Design
- Biome (Swamp) World Details

Figure 8 Before _sidebar



Clone repository

To edit sidebar click here

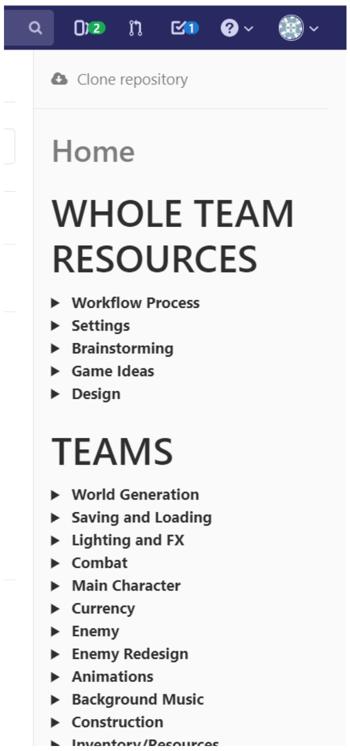
Home
- Story (Final)
- WIP Design Document

Design
- Style Guide
- Influences and Concept Art

TEAMS

World Generation
- Biome Generation
- Biome Ideas
- Biome Implementation
- Delaunay Package
- Perlin Noise Implementation
- Tile Implementation
- Tiles design
- World Details Design
- World Details discussion and evaluation of testing results

Figure 6 Customized sidebar added in Sprint 2 (link)



Clone repository

Home

WHOLE TEAM RESOURCES
- ▶ Workflow Process
- ▶ Settings
- ▶ Brainstorming
- ▶ Game Ideas
- ▶ Design

TEAMS
- ▶ World Generation
- ▶ Saving and Loading
- ▶ Lighting and FX
- ▶ Combat
- ▶ Main Character
- ▶ Currency
- ▶ Enemy
- ▶ Enemy Redesign
- ▶ Animations
- ▶ Background Music
- ▶ Construction
- ▶ Inventory/Resources

Figure 7 Collapsable sidebar is made from Sprint 3 onwards (link)

WHY: In ticket #169 , when I was browsing the Pyramid Scheme example project, I realized that Skyfall's wiki sidebar by comparison is less efficient for navigation due to the following reasons:

1. not all of our wiki pages are showing in the sidebar,
2. sidebar has no structure-related formatting,
3. sidebar cannot show nested page structure and
4. cannot customize the order of pages

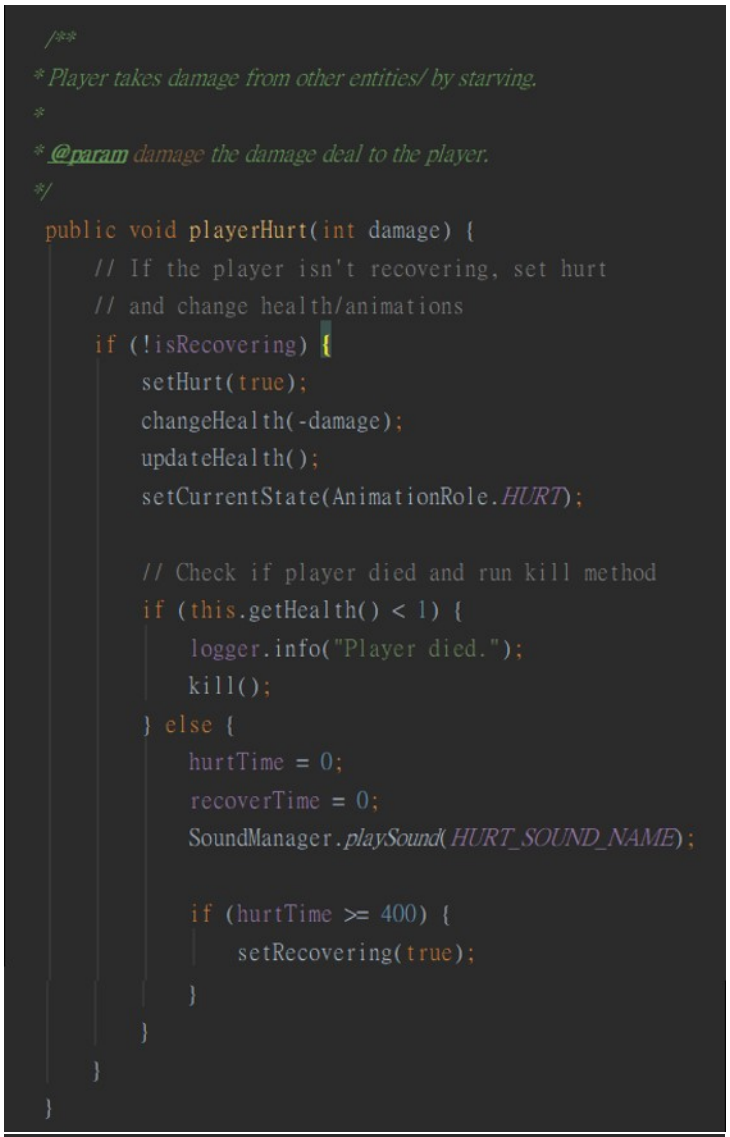
HOW: Doing some research online, I found that their sidebar used a **GitLab wiki build-in tool called customized sidebar page**, however it hasn't had the function to auto-update when one adds a new page/change name of a page. Therefore, during mid-Sprint 2 I **proposed the idea to gather support on Slack** and consulted tutors' opinions and permissions in studio and tutorial. The result turns out to be positive, thus I made a **GitLab ticket's milestones to track whether all the teams has put their wiki pages on the _sidebar page**.

After Sprint 2, along with the help of Harry@jhhuang7 (in putting remaining team's wikis into _sidebar), the customized sidebar is fully implemented with all the teams and important pages like style guides and story. It solves the above 4 difficulties to make GitLab wiki navigation more scalable, thus optimize communication speed between teams in the later 3 sprints.

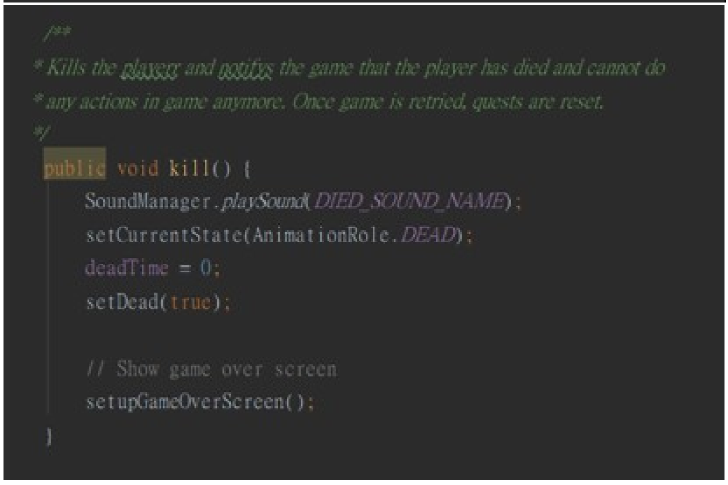
During Sprint 3, since I observed from studio's demo presentation that the teams have to scroll for a long time before finding their team's wiki page to present. Therefore, I **decided to improve the sidebar sections to be collapsible** using the combination of HTML and markdown language in GitLab editor.

REFLECTION: In general, I have made good use of GitLab's build-in tools and HTML knowledge to address the studio's communication efficiency. **For improvement, I could find GitLab tools to make the customized wiki sidebar auto-update with the actual wiki pages.**

2.3 Main Character's hurt and died methods



```
/**
* Player takes damage from other entities/ by starving.
*
* @param damage the damage deal to the player.
*/
public void playerHurt(int damage) {
    // If the player isn't recovering, set hurt
    // and change health/animations
    if (!isRecovering) {
        setHurt(true);
        changeHealth(-damage);
        updateHealth();
        setCurrentState(AnimationRole.HURT);

        // Check if player died and run kill method
        if (this.getHealth() < 1) {
            logger.info("Player died.");
            kill();
        } else {
            hurtTime = 0;
            recoverTime = 0;
            SoundManager.playSound(HURT_SOUND_NAME);

            if (hurtTime >= 400) {
                setRecovering(true);
            }
        }
    }
}
```



```
/**
* Kills the playerr and notifys the game that the player has died and cannot do
* any actions in game anymore. Once game is retried, quests are reset.
*/
public void kill() {
    SoundManager.playSound(DIED_SOUND_NAME);
    setCurrentState(AnimationRole.DEAD);
    deadTime = 0;
    setDead(true);

    // Show game over screen
    setupGameOverScreen();
}
```

<u>WHY:</u> In the game, one of the main character's goal is to "eliminate enemies in each location using your weapons and collected resources. Killing these enemies will reward you with gold pieces". This will require the main character's ability to be attack, hurt and die.

<u>HOW:</u> For this, I have read through *MainCharacter* class's parent class *Peon* for related methods (e.g. health-related methods like *changeHealth(-damage)* in *Peon* class so that I can re-use that in this subclass).
Next, I created 2 *boolean* setter and getter variables *setHurt()* , *setRecovering()* to define the state of player was hurt and then recover. Thus, in the beginning of the *playerHurt()* method, when player is NOT recovering, *playerHurt()* will be executed.

After that, I add the condition if player's health is under 0, kill() will be executed to play dead animation, sounds and show a Game Over Screen (with *setUpGameOverScreen()*)

Lastly, I added animations and sounds with *setCurrentState(AnimationRole.HURT)* and *SoundManager.playSound(HURT_SOUND_NAME)* .

<u>REFLECTION:</u>

For improvement, I should have set a schedule with the Inventory team to add the method of dropping gold pieces into the hurt method.

2.4 Fixing Code smells, duplicated codes, raising code coverage

WHY: Issues like duplicated code among classes would make the code tedious to read

HOW: During Sprint 4 and 5, I have **used SonarQube's dashboard to track code quality metrics** and **IntelliJ's SonarLint plugin to address code smells in the game's code.** By doing this, I realized that it is not only the functionality of the code that matters, but the code quality also matters since we have to be able to read and write another programmer's code within a studio collaboration project.

REFLECTION: I should have communicate more frequently on Slack and GitLab ticket about how to correct some of the code smells because changing some code may change the Junit test results in the code.

3. Group Work

3.1 Combat and Spells(Magic weapon) animation integration

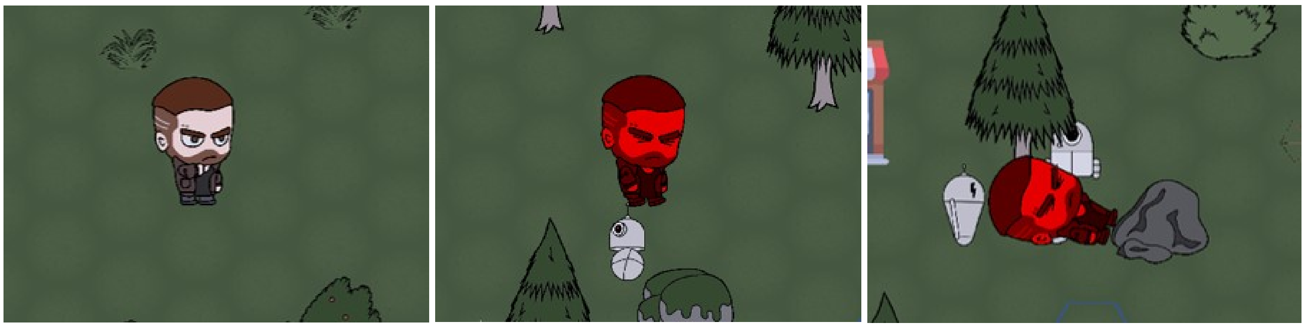


Figure 9 Combat animation (from left: AnimationRole.ATTACK, AnimationRole.HURT, AnimationRole.DEAD)

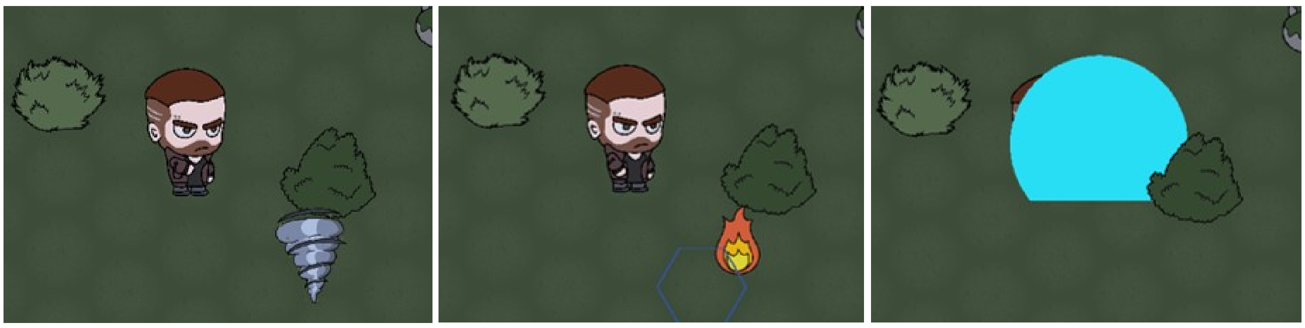


Figure 10 Spells animation (from left: Tornado, Flame Wall, Shield)

In ticket Combat: #130 #134 #142 Spells: #257, I have integrated designer's animation spritesheets, into the programmer's code.

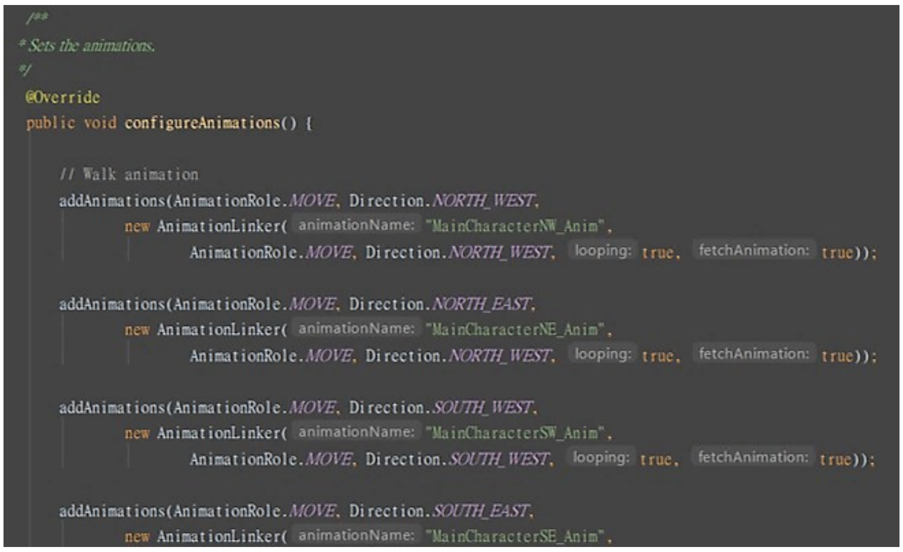
WHY:

It is important to the combat team since combat team programmers need character animations to visualize the process of attacking, hurt, and died in combat.

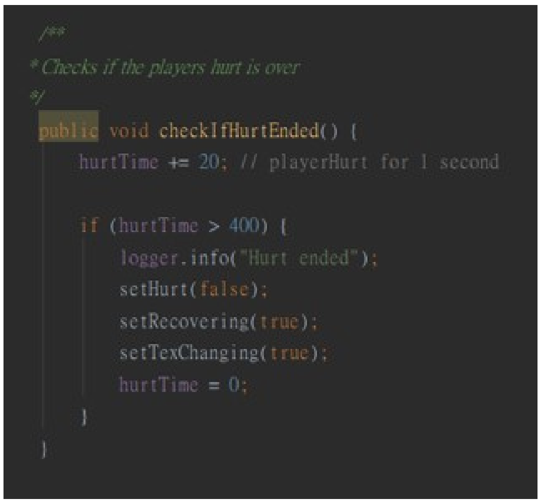
HOW:

During Sprint 2, I helped ***@ShaoQiTan***, who designed the main character's sprites, to integrate main character's MOVE, ATTACK, HURT, DIED animations into the class MainCharacter. Within the process, I have studied the Animation team's wiki page and code in order to import the spritesheet as texture (in TextureManager class) to convert it to animation sprite frames, then generate an animation object with it (in AnimationManager class). The animations are then imported into the targeted object and controlled through *configureAnimation()* and *updateAnimation()* .

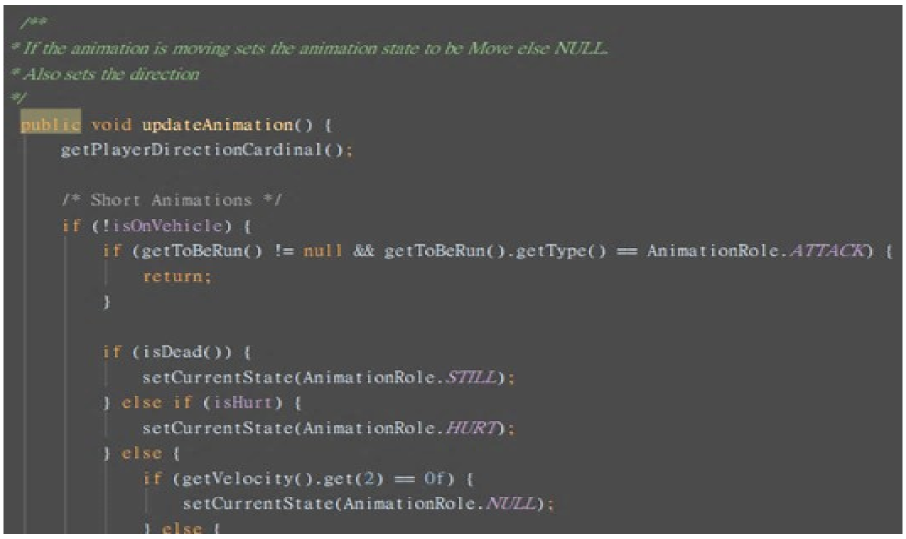
However, when I ran the game, I found that the animation did not apply to the player during combat. Therefore, I have asked ***@Kausta*** from the Animation team to solve this issue, which he created *hurtTime int* variables and checked the animation duration through *checkIfHurtEnded* . I have then further added *recoverTime* , *deadTime* in and *checkIfRecovered* to set the duration of each state for the attack/die animations (by the line *setCurrentState(AnimationRole.XXXX)*) and red tint (explained later) applies on the player.



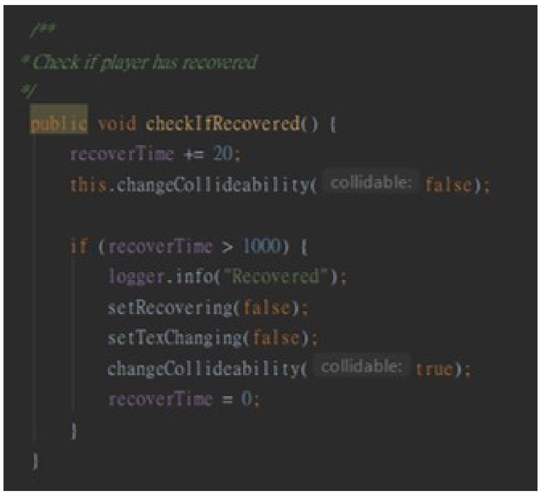
```
/**
* Sets the animations.
*/
@Override
public void configureAnimations() {

    // Walk animation
    addAnimations(AnimationRole.MOVE, Direction.NORTH_WEST,
            new AnimationLinker( animationName: "MainCharacterNW_Anim",
                    AnimationRole.MOVE, Direction.NORTH_WEST, looping: true, fetchAnimation: true));

    addAnimations(AnimationRole.MOVE, Direction.NORTH_EAST,
            new AnimationLinker( animationName: "MainCharacterNE_Anim",
                    AnimationRole.MOVE, Direction.NORTH_WEST, looping: true, fetchAnimation: true));

    addAnimations(AnimationRole.MOVE, Direction.SOUTH_WEST,
            new AnimationLinker( animationName: "MainCharacterSW_Anim",
                    AnimationRole.MOVE, Direction.SOUTH_WEST, looping: true, fetchAnimation: true));

    addAnimations(AnimationRole.MOVE, Direction.SOUTH_EAST,
            new AnimationLinker( animationName: "MainCharacterSE_Anim",
```



```
/**
* Checks if the players hurt is over
*/
public void checkIfHurtEnded() {
    hurtTime += 20; // playerHurt for 1 second

    if (hurtTime > 400) {
        logger.info("Hurt ended");
        setHurt(false);
        setRecovering(true);
        setTexChanging(true);
        hurtTime = 0;
    }
}
```



```
/**
* If the animation is moving sets the animation state to be Move else NULL.
* Also sets the direction
*/
public void updateAnimation() {
    getPlayerDirectionCardinal();

    /* Short Animations */
    if (!isOnVehicle) {
        if (getToBeRun() != null && getToBeRun().getType() == AnimationRole.ATTACK) {
            return;
        }

        if (isDead()) {
            setCurrentState(AnimationRole.STILL);
        } else if (isHurt) {
            setCurrentState(AnimationRole.HURT);
        } else {
            if (getVelocity().get(2) == 0f) {
                setCurrentState(AnimationRole.NULL);
            } else {
```



```
/**
* Check if player has recovered
*/
public void checkIfRecovered() {
    recoverTime += 20;
    this.changeCollideability( collidable: false);

    if (recoverTime > 1000) {
        logger.info("Recovered");
        setRecovering(false);
        setTexChanging(false);
        changeCollideability( collidable: true);
        recoverTime = 0;
    }
}
```

REFLECTION:

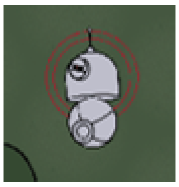
From my peers I have learnt that in a collaboration studio, teams have to communicate with good documentation (e.g. include a step-by-step guide) and an already implemented example code for reference.

If I have to do it again, I would have asked in Slack who wrote the rendering of animation sprites', so that the Shield's opacity is also rendered. When the player summons a Shield, a slightly transparent sprite should be applied on the player. However, I can't seem to modify the *Renderer3D* class to render opacity of the Shield sprite so I kept wasting time to do trial and error to find the solution while I could actually ask the author of the code how to do it.

Before leaving the Spells team for the Enemy Team, I should also have reminded the Spells team to fix the clash of Tornado spell summon having the same key control *C* as the one for showing tile name. I have found that the previous World Generation team have written the *C* key to show on the tiles the name of the biome, but the Combat team still has not communicates with them at the end of semester. Therefore, when the player pressed *C* in the game, there is still 2 different methods being called at the same time.

I should also help the studio to create an image indicating all key controls on a keyboard.

3.2 Enemy-redesign: Help in tidying up enemy-related classes, troubleshooting failed tests in the team



Being in the new Enemy team, we focused on graphic designs pivoting towards a Cyber-punk theme enemy, having levels of difficulty and random-spawning in the game as features.

---

a. In terms of ideating, **I cleaned up the original enemy classes in order to help the team to inspect possible codes for class redesign**. Ticket [#383](#383)

WHY: Since I am the only programmer in the team who have collaborated closely with the original enemy team for 3 sprints, I understand how the enemy's code depends on other classes. Therefore, when our team is discussing on the whiteboard of what methods we need, I tidied up the old enemy team's classes.

HOW: I **extracted the subclasses' duplicated methods to be inherited from** *Enemy* (e.g. *Spider*, *Flower*, *Treeman* subclasses both have the exact same *randomMoving()* algorithm, thus I extracted them into the *Enemy* parent class). This later helps the team in understanding how the enemy's combat process with just one parent class instead of five duplicated subclasses.
I have also **corrected mistakes when some programmers misunderstand the** *Enemy* **class's dependencies**. For instance, when one teammate wrote the *setScale()* method (set the difficulty of the enemy by setting its strength, speed……), I have noticed that *setScale()*'s strength parameter is a *float* datatype, which doesn't compile with the *playerHurt(int damage)* method that require an *int* datatype as parameter. Therefore, I have changed the parameter and fixed related tests.

---

b. In the aspect of programming, I **followed** ***@SamuelEagles*** **advice to use a string concatenation for optimizing import process of the animation objects into the** *Enemy* **objects**.

HOW: Though the process of working on this new importing system involved lots of hard work (changing all the names of the original enemy textures to fit exactly the same as the string concentration format: enemy1_Direction_state), it eased my pressure for later imports as I **shared my work with the designers** ***@Shao @Michelle @Fardeen*** **to help re-naming their animation sprites into the above format.**

---

c. In terms of testing, I have **frequently troubleshooted failed tests with** ***@Chalinda*** **before we merge our feature branch to master branch**. Ticket [#384](#384)

WHY: It is because the *Enemy* **class has dependencies on other major classes** like *MainCharacter* and *StatisticManager*, when we tried to rewrite algorithms of the old enemy class, some tests from depended class failed.
HOW: For example, when we replaced the enemy's subclasses *Treeman* with new types of enemies *Scout*, *Heavy* and *Abductor*, Chalinda remind me of the failed tests on Slack. After we **traced the errors with Gradle's clean build**, the failed tests are because *WorldDirectorTest*, *StatisticManager* class still uses *Treeman* as a tested enemy object.

REFLECTION: I should have added a GitLab task ticket with the Inventory team to make the enemy drop gold pieces after enemy died, since our Enemy team added the dropGoldPieces() method in a hurry just before the demo, and it caused an error. If I can redo it, I could have made a ticket labelled with Enemy team and Inventory team to remind both teams to add the method as the goal of the game said "eliminate enemies in each location using your weapons and collected resources. Killing these enemies will reward you with gold pieces".

4. Final Reflection and Future Improvement

What I have learnt?

Outside of the feature team's work, within the studio I have also learnt first-hand the actions in SCRUM methodologies like stand-up meeting, team showcase, team retrospect and a 2-week sprint interval. As a programmer, I have also learnt to use various integration tools like Git (continuous integration of code work), Jenkins (ensure successful build and combine other tools), SonarQube (for code quality debugging) and GitLab wikis (for communicating code usage).

What have I done good?

As part of the studio, I think I have made good of GitLab Kanban/Issue board, task ticket and wikis to manage and track my work with teammates. For example, in GitLab when one adds a ticket link reference "#233", it will be shown as "#233 (closed)" when it is done. Therefore, I use this feature to keep track of the designer's work when working on Enemy animations.

What have I done bad?

I should also have made a wiki page about how to import animation sprites. Since during Sprint 3-4, there are other designers who wanted to design the 2nd main character sprites don't know how to design animation sprites. Even though I have told them to take a look at the previous designer's work, they did not follow it. Making a wiki page about the animation sprite format requirement may resolve that.

In the future, I hope to use the above team process management tools and methods learnt into an actual production team.