Deep Raved - Final Report

DRATARI

June 10 2022 - Final Report

Brought to you by

Aymeric FAIVRE, Dorian WOLFF, Pavitran SAMBATH,
Christophe DE PONTAC

Contents

1 Introduction

Deep Raved is intended to be a fast-paced, mini-game based board game. We want to make a party game to give us more flexibility in the game we are going to make. Mini-games allow you to focus on creating short, fun experiences, rather than having to test and strain an initially interesting idea. Putting it all on a board makes the experience all the more compact and controllable. The name Deep Raved was born from an amalgamation of what we felt best represented the idea of the game we wanted to create. A rave perfectly encapsulates this idea of high energy and chaos.

1.1 Information Sheet

Product Type	Video Game
Genre	Party Game
Style	Action / Puzzle / Turn-Based
OS	Windows
Engine	Unity 4.1
Multiplayer Engine	Photon
Number of Players	1 to 4
Rated	T-M

Table 1: Technical Details

1.2 Case Study

Deep Raved, when completed, is meant to be fun above all else. What are video games if not interactive entertainment? As former Nintendo of America COO Reggie Fils-Aimé said, "If [the game] is not fun, why bother?". We want our players to have fun, but we also want them to hate each other during a game.

Our goal is to give players an adrenaline rush through a fast-paced, chaotic game that is not too demanding in terms of brain-power but still fun to play with friends, so it is in our interest to make Deep Raved as polished and interesting as possible.

For us, this project is a great learning opportunity. First of all, it is an opportunity to work as a group, which means an active effort on our part to be coordinated and work smarter towards the completion and quality of the project. Secondly, it allows us to use what we have learnt so far in a concrete environment, namely video game development, to really test our skills and learn new ones.

1.3 Our Team

Even though the game is in a nightclub he had never put a single foot in a party.
The reason ? Constantly sleep deprived and “is too old for that”.

The stand-in for our absent mascot. An eccentric focused on making this project work

Also known as the bad joke machine, has a creative spirit and keeps on polishing his programming skills. However, he can be quite short sighted.

Needs to go to sleep rather than write this sentence.

2 Project Description

2.1 State of the art

The first party-games were not really party-games per say. In the early days of home video game consoles, sports games and what are commonly called family games took the place of modern party games, with a similar interest in multiplayer and fun for all. The Games series and Track and Field series, which came out in the early to mid 1980s, mainly on the NES (Nintendo Entertainment System), were probably the most famous early examples of the genre. It was a simple but revolutionary game at the time: a collection of sports-based mini-games that could be played by several people. If you are looking for the first party-game, there are not many choices one can make, it is either an adaptation of a TV game show or simply Mario Party.

The most popular games in the genre are staples of what a party-game should be, such as Mario Party, Wii Sports and the Jackbox Party Packs. All of these games have an indirect effect on our project for one obvious reason: they have the soul of a board game. The Mario Party series is the most popular of the two, where Nintendo used its famous characters to draw attention to this simple but effective virtual board game. They really revolutionised the concept of board games in the way they implemented the board. While the mini-games are the core of the game, they are not entirely necessary to win the game, allowing anyone to play it, as it requires no skill level. There are however some drawbacks to this formula, for example the mini-games and the game in general can easily become slow if they stagnate.

Wii Sports is another famous gem from Nintendo, intended as a family game that exploits some capacities of the new Wii remote, it still is a widely popular party game. Since the game is a collection of sports related games, it still attracts a lot of people without having to be extremely over the top. Though popular, Wii sports is not perfect, there is no dedicated mode to play all the games and the reliance on motion controls can make the experience somewhat uncomfortable. The Jackbox Party Packs are way more recent than the other examples, their popularity being strongly influenced by streaming culture. Jackbox Party Packs are ensembles of mini-games that heavily rely on the players' ability to entertain the others,

most games being text-based and asking the players to actively write something or choose the best line in the given game. These Packs constitute virtual board games on their own, it is not surprising to see how popular they are. The problem with this reliance on language and the player is that it limits the scope of potential players. For one the games are mostly exclusively in English because certain references and puns would not apply otherwise. Secondly the players can simply be at fault for not making the game interesting enough, if you are not funny the game will quickly become stale.

All of these are great inspirations for what this project should become.

2.2 Synopsis

Deep Raved is a 2D party game, similar to Mario Party. The main characters enter a rave organised by a certain Quetzalcoatlus Tardatus, also known as **Drattardé**. What they do not know is that the rave takes place in a seemingly labyrinth-like complex and that they are trapped there at the mercy of Drattardé and his strange games. They will have to play these games and gather what they can to find a way out of this place, even if it means fighting each other.

In Deep Raved the players advance on a board with different tile effects. From time to time they will land on a tile that will launch a mini-game, a bonus, a shop, or even a curse if they are unlucky. As they move through the mini-games and the board, they will earn a currency that will allow them to obtain equipment and talents. If they are lucky or good at the mini-games, they can also get quality equipment for free. After a certain number of rounds, players will find themselves in an arena where they have to use their equipment to fight a boss.

3 Game Presentation

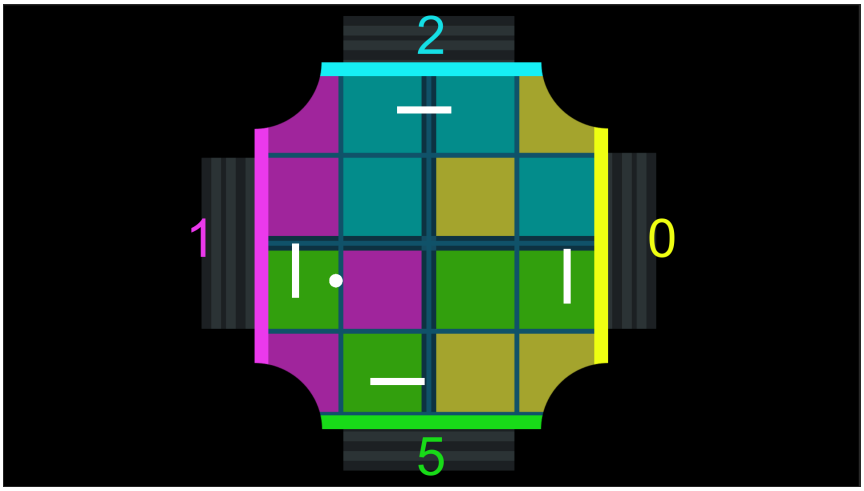
3.1 King-Pong

Made By Pavitran

Rules

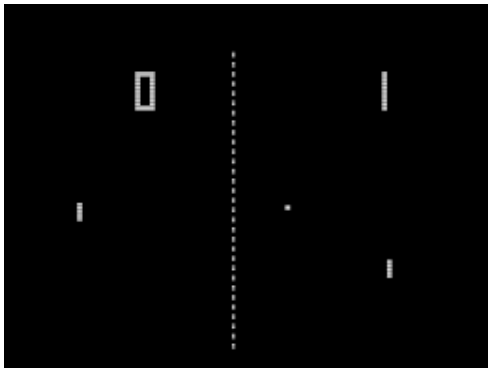
- Number of players : Any
- Goal : Protect your Goal

King-Pong is a 4 players game of pong where, the ball slowly increases speed as it bounces. The way scoring is handled is reversed. In King-Pong you need to protect your side of the floor by preventing the ball from passing your side, the arena is square shaped with corners to prevent unwanted bounces. As soon as someone as taken 11 goals the game ends.



Inspiration The idea came from a concept we initially had for the project but that did not come into fruition, the basic idea was a ball sport where the ball would accelerate upon touching anything with emphasis on being able to hit the ball with any part of the character. Since we were somewhat attached to the idea, we converted it into a more dumbed down Pong version. With the theme of the rave, I also tried to make the all experience a bit more noisy to fit with the aesthetic, the background is bright, the music is dynamic and the speed of the paddles has been calibrated to feel a bit too fast to simulate tipsiness, similarly the AI has been messed with to seem more chaotic and unpredictable.

Conceptualization Making a game of pong is obviously simplistic enough, it is just a matter of laying down the physics of how everything should interact with each other. What was complicated was to figure how we should handle the 4 players, it is impossible to determine which experience is better but we decided to settle down on a 4 player versus principle for easier apprehension.
Another problem was to make the game interesting; Pong is the most basic concept to elaborate from so it was complicated to add things without making the game needlessly complicated or not fun.



The way we handled the 4 players helped to improve the novelty a minimum and the accelerating ball, though kept to a slow ascension, also contributed to it. I decided to change the corners of the arena into quarter circles to prevent players from hitting each other and limit endless bounces on the ball, though I could have just limited the paddles' movement for a more focus experience, changing the shape of the main arena lead to a more interesting behavior when

the ball bounced making it more unpredictable. One major change we decided upon was reversing the point system, rather than aiming for the maximum of points, you need to protect your goal from taking 11 goals because we considered that making the ball "belong" to someone before scoring would make it needlessly complicated to track as well as making it unfun for those struggling to score.

Historic After having the base game of Pong, it was a matter of implementing all the other features. I began with the acceleration factor of the ball since it was a key component, and while testing I decided to limit the acceleration factor from doubling the speed to adding onto it, or else the ball would accelerate too fast to make the game playable. A quick failsafe was also implemented to prevent stalemates where the ball would bounce endlessly regardless of player interaction.

AI Because of aforementioned multiplayer apprehension issues, I then started implementing AI for the game. It consisted of a tracker that could read and follow the relative position of the ball and moved along its given axis. Surprisingly enough it was hard to conceive the AI without handicapping its speed heavily, without this the enemy AIs would play perfectly and never miss the ball. But upon lowering the difficulty the game became too easy and less visually stimulating, to remedy that I choose to put back the initial unbeatable AI and add a factor that would occasionally randomize their movement, though we kept a modular difficulty if necessary.

Problems There were multiple physics related problems regarding the ball, for example collisions felt somewhat loose depending on the angle of contact, the ball had a square shape so bouncing on moving paddles and the curved corners would sometimes have unexpected results. Rather than making the ball stiffer I decided to double down on this non-predictably for more chaotic gameplay, I made the ball round instead since the visual homage to Pong was less important and added a torque effect upon bounce, though to make it a bit easier on the players I increased the size of the ball. Scoring was as simple as tallying the scores on each goal. Upon contact with the ball the score increases for the given goal and the ball resets its position. As soon as any goal's score reaches 11 the game stops.

Visuals After realizing that it would be infinitely better to separate the visuals from the base Pong at least a minimum, I began to make sprites for the game. Since in essence it is still a Pong-inspired game I kept the paddles and ball the same to keep this attachment. The main asset was the background which represents a dancefloor with each lights in accordance with a player's color. The corners were turned black to make the shape more pleasant looking.
UI was added to display the scores and if the controlling player had lost or survived, using crude internet lingo to fit with the aesthetic, namely the expressions "L + RATIO" and "GIGACHAD" respectively representing a loss or win for the controlling player.

3.2 Spam The Line

Made By Christophe

Rules

- Number of players : 1 (vs AI)
- Goal : Kill as much enemies as possible within time constraints

Spam The Line is a 2D survival game were you fight demons by typing the corresponding letter above their head. For example if a monster has a "K" above his head, then pressing "K" on your keyboard would kill it. You have to survive for the given amount of time, if you get hit by an enemy, your character dies and you lose.

Inspiration Spam The Line is based on the game Typing of the Dead (See pictured above), in which you have to press the highlighted keys to complete the word or sentence to kill the enemy. In essence the House of the Dead and subsequent spin-off Typing of the Dead are very famous arcade style games so of course their style of gameplay greatly fits with our party game. To not plagiarize entirely the game, I made it way more simplified, instead of a rail shooter I opted for a single 2D screen where the player can move and instead of words and sentences, there are only single letters.

Historic Initially, I managed to gather all the sprites and game objects needed to make a first working version of the game, namely the player, the enemy, the background and, finally, the keys.

The game has been constantly evolving throughout the semester, in terms of the project structure, sprites and prefab, and many other things.

Let's start with the movements. Over time, player movement has improved, and different spawn techniques have been tested. For example, the player can now move anywhere on the screen, whereas previously they could only move up and down. At the time, I felt that the player's movements were too limited, which worsened the user experience in that it was impossible to stay alive for more than 10 seconds. The game was tedious anyway at that time. Therefore, I decided to redo it completely, changing the way the enemies spawned and adapting the necessary elements in the scene.

There used to be an HUD (See pictured above), which could not be uglier, that was used to retrieve some data from the user, so that they could try to kill enemies. Another visual change came about when I realised that this feature was more a freak-ture of nature than an actual feature. So, I decided to remove it and let the user figure out how to kill the enemies by themselves.

Furthermore, the enemies are now perfectly rendered on screen. And here is the **AI**: the enemies are designed in such a way that they constantly follow the player, making it harder for the user to stay alive, which improves the overall user experience, as the game feels more genuine, engaging and addictive.

Additional Features In addition, there are now two different types of enemies, namely the Big Enemy and the Small Enemy. Both inherit the same data class but have a different speed and size. For the player to kill an enemy, they must press the button displayed above the enemy.

I first decided to update the keys as fast as possible (at every frame), which made my game funnier to play. The tricky part is that the keys were updating so fast that killing an enemy was only a matter of luck. My way to pinpoint this issue was to add a simple efficient timer to adjust the key change span so as to keep the fun and the engaging aspect of the game.

Another deathly hidden feature in my game is the speed: the more you kill, the faster you move. That was the moment when I discovered how to share data in a much more efficient manner than before - I tend to do the same computation repeatedly for each script and objects, rather than storing and fetching the data.

Note that the player's speed depends on the number of deaths, speeding up the player when killing an enemy. A time limit has also been set to make the game more attractive, in the form of a countdown timer displayed in the upper right corner of the screen. The player's movements are limited by the virtual camera, which takes up most of the screen, regardless of the device. On the other hand, enemies can appear outside the screen, making it difficult for the player to avoid them.

What could have been improved? I would have liked to put in more animations, to make the game even more alive. Anyway, I'm very proud of what I accomplished this semester, especially the way I coded my mini-game: the structure and the code are the most technically successful things in this game.

What should I keep of this Project? Working on this project has been a great learning experience for me. Not only did I learn about Unity and game development, which I was very unfamiliar with, but also about how C# is used in a real-world tool (which is something you can not find in regular programming classes at EPITA). I am glad to have learnt so many things I could not think about before. Even though I sometimes play video games with some friends, I am not fond of them, which happens to be a great caveat when it comes to designing and making my own mini game from scratch. Nevertheless, I benefited my teammates' help to think it through and eventually, we devised a not-so-simple fun attractive game – "Spam The Line".

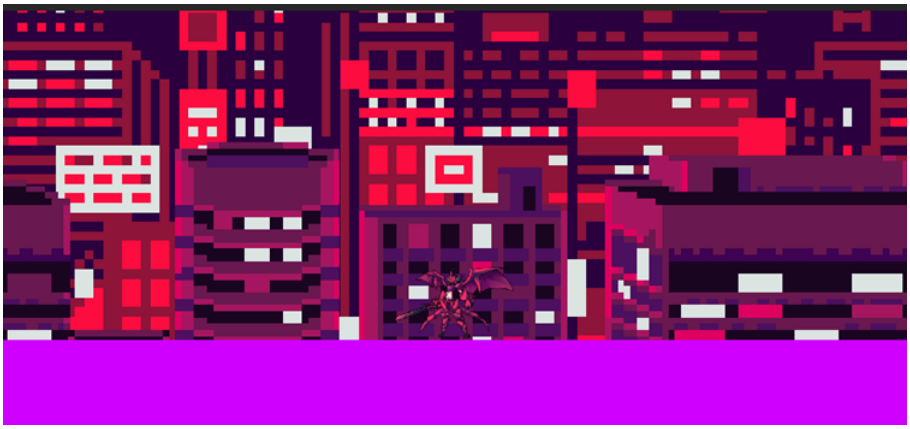
3.3 Anarchy Road

Made By Dorian

Rules:

- Number of Player : 1
- Goal : Get to the end of the map

Anarchy Road is a game where the player controls a red character and has to get to the end of the map as fast as he can. The gameplay is quite simple, as the user simply uses the "up arrow" or "w" key to jump, "right arrow" or "d" key to go right, or the "left arrow" or "a" key to go left.



The player is spawned on a map, where they progress through a randomly generated horizontal obstacle course. The graphics are inspired from futuristic punk-like cities, where worlds like Alita and Blade Runner collide. Since our main game theme is that of a rave, we thought any kind of weird people can end up inside, including the generic red dragon type character. The player has a certain amount of time and unique abilities they can utilize to gather items and points, hence paving their path for victory. Colliding with some obstacles will decrease your remaining time so do your best to avoid them! Some items may increase your stats and enable you to win the race! This mini game is a combination of skill luck and strategy as you progress through the obstacle course. Good luck!

Struggles "As the game developer of the Anarchy Road mini-game, the main hardships I faced were certainly understanding the functioning of built-in Unity functions. At some points, our whole team had to gather to solve one issue, which could have even be seen as small at first glance." - Dorian Wolff, Anarchy Road's core developer.
Other struggles encountered while creating this mini-game are animations for the player, core game mechanics such as fighting and obviously understanding the functioning of Unity at first, since this was our first ever game project.

Evolution Decisions have been made, problems avoided, initial ideas improved. This mini game has drastically evolved over the last few months as unplanned features and improvements (such as different skills and stats to players' characters) have been added. It started out as a single player game where obstacles randomly generate as you progress through the map. However, once the solo version was almost finished, I had to convert it to a multiplayer game. And as a warning to all future game developers, converting a solo game to a multiplayer game is much harder than creating a new one with a solo base. So that is exactly what I did, I rebuilt the game from scratch but as a multiplayer game this time and using all the skills and knowledge I used while creating the first version.

3.4 Can't Run Straight

Made By Aymeric

Rules

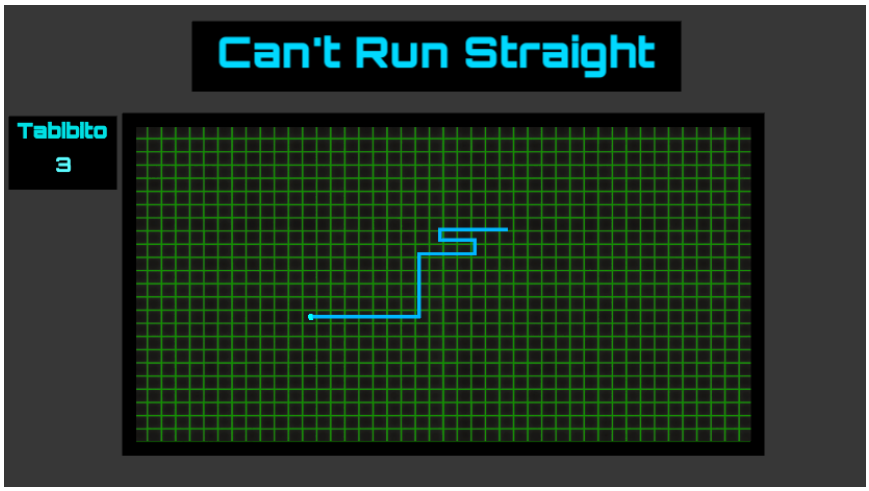
- Number of players : 1 versus All (up to 4 player)
- Goal : Be the last one standing

You are against three other players, you can move using either the arrows or if you are a REAL gamer you can use 'ZQSD' (on European keyboard), and you can create a trail that will be your weapon to defeat your enemies. There are many use for the trail, you can set a trap, force them against a wall... But still be careful speed is both your ally and your enemy.

Inspiration : The inspiration to this game came from a movie that came out in the 80's called Tron (See pictured above). The plot is wrapped around an arcade game where two players confront each other using a futuristic motorcycle on a board called the 'Grid' in a black and neon ambiance and that is what I wanted to recreate.

Historic : I began this game by conceiving a box made of multiple black sprite squares stretched out with a Box Collider 2D component (This will be explained later). I continued by implementing the UI and created an action text that will display the instructions on how to play the game or if a player died. Then I created a player HUD with a space for the player's name and the number of life that they have left and for the final touch I put a background image of a grid.

After that I made a player Game Object and added 2 things; a RigidBody 2D so that the player can move and a script to define the input the player can receive to move ('ZQSD' and Arrows) Now the tough part was limiting the movement to a straight direction since only 2 axis of movement are possible. To solve this problem in the player's movement script I modified the FixedUpdate method that allows the player to move and I rounded the transform component to the .5 upper or lower.



At that point the player could move freely without any barrier to restrain it, and this is the part where I learned how to use the Box Collider 2D.
Even though freedom is important, rules are necessary to not overstep other freedoms. In this game rules are materialized as boundaries. I added to both the sprite and the player a Box Collider 2D, in Unity a Box Collider 2D allows a Game Object to know if it was in contact with something, but more precisely for my game if it was a wall or if it was a player. In order to know what the Game Object was touched with you need to add tags and give the tag to the thing you want. In the present case I gave my boundaries / walls the tag "Wall" and to my player the tag "Player". I could have done it differently, but I did it that way because I wanted players' death messages to be different depending on what players died to. Finally, I began working on the trail.

The trail is made of multiple small sprite squares following each other. In the Update function there is an if statement reading if the space bar has been pressed, if so the program adds a new trail piece to a list and reads the position of the last element before the new trail piece and position, the new trail piece at those previous element coordinates plus the size of the square. Lastly, I added to the trail piece the tag "Player" so that a player touches a trail whether it is his own or an enemy's one.

Ultimately, I did some tests and obviously they were few problems that make the game chaotic.

- Speed:
  The speed was to high and made the player movement too fast so I reduced the speed by a quarter.
- Collision:
  If you decided to go to the left and then to the right you would have encountered your own trail and thus died. So to fix that I created a variable taking the direction of the player and the disable the input going the opposite way (for instance if yo go left you cannot go right just after that, you have to go up or down then right)

And with all of that the 'Offline' part of the game was finished but the struggle was not over yet because it was now time to implement it in multiplayer but I will explain this part in chapter 5.2

3.5 Focus Or Else

Made By Aymeric

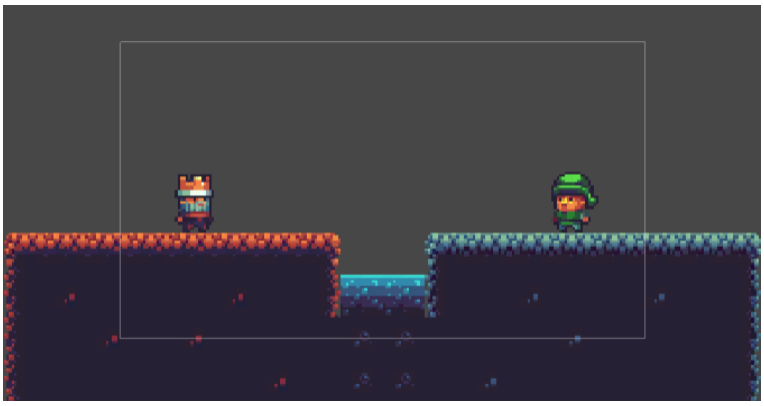
Rules

- Number of Players : 1 versus 1
- Goal : Eliminate your opponent This game is a turn-based duel game similar to Rock-Paper-Scissors. The objective of the game is to bring the enemy's HP down to 0. In order to do so you have multiple choices :
  - Load : Make the player load a bullet, this action is necessary to Attack another player. You can stack up to 3 bullets and each time you Attack you lose 1 stack.
  - Attack : Deals a set amount of damage to the opposite player. If you do not have a bullet (if you did not previously Load) you will not be able to Attack thus dealing no damage.
  - Defend : You create a shield that disables the opponent's ability to make you loose health. Can only be used once every 2 turn.

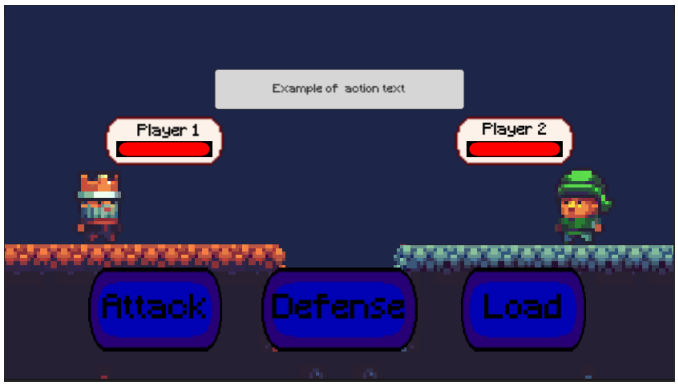
  The games ends once one of the two players died and the winner being the one who remains.

Historic : The first things that I implemented was the background. I searched for a template containing all the elements I needed and then I created the floor. I learned how to use the sprite editor and how to create multiple planes using this sprite editor. The next thing I did was finding a template containing some player animation. With the freshly acquired knowledge of the sprite editor I split the template to multiple single frames of animation I then learned how to use the Animator. It was not much, a simple loop of two frame making the player look a bit more alive.

Next, I studied how to use UI in order to implement all the buttons that will, in the future, do the action of attacking, loading and defending. After that I implemented an action text that will show the action of each player during a turn and I looked up how



to change a font because those that were proposed were not in the theme of the game. Finally I did a HUD containing the player name and its health bar made of a slider [1]



It was now time to make some code.
I began to create players script that, at the time, only took as input their health, damage and name. The health input was used to synchronize the health bar from the HUD and the in real time life of the player represented as a number. I then created scripts for each button; an attack script for the Attack button, a load script for the Load button and a defense script for the Defense button.
This is the part where my lack of knowledge, at that time, of C# and of Oriented Object Programming will be the source of a big loss of time later.[2]

[1] HUD or Head Up Display is panel of information that allows users to see some piece of information

[2] cf 5.2

The most difficult part of this mini game was the turn-based system because, at the time, I did not start implementing in multi-player and thus making a turn was difficult. So in order to simulate a turn I used the Invoke method and the IEnumerator to have a delay between multiple player actions like if a player chose the Load button the action text will change to : " '*player name*' is loading its attack !" and then the player that did the action had its buttons hidden and then the action text switched to : " It is now '*player opponent name*' to play " and the opponent's buttons showed up with a few millisecond of delay. But when I did some testing, I noticed that the buttons were clickable multiple times during a single turn and that there was too much delay between the text and the action. So, I had to modify the script so that right after the player clicked a button all the buttons hide and right after the opponent buttons appeared.
Synchronization between turn was done and the game was almost done so I asked one of my friends to test it and he said that the Attack action was a bit boring because you could only see the opponent health going down without any visual interaction. I then asked him what was in his opinion the best attack animation for this, he answered me "Would it not be funny if they blasted a laser". And that is what I did.
First, I designed a laser then imported it in Unity to make a prefab that I will Instantiate multiple times until it touched the opponent and then make the clones disappear. To do so, I got its dimensions and then put it in a loop with an adjustable number of iterations that Instantiate laser prefabs one after the other and put each of the prefab in a list. Then I used the list of prefabs to make every prefab disappear and clean the list. Now the game is done.

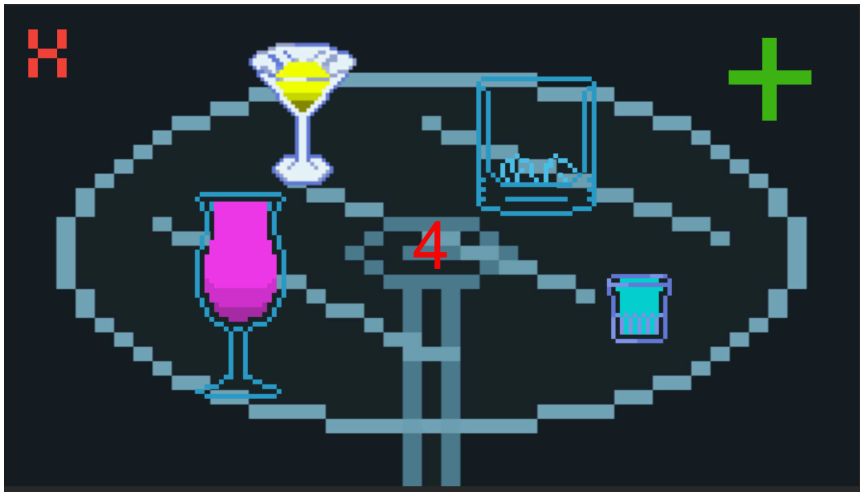
3.6 Spawn Colors

Made By Pavitran

Rules:

- Number of Players : Any
- Goal : Complete the sequence of cocktails

Colors is a game in similar fashion to a "Simon Says", the players create a sequence and need to reiterate the previous sequence before adding onto it, upon failing the sequence the player is punished. All gameplay is done by moving and clicking with the mouse. In Colors you must make a sequence of cocktails to drink in succession, you are given 5 seconds to complete or add onto the chain and 4 cocktails to choose from by left-clicking on them. If you mess up the sequence or the timer runs out, you accumulate mistakes and after 3 mistakes you lose the game, otherwise an indicator shows that you have added a cocktail or chose the right one.



Inspirations : This was one of the first mini-games to be conceptualized for how easily it fit our theme. Replacing the concept of adding words or actions by cocktails is a surprisingly new experience if given the right atmosphere. The eventual addition of a timer is what builds tension on longer sequences, since the visually colorful and bright "buttons" allowed for easier memorization so there was a need for an incentive to not idle.

Conceptualization : The first phase of experimentation for implementation was done purely through code, to really get into the core of the mini-game. It was a slow process of understanding how to translate the concept of a "Simon Says" into code, though some aspects are probably not as optimized as they could have been, namely the way the cocktail objects were implemented, they were optimized for easier development. In fact, most of the technical aspects were done first, the skeleton for all features of Colors was built first since there are really only two logical components, the overall observer of the game and the cocktail object. So, every aspect of the code could be overseen in advance.

Historic : At first the main part was the sequence building management as in the beginning we were unsure how multiplayer would function. I choose to make each cocktail a separate gameobject that could be recognized when clicked, a script applied on the camera then allowed to interpret each individual cocktail on the scene as well as the building of the main sequence, which consisted of checking on an array if the given cocktail that was clicked corresponded to the current observed cocktail in the sequence.

Then next to implement was a timer, though as explained before some aspects of it had already been laid the ground-work for, so it did not take as long as expected. It was simply a matter of deciding a maximum interval of time between each turn, I settled on 5 seconds per turn. The timer then worked by removing the time of each frame to the previous time value, for a more precise measure, until it reached 0. After that the mistakes counter was to be implemented, for that I choose a limit of 3 mistakes and/or time limit breaks even if on early turns this limit seems rather high it is on subsequent turns that it becomes more fair since you do not know

which was the right choice.

Visuals : At that point I struggled to implement the multiplayer component in a concrete manner, so I proceeded to work on improving the user interface, since at the moment there was only code. It started with simple buttons without any flavour. Then I made my first sprites for one of the cocktails and proceeded to learn how to animate through Unity. Though crude and rather undetailed the sprites and animations I made for Colors were representative of the overall atmosphere we decided on, even if the background is probably a bit too weak compared to the rest, namely a plain table. Overall though the process of creating these sprites was rather simple I then continued with the rest of what was already there, I added a minimalist visual timer on the middle of the table for better visibility and a visible sign to show that you lost or made a mistake. On further test plays I realized it was not that clear when you made the right choice or added on the sequence, so I added visuals for these as well.

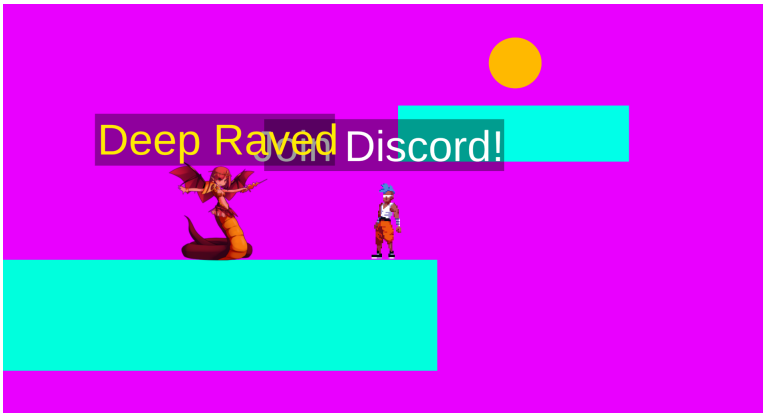
3.7 Arena

Made By Dorian

Rules:

- Number of Players : 4
- Goal : Get the most points

Arena is a game where the player controls a selected character and has to collect as many points as they can. The gameplay is quite simple, as the user simply uses the "up arrow" or "w" key to jump, "right arrow" or "d" key to go right, or the "left arrow" or "a" key to go left, as well as the "space" key to hit and damage other players.



Originally thought of being a boss fight, the Arena mode was selected as the last mini-game on our list of games. It uses the same avatars as in the mini-game [Anarchy Road] but, as its name suggests, enables players to fight among themselves and hence, creating an ultimate champion! Players compete against each other on a fixed platformer map and gather skills and items to take other players down. This game stage is where each one of your previous decisions really pay off and where players experience different levels of emotions, ranging from stress to delight without forgetting hate. The final ranking is determined by your survivability in this mini-game. If no players die within the time limit, previous game rankings will be taken into account for the final score result.

Struggles This mini game encountered many similar struggles as Anarchy Road as it has the same game base, such as animations for each character, but adding to that, comes implementing multiplayer with Photon, which was definitely a core and game-determining function. Additionally, I had to add hitting mechanics which can be tricky in multiplayer but it eventually worked out and now the combat system is fully fledged.

Evolution This mini game was originally supposed to be the end of the line; a.k.a. a boss fight. And even though it is the last game where players fight among each other, there is no actual "boss" apart from other players. However, we already were considering making some sort of arena instead of a boss fight at the beginning of the project, and we safely changed the endgame to match this idea. The arena is one of the key assets the game has, and since it is a platformer, it has a similar vibe to the "Smash Bros" series. Apart from

changing its original footing, the game is exactly as it intended to be, only things like balance changes and more character releases are left for this game to function flawlessly.

3.8 Technical

Audio *Made By Pavitran*
Audio has been implemented but sound effects in particular were not added to any of the games to focus more on the functionality of the mini-games. Music has been selected very early for everything but mixing and looping took quite some time, the following tracks were added :

Main Menu : Express - luxury elite *(sampled)*
King-Pong : Club Ezekiel - Toshiko Tasaki Tsukasa Masuko
Spam the Line : Hacker (instrumental) - Death Grips *(sampled)*
Anarchy Road : Le perv - Carpenter Brut
Can't Run Straight : The Game Has Changed - Daft Punk
Focus Or Else : Ghost - Machine Girl *(sampled)*
Spawn Colors : Voyager - Jasper Byrne
Arena : Robot Rock - Daft Punk *(sampled)*

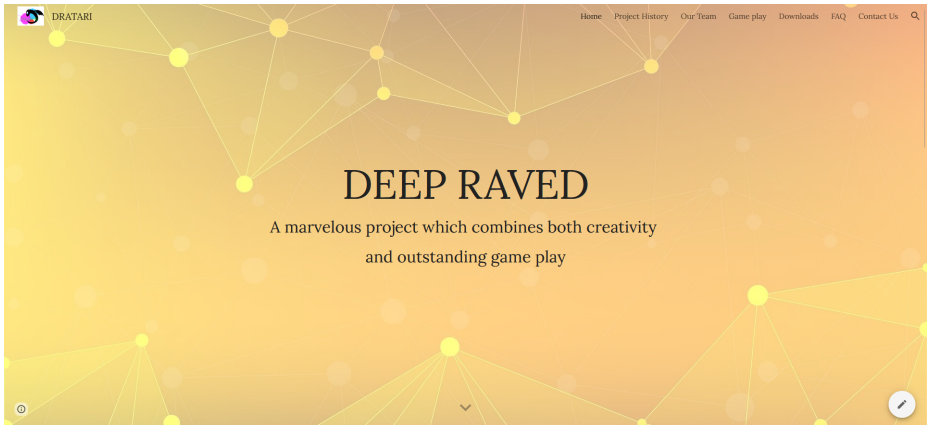
Visuals For the most part visual assets were imported from free-to-use assets on the the Unity Asset Store. Otherwise the sprites were made by hand.

4 Socials

4.1 Website

Made By Dorian & Christophe

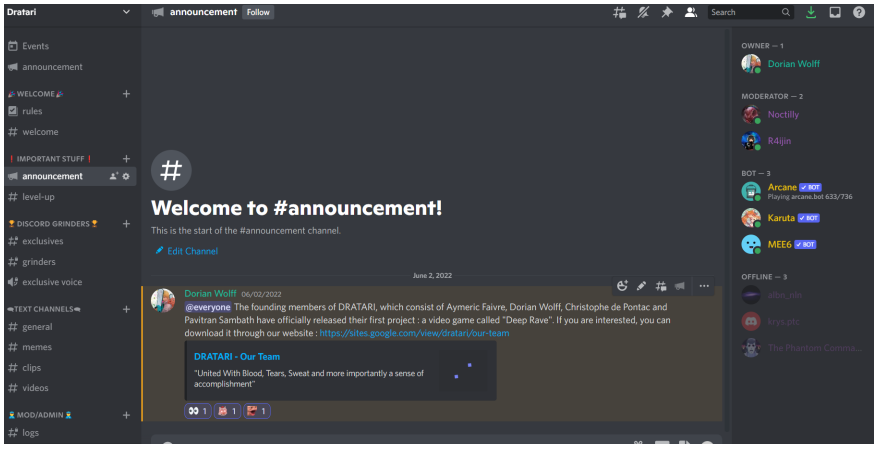
The website is a key feature to promote and advertise the game. The website has many functions and serves as a pivot to our growing game. Ranging from useful links to Easter eggs, without forgetting its clean aesthetics our website has enough features alone to keep you entertained! The different categories of the website are intuitive and useful. The home section briefly presents our project and the game as a whole, while the Project History part sums up our struggles and the evolution of the game while providing visual aesthetics. The Our Team section provides information on our team as well as personal developer notes, the Game play section enables users to watch game play videos on our YouTube channel? Finally, the Download section's purpose is to download the game, which will automatically happen, and the Frequently Asked Questions and Contact Us categories are self explanatory and provide a polished part with hyperlinks to solve your problems. We also have a search system which provides the user with a search tool on the website.



4.2 Discord Server

Made By Dorian

Also, as any good video game we created community platform to give support or advice to new comers or other game developers. And on what platform do programming and gaming communities collide? That's right, discord. Our hand-made website brings many key details and evolved with our project, hence creating a dedicated community. I created an open discord server with many usages and plenty of entertaining features, such as bots to manage and add key security functions (Arcane and MEE6) but also play games (karuta bot) to mix fun and help.



4.3 Other Socials

We have created a YouTube channel which is linked with the other media platforms. There are currently only a few videos but we plan on making the channel and the community as a whole grow, with the aspiration of reaching one day a few thousand subscribers.

5 Difficulties Encountered

5.1 Git

Using git was a first for us and we did not know anything about it, so we had to learn. First creating a new repository git and connecting everyone to it was the easy part but the first problem appeared. Sometimes we are disconnected form the git repositories thus each time we had to redo all the process of the beginning we did not found a solution to fully fix this, but we found a way to reduce the process to only 2 lines of command. Due to our lack of knowledge, we did as we usually did and push all our games on the same branch, but it did not work and some of our friends advised us to create a branch for every one of us. We emptied the git repository and created branches as recommended and the commit and pushed everything. This decision caused many issues especially at the end when we had to merge our project for the multiplayer implementation. The merge process was the point in time where we lost most of our time because we did many false manipulations that made us loose some precious time because during that process, we lost many elements for the games that were merged making them unplayable. Our insufficient knowledge of git made us recreate some games multiple times and again with the aid of one of our friends we learned that we could came back to some of our old commit. But not everyone had a commit of their latest version of the game we had to redo some part or most of our games. In one hand commit difficulties and on the other hand merge conflict solving caused a delay of almost a week in the schedule. Finally merging the multiplayer's games were a long and rough task because a lot of unknown errors appeared and thus, we had to find new ways to solve those problems losing again some precious time.

5.2 C# Coding

We began creating this game at the same time we learned how to code in C# and in Oriented Object Programming in general so we could see the evolution and the application of our work with this project.

Aymeric During the whole year I had trouble understanding how OOP [3] worked and gradually I started to figure out how classes worked how keyword worked. The game that was the pinnacle of all this knew acquired knowledge was Focus Or Else. At the beginning I created a method for every and each action in the game and put it in the same file so reading the program was awful and implementing the game in multiplayer was almost impossible, so I rewrote all the code to make it more readable and easier to implement in network. This is where I understood how much I have evolved I was able to clean the chaotic containing all the method. This took me some time because by cleaning this file I created a lot of other problem some major like laser not appearing to smaller detail like the action text not having the right input. If I developed this game like I did at the end of the project this game could have been working online.

Christophe C# Coding was not the trickiest part to me. Overrides were the only part of C# that has not been covered in regular programming lessons, when it comes to the syntax. As for the rest of the coding, I struggled to grasp the working of Netcode, which was left out in the cold. I also spent a lot of time getting familiar with Photon documentation, which was easier to manipulate at first glance, yet a bit tricky despite appearances. I felt that probing documentations was a great training, even though it is sometimes daunting to see how vast a documentation can be.

[3] Oriented Object Programming

5.3 Multiplayer Implementation

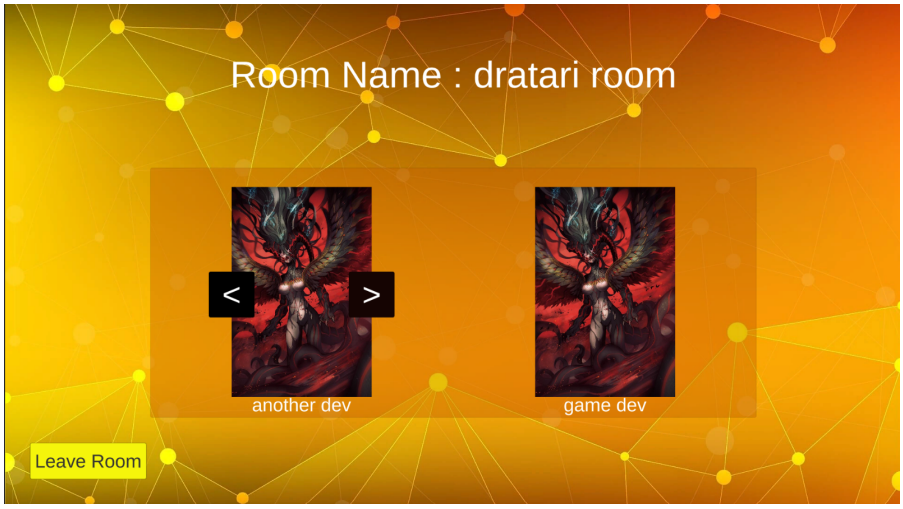
When we started searching for ways to implement our games in multiplayer the result that came up first was Netcode, Unity's network environment. So those who finished at least one game started looking on how to implement their game in multiplayer, but it was a real struggle because the documentation was vague, and tutorial were few. Even with those constraints we still tried to continue the implementation, but the process was slow and dreadful. Thus, when everyone completed their game, we decide to switch to Photon 2 which had everything that Netcode did not had, such as a clear documentation, forum to answer our question, multiple tutorials and it was far simpler to implement overall. Now let's hear what everyone has to say about the multiplayer implementation.

Aymeric Just after finishing Focus Or Else I started seeking information on how to conceive a game in multiplayer at first it was Netcode so I tried to both create and implement Can't Run Straight with Netcode but the task was harsh because I had to learn all the network mechanics. It took me a while to merely begin to understand what methods did, how they work etc. I went through all that process because I wanted to prepare myself to solve potential future error. When I started the creation of the lobby, I realized that it will take much more time than anticipated in the schedule. But I still manage to create the lobby (where player can choose in which room [4] they go) and the player manager (the script that instantiate players and make them appear in the same room) but even being hands deep in the multiplayer there was still too much haze especially for the UI and the Animation part where I did not find any documentation or tutorial.

[4] A room is an online space where multiple player can interact with each other

Pavitran Multiplayer was the hardest part for me, I had the least preparation on how to implement online, considering that I overviewed both audio and most of the graphic design combined with poor time management on our part, I never was fully accommodated with this part of the project, so I did not manage to add multiplayer components to my part of the games.
A lot of the ideas I had for my mini-games were interesting in concept but too complicated since we had to make sure everything worked on multiplayer. For example I removed camera effects on both games, such as frantic zooming and spinning and soft blurring to add onto the "rave-like" experience, because I was not ready to port it into a multiplayer situation.
After watching countless tutorials for both Netcode and Photon2 with diminishing returns, I decided it was better to have a functional solo game than a buggy unoptimized multiplayer mess.

Dorian After going through the menu, connecting to a server then joining a room, players are asked to select the character of their choice.



These characters all have their own in game stats and abilities which will come in handy for the rest of the game. The different character options are as many as 6, scrolling from left to right : Tiamat, steampunk, cyberpunk, cybergirl, anubis and lamia. Each of these character models come with their own in-game avatar which

you players will be able to manipulate through mini games like anarchy road or the arena. Notice the "play" button only appears on the hosts' screen and once exactly 4 players have joined the room. Now let the game begin!

Christophe I first tested Netcode as the main tool to implement multiplayer, and I found that Netcode - especially the beta version - was not documented enough (the beta version being the only one that fits our needs). Hence, we decided to use Photon instead. The change took some time, and we had a hard time adapting the mini-games to Photon. My game [Spam The Line] actually required to spawn each player in different scenes for the same game. The problem is that Photon has a property called *AutomaticallySyncScene*, that you can only set before connecting to Photon servers, which forces each player in the room to join the same scene no matter what. Despite all the time and effort put to find a workaround, I was not able to add Photon multiplayer to my game. Faced with my failure, I moved onto working hand in hand with Dorian on the menu and the waiting room in multiplayer. We did a great job implementing this part of the game. We even tried to fix multiplayer-related issues in our teammates' games, and I think we could have finish the game as expected if we had an additional week or so. Working on multiplayer was certainly the most captivating task of all.

6 Personal Feedback

6.1 Aymeric

My feedback to this project can be explained in three main section : Communication / Demotivation / Leadership

Communication I think we can say that one of the main troubles of our team, apart from our technical problems, is that we did not communicate enough with each other. Indeed, the technical difficulties encountered on Git are linked, from my point of view to the lack of communication between us about what each of us was doing and when we were doing it. We were also not enough transparent about the work we were doing, the difficulties we were encountering the difficulties we encountered, and we did not always know how to ask for help from the other We were also not transparent enough about the work we were doing, the difficulties we were encountering, and we did not always know how to ask for help from other team members when it was needed. To remedy this, the team tried, at different levels, according to the personal constraints of each member, to meet personal constraints of each member, to meet on Fridays to bring this moment of exchange and sharing. Since not all team members were able to attend, it was only beneficial to those present, but it had the merit of moving the project the project to move forward.

Demotivation The struggles described in the previous paragraph, the slowness of getting the project back on track after crashing our code several times, the frustration of not seeing the project move forward coupled with the school problems I faced during this semester led to a loss of motivation at times which slowed my progress on the project. Nevertheless, for the sake of the commitment we had made in the team to build this project, I continued developing the mini games that I had to complete. I was also able to see that the other members of the team suffered from the same demotivation which, for some of them, had a more negative effect on the project and which explains, in that this project presented in the initial specifications is not finished.

Leadership When we chose the roles at the beginning of the project, the team leader role was given to me because of my fluency in speaking. As the project progressed, it turned out that this role also required leadership, which I did not show at first because it seemed to me that my friends were autonomous and knew what they had to do. Moreover, having gone through a demotivation phase myself as mentioned in the previous paragraph and not having technical leadership, it was not always easy for me to find the right method to re-mobilize the team. In the same way, when I noticed the demotivation mentioned in the previous paragraph, it was not easy for me to find the right method to re-mobilize the team. As previously said in paragraph, it was not easy for me, as a team leader, to find the right words so that the demotivation would not impact our project.

Conclusion Even if this project has a bitter taste of not being finished, I learned a lot of things to do it. First, I had to learn how to use Unity and even at the end of the project I have the feeling that only scratched the surface of this software. Second I had to study git in order to have a clean workspace even though it has failed for this project I will be prepared for the next group project. Third due to us choosing Unity Netcode I learned not only the code and how to implement the game but also how we can create a server only with our computer thus studying how computer port are working. Finally the most challenging thing was to work autonomously and manage to finish my work in the limit of the schedule.

6.2 Christophe

I would like to present my feedback in 4 main parts: Autonomy, Cooperation and communication, Assistance.

Autonomy It cannot be denied that our S2 Project is deeply geared towards autonomy. Despite the fact that I had enough knowledge and adaptability to create such a project, I had never been involved in a project of this scale before. Yet I saw it as an opportunity to challenge myself, which I think was the reason why I did not give up on the project. Besides, being in a group makes it all the more exciting and captivating. I was able to group all the projects together, as well as helping my teammates working on their games, especially Moreover, we struggled to come up with a game idea that everyone would like. And I consider that the idea of creating a mini game for each person in the group actually had not been without unexpected consequences, because the autonomy we initially had as a group rapidly turned into complete autonomy. That is to say, the fact that everyone was working on a different game made us unaware of each other's progression. Maybe we should have used a version control software such as Git right from the start to force us group together. And that point leads us to the next feedback that I want to share:

Cooperation and Communication At the beginning, we did not make communication a priority, as we should have done. Our lack of technical knowledge regarding Git did not help either. Yet we ended up communicating much more (and better!) after the second presentation, which could not have gone worse. The last part of the semester was thus the most productive. As we managed to make up for lost time, I consider that we have made a lot of progress at the end of this year in terms of communication and, therefore also of cooperation.

Assistance As our communication improved, I started to help my teammates fix the constant bugs we were facing. Fortunately, we became more efficient and my motivation could not have been better. I also received help from my teammates, and it is in these moments that you realise what teamwork is. Assistance is key to keep things moving efficiently and in a clever way, as four brains are definitely better than one. I feel we could have done better if we had changed our way of working, but I am sure next time we will know what to do.

Conclusion I am glad of what we have achieved and learned throughout the semester, and the least I can say is that I am looking forward to do another project soon!

6.3 Dorian

In my eyes, this project will forever be a source of inspiration and hard work.

Work Time and improvement From having barely touched a computer to a fully-fledged game developer, this project's journey was a long and productive one, that I would have had trouble finishing if not for my fantastic developer group. Staying over until past midnight, overworking hours for sometimes a deceiving result. However all this time spent paid off, as we now can present to you the fruit of our combined efforts and a great step in our carriers as programmers. And looking back, I would gladly spend over 250 hours again.

A learning experience Throughout this project, I touched the surface of many tools, such as sprite editors, built-in animators, artificial intelligence in the mini-game Anarchy Road and, of course, multiplayer (which I touched more than the surface) in Arena. I also learned how to manage my time, improved my overall programming skill set, understood object oriented programming and actually genuinely enjoyed working. I was in charge of multiplayer for my games, both Anarchy Road and Arena, the menu, loading, connection, and character selection scenes, as well as creating the website, linking it with a newly created discord server and a YouTube channel. Throughout this process I learned a lot from each of these tasks, having never once touched these subjects previously.

A source of inspiration and motivation This completed project on which contributed, like suggests the heading of my dedicated part, is a great source of inspiration, in the way that I had new game ideas while programming and in the way that I will look back to these days once facing a virtual obstacle and confidently tell myself "I can do it, I have gone through worst". Also, this project was a real source of motivation, since I will create a new unity game during the summer break, which will be a turn based card game (I have started working on the designs with a relative). I also hope I can implement my games on different devices such as phones and make them easily convertible from one one to another.

Conclusion This was probably the biggest project I have worked on in my life, and I know for sure it is not the last. As I continue my journey as a programmer I hope I will encounter many of these projects which actually make me look forward to dedicating my time and effort and see them bare fruit.

6.4 Pavitran

I am biased in saying this but even if the project did not achieve its full potential, its mere existence is a worthy success for us. It may not be the best game ever but it is our product and we have assumed its responsibility through its development.

Creativity Creativity and Problem-solving are two domains that go hand in hand when creating a game or making algorithms and programs. Through creating things for the project, whether it would be assets, code, or even rough drafts, the process of thinking and imagining solutions for every single detail was honestly exhilarating for me. I personally love the feeling of making something and then finding out that it works, I find that feeling a lot when programming but even more when it was for a project we conceived from the ground up. Even if it was a struggle more often than I would like to admit, I would not change the process of making this game very much.

Failure What will probably be remembered way more vividly will be the short-comings of our project, for most of what we accomplished, a glance at our beginning predictions for progression shows that a lot of our expectations were not met. It is without a doubt that I admit that Deep-raved should have been better, but mostly due to our outlandish expectations and very inefficient time management it remains in a state of semi-completeness, while it is a functional and rather enjoyable experience it is not what we aspired it to be. It is a shame that the short-comings of the project more or less shadows the genuine product it has become, for our perfectionist mindset it is unbearable to show any sign of failure even if these failures are merely more visible through the success in the rest of the project.

Teamwork This project is the first active group project I took part in, by that I mean that this group project truly called upon our collective skills as a group to think and create. In opposition to most school projects, this one gave enough freedom so that you had to actively partake in it to make it more interesting/better quality. In no regards will I ever discredit my fellow DRATARI members, in fact I would argue that the work they output far outweighs anything I have done. It is an understatement to say that the help we provided each other helped us progress easier, the struggles we faced were mostly handled as a team in jolly cooperation and tearful late nights. This project has throughout its all development greatly strengthened our bonds has well as our teamwork and it is undeniable that its current state would have never been achieved if not for our group chemistry.

Conclusion Though I am heavily disappointed that we did not entirely finish Deep-raved, the experience of creating and conceptualizing the game though straining was very much enjoyable. It was my first experience with a project organized in this manner so it will remain alive in my memory for both the bad and good. After post-editing this entire final handout I am still proud to say that I was a part of this project, of the DRATARI team.

7 Conclusion

This project is not representative of our actual level, so it has left us a bitter-sweet taste. We are proud of us for making our mini-games because it was a first for everyone and the satisfaction of finishing was immense, but we are also disappointed that we could not make them work together as initially intended. We faced a lot of issues coming from everywhere, but we found a solution for every of those issues without giving up or loosing faith in this project, thus learning how to surpass ourselves. We collectively think that with 1-2 weeks more we could have fully finished this project but sadly time was against us.
To conclude we would like to quote Ralph Waldo Emerson, "It is not the destination but the journey" because it represents well our project. Even if we did not manage to fully complete this project, we learned a lot of things along the way from more technical parts like Git and Photon but also some human part such as working in a team, exchanging our points of view, helping one and the other and working in autonomy. We grew at each of our mistakes and those mistakes made us stronger.
And that is what we will remember from this project.

Appendices

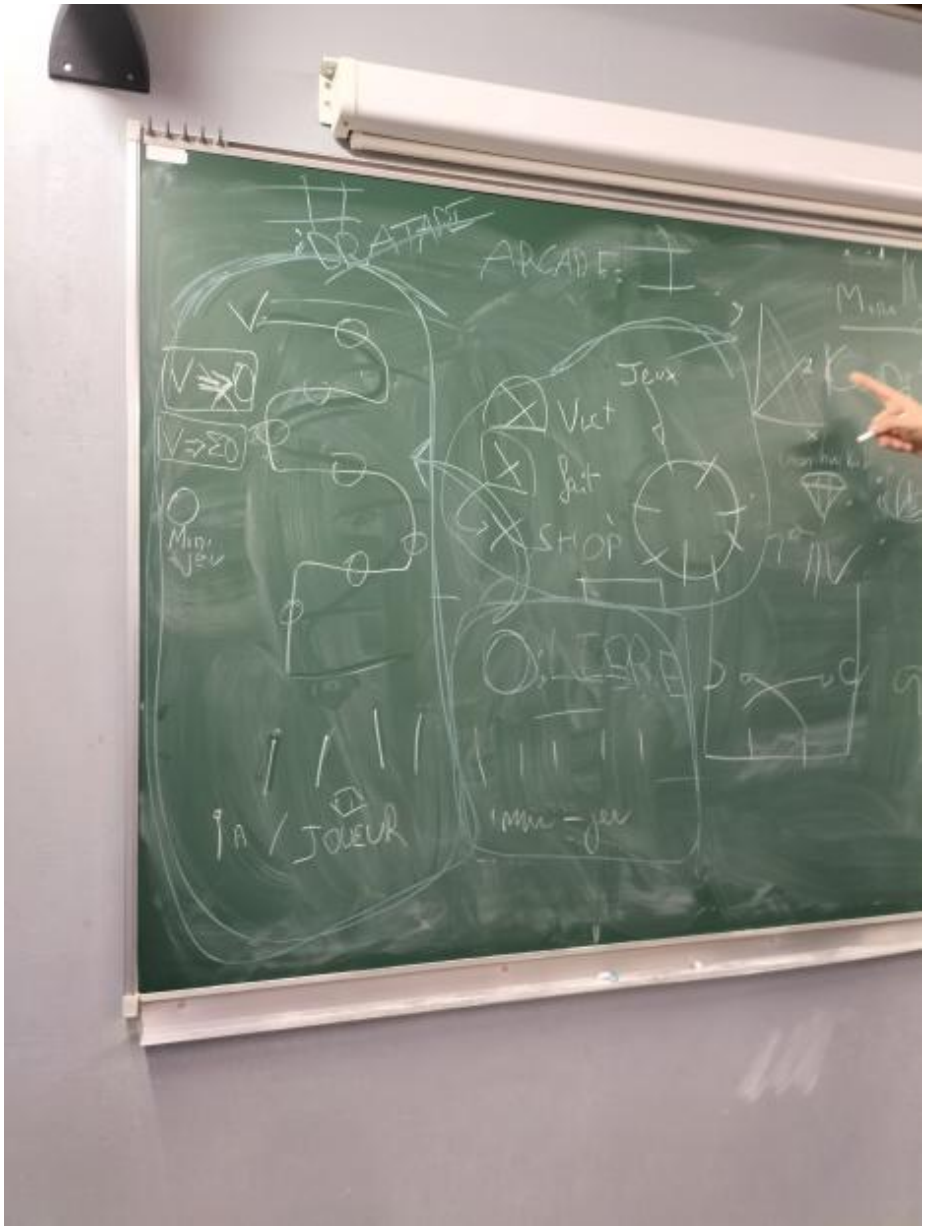


Figure 1: Only insiders can understand.

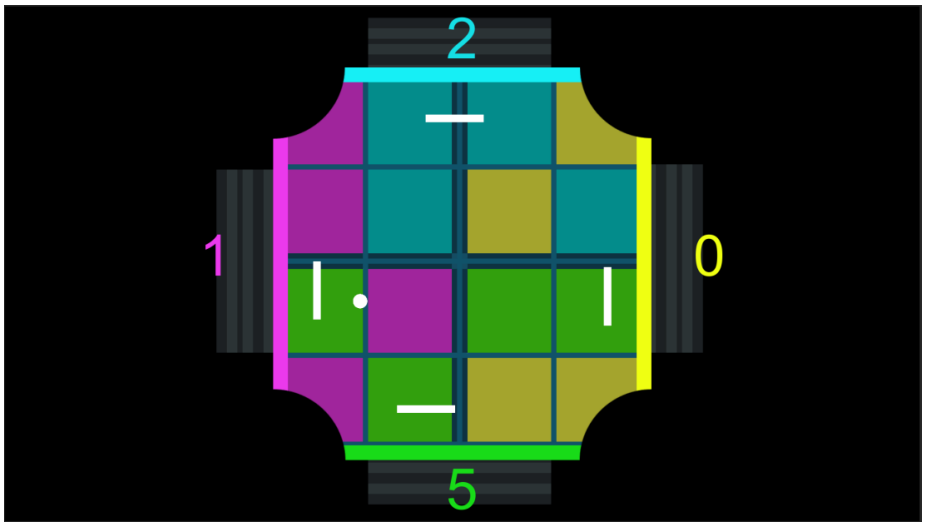


Figure 2: King-Pong Final Version

Figure 3: Spam The Line Final Version

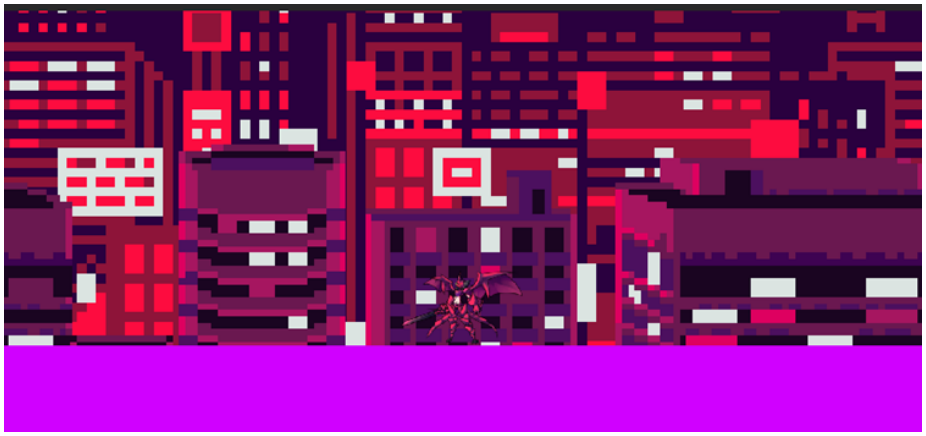


Figure 4: Anarchy Road Final Version

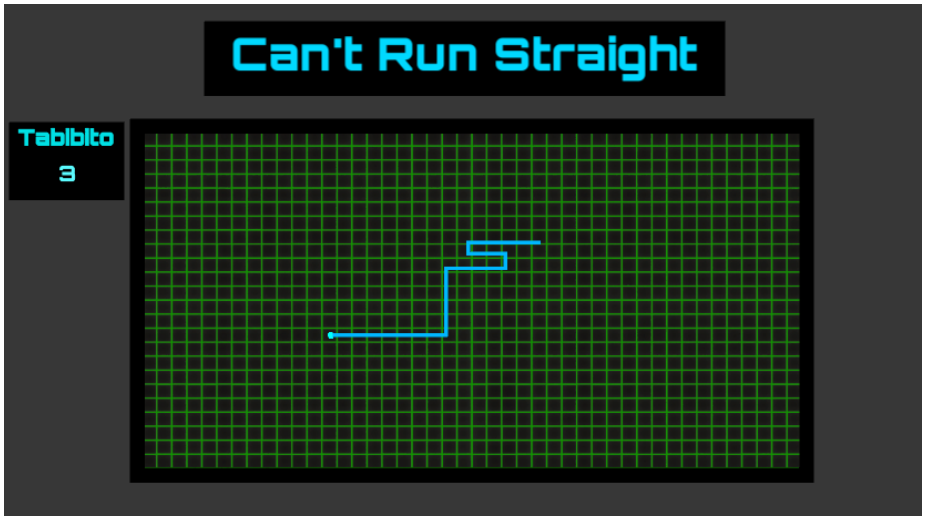


Figure 5: Can't Run Straight Final Version

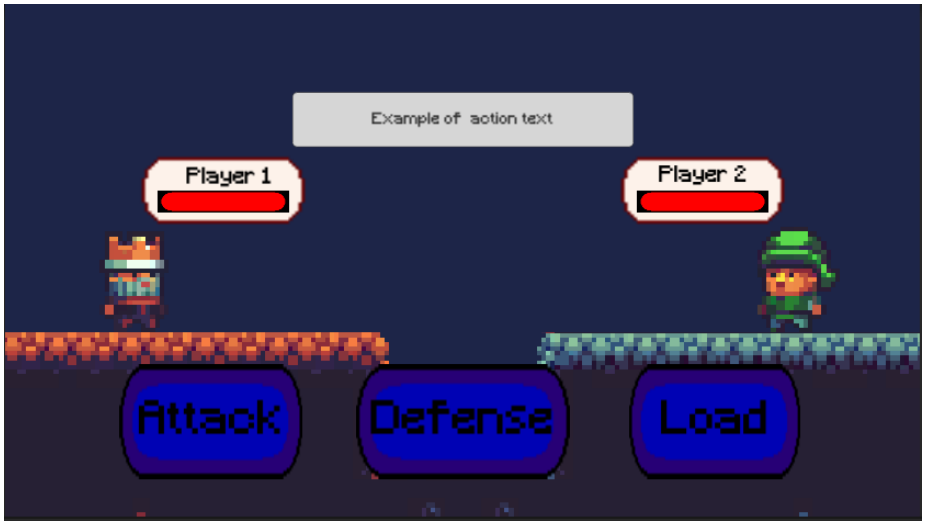


Figure 6: Focus Or Else Final Version

Figure 7: Spawn Colors Final Version

The following was mostly made by Dorian, backed up by Christophe

Figure 8: Main Menu

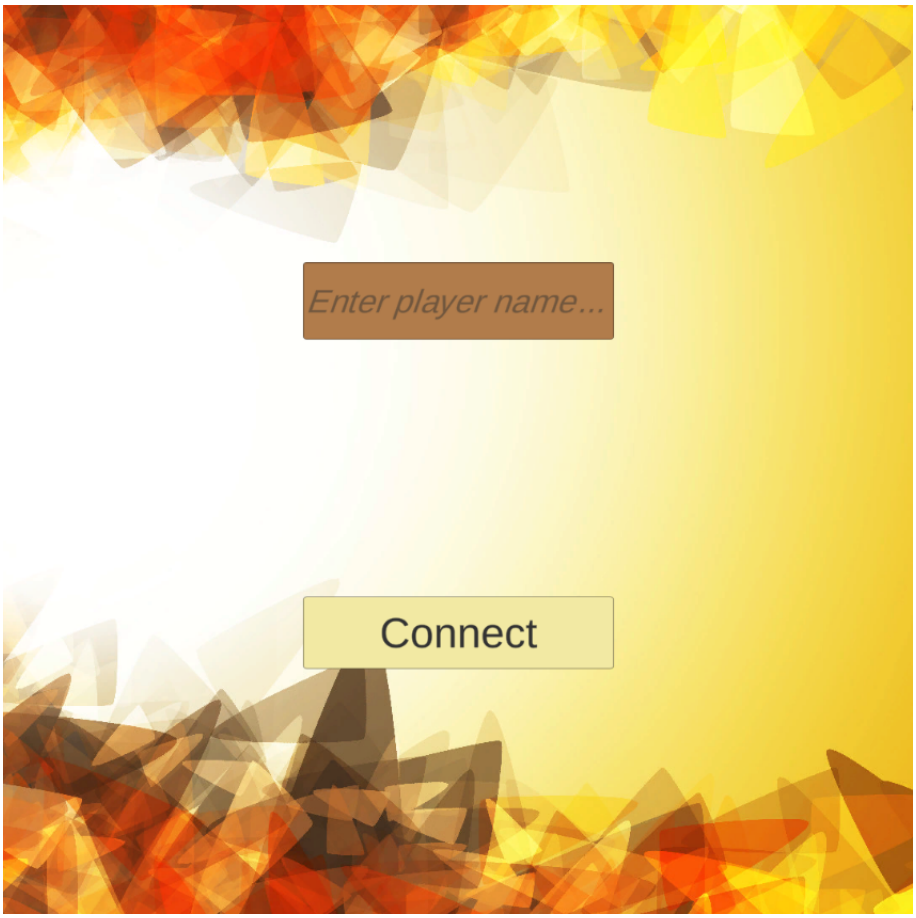


Figure 9: Connect Screen

Figure 10: Character Selection