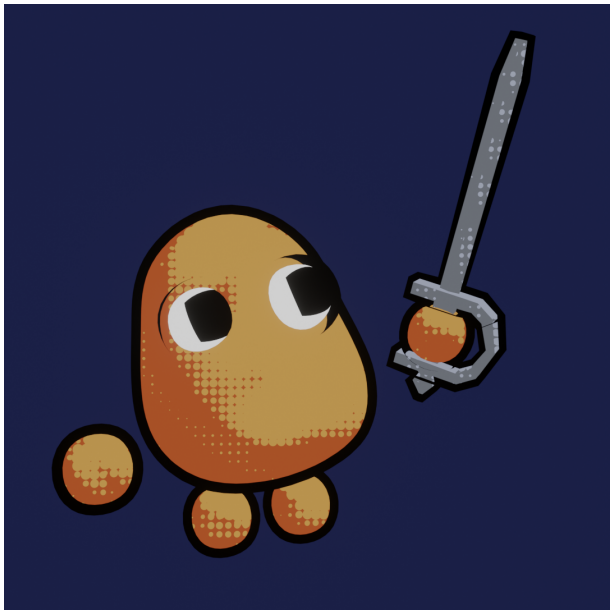
BattleFields
Return of the Potato



Julie Fiadino Paul Dufour
Auguste Charpentier David Goncalves

Prepa S2

Contents

Introduction

This project is a two players PC VR game named:
"Battle Fields: Return of the Potatoes".

The game is a Tower Defense: potatoes will spawn in different places and will try to reach the player's base. The players must prevent this and hold as long as possible.

Two players are required for this game :

- The **VR player**, with a headset and controllers. He has a bow, on top of a tower, and has to aim and shoot the incoming enemies.
- The **PC player**, with a mouse and a keyboard. He has a view from the top of the map and can purchase upgrades.

Both players have to communicate together to survive through the endless waves of potatoes, wanting to steal all of our resources.

Game Story

The game needs a story to keep the player aware of what is happening and the state of the world he is in. Here is it's current state:

The Potato Kingdom was going through a rough time. They were lacking the minerals necessary for their life. In an desperate attempt, The Potato King asked his neighbours for some of their goods.

The other kingdom, The Green-Plants Camp, was also trying it's best to survive. They decided to refuse the offer. The Potato King returned home, defeated.

Some months later, the green plants managed to gather enough resources. It wasn't the same for the Potato Kingdom.
The Potato King was angry. He felt betrayed. Thinking the Green-Plants Camp Leader lied to him, he declared war.

The Potato Army is now coming, and we need all of the archers ready for battle.

Evolution of the specifications

1.2.1 Workflow

1.2.2 Task Repartition

Leader **Contributor**

Task	Auguste Charpentier	David Goncalves	Julie Fiadino	Paul Dufour
Multiplayer				
VR				
PC				
Menu				
Map				
Entities				
3D modeling				
Music/SFX				
Website				

Figure 1.1: Initial Task Sharing

Task	Auguste Charpentier	David Goncalves	Julie Fiadino	Paul Dufour
Multiplayer				
VR				
PC				
Menu				
Map				
Entities				
3D modeling				
Music/SFX				
Website				

Figure 1.2: Final Task Sharing

1.2.3 Progress Forecast

Progress forecast (in %)			
Progress	Presentation 1	Presentation 2	Final
Multiplayer	60	80	100
VR	60	90	100
PC	50	80	100
Menus	20	60	100
Map	60	100	100
Player	60	75	100
Enemies	30	60	100
Longbow	90	100	100
3D modeling	60	80	100
Sound Design	0	60	100
Music	0	60	100
Website	50	70	100

Figure 1.3: Initial progress forecast

Progress forecast (in %)			
Categories	Presentation 1	Presentation 2	Final
Local Coop	20	80	100
VR	80	95	100
PC	20	70	100
Menus	20	40	100
Map	90	100	100
Gameplay	40	80	100
Enemies	20	70	100
Longbow	100	100	100
3D modeling	60	90	100
Sound Design	0	70	100
Music	0	60	100
Website	50	85	100

Figure 1.4: Final progress

Game Design

When starting the project, we needed to set clear rules that would define the game. Those are not really far from the ones described in the specifications, but they became more and more precise with time.

Main Goal

The game has an infinite duration. The goal is to last as long as possible through the endless waves of potatoes.
For that, we can find two different types of buildings:

- Towers, used to target the enemies
- Warehouses, used to store money resources

Warehouses are only for storage purposes, but if destroyed, the game is over. To protect them, we use towers. They will serve as shooting points for the archerer.
Those buildings must be placed very carefully to ensure victory as enemies will try to attack both of them.

Cooperation

To protect the warehouses, 2 players will share the different tasks: the Soldier and the Commander

2.2.1 The Soldier

The Soldier is the player with the VR headset. His goal is to shoot potatoes with a bow from the towers. With cooperation of the Commander, he can also set his arrows on fire to make even more damage. This position requires precision and accuracy.

2.2.2 The Commander

The Commander is the player on pc. He is the one in charge of strategy. His goal is to place towers and eventually additional warehouses to the map. He can also repair buildings and add torches for the Soldier, but with a cost. This position requires good management skills.

2.2.3 Money System

Money is earned by the soldier when eliminating enemies. It is then stored in the warehouses. If one of them is destroyed, all of it's money is then lost. This money can then be used by the commander to buy upgrades and new buildings

Visual Identity

2.3.1 3D models - *Julie Fiadino*

The models were done with the Blender application. Each object was created with a unique texture and the potato with a unique set of weapons. Textures were generated directly in Blender and baked to the models.

Figure 2.1: Potato Model

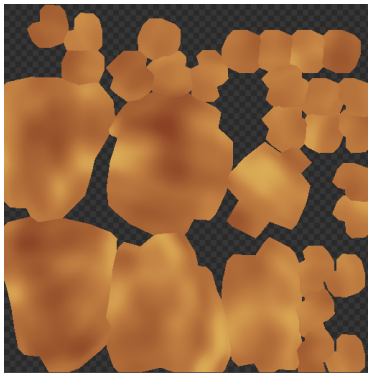


Figure 2.2: Potato Texture

A bone structure, commonly called a rig, is then created for the potato. This rig helps moving the character to create animations.

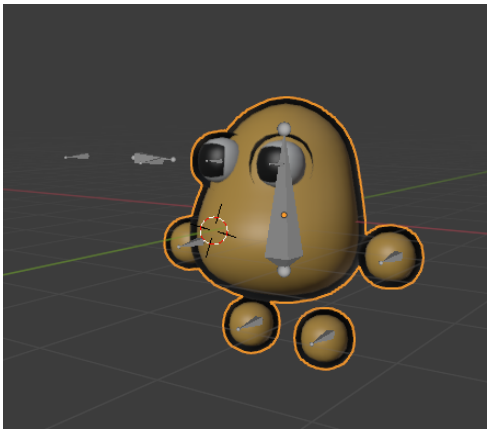


Figure 2.3: Rig of the Potato

The character comes with a set of animations We can separate 2 types of animations: state animations and weapon animations.

- State animations are those defining the state of the character. It is for example the walk cycle of the attack animation.

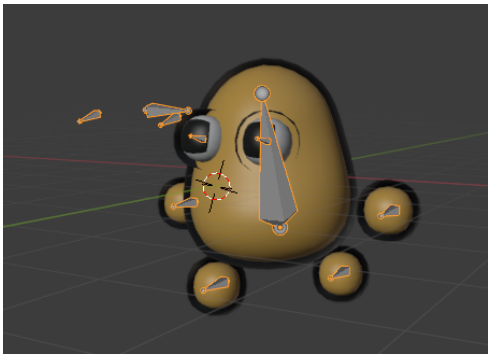


Figure 2.4: Potato walk cycle

- Weapon animations define how the character should grab an item. They are played simultaneously with the state animations to allow a vast freedom of movement.

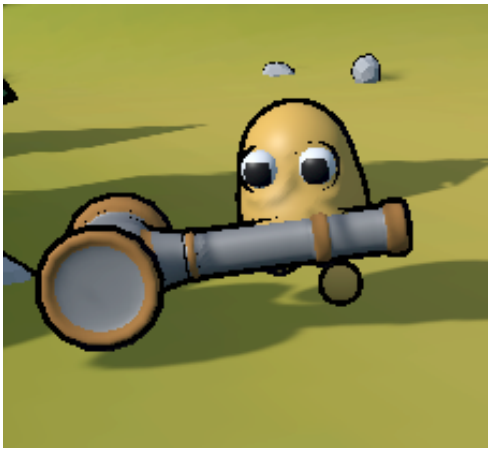


Figure 2.5: Potato with Hammer

Those models were exported in the .FBX format, which is the format used by unity. It allows importation of textures as well as the rig and animations.

2.3.2 Animation - *Julie Fiadino*

Animation integration

To allow simultaneous use of the animations, Unity offers an object called an avatar mask. This object allows to set different parts of the body that will be overwritten by the animation.

Firstly, an avatar must be created. It is a representation of the bone structure of the model. It is not generated by default and must be set in the import settings of the object.

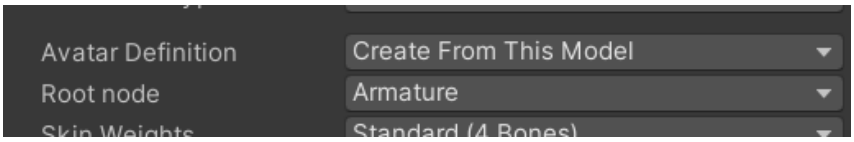


Figure 2.6: Settings of the avatar

Then an avatar mask is created for each weapon used by the character. The mask takes the bones used for the weapon, which corresponds to the bones that will be overwritten during the animation.

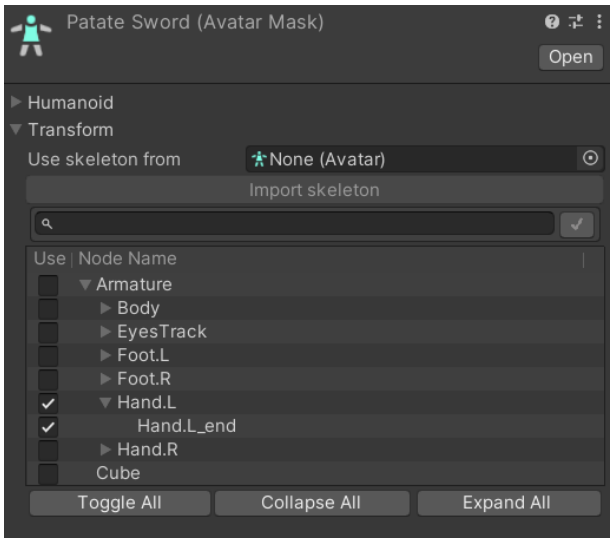


Figure 2.7: Settings of the sword avatar mask

Those masks are used in the animator controller, which will manage and play all the animations. They are used to create layers which come with a weight factor that defines how much the animations will be overwritten.

Figure 2.8: Differents layers of the Potato

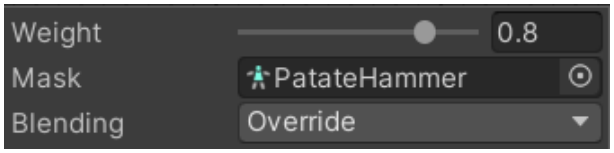


Figure 2.9: Hammer layer settings

All the animations are then connected and the transitions are limited with parameters such as booleans and triggers.

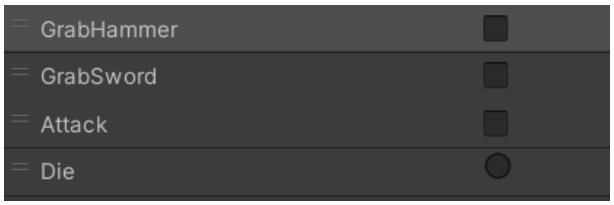


Figure 2.10: Differents parameters of the Potato

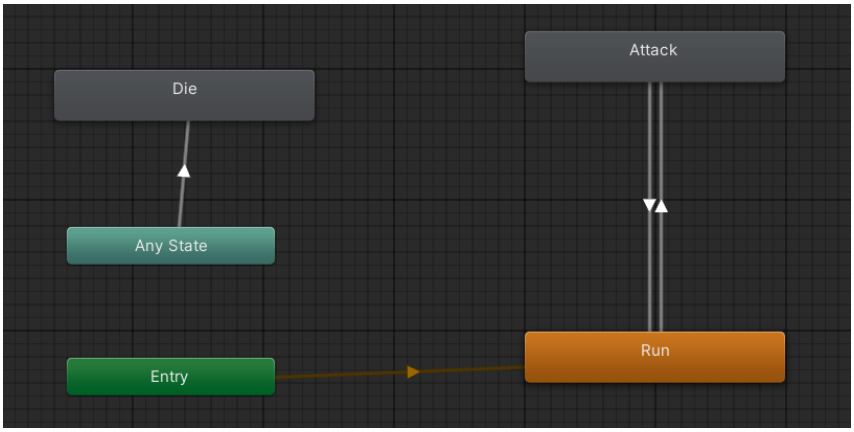


Figure 2.11: Base animation layer

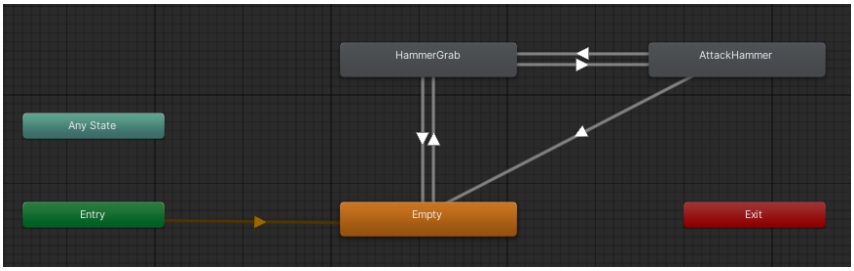


Figure 2.12: Hammer animation layer

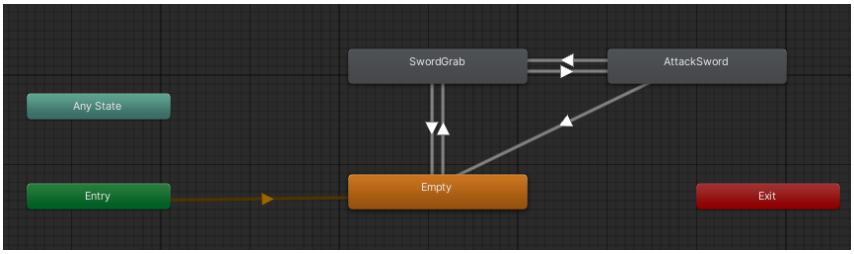


Figure 2.13: Sword animation layer

Animation Controller - *Julie Fiadino*

To control the animations, specifically potatoes that are the main AI of the game, the prefab is used with a StateManager.

The script contains the definition of an `enum WeaponType`, which lists all weapons available. This enum is used to know the current weapon type and which animation to use.

The main class is then composed of 4 public functions:

```
/**
 * <summary>
 * Start Attack animation
 * </summary>
 */
public void StartAttack();
```

```
/**
 * <summary>
 * Stop Attack animation
 * </summary>
 */
public void StopAttack();
```

```
/**
 * <summary>
 * Change weapon type and load animations. Also instantiate said weapon as a
 * children of the prefab
 * </summary>
 *
 * <param name=weapon>The new weapon that needs to be set</param>
 */
public void ChangeWeapon(WeaponType weapon);
```

```
/**
 * <summary>
 * Deactivate all states and start death animation
 * </summary>
 */
public void Die();
```

Those functions can then be called by the behavior script of the AI.

2.3.3 Skybox - *Julie Fiadino*

To create an immersive environment, we needed to change the default unity background. That's when we started creating a skybox.
This skybox is set as a background and allows the game to change its settings.
The created skybox uses unity's node editor. It allows creation of shaders (programs compiled by the GPU) which include skyboxes.
The one used in the game has multiple parameters such as sky and sun color, cloud density and speed, stars quantity, etc...

Figure 2.14: Skybox settings

This skybox is then used as a texture and pasted onto a sphere

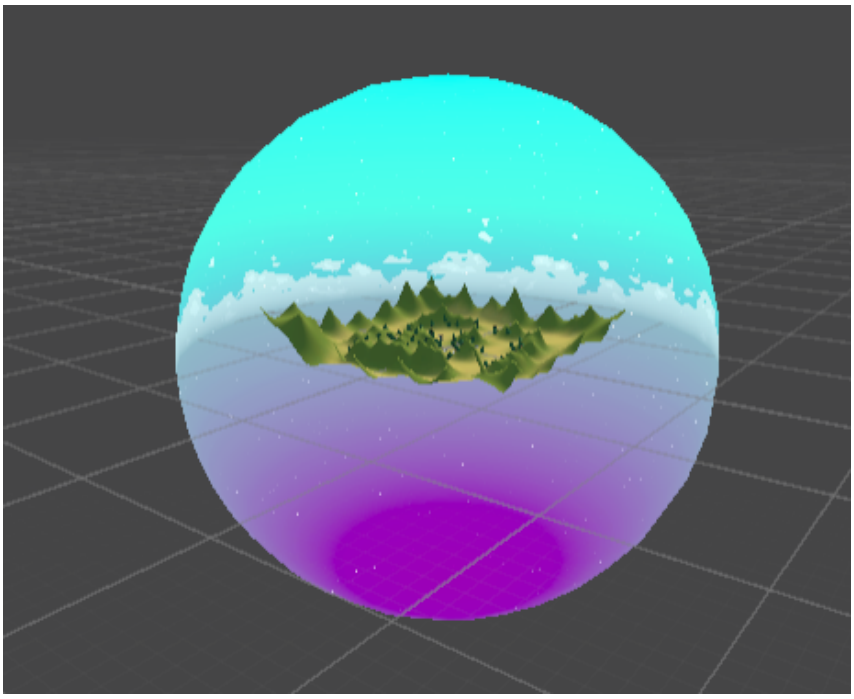


Figure 2.15: Skybox Sphere

The last thing needed, is a day-night cycle. The properties of the skybox can be accessed by script, which allows the creation of a time management system.
The skybox will then alternate between 3 states as time passes:

- Day
- Noon (being used to transition between night and day)
- and Night

Figure 2.16: Our game at daytime

Figure 2.17: Our game at noon

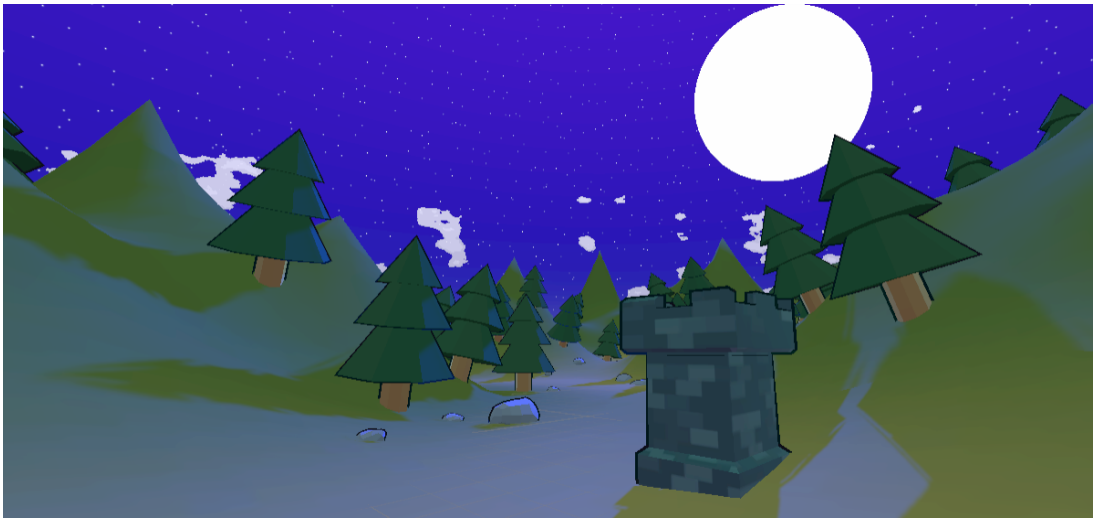


Figure 2.18: Our game at nighttime

2.3.4 Effects, Fire and Explosions - *David Goncalves*

When something gets burned, something is required to show this. This is where effects comes in.To make effects in Unity, we first create a ParticleSystem.

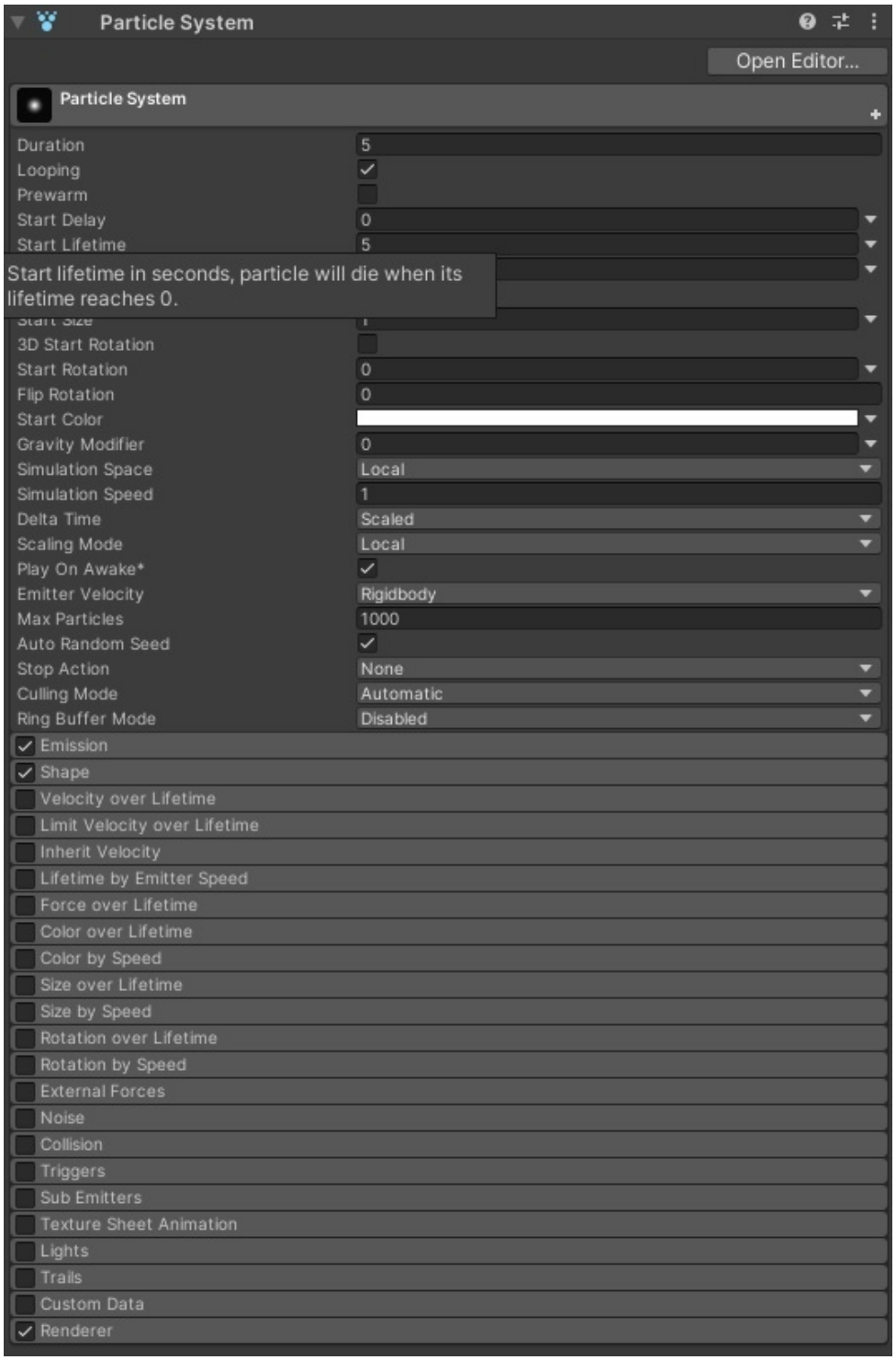


Figure 2.19: Particle System Menu

From a lot of of things can be changed to create effects like the Red fire from SteamVR that is used to indicate fire on everything that is on fire.

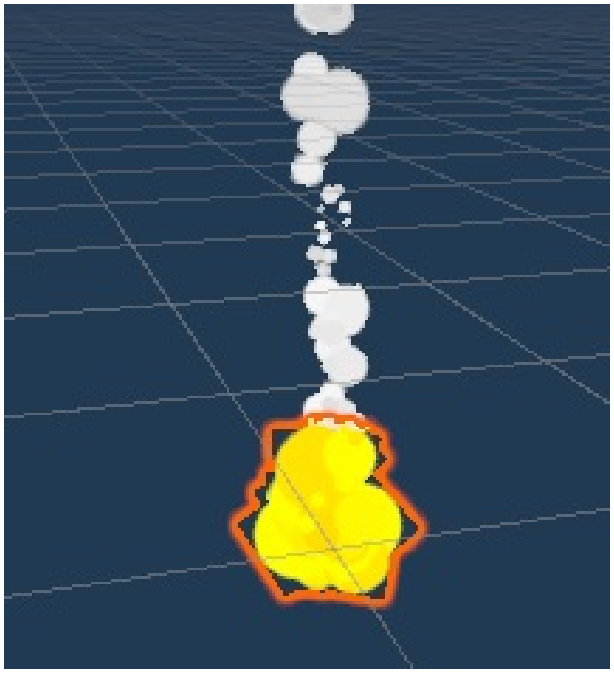


Figure 2.20: SteamVR's Red Fire effect

An effect is as well used to show explosion. That's where a hand maid effect using inspiration from the Red fire effect was made.

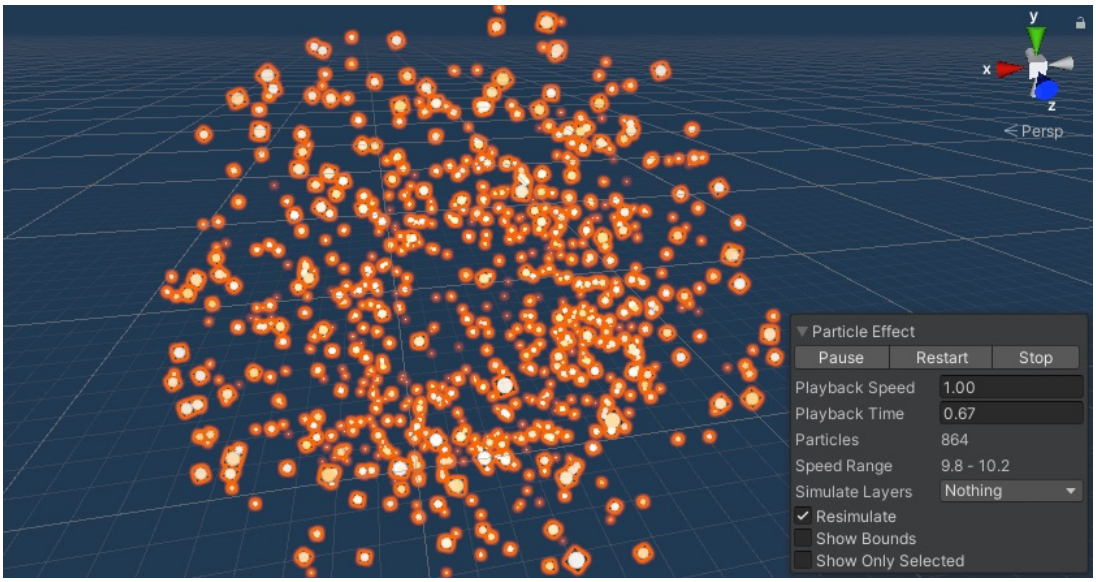


Figure 2.21: Explosion effect

Furthermore, if a potato is on a fire or die from an explosion, fries get launch in the air.

Figure 2.22: Flying Fries

2.3.5 Drawings - *David Goncalves*

For our project, multiple drawings where made.

The first one is for the menu and loading screen of the game.

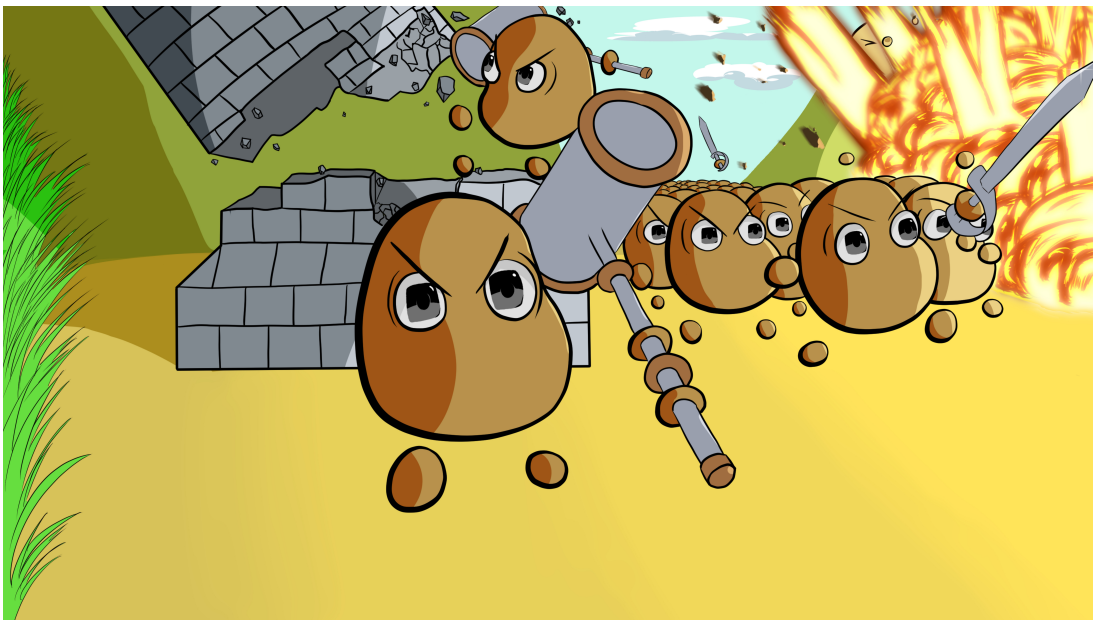


Figure 2.23: Menu Background

This one was for the web site.

Figure 2.24: Website Welcome Drawing

Here is the company logo.
And this one for the shop keeper.

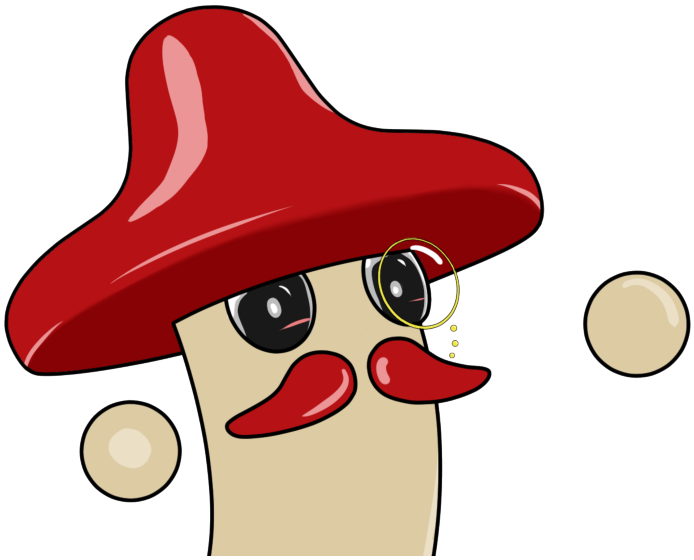


Figure 2.26: Mushroom Shop Keeper

As a small bonus, here is the redrawn texture of the grenade (which will be presented later)

Figure 2.25: Wyvern Logo

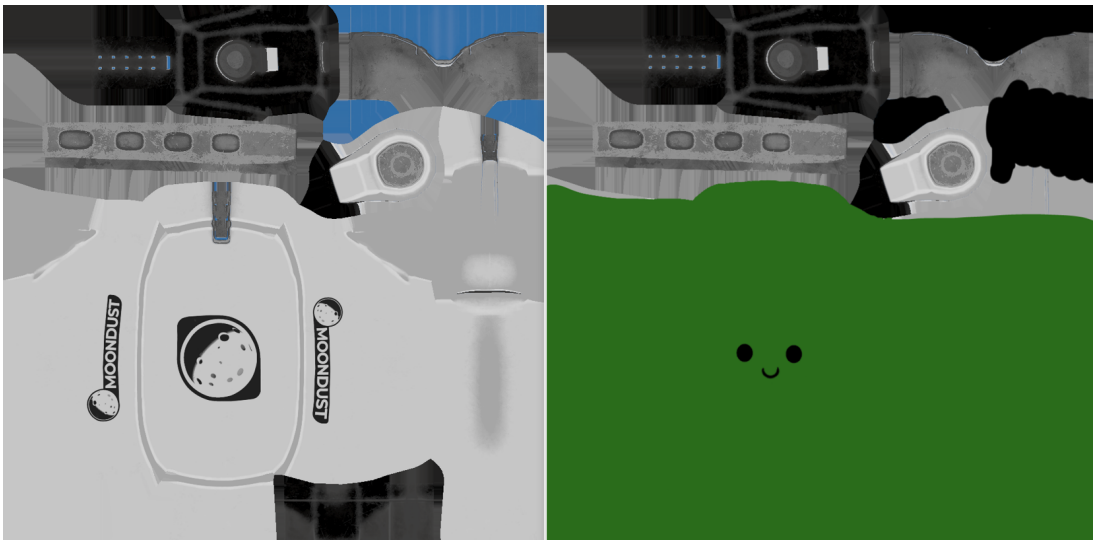


Figure 2.27: Old texture vs New texture

All those pictures are the first time I really draw with colors and I'm a quite proud of what I was able to achieve during this project. By providing drawings to the game, I upgraded my skills, learned how to make textures and was able to enjoy a lot the process.

Virtual Reality

The first player, namely **The Soldier**, plays in VR and must aim a bow to shoot the incoming ennemies.
However, you can't just snap fingers and have a VR-ready game, it took a lot of time to think about how to integrate gameplay elements in virtual reality, with the new possibilities offered.

Developing a VR game

VR in gaming has started to become serious with the release of the HTC Vive in 2016, that's also when the development of VR games started to become more common.
Valve, the company behind the Half-Life and Portal games, and the game launcher Steam, invested in VR development with SteamVR, which allowed us to present you this game.

We are therefore using the SteamVR for Unity package, providing us with all the necessary tools and examples to interact with.

3.1.1 The design phase

Before coding and implementing stuff, we had to take some time and think about what we should do. We already explained the design of our game in general, but I want to explain how we came to this result.

First, we wanted to play some of the most original and interesting VR titles, understand how and why they work:

- **The Lab:** Made by Valve for the release of the HTC Vive, it's a collection of small VR experiences. Not only is it new and fun for players, it's also inspiring for us, developers. One of the mini games we really liked is the longbow one, where we shoot ennemies with a bow.
- **Keep talking and nobody explodes:** A game that was ported to VR where one player has a bomb in his hands, in game, and the other player(s) have a bomb defusing manual, in paper. The players therefore have to communicate well to complete the game.

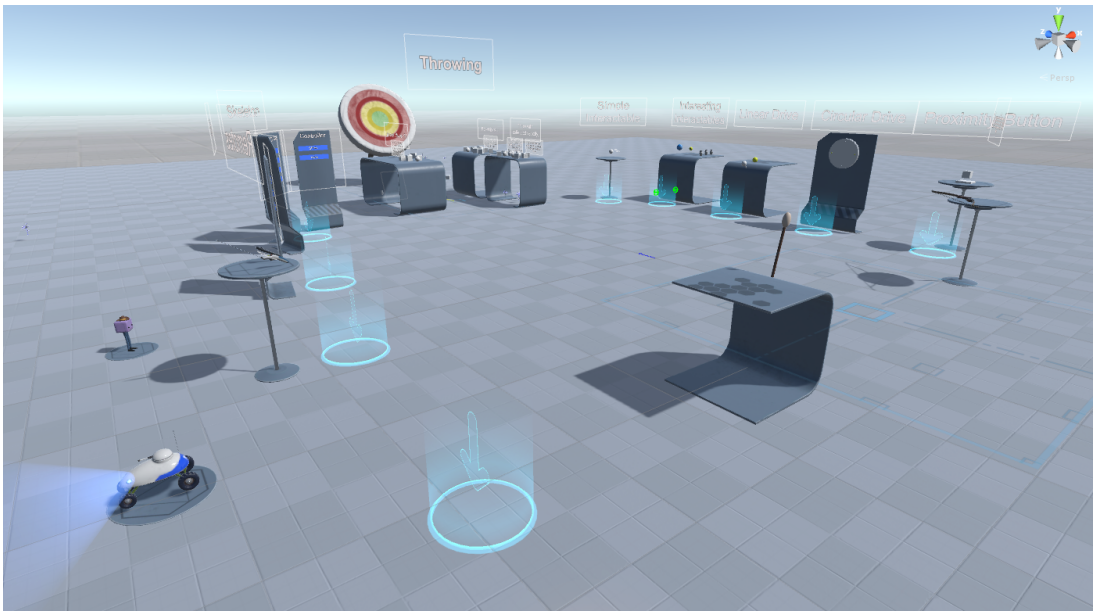


Figure 3.1: SteamVR interactions showcase scene

Battle Fields' initial idea is inspired from these gameplay elements.

We also found inspiration from a tool provided by Valve, an example Unity scene featuring different interactions in VR, with open code and assets (figure 3.1).

3.1.2 Coding in VR

Once the ideas are clear, it's time to develop the game.

At first, development was surprisingly **easy**, thanks to the tools available!
Dragging the Player object in a scene made **everything ready** for VR, with initialization, tracking and movement, hands and interactions, all of that **without an issue**.
More objects were available and could be added seamlessly, like a teleportation system, or some example "interactables" objects.

However, when we wanted to customize some behaviour, we had to **understand** the code and **modify it**.
This is probably the main thing that differentiates this game from the others in the way it was coded: instead of coding everything in our scene, we had to **rely on existing code**, understand it, and modify it without breaking it when necessary. Most of the time spent coding features was not writing but **understanding** others' work.

3.1.3 The issues we had to deal with

VR doesn't benefit from the same maintenance as Unity in general. The plugin we're using wasn't frequently updated, and we experienced some bugs.
Most notably, we encountered the following problems:

- The skybox component could not be rendered. This was a low-level problem, as the skybox was not just black, but it took the content of the previous frame, creating a weird effect. We solved this by using a big sphere instead.
- The Unity editor randomly decided not to detect SteamVR. During the first presentation, we could not launch the game because of this and didn't know the fix, which is to untick and tick a box back.

Dealing with the problems and fixing them took us a lot of time, which makes us even more proud to present you this stable game!

PC player

Movement across the map *Paul Dufour, David Goncalves*

The Commander is able to move across the map by draging the screen in the direction he wants to move. This operation can also be done with the directional arrows.

He is also able to zoom in and out with the mouse wheel.
This permits him to have a full view of the map, and prevent enemy attacks.

Money System *Paul Dufour, Julie Fiadino*

For our game we have a shop part where you can buy things, however, no payment part existed.

So we decided to add warehouses. These warehouses will be used to store money and the towers will be in charge of protecting them.

We start the game with 20 French fries.
The only way to win some afterwards is to kill potatoes.

- 2 for normal potatoes
- 3 for the potatoes with swords
- 4 for potatoes with hammer

Figure 4.1: WareHouse model

Each warehouse can contain 100 french fries. They can then be upgraded for 20 French companies, which allows to store 20 more potatoes[1].

We start the game with one warehouse. Then the player has to manage his money to repair the turns and manage his money (the fries)

4.2.1 The prices

A kill can bring you

- 2 for normal potatoes
- 3 for the potatoes with swords
- 4 for potatoes with hammer

You will need 15 French fries to add a tower and 25 to add an warehouse.
The towers can be improved :

- a torch for 2 potatoes
- repair 20 points of life of the tower for 10

For the warehouse, we can

- improve the tower for 20 French fries
- repair 20 points of life of the tower for 10

[1]Did you known : the history of the potato (Solanum tuberosum1) begins with that of the Amerindians who lived more than 10,000 years ago2 in the coastal area of present-day Peru and southwestern Latin America.

Building placement

Paul Dufour, Julie Fiadino

To indicate where to place buildings, we decided to make a grid that covers the entire map up to our game borders.
Each possible tower slot will therefore be a tile on that grid.

To do that, we made a texture to show the different tiles:

1. In green the ones available
2. In orange, the one unavailable
3. In grey the borders of the boxes

Here is the result of the texture once applied to the map.

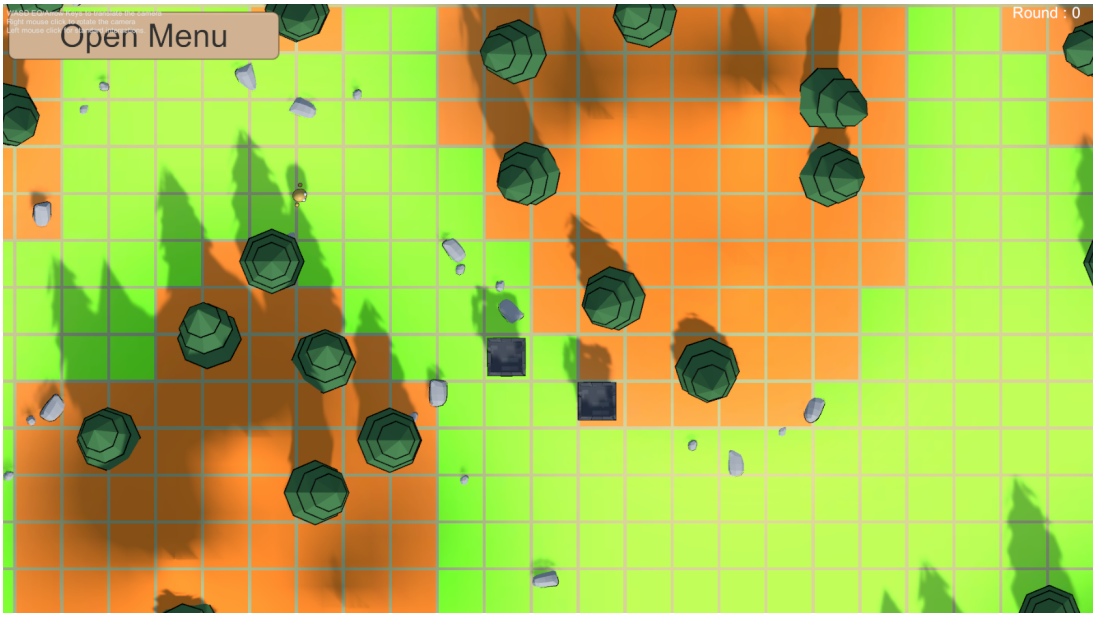


Figure 4.2: Grid for tower placement

All that remained was making sure that the buildings were placed correctly on the map according to the grid.

In practice, it was a little more complicated. We needed to calculate the position of the tower on the grid according to the mouse position.
We then had to define the tiles with mountains or rocks, to prohibit any placement.

Moreover, we made the warehouses take 4 slots on the map, in order to let the placement be more difficult for the player.

The menu

Paul Dufour, Julie Fiadino

The menu is made of 3 different tabs:

- The general tab
- The tower tab
- The warehouse tab

4.4.1 General tab

From this tab, the Commander is able to add towers and warehouses. He can also view the current wave and the money stored in all of the warehouses.

Figure 4.3: General tab

4.4.2 Towers tab

The towers tab allows us to preview the stats of each tower on the map. From this menu, the Commander can add a torch to the tower (which goes up to 3 torches) or repair the tower. But each of those operations will cost money.

Figure 4.4: Towers tab

4.4.3 Warehouses tab

The warehouses tab is quite similar to the towers tabs. Both allows to preview each building on the map. Each warehouse is also described by a french fries bar, indicating the amount of money stored in the warehouse.

Figure 4.5: Warehouses tab

4.4.4 Animation

We then tried to implement the animations allowing the different elements to appear on the scene.

For this, we used the "Animator" tool in unity:
There are two stages:

1. Panels out of the player's view

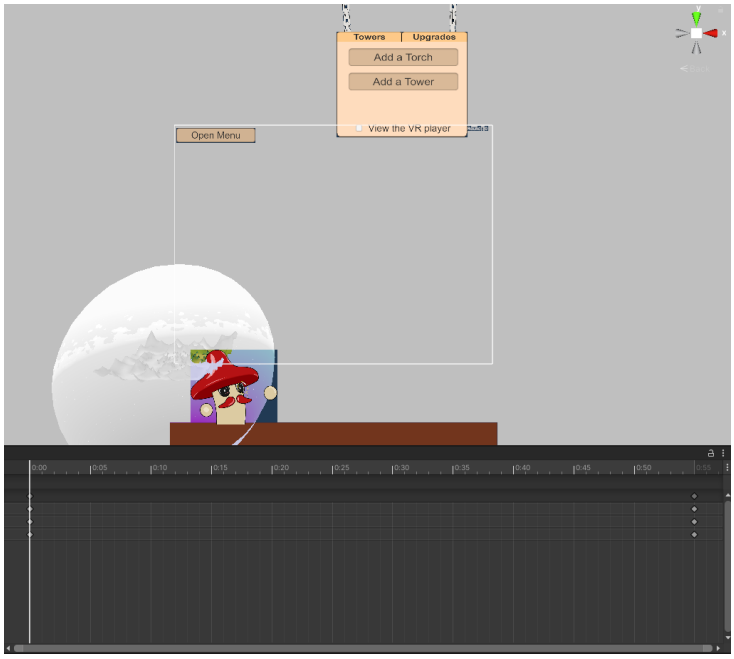


Figure 4.6: First animation state

2. Everything is displayed with a moving transition (when the shop is active) :

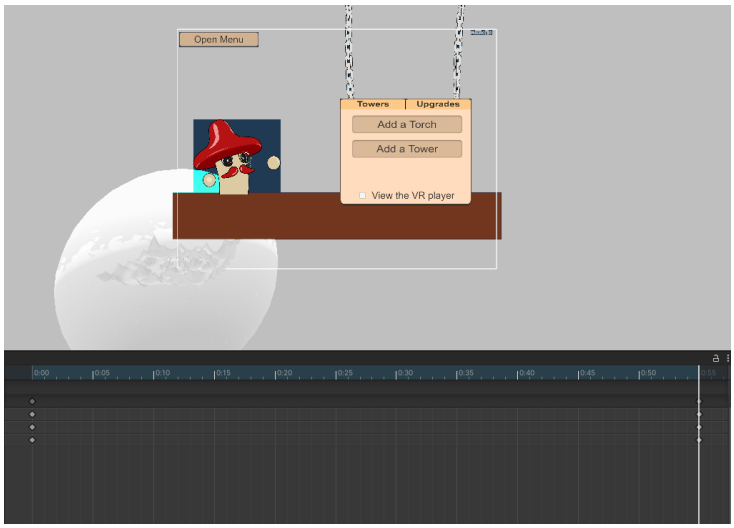


Figure 4.7: Second animation state

Then we had to manage the animator, with a boolean variable to tell if the menu is open or closed and the trick is done.

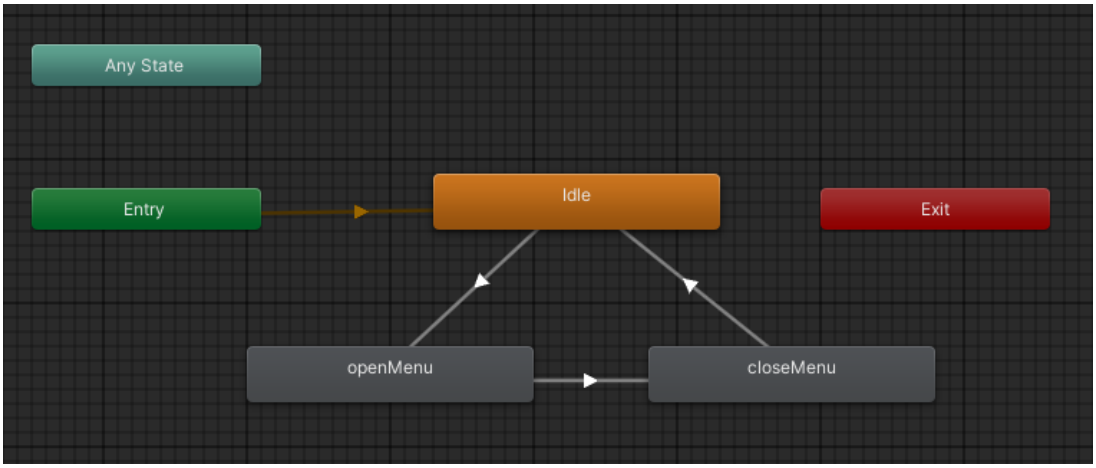


Figure 4.8: The animator

Enemies and Weapons

Artificial Intelligence

David Goncalves

5.1.1 Movement

For the AI, Unity's NavMesh is used. It allows us to easily define the space our enemies (potatoes) will be in and move in.

The enemies will spawn randomly on defined spawn points, then will choose a target between the one that are available depending on which one is closer.
Then Unity will find the shortest path, taking into account the obstacles and height of the ground (see figure 5.8).

Figure 5.2: NavMesh Zone in blue

Figure 5.1: NavMesh pathfinding

Once set, the potatoes will find the fastest way to access the target. If a building appear and it's closer than the one they were aiming for, the potatoes will change target. It as been made sure that potatoes don't overlap with each other or with constructions.
When they are close enough, the potatoes will attack the target.

Each wave will spawn after a certain amount of time; the higher the number of rounds is, the higher the numbers and power of the enemies are.
In terms of stats, the towers contain only health while the potatoes have attack damage, health and attack rate.

5.1.2 Types of enemy

There is three types of enemy:
- No weapon potato : simple potato that deal low damage and have only one HP.

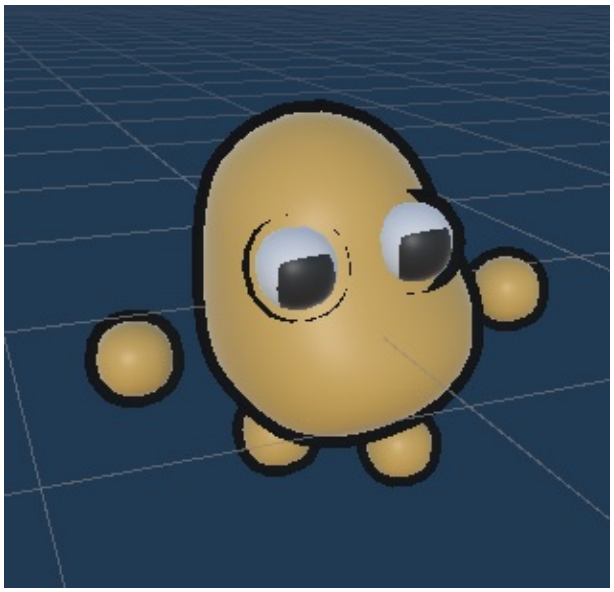


Figure 5.3: No weapon potato

- Sword potato : a potato with a sword that deal medium damage and have two HP.

Figure 5.4: Sword potato

- Hammer potato : a potato with a hammer that deal high damage and have three HP.

Figure 5.5: Hammer potato

Weapons *David Goncalves, Auguste Charpentier*

5.2.1 Bow - *Auguste Charpentier*

The bow is the main weapon of the VR player. With the simple assets of SteamVR, the weapon is ready to go.

Figure 5.6: Bow

Still, it doesn't do damage to potatoes. With the use of a collider for the potatoes, I modified a function called ArrowCollider that normally cause balloon to explode to deal one point of damage to the enemy it hits. Furthermore, because the arrow bounce from collision, it can ricochet.

5.2.2 Fire arrow - *David Goncalves*

When there is a torch on a tower, it is possible to put a arrow into it and create a fire arrow. It supposed to work the same than the normal arrow but only add a burning effect to what it hits. The fire can as well spread itself to targets around it and so create a magnificent barbecue.

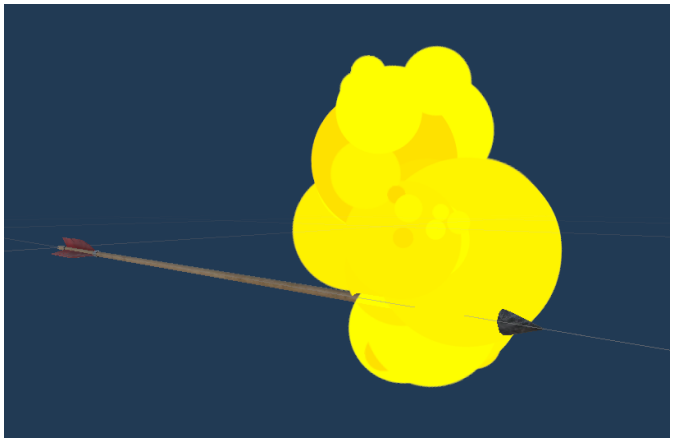


Figure 5.7: Fire Arrow

The reason why it only apply a burning effects is that if the targets lose one HP when hit

by a burning arrow and the enemy have only one HP, it simply kills the enemy without spreading the fire.

5.2.3 Bomb - *David Goncalves*

SteamVR has a grenade asset that has been used for making the bombs. But differently to the bow, there had to be a lot more of added code to make it work, as the original grenade only spawned flowers.

Overall, when a grenade have an impact with the environment or a potato (not the towers nor the player), they explode. Everything in a certain radius of the impact get killed or burned.

Figure 5.8: Grenade

5.2.4 Collision and Pain - *David Goncalves*

I must point out that understanding collisions in Unity was way harder than it should have been. Normally, when Unity detect a collision, a Collider is given. From this Collider,

we can do Collider.gameObject to find which object the collider refers to. And from this gameObject we can do gameObject.CompareTag(tag) to check if the object have a tag and know if we want to interact with the object.

But, for some reason, it does not work. What you need to do is the Collider.collider.CompareTag(tag) and then it work.

And it frankly feels stupid to search for the collider of a collider, and alas, there is more of them. A call to a SteamVR function that work, until for no reason, it simply doesn't exist anymore for Visual Studio, and so for Unity. A code that has an OnColliderEnter (use for detecting collision) that just doesn't work, but if you do another script and copy and paste the same code, it works.

Overall it really slow down the process of creation and forces you to go through some length you feel should not be reaching. But still, once you "understand" how it work, it's very satisfying to see the result.

Website

There are a lot of options to do a website. But we wanted to be creative. So we decided to make the website by hand, without Bootstrap, or Wordpress, but just with **REACT**.

React

Paul Dufour

React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.

With React technology, we can first we created components : headers, footers, cards, button, ...
Then we made the whole site responsive.
Once the base was created it only remained to add the content of the site.

6.1.1 Component

React works with the component, so I have created a few components like headers and footers.
For the Author page, there are only four times the card component to display properly our different names , roles and photos.
For the credit page, We also have made a card component, with 4 settings : a logo, a website, a name and a tag.

6.1.2 Responsive

We wanted our site to be entirely responsive.
We have used a lot of my time, just to try resolve problems of the header scale, menu

scale, or footer scale. We finally decided to make a button if the scale of the screen is too small to resolves all the problems. When you click on the button, a full size div should be shown.
The hardest work was to make the credits part work properly. Indeed the number of elements displayed per line depends on the width of the page. We solved this problem by creating multiple divs that allow to place the elements separately.

The different pages : *Paul Dufour*

6.2.1 Main page

The main page contains the most important information: a link to download the game, a short presentation and also our different project reports.
The different parts of the page are separated by waves.

6.2.2 Info page

We have made a gameplay presentation page where we explain the roles of the different players
First there is the role of the person in VR: **The Soldier**, then the person on the computer: **The Commander**

The Soldier

Soldier, you should be proud! You will be leading our nation to victory! With your bow, shoot at the enemies from the towers. Be careful, they will try to destroy them.
We asked one of our greatest commanders to help you with this task. Protect the warehouses at all costs, or the kingdom will be doomed!

The Commander

Commander, we need your help in this battle. Use our resources to build towers and warehouses to sustain the troops. Place them intelligently and the victory is ours.
The shopkeeper will always be here if you need help. He can hire forces to help us maintain those towers alive. I heard he can also provide some useful gadgets.

6.2.3 About Us page

This page contains all the project members, with a short description, and their roles for this project.

6.2.4 Credit page

The credits page contains all the tools we used for this project, with their logo and a redirection on click.

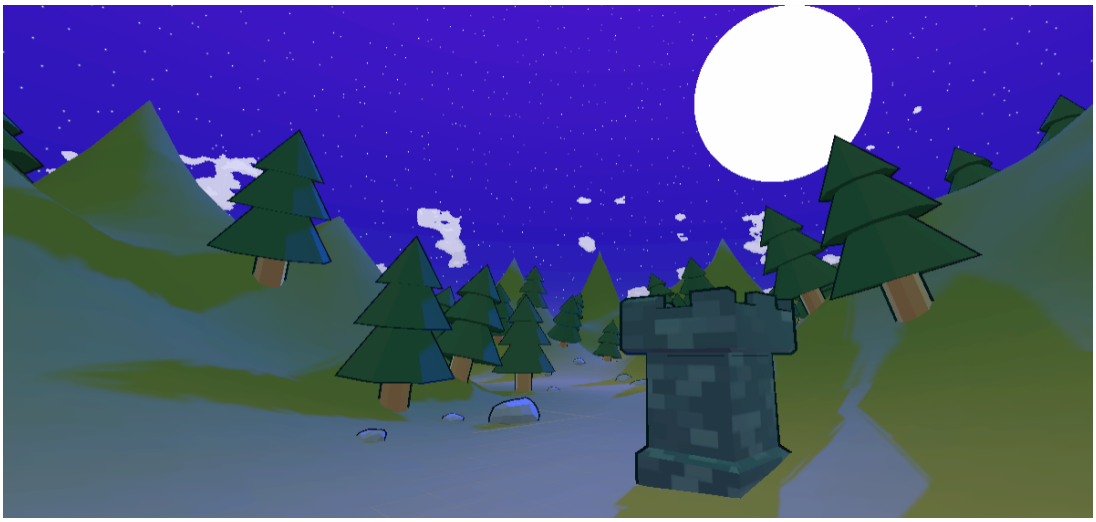


Figure 6.1: Our game at nighttime

Conclusion

The game is starting to look good now!

Provided our game doesn't crash once again during the presentation, we are very satisfied with the progress we have done.
The game has now reached a functional state, it is mostly working with the flagship features we want.

Of course, there is still something big remaining: fun. The most important part of creating a game is the final one, details and little things that will make the game fun to play.
Interface is very basic, the stream of waves is continuous, there are almost no upgrades possible, overall the game is bland.

The goal from now on is to find fun things to add. Upgrades, ammunition and money management, tinkering the UI, game and level design adjustments...

Publishing the game to Steam as a free title is still considered! Last in our priorities, but an achievement we would be proud of.
It's quite impressive to see what we have done

Progress forecast (in %)				
Categories	Presentation 1	Expected Progress	Final	Current Progress
Local Coop Paul/David/Auguste	20	80	100	100
VR Auguste	80	90	100	100
PC Paul/Julie/David	20	80	100	100
Menus David	20	40	100	100
Map Julie	90	100	100	100
Gameplay Paul/David/Auguste/Julie	40	75	100	100
Enemies David	20	60	100	100
Longbow Auguste	100	100	100	100
3D modeling Julie	60	80	100	100
Sound Design Paul/Julie	0	60	100	100
Music Paul/Julie	0	60	100	100
Website Paul	50	70	100	100

Figure 7.1: Current advancement and expected progress