STAR

STARFALL REBELLION

WARS

PROJECT DESCRIPTION

In Lego Star Wars: Starfall Rebellion, the player will help a rebellious leader and team up with an ex-convict in order to defeat the oppressive organization that is ruling over their world and their abuse, but not with the outcome that it would be expected.

MAIN CHARACTERS

During the duration of the first level, the player will only control Bix and her abilities, but upon reaching the end of said level, she and Allura will team up, and they will both be available to be controlled, so the player could decide which one to control in each situation.

BIX

A consumed pirate and scoundrel, Bix has always relied on her wits and talent for stealing to get through a galaxy ruled by the empire. Cynical and temperamental she will do anything if it serves her well, even lying about her being a padawan in her younger years.

Main weapon:
A lightsaber.
Basic abilities:
Advanced mobility and melee combos.
Special ability (out of combat):
Force powers.

ALLURA

A rebel agent hailing from Corellia, a cheerful smart woman who has a strong sense of justice and sees the good in people even when others do not see it. She wants to liberate the galaxy from the clutches of the empire and that's why she will join forces with Bix.

Main weapon:
A submachine rifle.
Basic abilities:
Advanced mobility and ranged combos.
Special ability (out of combat):
Hacking.

STORY

This story unfolds on the oppressed planet of Coltran, where an Evil Corporation, backed by the Sith Empire, has cultivated an environment of crime, poverty, slavery, and lawlessness.

Our protagonist, Bix, arrives with aspirations of wealth, only to succumb to the pitfalls of greed. Betrayed by the corporation's Top Man, she loses her prized ship and is cast into the sewers to meet an apparent demise.

Enduring five harsh years in the sewers, Bix emerges with a burning desire for revenge. Escaping, she makes her way to the hangars to reclaim her ship and embark on a journey into space.

With a plan to eliminate the Top Man, Bix infiltrates the corporation's space station, the Hand of Dominion, through a maintenance tunnel. In the hangar bay, she encounters Allura, her new companion in the quest to dismantle the corporation and free the planet from its grasp.

Through battles and traversing the station's bridge, the protagonists succeed in an epic confrontation, bringing an end to the Top Man and marking a pivotal moment in their mission to liberate Coltran from the Evil Corporation's influence.

THEME

This is a game about ambition and rebellion. There are action and hectic moments, as it is a hack and slash game, but it will also have a sense of humor and be kind of funny looking (LEGO).

WORLD AND AREAS

This whole story will take place in a far away galaxy from the Star Wars universe, which is why this game will have some resemblances that will make the player feel like they are in the same world as the movies/comics.
The player will go through different areas during the game, which will be:

The cantina and the sewers:

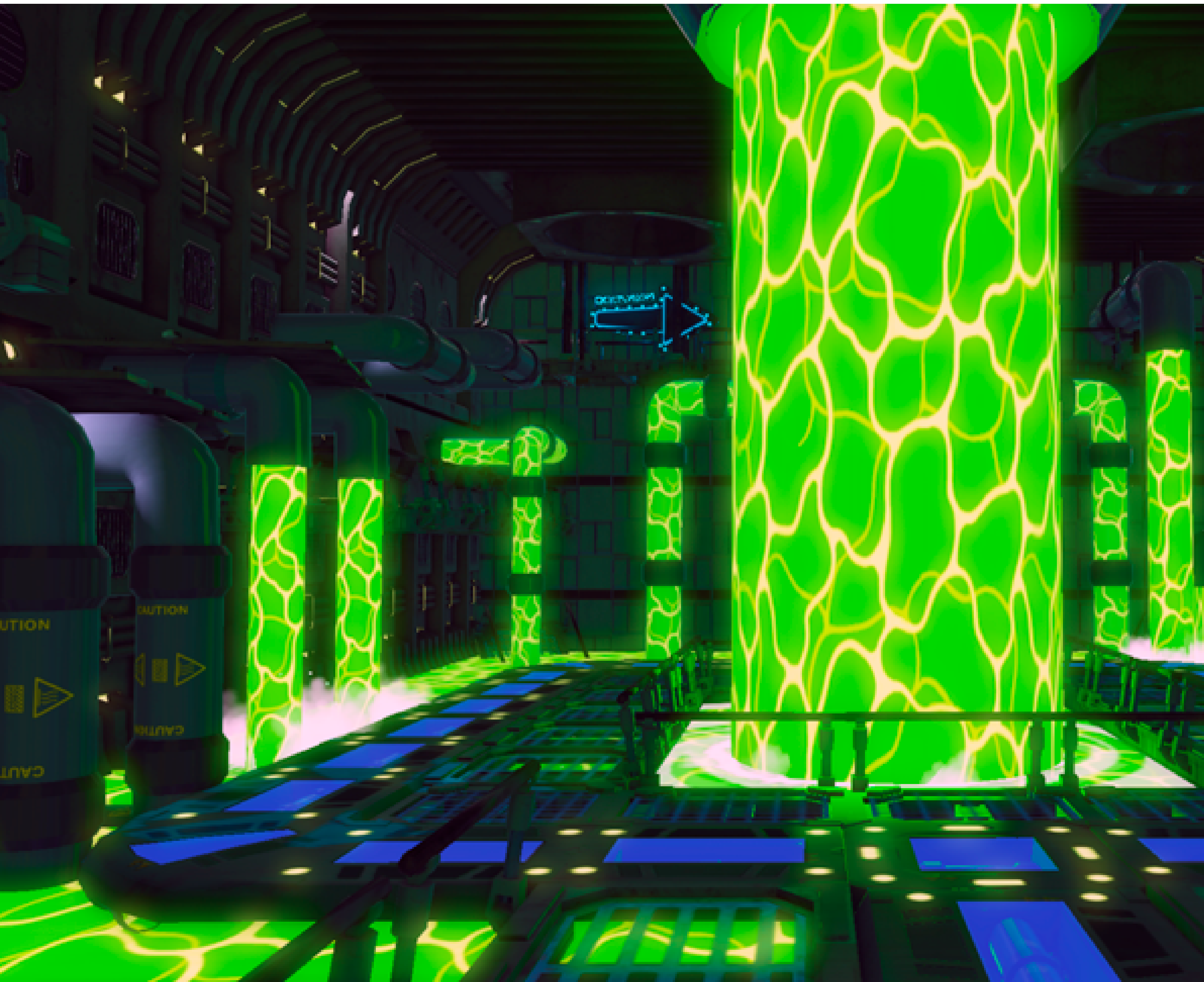
The place where Bix was thrown when Top Man defeated her. A place done for those who have been exiled by the evil corporation.

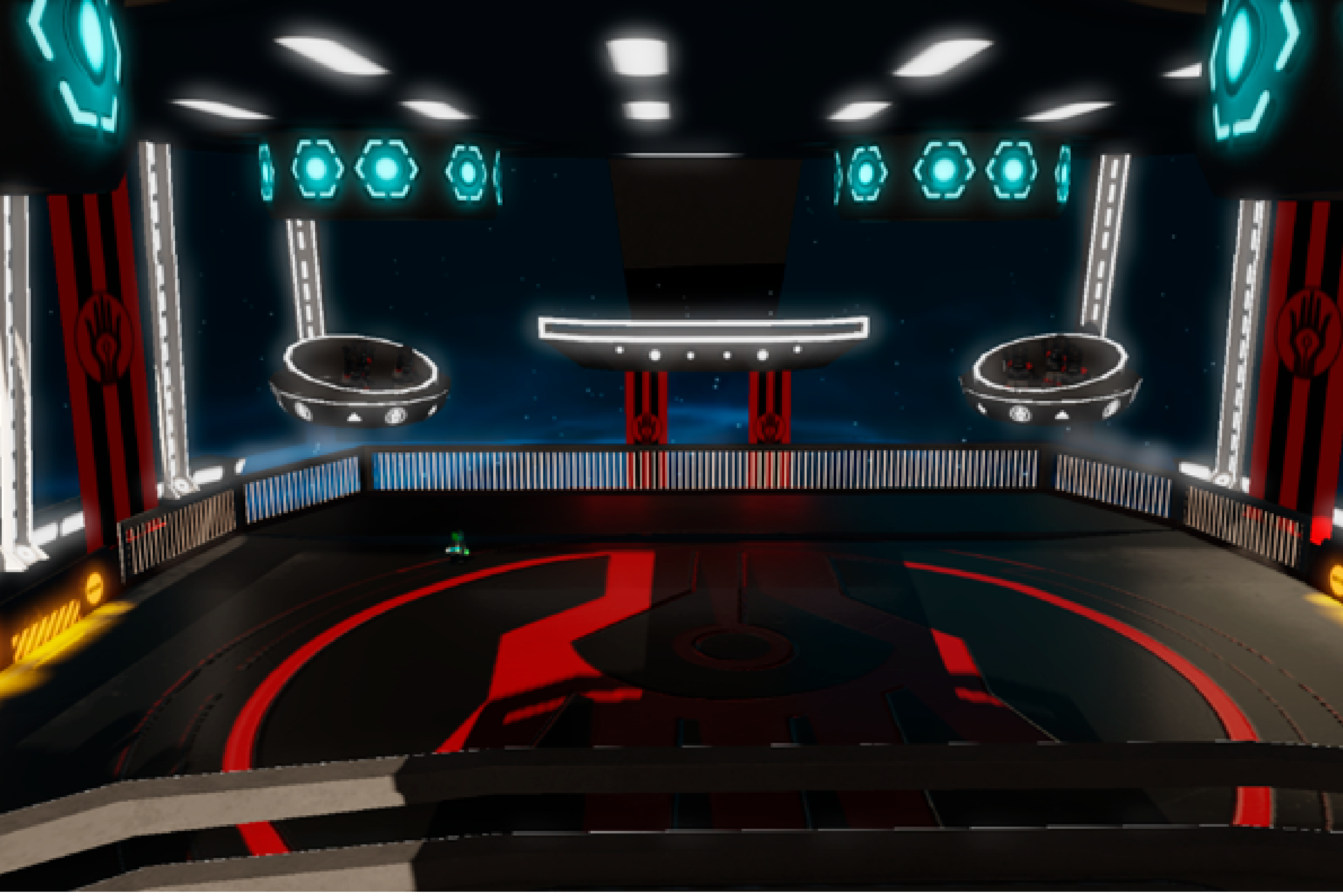
The Spaceship:

This is where Bix will obtain a spaceship and fly to the evil corporation's HQ to defeat them and exact her revenge.

The hand of dominion:

Here Bix will meet Allura and they will open a path to where the Top Man is, giving him defeat and unlocking the final battle that will face them with each other.





STORY PROGRESSION

The locations mentioned above will be connected linearly and will follow this succession:

[SEWERS -> HAND OF DOMINION -> FINAL BOSS]

From the cantina that is located in the sewers, the player will traverse the whole sewers, defeating enemies and going through some parkour areas to finally meet the miniboss of this level.

After defeating the enemies of the sewers, the player will reach the hangar, and here Allura will appear and she will team up with Bix to destroy Daxus' organization, they will fly together towards the Hand of Dominion using a spaceship.

Once they arrive at the Hand of Dominion, together they will have to go through several enemies and a miniboss and complete a hacking puzzle with Allura's help to finally reach Daxus and challenge him to a battle.

GAMEPLAY

GOALS

In the grand scheme (long term), the overarching objective is to vanquish Daxus and determine the fate of the ruling organization.

On the immediate front (short term) of gameplay, the focus is on conquering adversaries, progressing through diverse areas, and unraveling puzzles that impede your path.

GAME MECHANICS

3RD PERSON COMBAT

In addition, attacks will have only one target (they will not deal AoE damage), and we'll have a marker in the nearest enemy so the player knows which enemy is going to hit.

This marker will be much more visible for Allura, as her being ranged means that she will love to have a visual target of what she's hitting from the distance (e.g. Bayonetta 3 - Devil May Cry).

It will work the following way:

There will be two types of attacks that the player can perform, light attacks and heavy attacks (light attacks will be bound to the left click and heavy attacks to the right click). The idea is to have combos of up to three inputs (e.g. left click, left click, right click; that would be a combo). The motivation for performing these combos (aside from doing damage to the enemies) will be to fill a special bar that the player will have on the screen and which will allow them to perform a special attack once a combo is fully finished and the bar is fully charged.

In the game, a player can fill the bar of the special attacks bar by performing combos. Light attacks contribute more to filling the bar than heavy attacks, preventing abuse of heavy attacks. The bar slowly decreases when no combos are executed unless it's at maximum, staying constant until a combo is completed. When the bar is fully charged, the player gains the ability to unleash a special attack, with the specific attack determined by the key pressed in the controller.

LIGHT FINISHER:
Stun Special. Bix employs a force wave or force push to immobilize adversaries, disrupting their movements. Allura utilizes an electric grenade, strategically thrown to stun enemies, providing a tactical advantage in combat.

HEAVY FINISHER:
AoE special. Bix throws her lightsaber and it bounces between several enemies, while Allura shoots a heavy bullet and it bounces between several enemies.

The camera dynamically switches between exploration and combat modes. Designated "combat areas" feature a more flexible camera suitable for the combat system.

After clearing enemies from a combat area, it transitions into a standard exploration zone, with no further camera changes within that area.

MOBILITY

Focus on mobility skills, such as dashes or rolls, to avoid incoming damage or get to unreachable areas.

Bix will have a dash so she can quickly run from enemies if they heavily outnumber her. Allura will have mobility as well, but instead of a dash she'll have a roll so both movements feel unique for each character.

References: Mass Effect (Biotic Charge), Death's Door (Dashing), Final Fantasy XV (Warp Strike)

FORCE POWERS

Bix can use some force powers to move objects around and create new pathways to solve puzzles.

During the course of the game, there will be points blocked by some rocks or platforms that are too far to reach with a simple or a double jump, and in this situations Bix could use her lingering force powers to open a path through the debris or to draw near platforms so she could reach them by performing a simple or a double jump.

References: Control – Lego Star Wars Saga

SWAPPING CHARACTERS

In the game, Bix, armed for close-quarters combat, complements Allura, a ranged combat expert. The deliberate design prompts players to smoothly transition between them, injecting an immersive element. This not only enriches the combat experience but also unveils unique abilities specific to each character. The strategic interchange between Bix and Allura becomes pivotal, demanding players to swap characters at crucial moments to conquer challenges and propel advancement within the game's narrative.

Referents: The whole Lego Saga

HACKING

Allura, as a trained rebel agent, can hack into remote systems to open doors or disable security measures.

This hacking will be performed by showing a fast and easy minigame (like a kind of quick time event) in which the player will have to press a succession of buttons without missing or pressing a button a several times, all this with a time limit that will not make the hacking an easy task (check this as a reference: Hacking Reference (P5R)), so, by following this mechanic we could "reuse" a bit the combo system we'll have for the attacks and apply it here as well.

By completing this hacking minigame, the player would either win access to new areas so they could carry on with the game or gain certain benefits such as a powerup or something that would feel satisfactory to receive.

References: Deus Ex (Remote Hacking), Watch Dogs: Legion (Drone Hacking), Warframe (Panel Hacking)

ITEMS AND POWER-UPS

When facing different NPC enemies, some of them will be equipped with objects that will boast their combat capability, and when defeating them, they will drop their power-up (we could do this chance based). These power-ups will give a certain boost for a limited amount of time.
There will be four types:

DAMAGE UP:
Special stone that increases the main weapon's damage.

DEFENSE UP:
Spherical shield that blocks damage.

SPEED UP:
Rocket boots that increase the movement speed.

HEALING:
A healing kit that will recover the player's health.

Those power-ups will be dropped by the enemies in their position once they die (they will disappear after some time).

Also, these power-ups cannot be stacked, if you already had one saved and you pick a new one, the new one will replace the one you had saved. Also, you could not activate a saved power-up until the one you are using expires.

PROGRESSION AND CHALLENGE

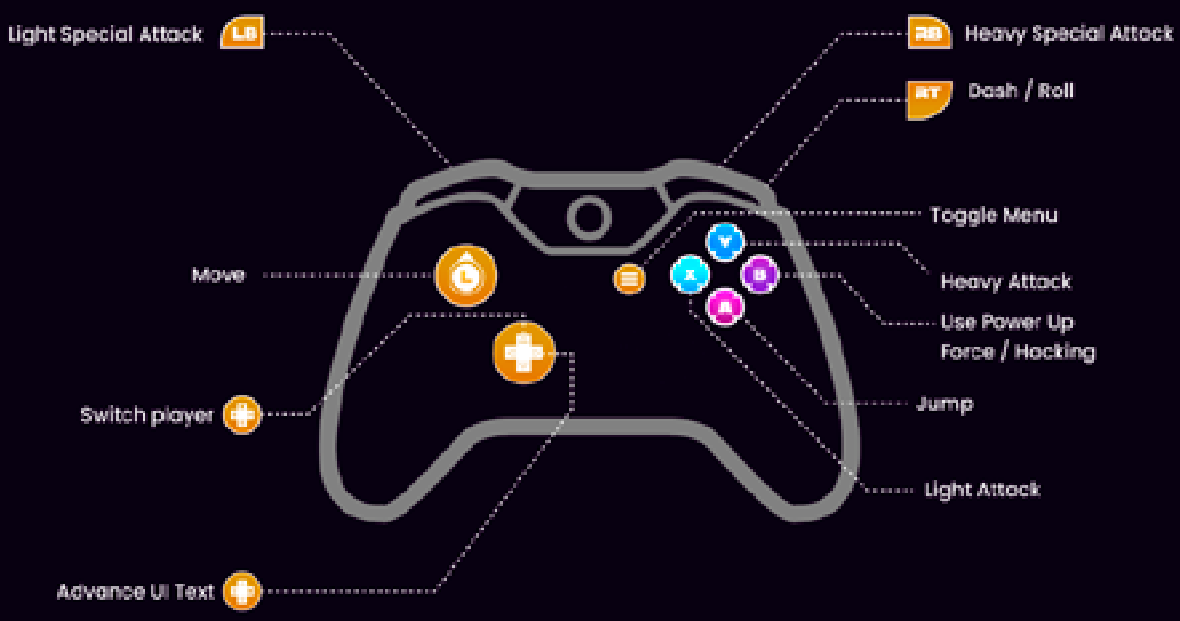
Due to the short duration nature of the game, there will be no noticeable improvements or any evolution for the player during the game, beyond those exposed during the game's story.

LOSING

The losing conditions in the game include having enemies reduce your life to zero, falling off a parkour area, or succumbing to poison. Upon losing, a Game Over screen featuring a laser saber is displayed, allowing the player to choose between retrying the game or returning to the main menu.

CONTROLS

Being a hack and slash game, its gameplay feels much easier with a controller, which is why despite being a computer game the main controls are XBOX. However, for all those who love the keyboard it can also be played this way.



ENEMIES

BASIC ENEMIES

These are the enemies that would recurrently appear throughout the duration of the game:

FLYING DRONES

Daxus' reconnaissance bots, a flying drone that will follow the player's trail and shoot them each certain amount of time.

Main Wapon: Laser gun.

VENOMITE (EVIL CORPORATION NPC)

Under Toxicus' command, Venomite faithfully follows orders, attired in a worn dark green tunic and a simplified gas mask resembling Toxicus'. Venomite's agile physique enables swift battlefield maneuvers, embodying a blend of obedience and adaptability in combat scenarios.

Main Wapon: Metallic baton / Submachine rifle.

MINI BOSSES

These are the special enemies that would serve as the end of level final challenge, increasing the clímax of the game (we wanted to add another one at the end of the level 1 but, due to lack of time, we finally couldn't).

MINIBOSS LEVEL 3 I X-27 SENTINEL

X-27 is a robotic support companion built by the same creator who designed Daxus. Unlike Daxus, X-27 has a more streamlined and agile frame, designed to complement his counterpart's abilities. His appearance is similar to that of a humanoid, but with mechanical details and tough armor.
Reference: Ebony Maw - Marvel

Abilities:
X-27 possesses the ability to generate temporary energy shields that absorb damage, providing a brief defense against enemy attacks. While shielded, X-27 is vulnerable, prompting the emergence of additional enemies to defend him during this period of heightened vulnerability.

Personality:
Unlike Daxus, who is ruthless and aggressive, X-27 has a more balanced and thoughtful personality. He is loyal to his creator, Daxus, and acts as a voice of reason and strategy on the battlefield.

Weapon:
Energy Beam: X-27 is capable of firing bursts of energy beams from his gun. These beams can track and strike enemies with precision, causing moderate damage.

FINAL BOSS I DAXUS, THE DEVASTATOR (TOPMAN FOR FRIENDS)

Daxus is an imposing being, a towering robotic creature with a battle-worn, blackened metal armor frame. His size is considerably larger than an average human, making him an intimidating figure in the hangar. His glowing eyes emit a sinister red light, while his right hand is a lethal cybernetic weapon.

Weapon:

Chaotic Plasma Hammer: Daxus' main weapon is a massive plasma hammer charged with chaotic energy. The hammer has a massive head with jagged metal spikes that emanate a red energy glow. Each blow of the hammer releases a shockwave of energy that can damage nearby enemies.

Abilities:

Brutal Charge: Daxus can charge at his enemies at high speed, ramming them with his metallic mass and knocking down those who get in his way. This charge will make some debris fall into the arena, blocking some of the player's path.

Energy Shield: Daxus can activate a temporary energy shield that protects him from enemy attacks. While the shield is active, enemies will keep spawning in order to defend Daxus. To deactivate the shield, the player will need to throw its light finisher to any rock above Daxus and make it fall over him.

Last Wish Missiles: When Daxus' health decreases to a critical point, he will disappear from the battle area and, from a control room nearby, he will start shooting missiles into the battle area and the player would have to avoid them until they stop spawning (this will likely only happen once).

UI AND HUD

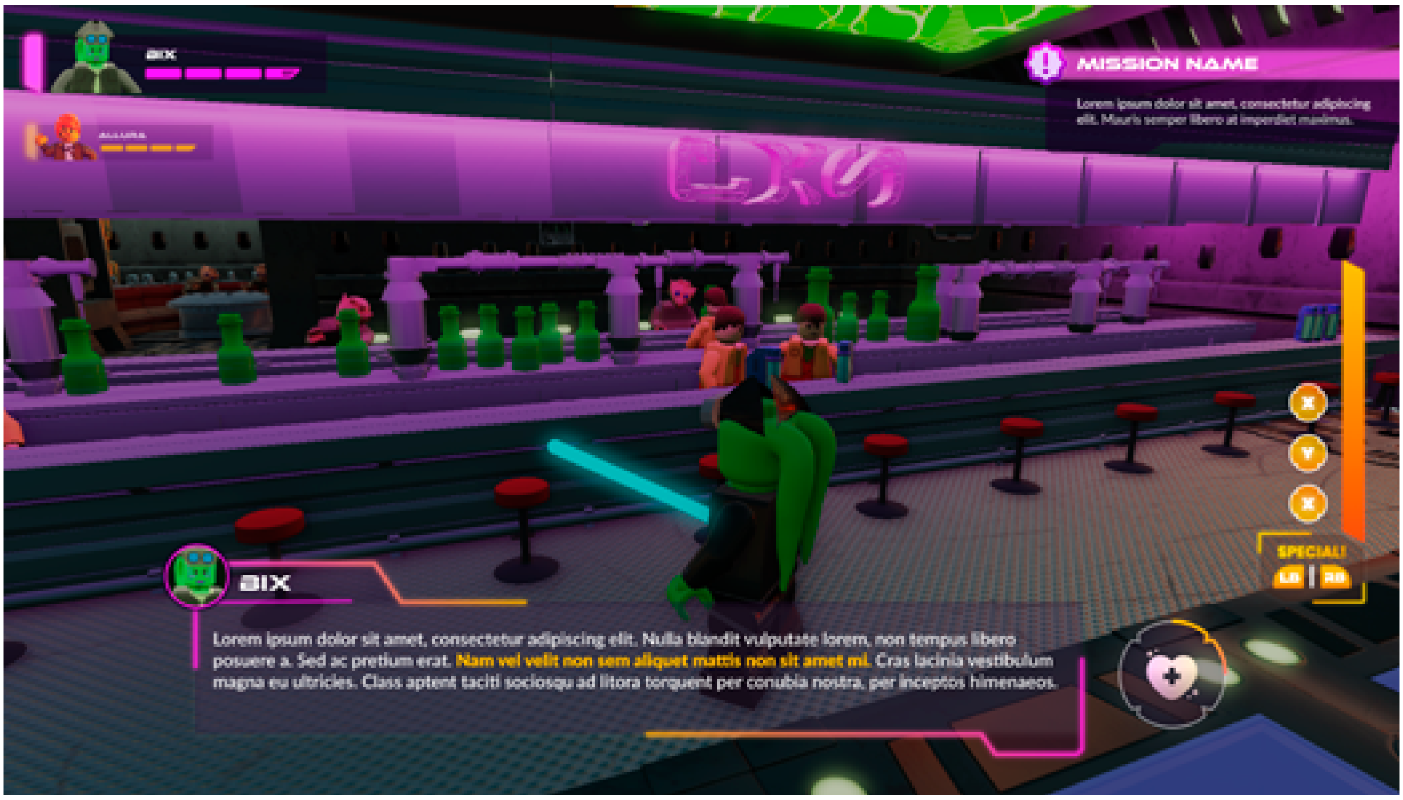
The art and design of the user interface should be based on the following guidelines:

Lego Games
Star Wars Universe
Classic Science Fiction

The design should seamlessly combine a clean aesthetic with excellent functionality, allowing users to easily navigate the interface and access information at each stage of the game. This can be achieved through the use of eye-catching animations and color schemes that reinforce the overall design concept.

The intention is to use vibrant colors like purple and electric blue to highlight key interface elements, while a mix of light and dark colors ensures good contrast for readability.
To draw attention to important information or functionalities, powerful gradients, and dynamic animations can be employed.

In summary, our primary goal is to create a unique and exciting visual experience, capturing the essence of the Lego games and the Star Wars universe in a harmonious and attractive way.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris semper libero at imperdiet maximus.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla blandit vulputate lorem, non tempus libero posuere a. Sed ac pretium erat. Nam vel velit non sem aliquet mattis non sit amet mi. Cras lacinia vestibulum magna eu ultricies. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.

ART STYLE

VISUAL STYLE OVERVIEW

In Star Wars Starfall Rebellion we aim to blend the nostalgic essence of the original Star Wars universe with a groundbreaking artistic approach.

This game fuses iconic Lego characters, reimagined with a vibrant cartoon-like flair, with environments crafted using realistic shapes and materials.

ARTISTIC DIRECTION

Our vision is to merge the classic Star Wars aesthetic with a contemporary twist, steering towards a cyberpunk style. This unique blend will transport players into a familiar yet refreshingly novel universe. By integrating the whimsical charm of Lego characters into this hybrid world, we ensure a playful yet immersive experience.

The environments, while maintaining realism in their form and substance, will contrast with the playful nature of the Lego characters. This juxtaposition not only honors the legacy of Star Wars but also introduces an edgy, futuristic vibe akin to cyberpunk aesthetics.

OBJECTIVE

Our goal is to create a visually captivating world that resonates with both long-time Star Wars fans and newcomers. By balancing the original themes with innovative cyberpunk elements, we aim to redefine the boundaries of the Lego Star Wars experience.

GAME LEVEL DESIGN OVERVIEW

LEVEL 1: CANTINA, SEWER, AND HANGAR

The first level diverges significantly from the classic Star Wars aesthetic.

The Cantina is lit by tubes filled with a radiant, radioactive liquid, simulating neon lights, casting an eerie glow over the entire scene. These lights not only illuminate the space but also add a sense of danger and intrigue, hinting at the hazardous materials that sustain this underground world.

Visual Reference: Cyberpunk 2077

Sewers is designed to be oppressive and perilous, creating a sense of danger and urgency for the player. To survive, players must adeptly utilize the elements of the environment, performing parkour across pipes and grates, and using the Force to maneuver objects, navigating the hazardous sewer landscape.

The Hangar serves as a pivotal transitional area, connecting the Cantina and Sewer levels. It marks a distinct shift in the visual narrative of the game, moving from the oppressive, neon-lit underworld of the Cantina and Sewers to a setting more reminiscent of the original Star Wars universe.

Visual Reference: Crash Bandicoot

LEVEL 2: SPACECRAFT

The second level transitions into a spacecraft setting. This level is designed to be simple yet foreboding, with an emphasis on creating a sense of menace and exhilaration.

The visual design combines sleek, futuristic elements with subtle hints of danger, making the spacecraft a thrilling and suspenseful environment to navigate.

LEVEL 3: CLASSIC STAR WARS INSPIRATION

The third level draws much closer to the traditional Star Wars theme, inspired by the Death Star and original spaceship designs.
This level pays homage to the classic Star Wars universe, with visuals that resonate deeply with the franchise's roots. Players will recognize iconic architectural elements and design motifs, reimagined within the Lego aesthetic, providing a nostalgic yet fresh experience.

LEVEL 4: BOSS LEVEL

Continuing the theme from the third level, the Boss Level combines the classic Star Wars style with added aesthetic flair. This level is designed to provide a closed, intense environment ideal for the final showdown with the Boss.

The art style here is refined and more detailed, heightening the dramatic tension and providing a visually stunning backdrop for the climactic battle.

MUSIC AND SOUND EFFECTS

The audio design of the game aimed to capture the spirit of the Star Wars saga while reflecting LEGO's personality as much as possible in terms of music and SFX.

Original music compositions, including menus and all the levels, were created specifically for the game. Throughout the entirety of the game, music played a vital role in shaping the overall game experience, serving as a fundamental element in creating the atmosphere of the LEGO Star Wars Saga. In some cases, unique tracks covered the entire level, while in others, changes occurred between combat and exploration modes.

All the musical aspects were created by Nicolas Franza & Antonio Zimmerman, students from ESMUC. Musical themes for the game were defined to convey the exact mood intended for each level.

Original sound effects were also created for the game, produced by the composers from ESMUC, either entirely original or with the assistance of online libraries like Soundsnap, provided by the university UPC.

In the game, SFX were incorporated in a logical manner, categorized into two main types: diegetic SFX and non-diegetic SFX.

Diegetic SFX encompassed all sounds originating from the game world, covering various elements such as character sounds, NPC footsteps, water noises, or machine noises.

On the other hand, non-diegetic SFX included sounds associated with interactions with the game's graphical user interface elements, like pressing a button in a menu.

The team worked with the music and SFX asset list in an Excel document to have all the information in one place.

Voice lines were not included in Lego Star Wars Starfall Rebellion.

TECHNICAL DESCRIPTION

The game was exclusively designed for computers, with no plans for a console release. Its control system was tailored for a controller interface.

To develop the game, we employed our proprietary engine, Axolotl Engine. Additionally, Unity was utilized for creating level blockouts, which were later integrated into our engine.

For efficient project management, we utilized ClickUp, a user-friendly platform that seamlessly integrated with GitHub.

Index

1. My contribution to the Project
1.1. Engine Camera Fixes & Improvements
1.2. Helping with Light Implementation
1.3. Normal Maps
1.4. Transparencies
1.5. Implement Wwise
1.6. Audio Meetings
1.7. Audio Modules & Components
1.8. SFX & Music Implementation
1.9. Attack Mechanics
1.10. Enemies Improvements
1.11. Second Mini Boss - X-27 Sentinel
1.12. Utils
1.13. VFX
1.14. Spaceship Level

1. My contribution to the Project

My role in this project was: Lead audio, gameplay, graphics & engine programmer.

1.1. Engine Camera Fixes & Improvements

One of the first tasks I did for the project was, after selecting one of the engines of the classmates to have as a base and start developing the engine of the group, was improving the engine camera, as I got a good grade on the individual part, I thought it would be a good idea to take that role.

- Mouse rotation fix: One important improvement was resolving a mouse rotation issue, ensuring a smoother and more accurate rotation for the camera.
- Limit angle while rotation: Additionally, I implemented a feature to limit the rotation angle vertically, providing better control and precision.
- Changed zoom to work with translation instead of FOV: To get a better user experience, I made a change by modifying the zoom functionality to work with translation instead of Field of View (FOV), resulting in a more intuitive and user-friendly interaction.
- Move camera with middle button mouse: Another important contribution was enabling the movement of the camera with the middle mouse button, adding flexibility and an easier navigation.
- Helped with the focus camera: I also participated in refining the focus camera feature, together with Oriol Sors and Alexis.
- helped with the unlimited screen cursor: On the other hand, I gave feedback to Alexis in the development of the unlimited screen cursor, which gave a better experience when trying to rotate, move, etc. the camera.
- Zoom with Alt + RClick: Another addition to the camera was the implementation of the Zoom with Alt + Right Click feature, providing an alternative method to control zoom levels.
- Mouse Cursors: I also worked on providing custom mouse cursors when using the different systems the camera had (moving, translating, rotating around an object, zooming).

- Change Camera velocity with mouse wheel while moving: In terms of camera controls, I implemented QWE hotkeys for translation, rotation, and scaling.
- QWE hotkeys (translation, rotation, scale): Additionally, I introduced the ability to change camera velocity with the mouse wheel while in motion, allowing you to dynamically adjust the moving speed for a more personalized experience.

Estimated Time: 10h

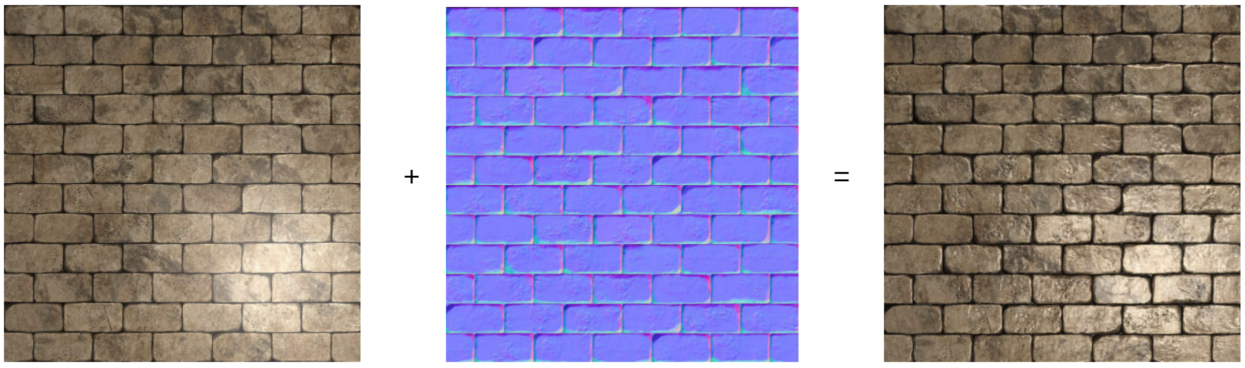
1.2. Helping with Light Implementation

I also contributed to the project by helping with small features or fixing bugs in the first stages of the implementation of lights in the engine. Some features I helped with were the implementation of applying changes on the lights only when the transform was modified, or the implementation of the engine UI for modifying lights parameters.

Estimated Time: 15h

1.3. Normal Maps

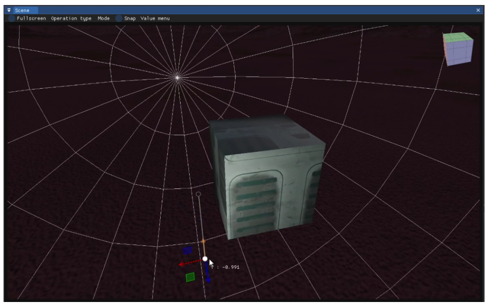
Normal maps are a type of texture very important for video games. These textures simulate details, bumps and shades by altering how the lighting calculations are performed on each pixel. The normal map is applied to a 3D model during rendering to create the illusion of fine surface details without the need for additional geometry.

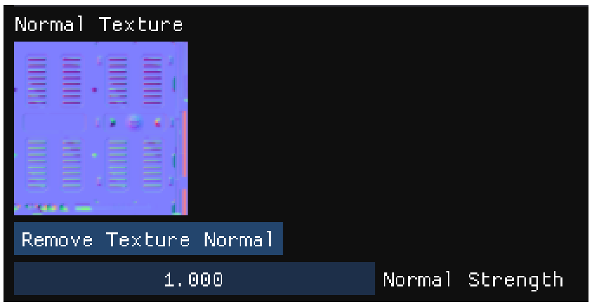


In a normal map, the RGB values of each pixel are used to encode the X, Y, and Z components of the surface normal vector. The normal vector is a unit vector that is perpendicular to the surface at that point. The red (R), green

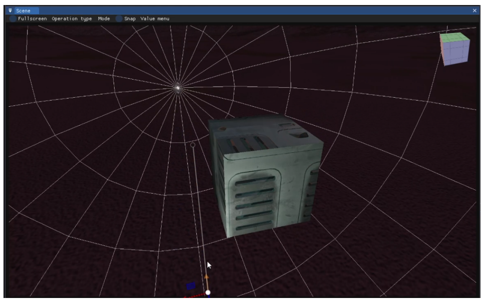
(G), and blue (B) channels in the normal map correspond to the X, Y, and Z components of the normal vector, respectively.

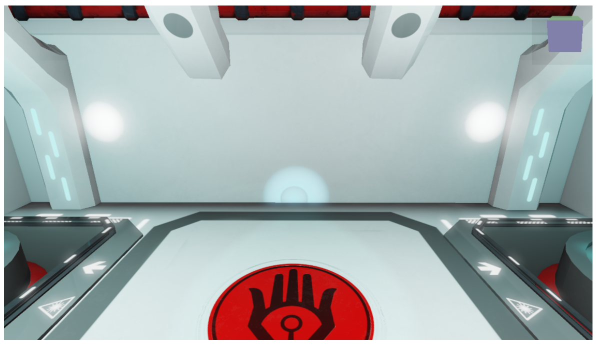
When applying a normal map to a 3D model, the normal vectors stored in the normal map need to be transformed from tangent space to world space, as tangent space is a local coordinate system that is defined at each point on the surface of a 3D model (It consists of three vectors: the normal vector (N), the tangent vector (T), and the bitangent vector (B)). This transformation ensures that the pixels encoded in the normal map affect correctly the lighting on the 3D model.

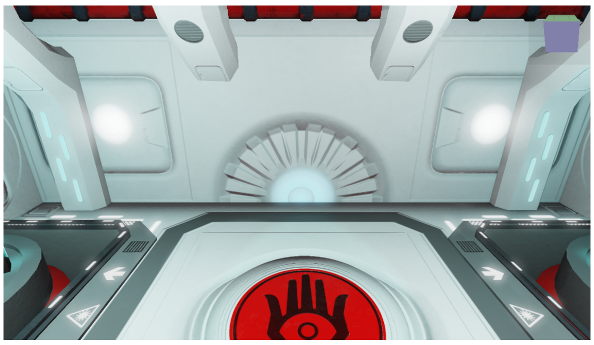




1.000 Normal Strength







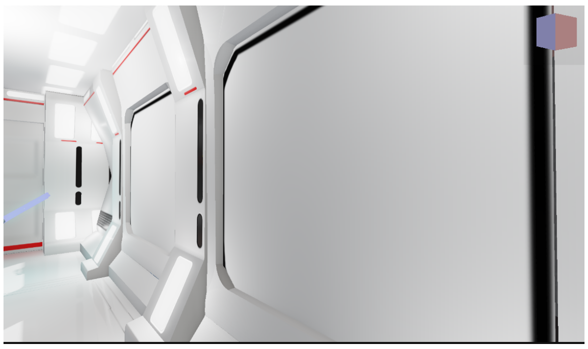
Estimated Time: 30h

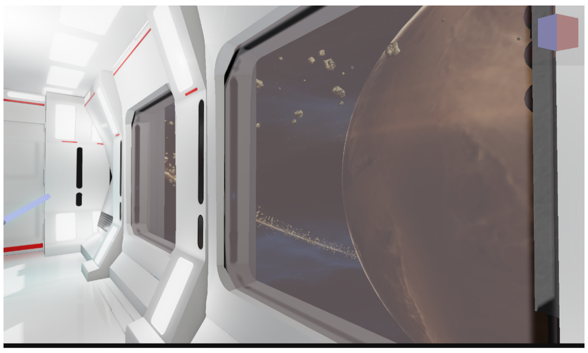
1.4. Transparencies

In the rendering pipeline, it is essential to prioritize the rendering order of opaque and transparent objects for optimal visual results. Opaque objects are rendered from front to back to prevent rendering a pixel more than one time. On the other hand, transparent meshes should be rendered from back to front to be able to see the transparent objects behind other transparent objects.
Transparencies in our engine:

- How to use: Transparent materials can be applied to objects by interacting with the MeshComponent attached to the respective object and changing the rendering mode of the material from opaque to transparent.
- Texture dependency: For objects with a diffuse texture, transparency is needed in the texture image to achieve the desired transparent material appearance in the 3D world. Alternatively, objects without a diffuse texture can control transparency using the alpha channel of the material's diffuse color.

As dealing with transparencies is complex, in our engine transparent objects need to be taken carefully. When dealing with large or multiple objects with transparencies, as the rendering order of transparent objects is calculated based on the distance from the camera to the pivot of the object, if the pivot is at a different distance than where the object should be rendered, visual bugs can happen due to objects being rendered in the wrong order.





Estimated Time: 20h

1.5. Implement Wwise

I successfully integrated Wwise into our C++ game engine by obtaining the Wwise SDK from their website and adjusting our project settings to accommodate it. First I adjusted our engine project settings to incorporate Wwise specific configurations, including the necessary headers and binaries. After that, I initialized the Wwise sound engine using the appropriate API calls from the Module Audio and setting up the necessary parameters, such as soundbanks or audio devices. Gerard helped me in this part with the implementation of libraries and headers.

Estimated Time: 25h

1.6. Audio Meetings

Audio meetings with the music composers and audio designers has been a very important role for me in this project. As a Lead Audio in Starfall Rebellion I've been meeting since the beginning of the project with Nicolas Franza & Antonio Zimmerman to explain to them what our objective was in the audio part of our videogame.
It all started with the selection process. At the beginning of the project some classmates and I met with the ESMUC (Escola Superior de Música de Catalunya) students to explain to them what our project was about and tried to convince them to join us as the audio team. After some days, some of them sent us their portfolios and experience in order to proceed with the selection process. After looking carefully at everything they sent us, we decided we wanted to continue with Nicolas Franza & Antonio Zimmerman.

During the beginning of the project I was on charge of defining and describing all the music tracks we would need for the game. The music was composed of 9 tracks:

- Cantina: This track aimed to make the player recognize that they were in the universe of Star Wars. We wanted to clearly make the player feel like this was very similiar to the famous Star Wars Cantina song. This was the only song of the project that needed to be diegetic, as it would be part of a stage with musicians playing.
- Sewers (Exploration & Combat): This track needed to create a mysterious and dark atmosphere, reflecting the dangerous and unknown place of the sewers for the player. Since it was the first interaction with enemies, we also required a combat version. The two versions shared the same structure to ensure a smooth transition between them.
- Hangar (Exploration & Combat): Originally, the Hangar was a place with many enemies where the player would have to use stealth to infiltrate and take control of Bix's stolen spaceship. Due to time constraints, the concept changed, and we decided to trigger this song at the beginning of the combat against the first miniboss, as we wouldn't have a combat segment in the hangar.
- Spaceship Escape: This track needed to have a lot of power as this would be a section of action where the player had to feel the strength of driving a spaceship while avoiding obstacles.
- Cargo Bay: At the beginning of the project we wanted an introductory level before the space station where the player could be introduced to Allura and have some time to play with its abilities. As we didn't have enough time, we had to cut this level out. What we did, as the song was already recorded, we used it for the main menu, which, surprisingly, fitted very well.
- Space Station: This level required music with a lot of tension, as the player was discovered after infiltrating the enemy space station. We wanted this track to have a lot of action, but not too much, as we needed to build tension for the final level, the final boss.
- Final Boss: For this track, our goal was to immerse the player in the most epic battle of the game. The music for this level needed to be among the best tracks, if not the best, to elevate the experience of the final of the game.

Once all the music was defined, the music composers started working on it, with me giving direct feedback to everything they made. Once every piece was composed, Nicolas went to Bratislava to record all the music with an orchestra, which, once it was mixed and mastered, made it feel like it was an OST for a real Star Wars game.

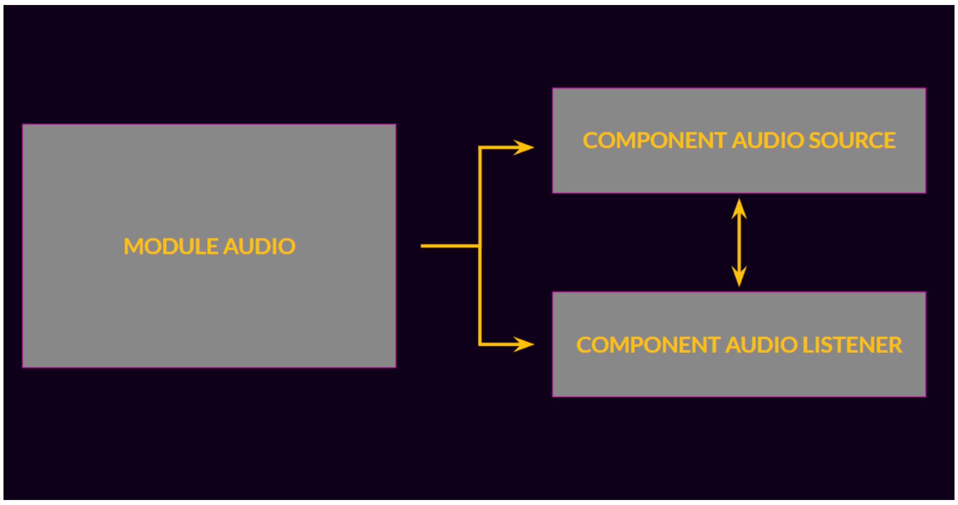
The same process was followed with the Sound Effects of Starfall Rebellion. After defining every SFX asset, Nicolas & Antonio would start working on them, which would result in around a hundred final assets, which later would transform to around a three hundred and twenty with all the variations of each sound effect.

Estimated Time: 100h

1.7. Audio Modules & Components

In a C++ game engine, the audio module plays a crucial role in creating and controlling everything that entails everything related with audio, and the Components Audio Source and Audio Listener help the module and the game fill audio systems.

- Audio Module: The audio module serves as the overarching system that manages and orchestrates all aspects of audio within the game. Its responsibilities include:
    - Initialization: Setting up the audio system, loading necessary resources, and configuring audio settings.
    - Resource Management: Loading and unloading audio assets, such as sound files and sound banks, efficiently during runtime.
    - Audio Control: Controlling the playback of audio sources, handling volume, pitch, and other parameters.
- Audio Source:An audio source represents a point in the game world from which sound emanates. Key functionalities of the audio source include:
    - Sound Play: Playing and stopping sound effects.
    - Attributes Control: Modifying attributes like volume, pitch, spatial positioning or looping. In our case, everything related with controlling these attributes was directly controlled from Wwise, so we didn't have to make difficult coding calculations.
    - 3D Positioning: Supporting the spatial positioning of sound sources. To do this, I created a function that updates the Transform of the object only when it is modified.
    - Event Triggering: Allowing the triggering of sound events in response to in-game actions.
- Audio Listener:
    - The audio listener represents the position and orientation from which the player perceives audio. As it is usually made, we had the Audio Listener Component on the Main Camera.



Estimated Time: 20h

1.8. SFX & Music Implementation

As Lead Audio of Horizons Games I have also been in charge of implementing all the music and sound effects into the game, with all that entails from Wwise to the game engine. Some of the most important contents and processes done in Wwise for this project have been the following ones:

Audio:

- Importing and Managing Audio Assets: In the "Audio" section, I imported and managed audio assets such as sound effects and music.
- Properties and Behaviors: I configured properties and behaviors for each audio asset. This includes defining loop points, adjusting volume levels, applying effects, etc.
- Spatial Audio Settings: Wwise allows users to define spatial audio properties, enabling the simulation of 3D sound within the game world. This allowed me to set positions, orientations, and attenuation parameters such as volume or filters for a more immersive audio.

States:

- Audio States: In the "States" section, I was able to define various audio states that represented different conditions or situations within the game. In our case we needed to know if the player was in Exploration mode, Combat mode, or Dead. These three states allowed me to control music in different modes depending on the level and situation.
- Transitions and Blending: Thanks to the tools Wwise has I was able to create smooth transitions between states, incorporating blending techniques to ensure a natural progression of audio elements as the game states changed.

Switches:

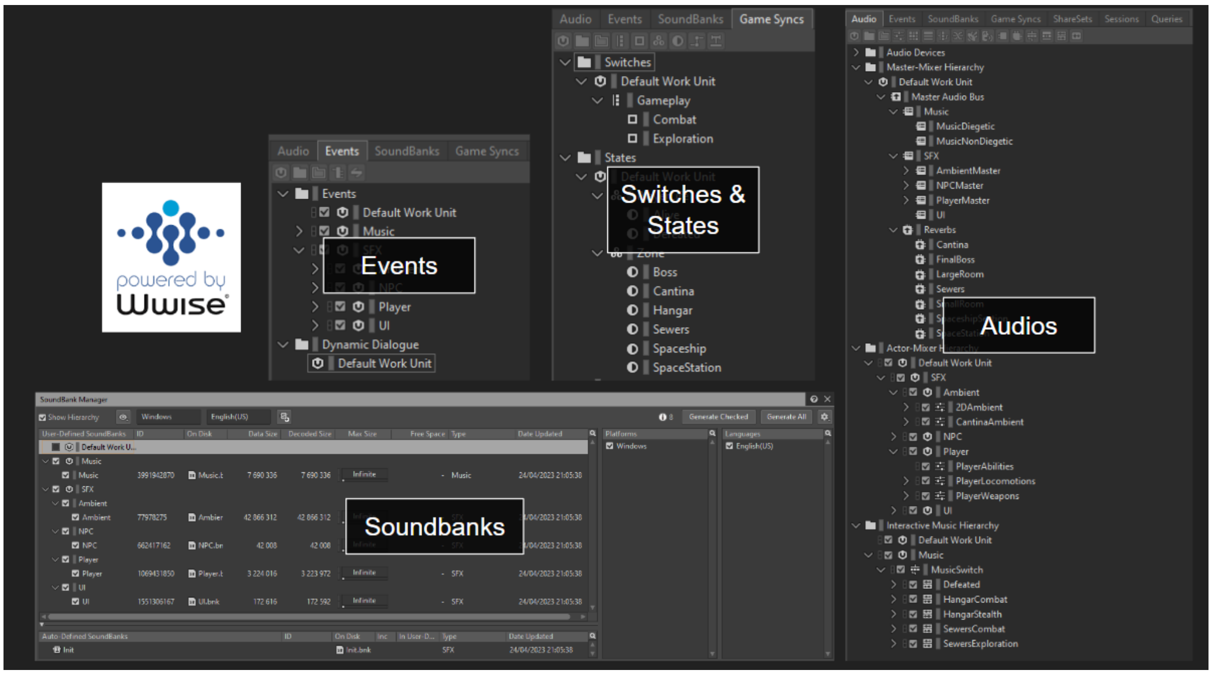
- Audio Switches: Switches are similar to states in a way that they can be used to create transitions depending on the situation in which the player is. However, the main difference lies in the fact that states are more generic and represent long-term parameters, while switches are more dynamic and specific. In our case, in the music section, we used states to define the player's state (exploration, combat, or dead); while, we utilized switches for the specific zone in which the player was situated. This allowed us, for example, in a level with two variations of the music depending on whether the player was exploring or in combat, to specify the zone with the switch and select one of the two tracks with the state.

Events:

- Audio Events: I defined audio events to trigger specific sounds based on in-game actions or events. Events serve as points in the game where audio changes occur, such as playing a laser shot sound when an enemy fires a weapon, playing a loop audio like an ambient sound or event stopping a sound.
- Link Events to the Game Code: Events are played in game by using a PostEvent() function passing the ID of the event as a parameter.

Soundbanks:

- Compile Audio Assets: The Soundbanks section is where I could compile and organize audio assets into soundbanks. Soundbanks are packages of preprocessed audio data that are optimized for runtime playback. Exporting the events in Soundbanks was essential if I wanted to use the events on my code.

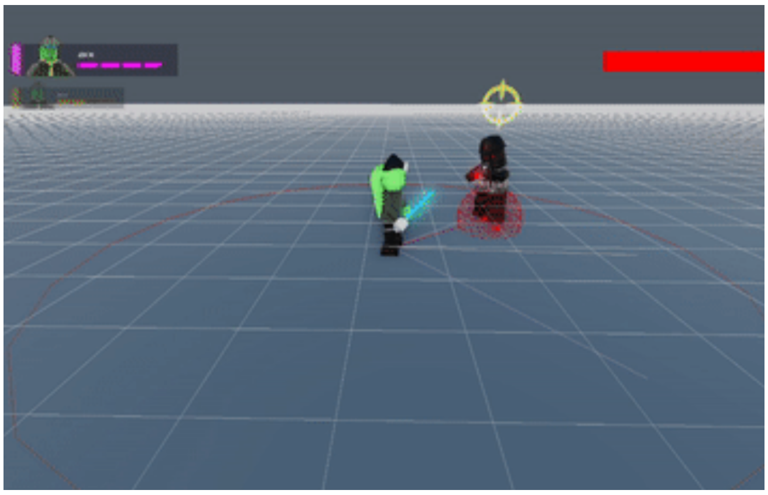


Estimated Time: 120h

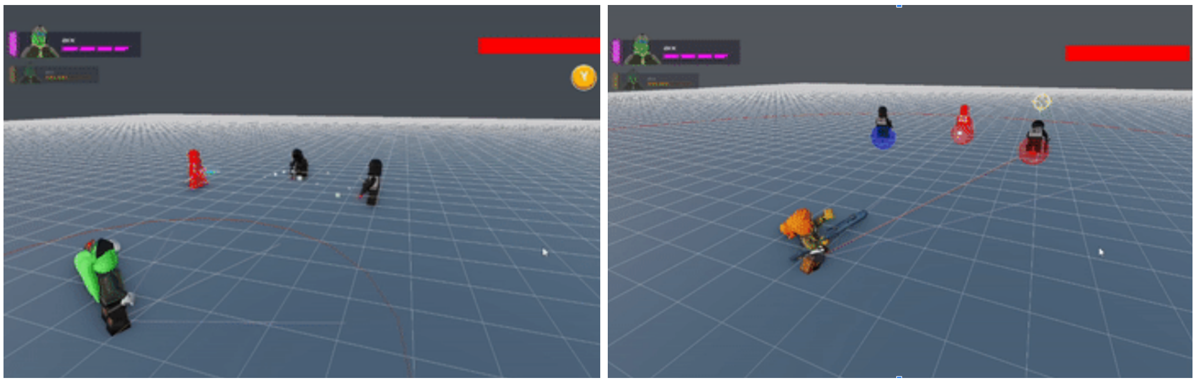
1.9. Attack Mechanics

I created and implemented different attack mechanics during the development of the gameplay of Starfall Rebellion:

- First implementation of the melee attack: I created the initial iteration of the main character Bix's melee attack. This implementation utilized raycasts to detect objects in front of the player. Later on, we modified this system to enhance the player's ability to hit enemies, after realizing that precise aiming was not always easy and with the player's intentions.
- First implementation of the ranged attack: I also developed the initial version of the ranged attack, allowing drones (enemies) to shoot bullets at specific intervals of time. This was quite challenging due to the limited tools in our early project phase, making it difficult to instantiate objects in real-time in our engine.
- Enemy Detection: In response to the need for an improved combat system, I created a system where, given an angle and distance, the player automatically detects enemies in front of him/her. This system prioritizes enemies at a closer angle in front of the player, ensuring that if two enemies are at different distances, but one is more in front of the player than the other one, this last one will be the selected one.



- Heavy Finisher (Bounce Attack): I implemented the attack mechanic that throws an object toward an enemy, causing it to bounce between nearby enemies. This attack is applicable to both Bix and Allura, with Bix throwing a lightsaber and Allura using an electric shot. The only difference is that, once the attack finishes, Bix's lightsaber returns to the player, while Allura's shot dissipates.



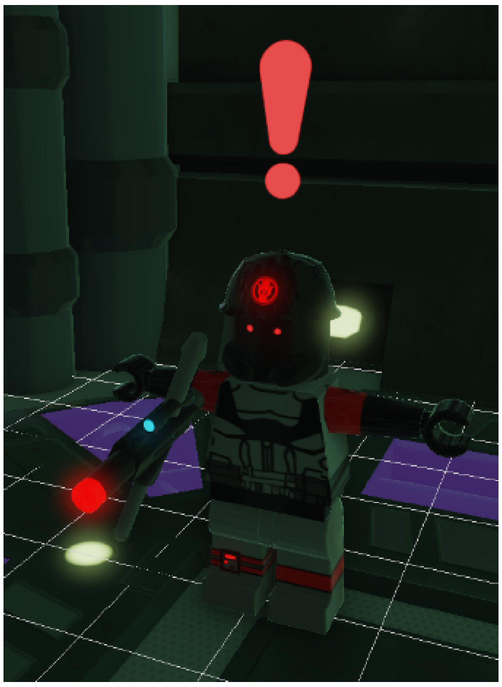
- Consecutive combo attack animations: I implemented a system where the main character automatically executes three consecutive animations during normal attacks. To deal with the short duration of attack animations and ensure fluidity, I incorporated a slight delay between each consecutive animation, allowing sufficient time for the player to input the next attack before the previous animation was already finished.

Estimated Time: 65h

1.10. Enemies Improvements

I worked on the AI of the enemies, improving some features and creating new ones.
First of all, I refactored all the code pertaining to Venomite and Drone entities, optimizing their functionality and making it easier to use for future usages. I also made the patrol behavior have a much better feeling by allowing the enemies to have an unlimited number of patrol points and introducing a stopped random time feature at these points. To have better enemy movement management, I centralized all related code to enemy movement into a single script named AIMovement, promoting efficiency and ease of maintenance.

A notable addition I made to the enemy behavior was the introduction of an alert state, visually represented by an exclamation mark above their heads when they detect the player. This alert state enhances player feedback and adds a layer of visual communication to the game.



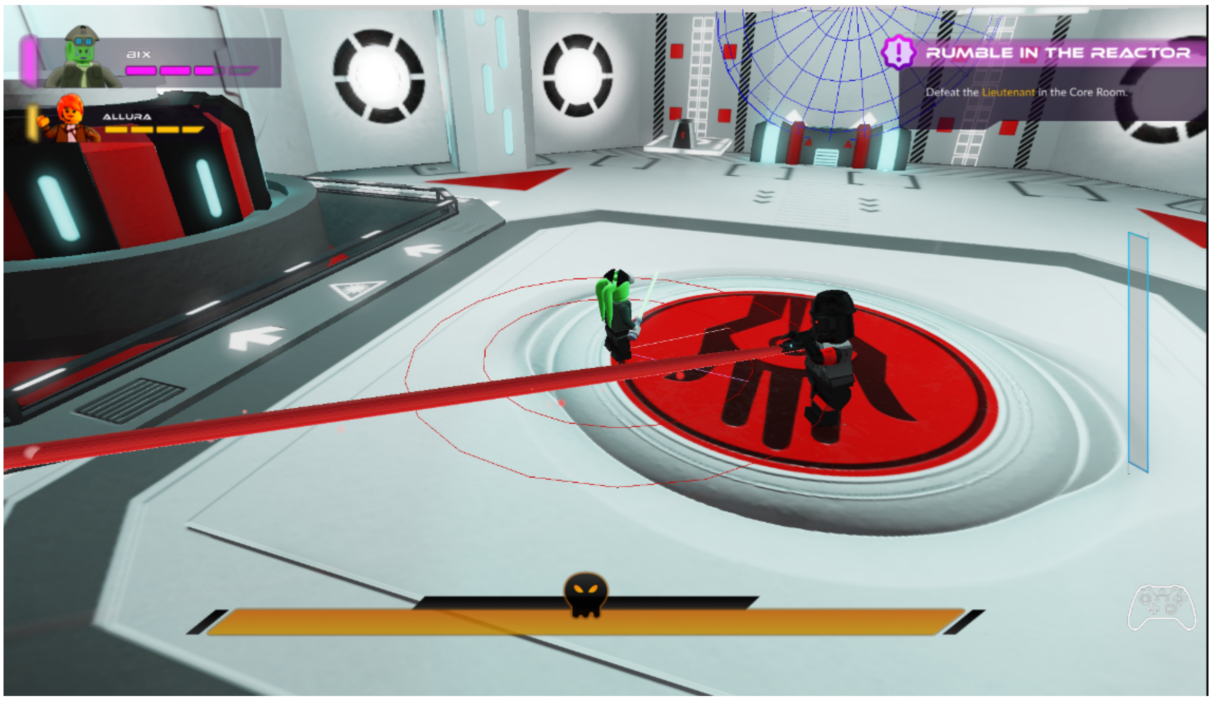
I also refined the combat mechanics; Venomite's melee attack is now triggered only when the player is in close proximity to the enemy, optimizing the use of this attack. Additionally, I made adjustments to the ranged attack to make the player understand it better. A particle system now plays before the shot, providing a visual feedback for better anticipation to the player, and the ranged attack has been improved with a system that introduces randomness in the shooting intervals, avoiding repetitiveness and adding an element of unpredictability to the enemy behavior. All these improvements contributed to a more polished and engaging gameplay experience.

Estimated Time: 30h

1.11. Second Mini Boss - X-27 Sentinel

I developed and implemented all the functionality of the second miniboss, from its behaviors to its attacks. In the part of the behaviors, is very similar to the Venomite attack, the only difference is that it has to take into account two additional attacks when interacting with the player:

- Energy Shields: This behavior wasn't created by me, as it was firstly created for the final boss, what I did was just reusing the existing script. In this case, the miniboss summons a shield to protect himself from the attacks of the player while at the same time, Venomites will come to attack the player.
- Energy Beam: As explained before in the Game Design Document, X-27 is capable of firing bursts of energy beams from his gun. These beams can track and strike enemies with precision, causing moderate damage.



Estimated Time: 25h

1.12. Utils

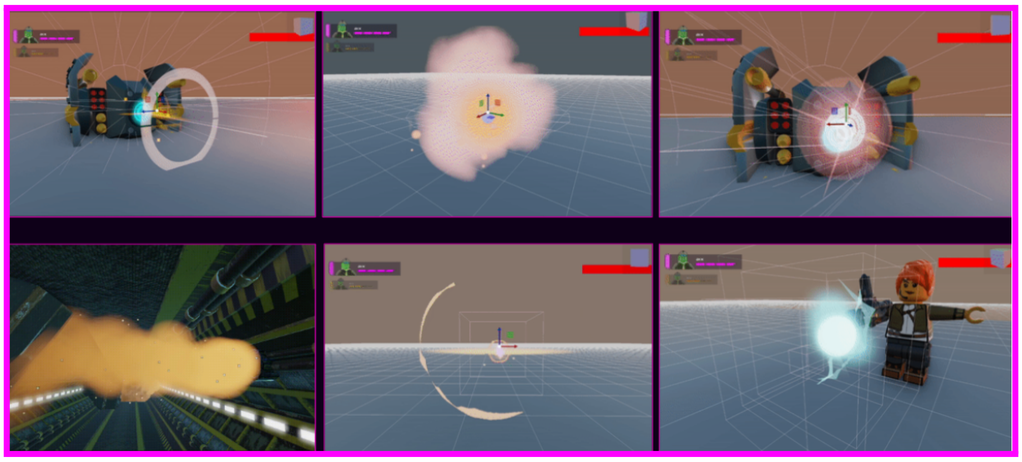
I created a space in our project for utility scripts, the idea was to have general scripts that can be used for multiple things. I started this by creating two scripts:

- Rotator Script: The Rotator script serves as a handy auxiliary tool designed to facilitate the continuous rotation of a GameObject. This script is particularly useful when there is a need for ongoing rotations to achieve a visual effect or dynamic element in the game world. By incorporating the Rotator script into a GameObject, developers gain a straightforward solution for automating rotational behavior without the need for manual adjustments or without the need of previously handcrafted animations in Blender.
- SendTriggerCollision: The SendTriggerCollision script is a very useful component, providing a convenient method to transmit OnTriggerEnter or OnTriggerExit events from one object's script to another. This script was firstly used for some obstacles of the spaceship level, where we had a main object that controlled multiple fire obstacles, and where we needed to get any collision that was detected on any of the fires controlled by this main script.

Estimated Time: 2h

1.13. VFX

I think it is also important to mention that I've been learning and using the tool to create Particle Systems in our engine, creating some final assets for the final delivery.



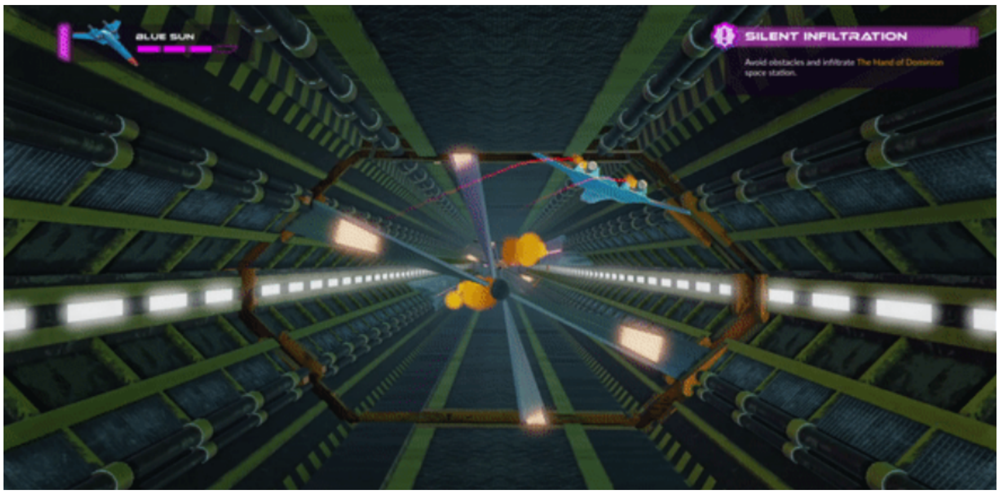
Estimated Time: 15h

1.14. Spaceship Level

The concept for the spaceship level emerged during the initial phase of the course when we divided in groups to brainstorm ideas for each level of the game. Contemplating the first segment of the second level, I proposed the idea of taking control of a spaceship to infiltrate the enemy space station. This choice was a good idea as the level primarily only involved moving the spaceship forward while avoiding obstacles. In Unity, I initiated the first implementation, iterating multiple times until achieving the final level design. Several months later, I took on the responsibility of porting the same level to our own engine. Throughout this process, I collaborated with the art department to discuss adjustments of certain elements.

The hardest task of this level was making sure that the spaceship object moved forward without generating any bugs with the collision system we have. Once this functionality was achieved, my remaining main task was implementing the logic of the obstacles previously designed in Unity:

- Basic Obstacle: A stationary element such as a wall or fire that inflicts damage if the player fails to navigate around it.
- Moving Fire: An animated fire element that traverses from one position to another, requiring the player to anticipate its movement.
- Multiple Fires Alternating: This obstacle has multiple fires activating and deactivating, consistently leaving open spaces through which the player must go.
- Turbine: Turbines in this level have four blades in constant rotation. Players must dodge these turbines to progress through the game.



Avoid obstacles and infiltrate The Hand of Dominion space station.

Estimated Time: 30h