Game Design Portfolio

by Enri.

Materials enclosed in this document are not to be released to the public. Please do not distribute. Thank you!

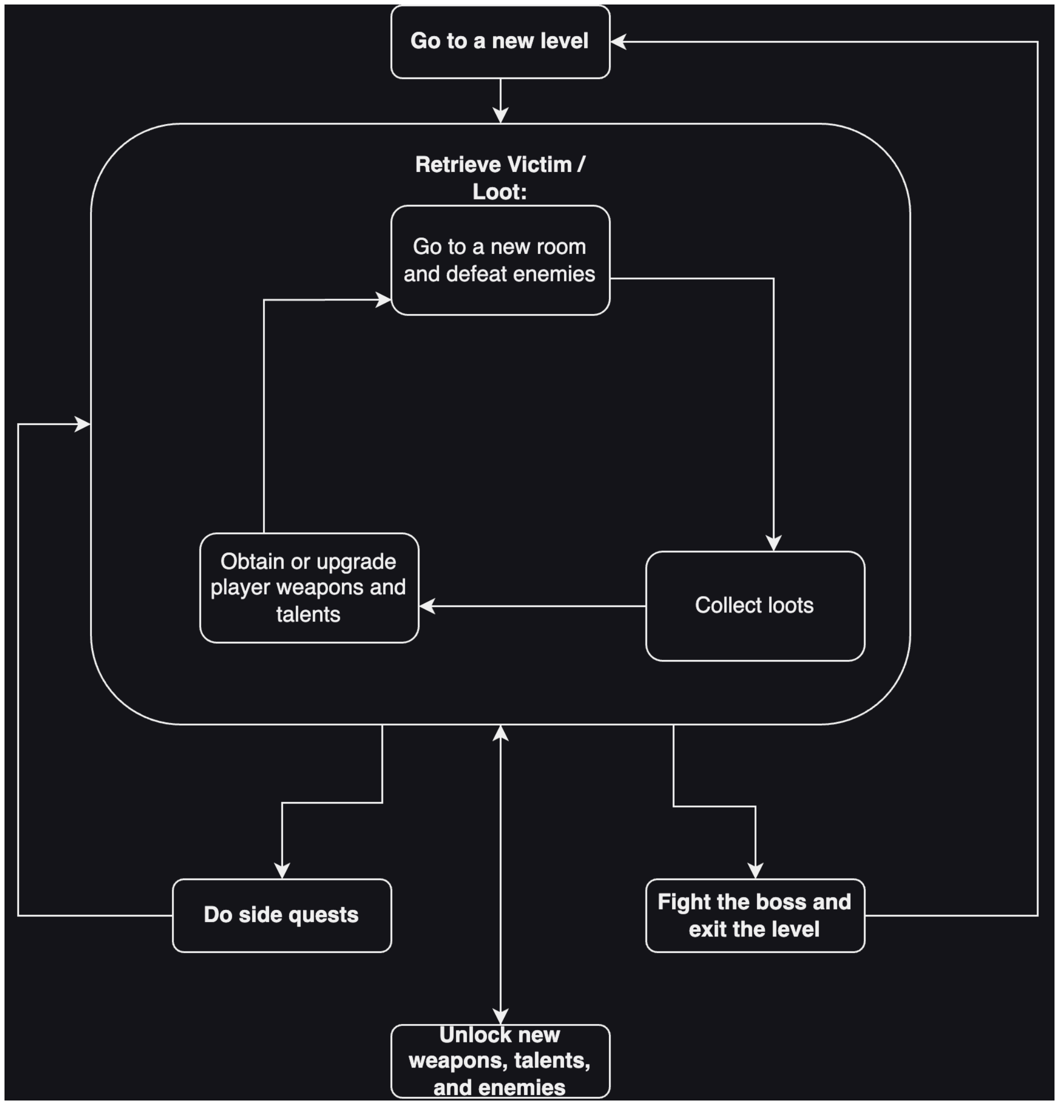
Renfield: Bring Your Own Blood

Credits: Skybound Entertainment

Game trailer here: https://youtu.be/ifnwsapdkCY

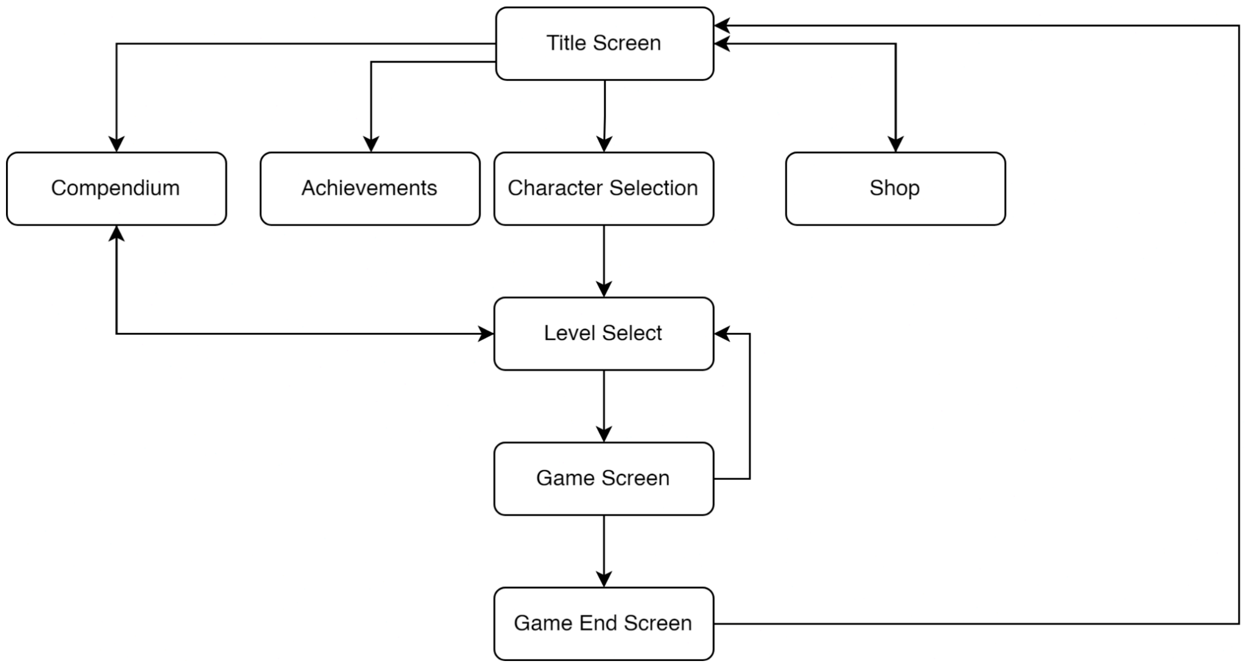
Renfield: Bring Your Own Blood is a 2D Roguelite game published by **Skybound Games** for Windows PC where the player plays as *Renfield,* a tortured slave to Dracula, whose goal is to retrieve Dracula's next *food.* To do this, Renfield must go through rooms, clearing waves and waves of enemies until they retrieve the victim.

Since we were targeting a game that is simple and non-alienating to casual gamers, I made sure that the core gameplay loop reflects this. The game loop focuses on entering a new level, defeating enemies, and unlocking things to get stronger. This constant stream of unlocks keeps the players engaged and thrilled.


Go to a new level
Retrieve Victim / Loot:
Go to a new room and defeat enemies
Collect loots
Obtain or upgrade player weapons and talents
Do side quests
Fight the boss and exit the level
Unlock new weapons, talents, and enemies

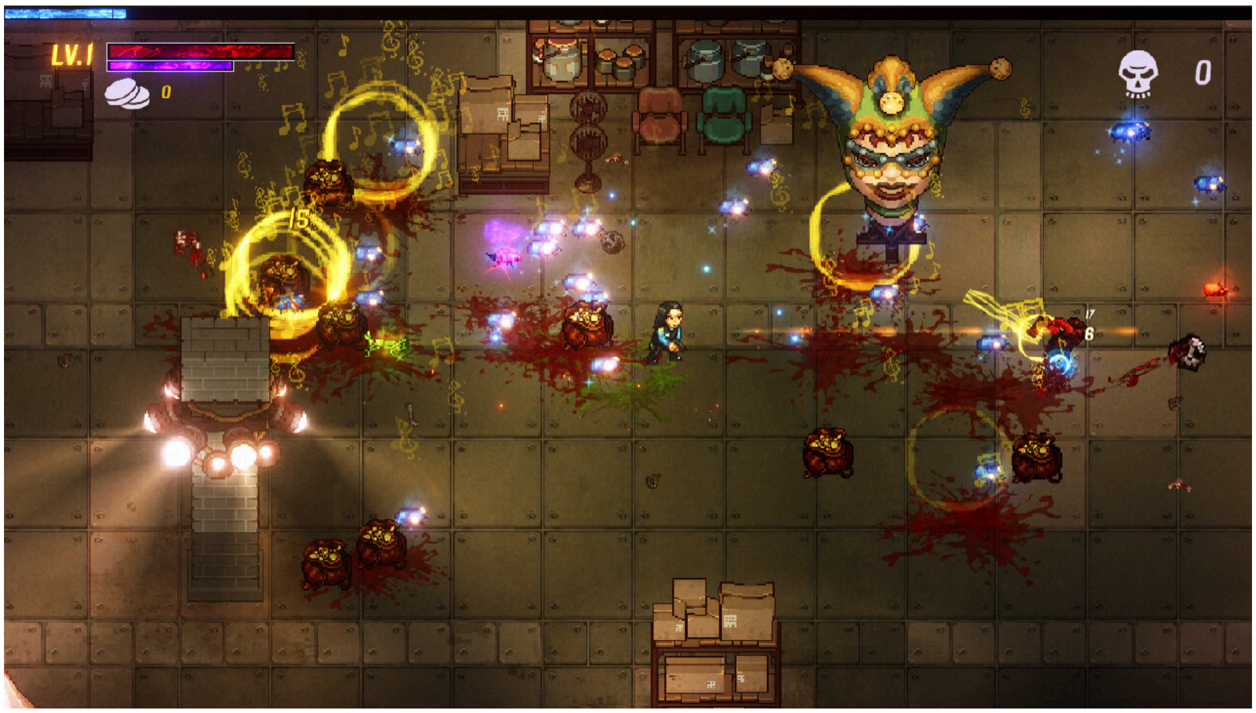
Gameplay Loop

Aside from that, I've kept the screen counts as minimal as possible to reduce the decision that the player has to make when opening the game. After all, we want them to focus on playing the game.

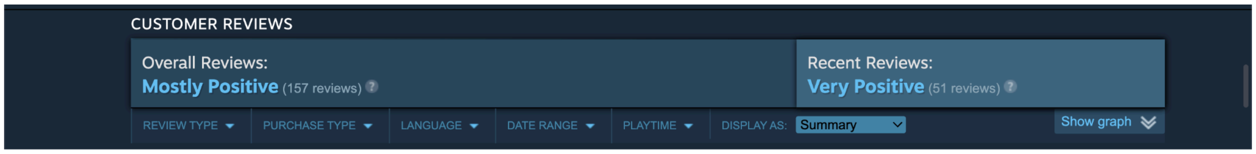


Screen Flow

The simplicity of this game is even more highlighted by its control system. Since the attacks in this game auto-fires instead of needing a button press to attack, we've removed an additional load for the players, allowing them to fully focus on the screen, seeing the cool VFXs as they kill enemies! All this, while creating an additional challenge to time and position properly to maximize their weapons.



I designed the majority of the gameplay system for this game. Through my contributions, we were able to release a game that fulfilled and went above the expectations of our publishers and buyers, earning a **Very Positive** rating of **81%** on its **Early Release Stage**.



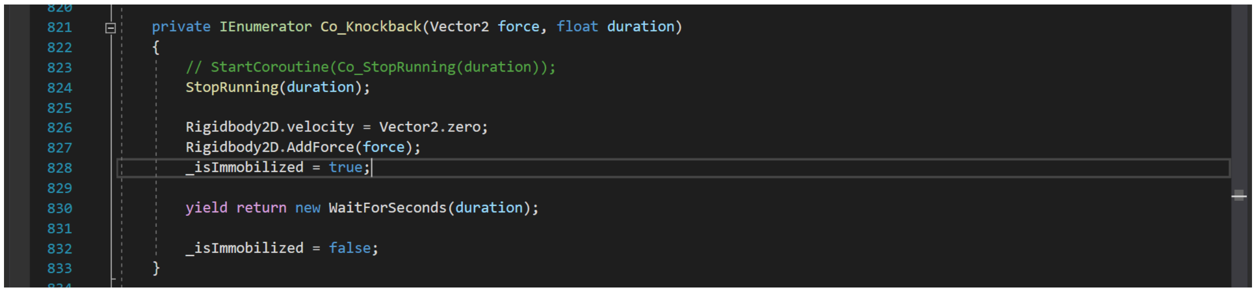
CUSTOMER REVIEWS

Overall Reviews:
Mostly Positive (157 reviews)

Recent Reviews:
Very Positive (51 reviews)

Credits: Renfield: Bring Your Own Blood on Steam

Production became streamlined due to my efforts in creating updated and concise documentations, and being very hands-on in communicating with our developers and artists. I also assisted the developers by consistently reviewing errors in logic, or implementing fixes in code.



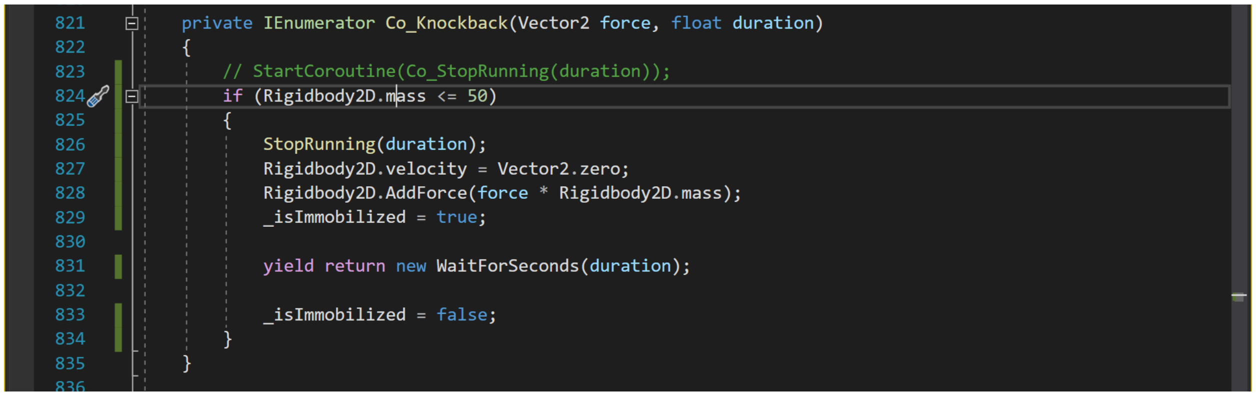
```
private IEnumerator Co_Knockback(Vector2 force, float duration)
{
    // StartCoroutine(Co_StopRunning(duration));
    StopRunning(duration);

    Rigidbody2D.velocity = Vector2.zero;
    Rigidbody2D.AddForce(force);
    _isImmobilized = true;

    yield return new WaitForSeconds(duration);

    _isImmobilized = false;
}
```

Example: These were the lines of code handling the knockback mechanic. It was not working as intended because the force value the function was taking was too low, and if we bump it, there would be other consequences in balancing and UX. V1.0.

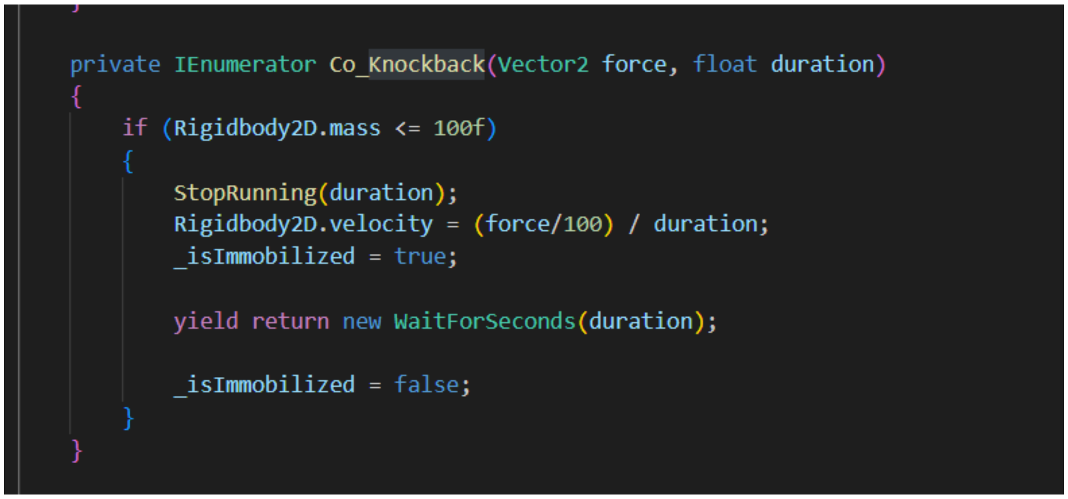


```
private IEnumerator Co_Knockback(Vector2 force, float duration)
{
    // StartCoroutine(Co_StopRunning(duration));
    if (Rigidbody2D.mass <= 50)
    {
        StopRunning(duration);
        Rigidbody2D.velocity = Vector2.zero;
        Rigidbody2D.AddForce(force * Rigidbody2D.mass);
        _isImmobilized = true;

        yield return new WaitForSeconds(duration);

        _isImmobilized = false;
    }
}
```

Initial fix for the knockback code. Aside from fixing the knockback without ruining the balancing, this also restricted knockbacks to only those that had below 50 (which were the non-boss enemies). V1.1.



```
}

private IEnumerator Co_Knockback(Vector2 force, float duration)
{
    if (Rigidbody2D.mass <= 100f)
    {
        StopRunning(duration);
        Rigidbody2D.velocity = (force/100) / duration;
        _isImmobilized = true;

        yield return new WaitForSeconds(duration);

        _isImmobilized = false;
    }
}
```

The initial fix was hypothesized to have caused performance issues because the game was computing large amounts of Newtonian Force over a large number of units every time a knockback happens. To optimize this, we opted for a velocity-based computation, leveraging the knockback duration parameter to make sure that the knockback still feels natural (or at least, not abrupt). V.2.0.

Contributions include:

- Designed the **Core Gameplay Loop** and the **Progression Systems.**
- Conceptualization of the interfaces, as well as the screen flows
- Designed the game's **major** features: **Economy, Enemies, Weapons, Shop Items, Characters, and Stat System.**
- Designed the game's **major** mechanics: **Combat System, Shop and Economy, Collectibles.**
- Researched on comp games for better insights on how game features can be improved during the **Post-Release** phase according to player expectations.
- Oversaw whether the features are programmed correctly, and revise the logic in scripts whenever necessary.
- Constantly tested the game to identify pain points, and heavily worked on balancing reward schedules and the game contents according to player and publisher expectations.

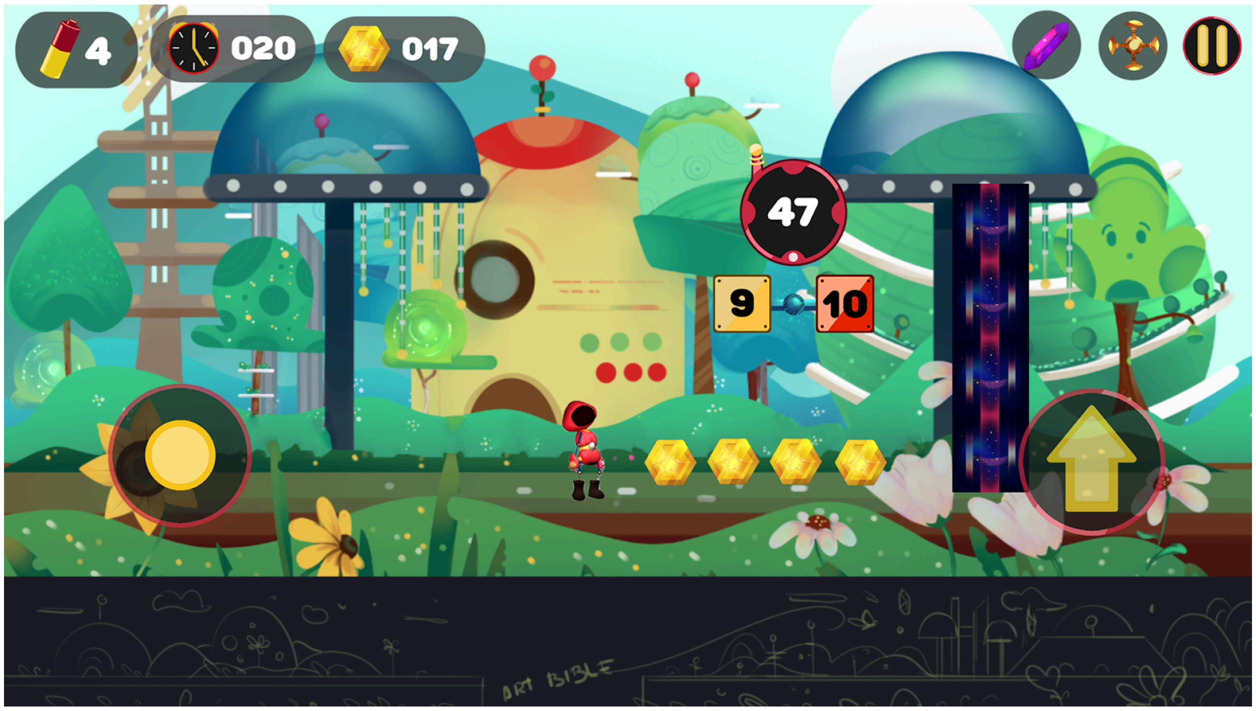
Operation S.Y.B.E.R.

OPERATION
S.Y.B.E.R
A 2D PLATFORMER GAME REINFORCING KNOWLEDGE OF
INTEGER OPERATIONS

Illustration by me

Vertical Slice Trailer here:
https://drive.google.com/file/d/1zbv7BZm450o1j3q9OIBioc7Kpz2NDfeG/view?usp=share_link

Operation S.Y.B.E.R. is a 2D Platformer Educational Game that aims to teach integer operations to Grade 7 Students, following the official curriculum for Math issued by the country's **Department of Education**. This was produced for our Research Capstone Project subject, where we attempted to create an engaging alternative to learning Math that is accessible to our target demographics in the Philippine context.



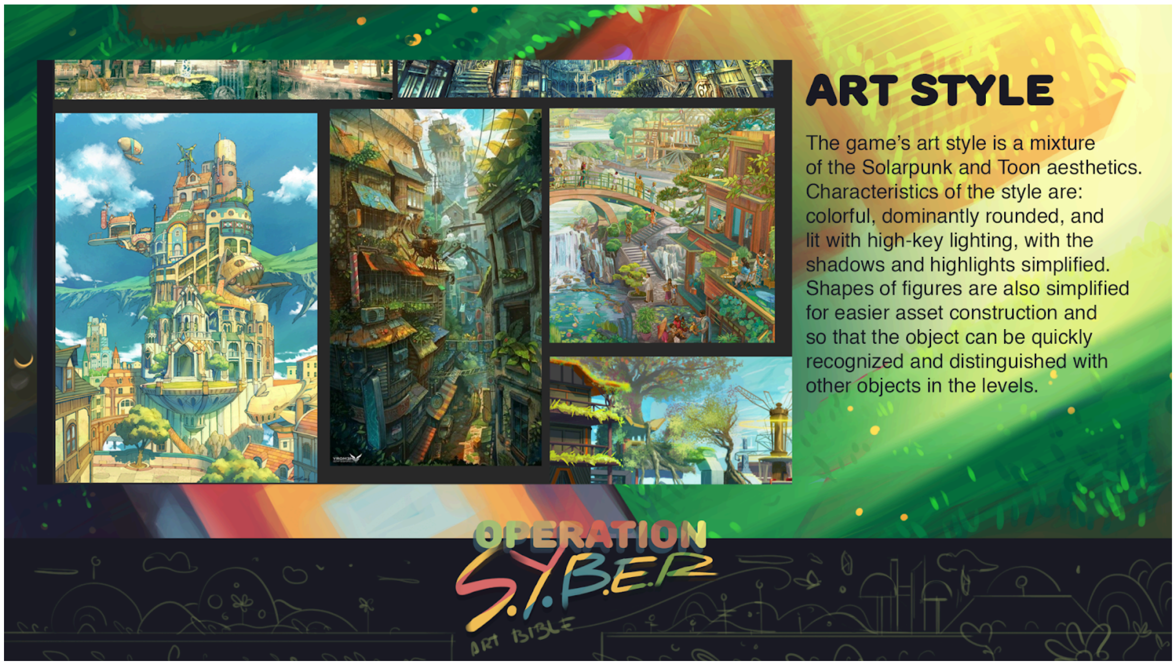
Abstract (lifted from our research paper):

"*Mathematics is a subject that many students usually find difficult to learn. Its technical nature, combined with societal expectations, contributes to the stress that students experience when working with numbers and operations. In contrast, video games are known to be more engaging activities for children to partake in. Most games immerse the player in fictional worlds that can stimulate their imagination or allow them to explore new skills through its different features and mechanics. It is from these two perspectives that the concept of game-based learning may be derived from. Game-based learning is an approach to education that utilizes the structure and mechanics of a game in order to teach a certain skill or subject.*

Given this approach, the researchers developed a learning game that would reinforce the knowledge of integer operations to Grade 7 students in the Philippines. The game was tested by both Grade 7 students and teachers alike, most of whom responded positively to it, citing its usefulness as well as being fun, engaging, and easy to commit to memory. The researchers then concluded that the game may be useful as an alternative learning tool which may be utilized by both teachers and students alike to further enhance their skills and knowledge on basic integer operations.

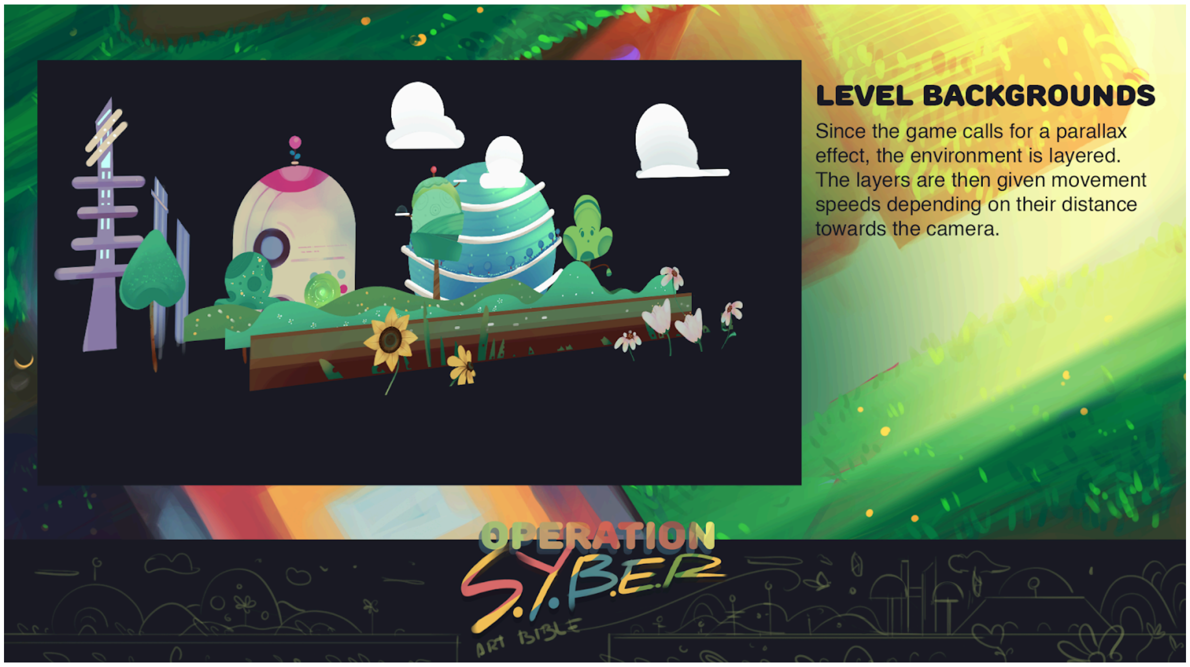
I created the entire design of the game– from the story and the characters, up to the gameplay systems and features. As this is a learning game, our content has to adhere to the learning outcome standards, and I made sure that we can achieve those by reviewing relevant materials and consulting the design with pedagogy experts.

I was the de facto *Art Director* as well. My efforts in researching styles and workflows / processes allowed us to complete the project without sacrificing the aesthetic cohesiveness and quality of the game, which were highly appreciated by the students and teachers alike.



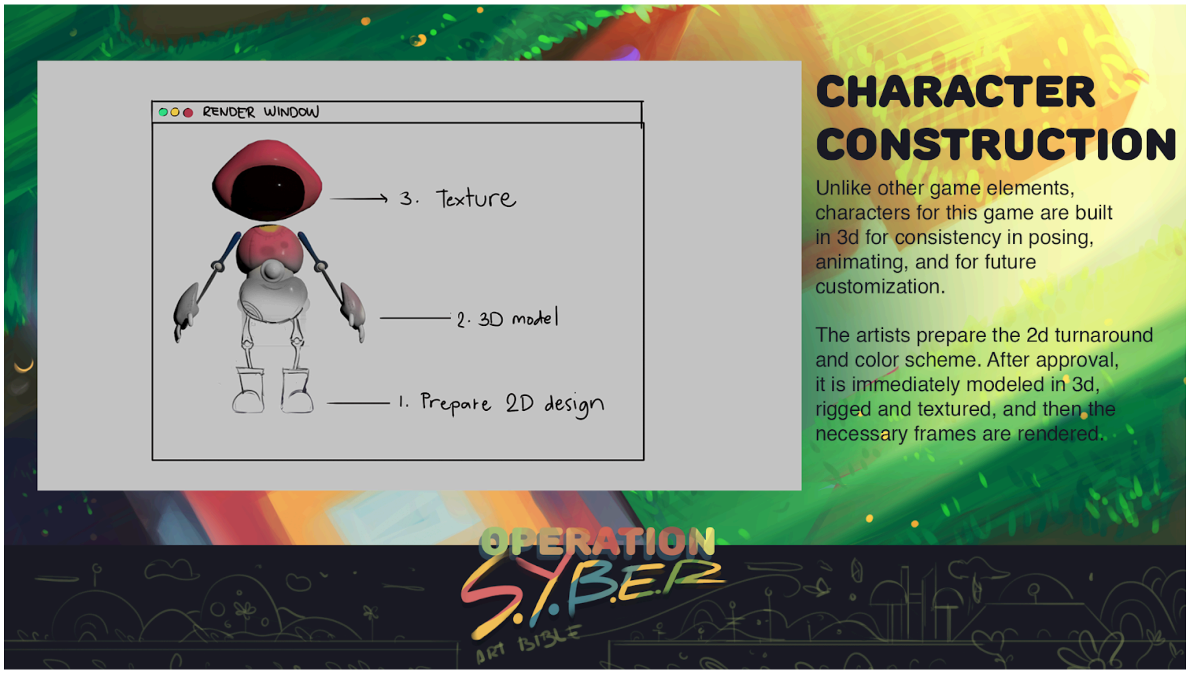
ART STYLE

The game's art style is a mixture of the Solarpunk and Toon aesthetics. Characteristics of the style are: colorful, dominantly rounded, and lit with high-key lighting, with the shadows and highlights simplified. Shapes of figures are also simplified for easier asset construction and so that the object can be quickly recognized and distinguished with other objects in the levels.



LEVEL BACKGROUNDS

Since the game calls for a parallax effect, the environment is layered. The layers are then given movement speeds depending on their distance towards the camera.



CHARACTER CONSTRUCTION

Unlike other game elements, characters for this game are built in 3d for consistency in posing, animating, and for future customization.

The artists prepare the 2d turnaround and color scheme. After approval, it is immediately modeled in 3d, rigged and textured, and then the necessary frames are rendered.

Sample pages from Operation S.Y.B.E.R. - Art Bible

I also jumped back and forth in the engine to implement the levels and review the implementation of our learning mechanics, and assist the programmers for it.

Contributions include:

- Entire game design.
- Art direction. I worked heavily on setting the standards of art assets that will be used for this project, as well as guiding my teammates in developing assets to comply with this.
- **X** (player character) concept art and game asset.

- Environment art and design.
- Game props art and design.
- Level design and implementation in Unity.
- Logic review. I worked with the team’s programmers to make sure that we avoid arithmetic errors (Division by zero, Integer Operations with float returns).
- Balancing the problem sets that we used to make sure that it adheres to an engaging game progression system **and** the competency expected from our target demographic.

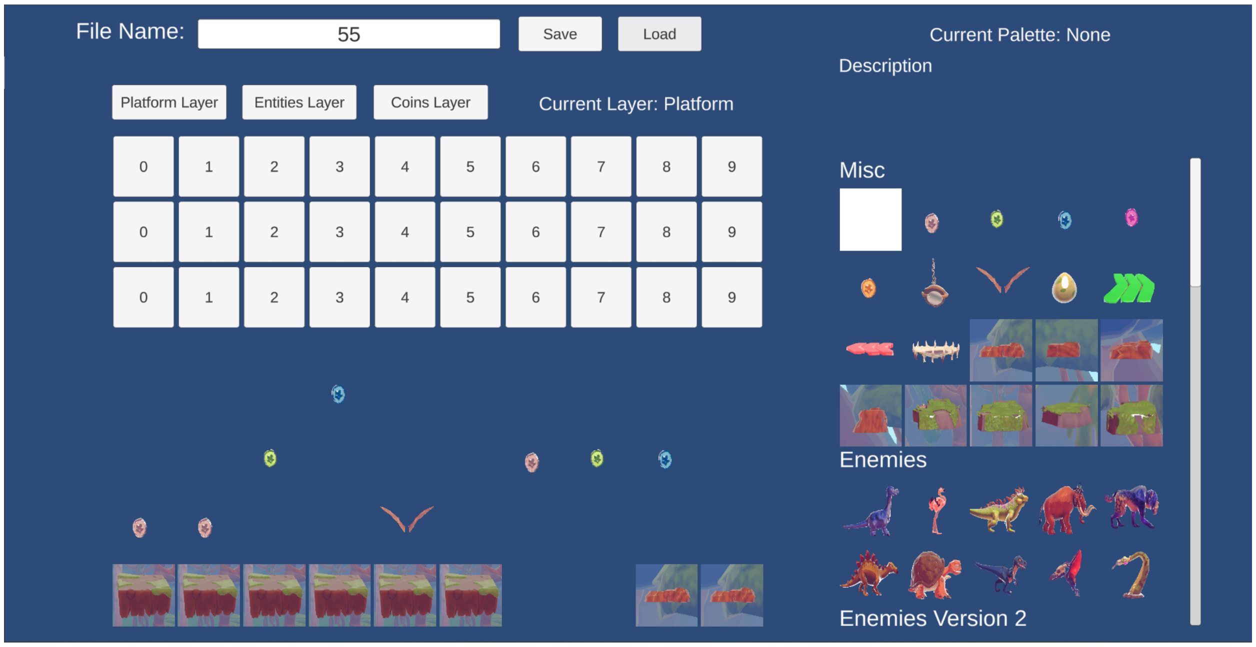
Dapper Dash

Credits: Dapper Dash Veecon Trailer.

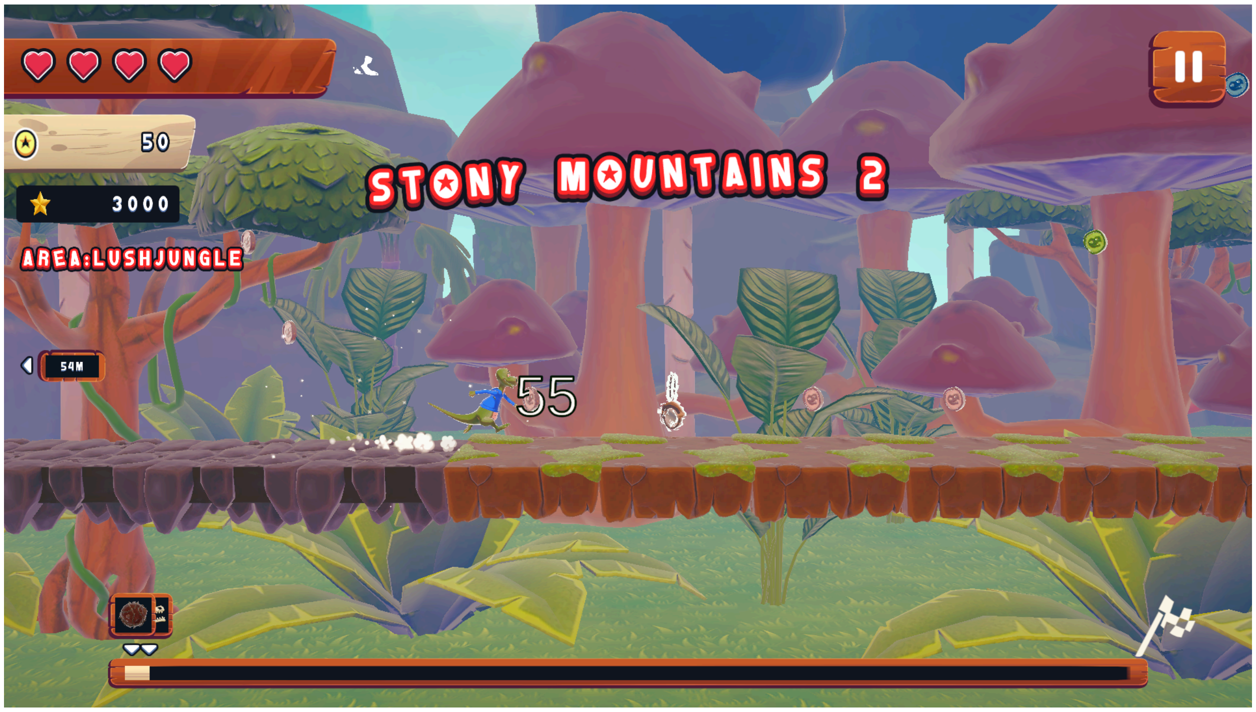
Game trailer here: https://www.youtube.com/watch?v=_0CV1vJSftI

Dapper Dash is a 3D infinite runner game for the Android and iOS mobile devices. In this game, you control a Dino, avoiding all sorts of hazards and enemies, while keeping distance from the giant boulder chasing you all throughout the level!

For this game, I worked on creating the initial level designs, as well as polishing and balancing objectives for the majority of it.



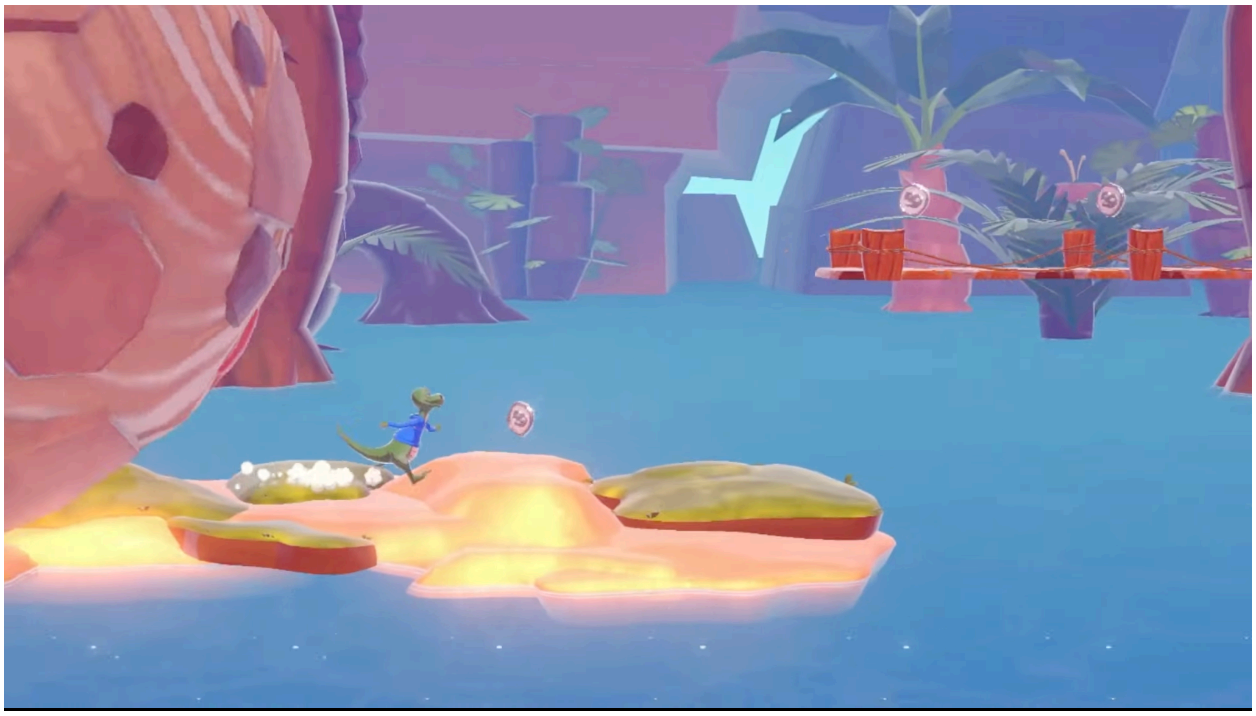
Designing using the in-house level editor…



… and testing it before sending it to production.

I've also established level design guidelines to make the quality of levels produced by ad-hoc level designers for this game unified.

To add more challenge to the levels, I worked on the design of a mechanic where a giant rolling boulder constantly chases the player and drives them to Game Over upon being rolled over. This mechanic, aside from the added tension, also made the *Speed-ups!* and *Slow Down!* powerups useful, which were totally useless before.



Lastly, I've worked on creating a sink for the in-game economy by designing purchasable powerups. These powerups promoted replays and helped create *burstiness* in gameplay results– which were extremely beneficial to us as we have a leaderboard system!

Contributions include:

- Level Design, Polishing, and Balancing.
- Created level design guidelines to unite styles.
- Designed the *Giant Boulder* mechanic to promote tension.
- Designed *powerups* to create an economy sink, promote replays through grind, and create randomness and burstiness in the leaderboard system.