Final Project

Draft Report

Student Id: 210168615

Introduction

I have chosen the Game Development - Arcade Game template for this project. I will develop a video game using Unity, in a similar style to the arcade games of the 80s and 90s.
The game will be a vertical scrolling “shoot-em-up” game, like *Space Invaders* and *1942*, but also incorporating elements from the “bullet hell” subgenre that appeared in the 90s, with seminal games like *Batsugun* and *DonPachi*.
The elevator pitch for a game like this would be: *Get ready to test your skills in this adrenaline-fueled bullet hell space shooter inspired by iconic games like 1942 and DonPachi*.
There are a number of characteristics that define vertical scrolling “shoot’em up” games. The first of all is the orientation of the game, the player’s character points toward the upper section of the screen and most of the enemies appear from there as well, moving downwards towards the player. A moving background creates the illusion of moving forward and adds dynamism to the scene. The main gameplay mechanics are moving and firing, destroying the enemies that attack the player while also trying to avoid being hit by projectiles, enemies or other bodies. There are other possible expansions that make the genre more diverse: powerups, upgrades, special movements, special attacks, sidekicks or weapons.
But what is the bullet-hell genre? “Bullet hell” or “bullet curtain” games are a subgenre of shoot’em up games, and are characterized for having a multitude of bullets being fired at the player, in complex and visually impactful patterns, such as helixes, spirals, circles, hexagons and clouds. To balance the difficulty, the player’s hitbox is smaller than in other shoot’em up games, often being a small part of the aircraft sprite. This genre is very popular in Asia, and especially in Japan, but does not have the same recognition in other countries.
The main reason I have chosen this template is due to my own personal interest in game development and in this game genre. I grew up in the 90s and have many fond memories of playing video games as a young kid. At that time a career in game development seemed impossible, but now the technology and know-how are widely available.
Another aspect I consider relevant is that these games have mechanics that can accommodate the requirements of the project and still offer many possibilities. These limitations may be useful to keep the scope of the project within the time constraints.
The input is reduced, so only a small set of actions is available, which fits a space shooter comfortably. In these games, besides moving, the player can shoot, use some items (like bombs) or special actions.

The game must be easy to pick up. To overcome this limitation I am leveraging the familiarity of a known genre. Controls will be easy to understand, as will be based on the two main standards used for playing with a keyboard: the directional arrows and the WASD keys.
Very short initial gameplay sessions due to high difficulty: Bullet hell games are inherently difficult, due to the number of moving elements on the screen, and complex patterns.
And it also proposes some interesting technical challenges: the bullets and enemies move following complex patterns, which can be solved programmatically, sometimes involving mathematical functions. There is also the difficulty of keeping a good performance when there are many elements on the screen, and making the game's components modular and extendable. Regarding game design, the biggest challenge is adding a novel mechanic to make the game interesting and engaging.
Of course, there is no sense in developing a game if there is no market for it, and while this game won't be released commercially, the public has looked for independent and small studios or even solo developers as an alternative to the expensive AAA games released by the largest companies of the industry[1]. The popularity of platforms like Itch.io and Steam for PC supports this idea, as they enable small studios to have a distribution chain without the need for physical copies. The invention and evolution of portable devices like cell phones and portable consoles was instrumental to the popularity of "hyper casual" games: games that are focused on one or two game mechanics, with very simple input, and aimed at massive non-expert audiences. The proliferation of this type of games contributed to spark the interest in vintage games, which are simpler and more focused, very different from the present day standard of open worlds, multiplayer oriented and convoluted games. There is also an increasing interest in games with higher levels of difficulty, like the *Dark Souls* series, evidenced by their success and the release of sequels and new titles.
As extensions to the project I will incorporate a 2 players option, a time dilation game mechanic (that will also help beginners to get used to the game), and a variety of enemies as wide as the time limitation allows.

[1] Anonymous, Accessed 2024-01-08. Wikipedia: Indie Game
https://en.wikipedia.org/wiki/Indie_game#Fears_regarding_saturation_and_discoverability_(2015%E2%88%92present)

Literature Review

In this section, I'll address the following topics:
What was the Golden Age of Arcade games? Which games from this period were most relevant and influential? And what was innovative or disruptive about them? I have taken the The Golden Age of Arcade Games.first section of the book "*The Golden Age of Video Games - The Birth of a Multibillion Dollar Industry*"[2] as our reference.
What methods or frameworks can we use to analyze these games? I took the article "*MDA: A Formal Approach to Game Design and Game Research*" as reference for this topic, and analyzed the game "DonPachi" using this framework.
Finally, one important question for this project is: What is the bullet hell genre? I have read Andrew Fan's guide[3] to have a better understanding of the defining aspects of the genre.

The Golden Age: When? Where? What?

It's not easy to determine when the Golden Age started, authors don't agree about the initial milestone, so the year can vary in the 70s, and some of the events that are considered as the most relevant are: the release of Computer Space, the first arcade video game machine (1971), the release of *Space Invaders* (1978) and the release of *Asteroids* (1979) one of the first games using vector graphics. There is more consensus about the end of the Golden Age, and it is usually associated with the crisis of the video game industry that took place between 1983 and 1985.
It took place all around the world, but the main centers of research and development were the United States and Japan. Other countries, depending on their economic and academic power, were either secondary centers (as many countries in Europe) or simply consumers of these new technologies, as Latin American countries.
Many influential games were developed and released during this period, some of them are still very popular and are still references for many game designers and companies (especially in the mobile gaming industry).
Some of the most important games developed in the USA were: *Asteroids* (Atari, 1979), *Centipede* (Atari, 1981), *Defender* (Williams Electronics, 1981). And in Japan: *Space Invaders* (Taito, 1978), *Donkey Kong* (Nintendo, 1981) and *Pac Man* (Namco, 1980).
Among the most innovative aspects we can mention: *Space Invaders* opened up the themes and game mechanics, most previous games were based on sports like *Pong* and *Gran Track 10*, it presented a feedback loop (the multiple lives per play), an evolving environment, and a high score, which greatly enhanced competitivity. *Donkey Kong* introduced characters that were clearly individualized and a very simple and easily understandable narrative, and was one of the first platforming games. *Pac Man* introduced a new type of interaction, where the player needed to avoid enemies in a maze and could obtain the power to temporarily

[2] Dillon, Roberto, 2011. The Golden Age of Video games - The Birth of a Multibillion Dollar Industry. A K Peters/CRC Press, USA.
[3] Andrew Fan, accessed 2023-12-01, Andre Fan's Code Dump. https://sparen.github.io/

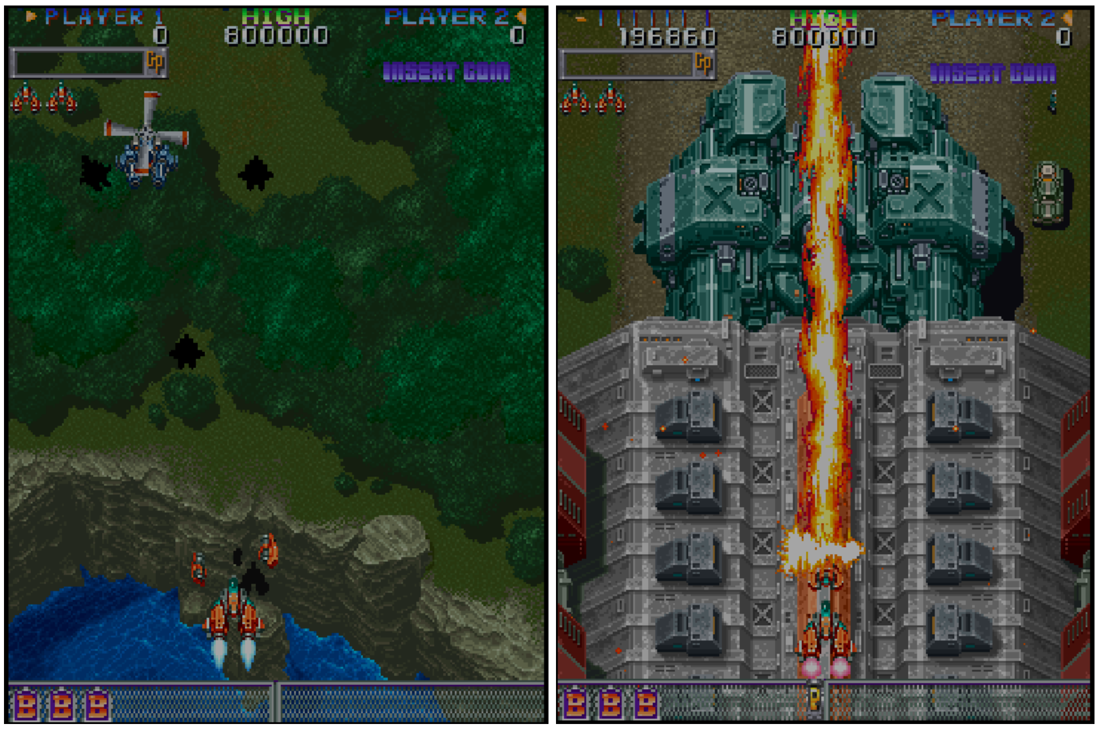
eliminate them. Pac-man was also a game where the enemies were clearly individualized by their color and name, and they had even different behaviors.
The hardware that powered the arcade machines also evolved quickly in those years: vector and color graphics allowed users to visualize more complex designs, generating the illusion of 3D movement and 2D graphics with a higher level of detail, characters could have a clearly defined image and more personality (like the ghosts of Pac-Man). This also influenced what kind of games could be developed. The evolution of home systems, both computers and video game consoles, gave birth to third party software development, with Activision being one of the first companies dedicated exclusively to game development.
But we could argue that besides the technological aspects that fueled these video games, the creative minds behind them installed video games in popular culture and set the foundations of what would be one of the biggest industries in entertainment.

How can we analyze these games?

Video game studies are a controversial area because there is not a unified definition of what a game[4] is (as seen in A book of Lenses, chapter 4 and Game Design Workshop chapter 2, among other textbooks), its scope and method are not clearly defined yet, and there is not even a consensus on where this study should be located, if it's part of Humanities, as multimedial works of art, in social sciences as a cultural phenomenon, or as a multi-disciplinary study that is not scientific in itself. For a long time, the only widely available articles talking about video games were the critical reviews that some experienced players wrote for magazines, which were mostly subjective, even if they tried to adhere to internal standards, like pointing systems and common areas to review (graphics, sound and music, replayability. difficulty, game modes, etc).
One framework designed to study video games is called MDA[5] (MDA stands for Mechanics, Dynamics and Aesthetics), and was proposed by Hunicke, LeBlanc and Zubek, in 2004. It's not the only framework, and it's not accepted unanimously by the community, but it can help us understand some aspects of the games.
The framework breaks down the games into smaller components: Rules - System - "Fun", which are the ways the player can perceive the game, and their counterparts from a design perspective: Mechanics - Dynamics - Aesthetics. The mechanics describe the components of the game, in a very low level of detail, like data structures and algorithms. Mechanics can be described as the actions that the game allows for the player and AI. The dynamics describe the behavior of the mechanics on a real play session, how they interact with other systems and with the player's input, ie. how the actions can be used in real life scenarios (rocket jumping, for example, is a possible move, but it wasn't designed intentionally, it's just a natural consequence of the interaction between the game's mechanics and the players' creativity). Finally, aesthetics are the expected emotional response from the player, evoked when she or he plays the game, and the product of the dynamics.
The framework includes eight categories that describe how a game is perceived by the player, from the immediate "sense-pleasure" and "challenge" to the more abstract "fellowship" and "discovery". The game can evoke multiple different sensations in the player:

[4] Some examples of definitions can be found in Wikipedia's page for Game. Anonymous, accessed 2024-01-08. Wikipedia, Game. https://en.wikipedia.org/wiki/Game#Definitions.
[5] Hunicke, Robin & Leblanc, Marc & Zubek, Robert. (2004). MDA: A Formal Approach to Game Design and Game Research. AAAI Workshop - Technical Report.

graphics, sound and music appeal to the dimension of the game as they directly stimulate the player's senses, but also if the game is cooperative (either with other players, locally or online, or AI), it can also create a sense of community or fellowship, thus being a social framework.
How can we use the framework to analyze shooter games?
As an example, I'll try to apply the framework to analyze the game "DonPachi". The first step is finding out what are the most relevant categories in the aesthetics dimension that we have to consider.



Images: In game screenshots from DonPachi.

Sensation. The game uses an abundance of color and rich textures to be visually appealing, the aircrafts, tanks and scenarios were carefully designed to keep a consistent futuristic style, there are multiple visual effects, different types of shots, bombs, explosions required by the genre. There are plenty of sounds, like gunshots, yells and explosions, that highlight the actions of the scene, and the fast paced music conveys a sense of urgency. Both the images and sound help to create an immersive experience.
Fantasy. The game presents a fictitious situation, where the player represents the only pilot able to face the challenges that the enemies present. While the plot is not very relevant to the game's continuity, the sense of danger is present throughout the whole game and the threat is constant, which is consistent with the back story.
Narrative. Most of the game is linear, with the missions ending and starting without any cutscene or dialog. This game is clearly not story driven, and the player doesn't have any say in the game's outcome.
Challenge. This is the focus of the game, the player's reflexes and skills are tested constantly and the difficulty of the game increases every mission.
Fellowship. The game allows 2 players to participate simultaneously, while it can be very helpful to have a comrade joining the fight, the sense of companionship admits only one additional person.

Discovery. The game's world is limited and doesn't admit any independent exploration.
Expression. The game doesn't present any opportunities for the player to affect the character's journey, or to customize the character in order to make it more adequate to his or her playing style.
Submission. The game is engaging, as it keeps the player's focus throughout the whole session. The challenge, the sound effects and the constant movement, make a very immersive experience.
How are the most relevant aesthetics of DonPachi modeled?
Enemies' behavior has two components: movement and attack (mechanic). The combinations of different movement and attack patterns generate different "personalities" for the enemies (dynamic). And depending on how they behave and how many of them there are in a group, the challenge the player faces will increase or decrease (aesthetic). If the enemies fire constantly and move too fast, the player won't be able to evade them (dynamic), creating a feeling of powerlessness and frustration. On the contrary, if the enemies are slow and don't fire (dynamic), the player may get bored quickly (aesthetics). In the case of DonPachi, there's a balance: some enemies move slowly but shoot faster, bigger rounds or homing projectiles, while other enemies move faster, but have limited shooting capabilities. In general, the combination of different kinds of enemies makes the game fast and keeps the player's attention, and prevents it from becoming monotonous. Projectiles' physics (and the aircraft's too) is not intended to be realistic, bullets normally would move much faster than in the game, even considering the difference in scale, and the controls of even the most modern aircrafts are most likely not as responsive as shown in the game (dynamics). This is done to make the game accessible and to create diversity (aesthetics).
Regarding sensations, the game features a colorful palette, stereo sound and background music (mechanics). The player can quickly see the bullets, because their color contrasts with the background (dynamic) and thus is able to avoid them, sounds like explosions and gunfire are signals that inform the players of the game's status, this makes the player's task a bit easier and enhances the immersiveness of the game, as the game involves different senses at the same time, and can also be used to call the potential players' attention when they walk through the arcade (aesthetics).

What are bullet hell games?

There are not many resources to design and develop videogames of specific genres. There are some examples of the most popular game genres, from official sources as Unity and Unreal and from indie developers, and tutorials that cover the basics of each video game engine in the market, but even tutorials that orient people to "develop" their own FPS or 3D platformer do not dive deep into how to create game mechanics that will make a game unique or how to design the levels to make it engaging, or how to balance the difficulty of a puzzle or strategy game, meaning that the most relevant topics of game design are not incorporated in these resources. Even though there are many bullet hell games, and it's a genre that started over twenty years ago, there is no authoritative body of knowledge or design guide or even a generally accepted definition of the genre, what components are required by it, how to use them. Other genres, even if they lack proper documentation, have at least examples so known that people immediately understand the reference (there are so many FPS and platformers that people understand how most mechanics work, and many

movements and techniques have widely known names like double jump, wall jump or rocket jump).
In this website, Andrew Fan tried to create a simple and flexible vocabulary to describe different patterns in *danmaku* and enable designers to discuss and analyze games of this particular genre[6], and also help beginners understand how to design a game that is properly bullet hell.
He proposes: “The first thing to note is that the differentiating factor [between regular shoot’em ups and danmaku] is NOT the number of bullets. You can have a lot of bullets, but if they aren't thoughtfully used, the pattern isn't exactly a bullet hell.”[7] There is a visual / aesthetic component that other games don’t have, and it makes a big difference in how these games are designed and developed.
This guide starts from the most simple element, the bullet, and works its way up to complex patterns (bullet groups like rings and stacks) to slowly build an understanding of how small differences in parameters like speed, spacing in a group, angles and number of bullets can have a big impact in games as fast-paced and complex as bullet hell games, both from a gameplay and an aesthetic perspective. For example, one of the first elements is the form of the bullets, and Andrew makes the point that depending on the movement of the bullet, a bullet with an angled shape may be better than a round-shaped one, as the player can perceive the direction of the bullet faster when there’s a visual hint.

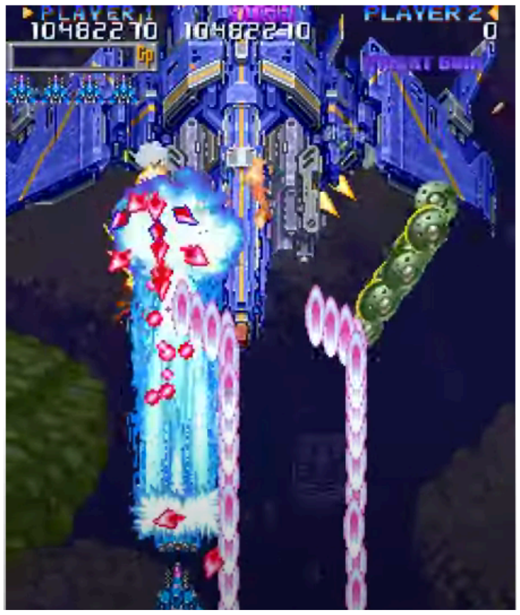
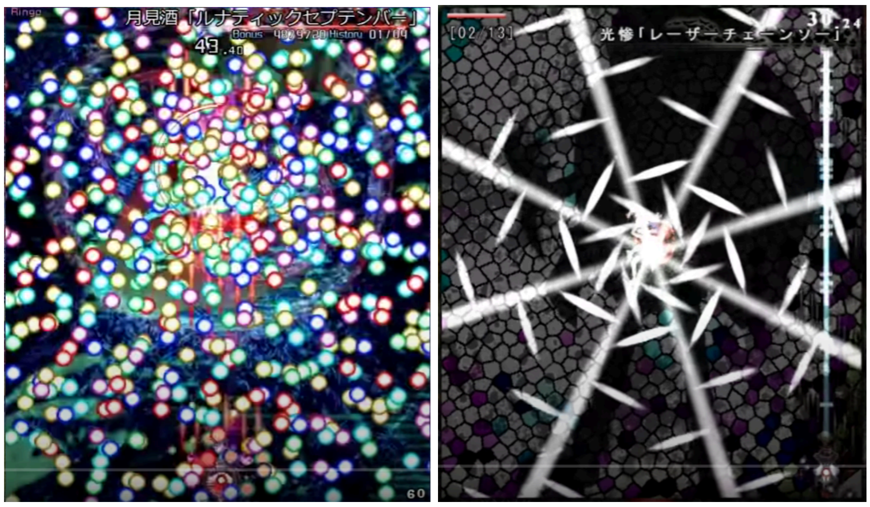


Image: In game screenshot from DonPachi, displaying 4 different kinds of enemy projectiles, each with their own movement style.

There are many elements to consider when designing a video game, even in familiar genres, and they lead to different approaches. In this kind of game, these elements can be used to design games with very different playing styles: slower games with higher bullet density or more complex patterns that rely on precise movements, faster games with smaller bullet density that rely on reflexes and shooting. This guide’s last concepts of bullet density and

[6] Andrew Fan, accessed 2023-12-01, Andre Fan’s Code Dump. Sparen's Danmaku Design Studio. URL: https://sparen.github.io/ph3tutorials/ddsg0.html
[7] Andrew Fan, accessed 2023-12-01, Andre Fan’s Code Dump. Sparen's Danmaku Design Studio. URL: https://sparen.github.io/ph3tutorials/ddsga2.html.

pacing, and spatial density, can be very helpful for developers to understand what makes these games work, and how to experiment and create new challenges.


Images: Left - Screenshot from Touhou 15 - Legacy of Lunatic Kingdom.[8] *Right - Screenshot from Len'en 2: Earthen Miraculous Sword.*[9]

Developing a good game usually requires a long time, a deep understanding of the genre's main mechanics, and several rounds of playtesting. Not all genres have this kind of material, but it is only an introduction and can only help to learn more from experience, make more insightful analysis and have a slightly better grip of how to tune the difficulty.

[8] "Jaimers", accessed 2024-01-08. Youtube. "Touhou 15 東方紺珠伝 ~ Legacy of Lunatic Kingdom - Legacy Lunatic No-Bombs 1cc". URL: https://www.youtube.com/watch?v=9D6fj5AUmnM&t=523s
[9] "MegamanOmega", accessed 2024-01-08. Youtube. "Len'en 2: Earthen Miraculous Sword - Extra Stage (No Commentary)". URL: https://www.youtube.com/watch?v=vq2tlX-p9Sk&t=252s

Design

Project's domain and users

The domain of the project is arcade action games, and the target users are video game players, from teenage to adults. As the game's difficulty level will be elevated, it's not likely to be played by casual players, most likely the audience will be gamers that are already interested in the genre.

Users and Domain's Requirements

One of the most important requirements of this game genre is to focus on the enemies and their attack. This is a part of the genre's definition, and to fulfill it, I decided to make both as configurable as possible, using the inheritance of the classes in C#. There's an abstract class with the most basic variables and functionality, and it's overridden by multiple implementations (movement, spread shots and enemies). And to store the configuration of these multiple elements, I'm using Unity's scriptable objects. Scriptable objects allow the creation of assets to store values and share them among multiple objects. Another useful resource for this requirement is Unity's modular and compositional structure; different aspects of a unit's behavior can be developed independently and added as components of a game object.
To have a visual impact, different types of movement must be implemented, and they have to be used in forms that have an aesthetic value, present a challenge to the player, but are not of an impossible difficulty. Balancing the difficulty, while still making the enemies attractive will require some extra time.

Project's Structure

Every game developed with Unity has multiple elements: handling the player's input, the main character's health and score, a graphic interface, sounds and music, projectiles, enemies and their attacks. This is backed by C# scripts, art assets like sprites and sounds and prefabs.
This is a list of some of the most important components of the game, from the coding perspective.

For the player:
Player Controller: receives the input and processes the corresponding actions.
Player Stats: Keeps the player's data: attributes, score, health points, and is used to inform the GUI.
Prefab: The player's ship with all the necessary components for it to move and interact with other objects.
Power ups: Healing and fire power increases.

Bullet controller: Collision logic.

Bullet prefabs: packages with different properties, to instantiate objects depending on the firing character’s capabilities.
Bullet spreads: groups of bullets created at the same time, settings for different implementations (radial and linear groups of bullets).
Attack patterns: settings for bursts of bullets fired by the enemies.
Movements: Abstract class and multiple implementations to manage the bullets and enemies’ movements.

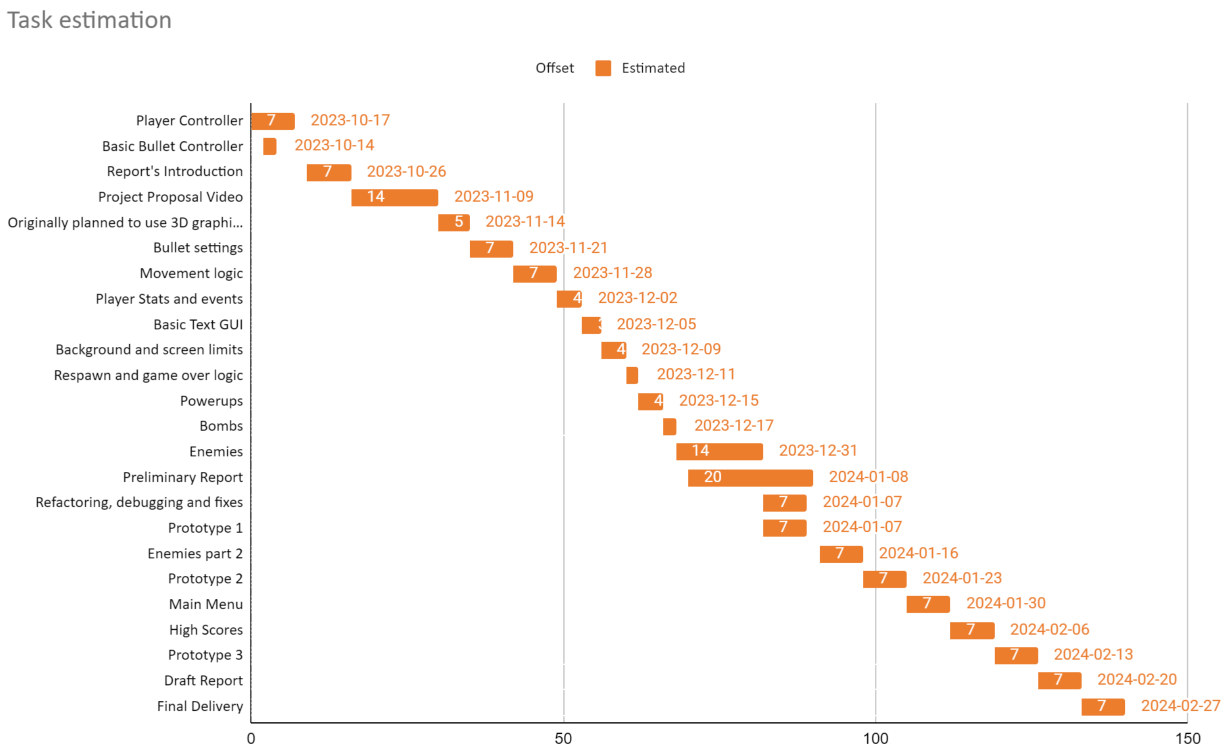
Game Controller: Controls some events of the game flow, like starting the game and ending it, player’s respawn and GUI.
Time Controller: Controls the time dilation logic.
Level Controller: Controls the enemy spawning.

For the enemies:
Enemy prefabs: Packages with the enemies properties embedded.
Enemy controllers: An abstract class with the base properties and methods, and different implementations for different enemies.

Considering the user perspective, the game consists of different elements: sounds, music and visual effects, a controller to interact with the game, data about the game’s status; and also a workflow: a title screen, a main menu, options and settings menu, perhaps a tutorial or instructions, and the game itself, that could be divided into smaller units like stages, levels or chapters. In order to develop a game of professional quality, it is important not only to work on the game mechanics, but also to set a mood through the game’s context and generate an experience as immersive as possible. As mentioned before, “danmaku”, or bullet hell, games have an emphasis on aesthetic appeal, so it’s relevant to create an immersive experience from the starting point of the game.

Work Plan

I have divided the work into smaller tasks and grouped them according to their content. I haven’t set strict deadlines besides the milestones of the module, as in many cases I would have to adapt to accommodate both my other responsibilities and all the learning and research done while advancing on the project. The main deadlines are the Project Proposal Video, Preliminary Report, Draft Report and Final Delivery.
It’s not easy to predict how much time each task will consume, as some topics are harder to understand and implement, and additional time might be necessary for research and doing smaller iterative changes in order to complete the expected outcome. Likewise, the scope for some features may need to be modified to address changes in the design that come from feedback or from the development process itself.



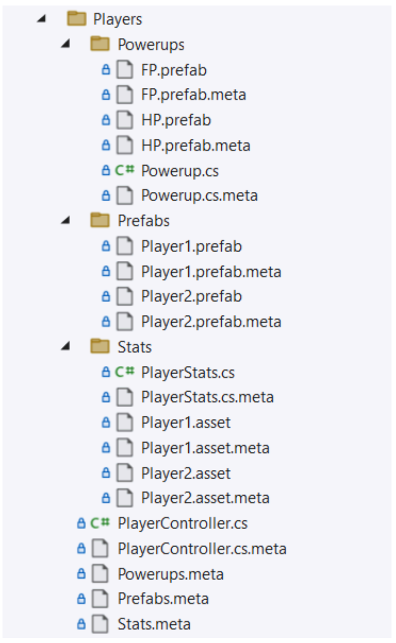
Graph: Gantt chart with the groups of tasks and their estimated duration.

Project's testing and evaluation

I have planned the release of one prototype of the project before the delivery of the preliminary report to test the basic gameplay elements of the game. Then I'll release versions with smaller changes to iterate on the level and enemy design, and to add the remaining functionality. Feedback will be considered to make further improvements.

Implementation

I have separated the development of the game in three sections:
Game Mechanics: This stage included all the components necessary to play the game: player characters, players' input and actions, movement control, enemies, bullets, bombs and special skills, collisions. The most relevant techniques used to make the coding more robust and effective are: prefabs and components, factory pattern, Unity's scriptable objects, object's polymorphism and inheritance.
Unity allows the creation of prefabs, similar to classes in OOP, that can be instantiated as Game Objects with certain predefined properties and actions. Typical examples of prefabs are enemies, items and player characters that appear during the game session. This way the developers have blueprints to create new objects and don't need to define each copy of a certain type of enemy.
The factory pattern was used as a means to have an object that is responsible for the creation of new instances of a certain type, in this case, it allows the player character to create bullets to fire, and is used to instantiate large numbers of enemies at the same time. Since factories can be used to instantiate different objects, they can also be combined with polymorphism, to create objects in similar patterns, like a 3-shot radial spread in 90 degrees, that can be made with enemy ships or with bullets.

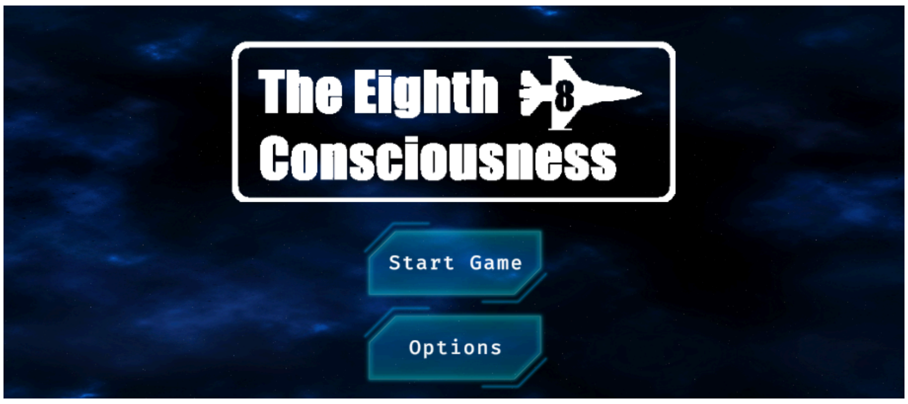


```
Players
    Powerups
        FP.prefab
        FP.prefab.meta
        HP.prefab
        HP.prefab.meta
        Powerup.cs
        Powerup.cs.meta
    Prefabs
        Player1.prefab
        Player1.prefab.meta
        Player2.prefab
        Player2.prefab.meta
    Stats
        PlayerStats.cs
        PlayerStats.cs.meta
        Player1.asset
        Player1.asset.meta
        Player2.asset
        Player2.asset.meta
    PlayerController.cs
    PlayerController.cs.meta
    Powerups.meta
    Prefabs.meta
    Stats.meta
```

Unity's scriptable objects were used many times: the players' status is saved in them, the configurations for enemies' movement and shot patterns are stored in scriptable objects too, and even how the enemies are spawned can be assets based on scriptable objects. This pattern enables data sharing and easily configuring new classes or variations of classes.
Player stats were created as scriptable objects, as they allow to have some basic configuration that will always remain static (initial values for lives, hp, bombs, etc), and also keep other values during the play session, even if the object that the stats are attached to is destroyed (when the player character is destroyed, we still need to keep account of the lives and score). Scriptable Objects were also used to store the configuration for bullets and for the way they are created, so both the properties that define the bullet movement and the way an object fires are stored in Scriptable Object assets, this can also make the configuration reusable as some patterns may be used by many different classes.
In the screenshot on the right side taken from the IDE, the contents of the *Players* folder are displayed. This folder contains the *PlayerController.cs* file, with the main MonoBehaviour that defines the players actions, and the folders to keep the prefabs and logic for the powerups (which increment Health Points and fire power), the folder for the player's Prefabs and the folder for the *PlayerStats* Scriptable Object and the assets for both players.

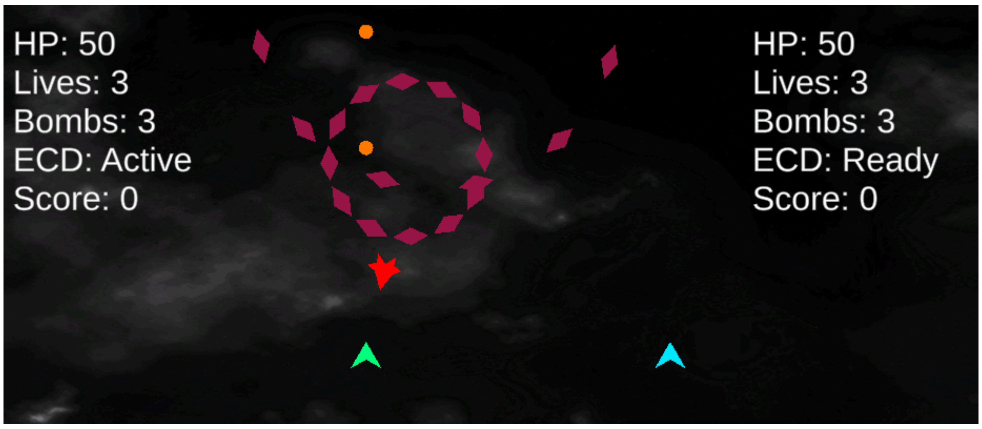
Inheritance played an important role to define behaviors that are inherently similar, but also different, for example movement. While it could be possible to define less but more complex

behaviors that are defined by a bigger number parameters, dividing them into easier to read and modify units can help the development, extension and maintenance of the whole solution. One example: I created an AbstractEnemy class that contains all the basic properties and initialisation logic (variables that contain the status of the enemy like HP, firepoints, timing, on death event, sounds and attack patterns), and the methods to get hit, updating the GUI and how to act when the enemy dies. All these variables and methods are common to all the enemies, and having them in one place reduces the code base and makes the code reusable. All the enemies inherit these base behaviors and can work on top of them to create more complex ones.
Game Status: The components involved in this section included the metrics that the game uses to determine the users' progress and their variation: Health points (for both the user and the enemies), lives, bombs, skill cooldown, score. Also, how this data is presented to the user and the events system that communicates them.
For the communication between different components, I used Unity's event system. It's based on the observer pattern: one component sends events and they can be read by multiple different observers and react in their own specific way. For example, when the value of the HP or score changes, the GUI can be updated automatically to display the new value. Another way to use events is actually the inverse: to update the points, we need to pass a value from the enemy to the player stats when the enemy dies. Instead of having multiple observers and one producer of events, this time we have a unique observer and multiple event producers. Also, the project needed two implementations of the event system, one is based on Actions, which can only pass one parameter, and the other uses custom events that implement the UnityEvent class and can pass up to four parameters. When the score is updated, we need to understand to whom the points should go and how many, and to keep the observer's status up to date we also need to know what enemy was destroyed, to remove the listener.
Game Aesthetics: Sound, music and visual effects that enhance the experience. The parts that I had to code from scratch for the game aesthetics were the title / main menu, with its submenus and the mini-tutorial. As I am not an expert in graphic or multimedia design, I used music, sound and graphic assets from Unity Marketplace to create the adequate look and feel for the game. This part was added relatively late in the development of the project, and from the feedback I received, adding music and sound gave the game a much more professional feeling. The volume for sound and music can be modified in the *Options* menu, Unity's PlayerPrefs functionality was used to keep the values through the game session.
The menu was developed using Unity's in-built GUI package.



Screenshot of the game's main menu.

As usual in the genre, the background image moves to create the effect of moving forward. This effect is constant, except when the special slow-motion skill is used. When it happens, the background turns gray-scale and the movement slows down. The change to gray tones helps the player to focus their attention to the elements that remain colored: the ships and the bullets.



Screenshot from the game while using the slow-motion skill.

Evaluation (max 3 pages)

describe the evaluation carried out (e.g. user studies or testing on data) and give the results. This should give a critical evaluation of the project as a whole making clear successes, failures, limitations and possible extensions.

There were a couple of feedback rounds with early prototypes, but as the project was still in development, much of the feedback was related to functionality that wasn't implemented yet (menu, sound, visual effects, balancing the difficulty, some small bugs that needed fixing).
For these prototypes, I allowed the players to express their opinion freely. I haven't used guiding questions or quantitative methods, as the feedback was used to evaluate whether the scope that was set at the beginning of the project was correct or if it needed big changes.
Regarding the special skill with the slow motion effect, some players mentioned that it was hard to use it in the first tests since they didn't know about the enemies and their shooting patterns, but it can be leveraged to plan better strategies when there are multiple enemies in the screen. The first prototype didn't have any visual effect at all, the change of the color of the background was added later and it was a very welcomed improvement as it made it more visually striking.
The UI was mentioned as one of the aspects that need more attention, as reading the status is not as natural as having visual cues. The special skill text-based indicator is not easy to read, and a filling bar was mentioned as a better option. Also there's no way to tell how long it lasts. The ships still seem to be placeholders for more artistic assets. They may be replaced if free packages with multiple spaceships are available, a first search in Unity's Marketplace didn't find any free packages with more than three ships.
Some of the most detail-oriented players said that the main menu doesn't have any way of being used without a mouse. While it's true, and it would match the controller input of an arcade machine, adding that option is not a priority at this stage of the project and will be included in the scope only if there's enough time.
In general, the aesthetics (menu, sound, visual effects) of the last prototype were praised as having improved the overall quality of the game. The gameplay was considered good, but the difficulty needs to be adjusted (how many enemies are present at the same time, how they fire and how much time there's between the waves of enemies).
There will be at least one more test round when all the development tasks are completed, and possibly one last test if there are changes in the difficulty.

Conclusion

The project is reaching its last stage and the game is almost complete. There are aspects to improve, but most of the requirements that were elicited at the beginning of the project were addressed. The game works as expected, but as a first attempt to build a whole game from scratch, it doesn’t have the quality that would make it a worthy competitor in the current market. Even inside its niche market, there are products that are better, and achieve this result by using design techniques and packages that are more adequate to the genre.
In spite of its shortcomings, the game has all of the features that distinguish it from other genres, and can be enjoyed by anyone interested in it.

Annex.

References:

Andrew Fan, accessed 2023-12-01, Andre Fan’s Code Dump. URL: https://sparen.github.io/

Anonymous, accessed 2024-01-08. Wikipedia, Game.
https://en.wikipedia.org/wiki/Game#Definitions

Anonymous, Accessed 2024-01-08. Wikipedia: Indie Game. URL:
https://en.wikipedia.org/wiki/Indie_game#Fears_regarding_saturation_and_discoverability_(2015%E2%88%92present)

Dillon, Roberto, 2011. The Golden Age of Video games - The Birth of a Multibillion Dollar Industry. A K Peters/CRC Press, USA.

Hunicke, Robin & Leblanc, Marc & Zubek, Robert. (2004). MDA: A Formal Approach to Game Design and Game Research. AAAI Workshop - Technical Report.

“Jaimers”, accessed 2024-01-08. Youtube. “Touhou 15 東方紺珠伝 ~ Legacy of Lunatic Kingdom - Legacy Lunatic No-Bombs 1cc”. URL:
https://www.youtube.com/watch?v=9D6fj5AUmnM&t=523s

“MegamanOmega”, accessed 2024-01-08. Youtube. “Len'en 2: Earthen Miraculous Sword - Extra Stage (No Commentary)”. URL:
https://www.youtube.com/watch?v=vq2tlX-p9Sk&t=252s