A2 - Linguagem de Programação

Eric Manoel Ribeiro de Sousa
Lucas Menezes de Lima
Victor Gabriel Haruo Iwamoto

November 2024

1 Introduction

shattered is a 2D platformer game developed in Python, designed as the second evaluation method for the Programming Languages course in the Applied Mathematics program at FGV. The game aims to demonstrate mastery of object-oriented programming (OOP) concepts and class usage, which were covered throughout the course.

2 Game History

In ancient times, the Freydis clan was revered for their bravery and skill in hunting legendary beasts. During an epic journey, the most powerful warriors, Erik and Stella, ventured into the ruins of the ancient palaces of Vermont with an impossible mission: to destroy the Three Calamities, mythological creatures of immeasurable power.

Balrog, the Lightning God, Ganon, the Indestructible, and Demogorgon, the Winter Demon, were responsible for the chaos and destruction that plagued the region. Though their courage was unwavering, Erik and Stella succumbed to the devastating power of the Calamities, their bodies consumed by the shadows of the beasts. With the death of the heroes, the palace door was sealed with mystical powers, as a warning: no living being could ever dare to challenge those terrible entities again.

Centuries passed, and the legend of the Three Calamities was nearly forgotten. However, in the present era, a new hero emerges: August, a direct descendant of the Freydis clan. Through an inexplicable phenomenon, he inherits the abilities of his ancestors, able to transform into them and access their powers to continue the unfinished mission of vengeance and redemption. His journey will be arduous and filled with challenges, as he will face monsters with unique abilities, which can only be defeated with the help of the spirits of his ancestors.

To break the seal of the Vermont palace and finally face the Three Calamities, August must locate the Obelisk of Eldrith, a mysterious artifact capable of

breaking the ancient magic that protects the palace. However, the path to the Obelisk is long and filled with immense dangers, and the young hero will need to master the powers of his ancestors to overcome these obstacles and reclaim the lost honor of his clan.

Will August be able to honor his family's legacy, defeat the Three Calamities, and restore the greatness of the Freydis? Only fate will tell.

3 About the Game

Shattered is a 2D platformer where players control three characters—Stella, Erik, and August—who alternate throughout the game. The objective is to defeat all the enemies, unlock the final boss door (obelisk), and defeat the last three enemies. Each boss or enemy defeated rewards players with points, and at the end of the game, the total score accumulated throughout the journey is displayed.

3.1 How to Play

Movement is horizontal: use A and D to move left and right, respectively. To attack, press the V key, and to interact with objects (obelisk) in the environment, press F. To switch characters, press Z to switch to August (the Knight), X to switch to Stella (the Yokai), and C to switch to Erik (the Ninja). All characters attack using the V key. Additionally, on Stella pressing W and V simultaneously causes the character to fire a projectile upwards.

3.2 Mechanics

In shattered [1], the enemies are designed specifically for the unique abilities of each character. As such, each character is essential for progressing through different scenarios in the game.

3.2.1 Characters

Each character has unique stats that align with their playstyle.

- **Erik**: A ninja wielding the Totsuka Katana, Erik excels in close-range combat with high dexterity, giving him great speed and the ability to double jump. However, his health is average. [2]
- **Stella**: A Yokai with the Housenka ability, which allows her to shoot fireballs. As a long-range character, Stella has the lowest health but compensates with high agility, enabling her to quickly move to various attack positions. She also has the ability to double jump, making it easier for her to reach high platforms. Her skill is especially useful against flying enemies or those at a great distance. [3]

- **August**: A knight equipped with the Yata Mirror, a shield capable of blocking any projectile. Due to his large size, August has low mobility and can only jump once, but he compensates for this with high health. He is best used against enemies that fire numerous projectiles, which are difficult to avoid. [4]

3.2.2 Enemies

- Dummy: A basic enemy that walks back and forth, changing direction when it hits a wall or at the end of a platform. It has no attack and only damages the player upon collision. [5]
- Mage: This enemy remains stationary, always facing the player and shooting projectiles. It also deals damage upon contact. [6]
- Flying: A flying enemy that occasionally changes direction and drops projectiles downwards. The player takes damage upon touching it. [7]

3.2.3 Bosses

- Demagorgon: This boss always follows the player. The player takes damage upon contact with the Demagorgon, and its primary attack is a punch. [8]
- Balrog: The Balrog moves randomly in the air. After a random time interval, it selects a direction and flies in that direction for a set number of frames before choosing a new direction. Its attack consists of lightning strikes falling from the sky. Before attacking, Balrog teleports to a spot where it cannot be harmed. Always follow Balrog to avoid damage. [9]
- Ganon: Ganon remains stationary at the edges of the screen. When the player approaches, Ganon enters an immune form and teleports to the opposite side of the screen. Ganon is immune to Stella's fireballs and invulnerable to Erik's sword due to its tough skin. Ganon's attacks consist of large rocks that are difficult to avoid. [10]

4 Structure

```
shattered/
├── src/                        #Main source code
│   ├── __init__.py
│   ├── assets.py
│   ├── boss.py
│   ├── camera.py
│   ├── enemy.py
│   ├── game.py
│   ├── ground.py
│   ├── interfaces.py
│   ├── main.py
│   ├── maker.py
│   ├── player.py
│   ├── score.py
│   └── weapon.py
├── tests/                      #Unit tests
│   ├── __init__.py
│   ├── test_boss.py
│   ├── test_enemy.py
│   └── test_player.py
├── docs/                       #Documentation
│   └── files .tex
├── assets/                       #Images and Assets
│   └── files .png
├── requirements.txt            #Dependencies of the project
└── README.md                   #General view of the project
```

5 What Each One Have Done

During the development of the project, all team members remained consistently active, as evidenced by the commits made. Every member contributed to all aspects of the game; however, we decided to modularize the code to improve workflow. As a result, some parts of the game were primarily developed by specific team members.

Eric Ribeiro was responsible for the initial structure of the game, including the character physics and the modularization of functions. He also implemented key mechanics, such as the obelisk (used to push the stone blocking the boss gate), camera movement, and the game design for the level (strategically positioning enemies, spikes, and platforms). Additionally, Eric worked on the

soundtrack and the score mechanic. Overall, Eric focused on the structural aspects of the game and provided assistance with other functions when necessary.

Lucas Menezes primarily developed the mechanics for the main character, including the weapons (sword, shield, and projectiles), animated the player's actions, and implemented the health bar. He also assisted in sourcing sprites. Furthermore, Lucas actively participated in the development of the menu screens and the implementation of the text elements and made the unittest.

Victor Iwamoto played a key role in defining the game's aesthetic identity, including sprite selection and menu design. He was responsible for developing the mechanics of all bosses and enemies (attacking, movement, and other behaviors), standardizing the sprite implementation method, and compiling the final report.

It's important to note that each member documented the code they contributed.

6 How to Run

6.1 How to play the game?

- Create a virtual environment.
- At folder root *shattered/* install the dependencies.
- Go to *src/* folder and run *main.py*

```
# In Unix
$python3 -m venv venv
$source venv/bin/activate
$pip install -r requirements.txt
$python3 src/main.py
```

If you are using WSL, the soundtrack may encounter errors due to the absence of an audio mixer. To resolve this, run the following code to install an audio mixer.

```
$sudo apt update
$sudo apt install pulseaudio
```

6.2 How to run unittest?

- In root *shattered/* go to *tests/* and run the following line code.

```
$ python3 file_name.py
```

7 Challenges

The entire game presented numerous challenges, as it was an ambitious project.

One of the initial hurdles was modularizing the game. Each element within the game interacts with others, making it difficult to develop a system where all components could function simultaneously without conflicts. The solution was to break the game into several distinct modules, ensuring better collaboration and maintainability.

Developing realistic physics was another significant challenge. This required implementing gravity, velocity on both axes, and terminal velocity. Terminal velocity proved particularly complex because, during jumps, the character would sometimes pass through the ground due to high speeds exceeding the ground's frame update. To solve this, a maximum velocity limit was introduced, preventing the character from clipping through surfaces.

Character swapping posed additional difficulties. The game features multiple heroes, each with unique abilities. However, maintaining consistent stats and positions during swaps was essential, which added complexity to the logic.

Another major challenge involved sprite handling and animations. Initially, characters were represented by rectangles for hitboxes, but incorporating animations required rethinking attack and movement logic. Additionally, the diverse range of characters, enemies, and bosses meant creating a modular system that could animate all of them without redundant code.

Camera movement also proved challenging. As the character moves, the camera must follow, which requires all objects in the scene to adjust accordingly. This synchronization was complex to implement. Furthermore, interface transitions, such as pausing the game to enter menus, required careful handling to ensure a smooth and seamless experience.

References

[1] Brullov. Sprite ground and background. `https://brullov.itch.io/oak-woods`, 2024.

[2] CraftPix. Sprite erik. `https://craftpix.net/freebies/free-ninja-sprite-sheets-pixel-art/`, 2024.

[3] CraftPix. Sprite stella. `https://craftpix.net/freebies/free-yokai-pixel-art-character-sprites/`, 2024.

[4] CraftPix. Sprite august. `https://craftpix.net/freebies/free-knight-character-sprites-pixel-art/`, 2024.

[5] MonoPixelArt. Sprite dummy. https://monopixelart.itch.io/forest-monsters-pixel-art, 2024.

[6] Xzany. Sprite mage. https://lionheart963.itch.io/wizard, 2024.

[7] Xzany. Sprite flying. https://xzany.itch.io/flying-demon-2d-pixel-art, 2024.

[8] Xzany. Sprite demagorgon. https://chierit.itch.io/boss-frost-guardian, 2024.

[9] Xzany. Sprite balrog. https://chierit.itch.io/free-cthulu, 2024.

[10] Xzany. Sprite ganon. https://darkpixel-kronovi.itch.io/mecha-golem-free, 2024.