Bartendu

Second report

Jean Bou Raad
Lou Lefebvre
Vincent Thirouin
Emeline Tichit

EPITA 2024

https://areas0.github.io/website/

Contents

Introduction

This second report will serve as a follow-up for the book of specifications and the first report, as well as a description of the tasks that have been carried out since the hand-out of the first report. Difficulties encountered and chosen solutions will be discussed.

As a reminder, *Bartendu* is a cooking bar simulation game in the making.

Les Cosmopolitains have chosen to undertake this project in C# on Unity. Other software and tools have also been carefully selected such as Blender.

The goal for players is to make drinks and cook to satisfy customers in a bar. Room (restaurant) management is also at the core of the gameplay.

For the second stage of our project, we aimed to produce more features while keeping in mind stability and quality. That means that we have finished singleplayer's mode core and introduced new gameplay.

1 General progression

Task	**Expected**	**Completed**
UI graphics	50%	75%
UI implementation	60%	75%
Character actions	90%	90%
Multiplayer	90%	90%
AI	70%	45%
Tutorial	0%	0%
Level design	60%	60%
Level implementation	40%	40%
Object implementation	45%	55%
Particle Effects	10%	60%
3D modeling	60%	70%
3D animation	50%	50%
Music	50%	60%
Sound Effects	15%	0%
Writing (Story-line)	10%	50%
Website	90%	90%

2 Vincent's Achievements

My part during this second period was to implement the singleplayer. I also did some more work on the multiplayer and the overall aspect of the game but those are almost insignificant comparing to the amount of time spent on implementing the singleplayer. When we first started the project, everyone told us that we had to start with the multiplayer first. . . And they were right. However, I did not expect to spend a huge amount of time on it and thus I was not prepared.

2.1 Singleplayer

2.1.1 From Multiplayer to Singleplayer

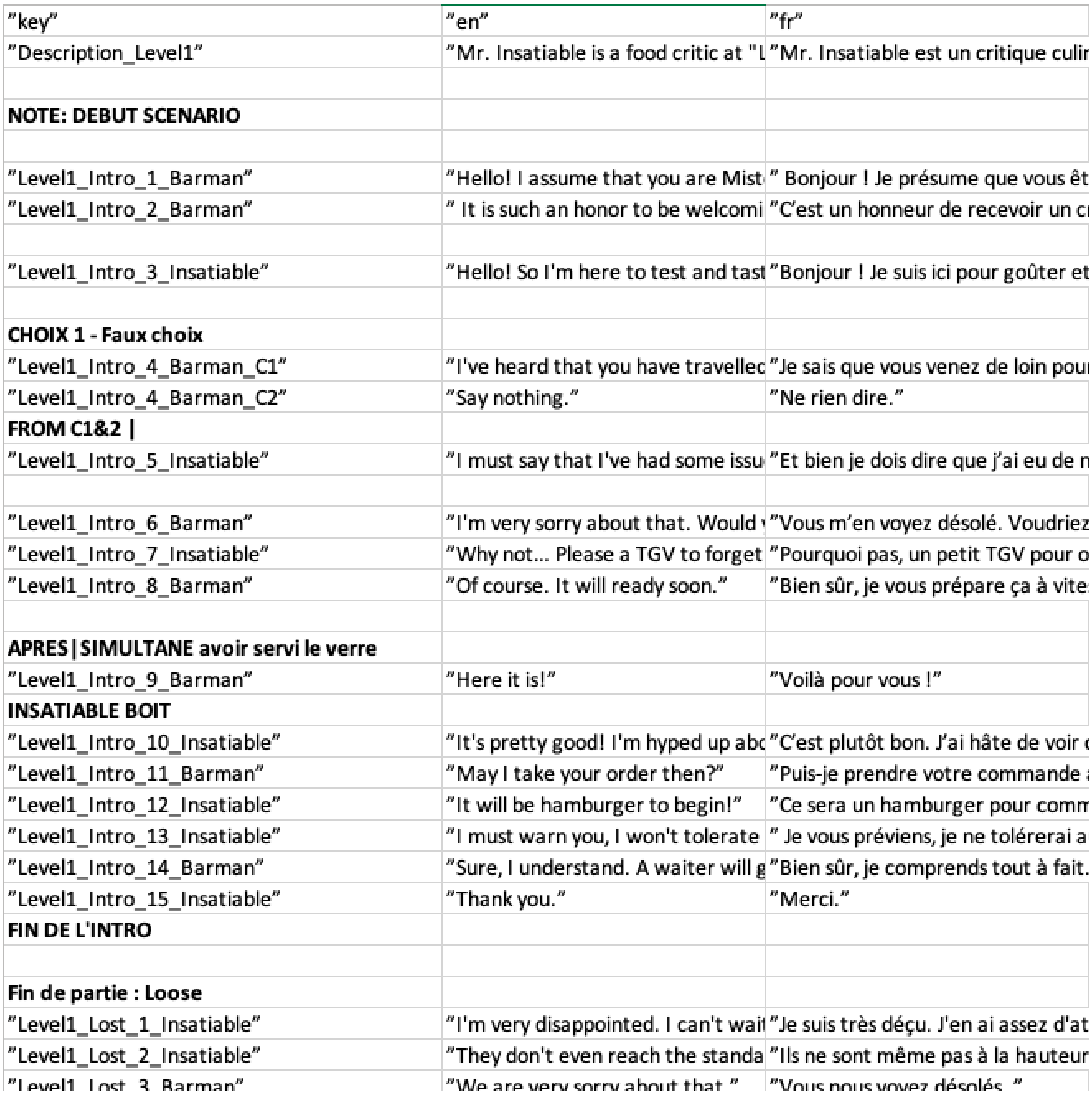
One would think that it is not very hard to make this change. But in fact, the problem was not the difficulty of the task, but rather the time it required. All scripts had to be partly rewritten in order to make sure that the scripts did not try to send data through the server. Moreover, each script was attached to an object, and this object had Photon Network's components. This component was responsible for sending, and receiving data via RPC. Thus, besides having to rewrite all scripts we also had to recreate every single asset of the game without network components. But that was not all, we also used scriptable objects to handle recipes and food properties. Those had to be recreated as well to update the asset variable they contained. After all this tedious work it was time to finally start working on one big aspect of our singleplayer: cinematics.

2.1.2 Cinematics

Our goal here is to create cinematics to immerse the player inside the story. In this part I will only talk about my work, the technical part. Jean will later tell, in his part, how the story was created. Instead, I am going to talk about the localization system, the dialogue management and the camera work.

The localization system
But what is a localization system? It is a script responsible for handling the different languages your game can handle. Indeed, when we decided that we wanted to have cinematics at the beginning of every story, it was important for us to make that cinematic as accessible as possible. We thus decided to create a localization system capable of handling English and French. I will now quickly explain how it works: I wanted to make this tool as easy to use as possible so that Jean would not have any problem inserting the text into the game. One fast and easy way to do it is to use Excel. Indeed, its interface is very easy to use and comprehend. On the leftmost column we would put the identifier key, the one Unity would understand and look for when trying to display a text. On the second column would be the English version and on the third the French version. Now having all this written in a block is not very readable. . . We had to find a way to say which text is a text to integrate into the localization tool and which text is just a header or commentary. For that, we decided to use a special character, the tilted double quote. As soon as the entire Excel document is formatted, we can then proceed to add it to the project. One problem, Unity does not accept ".xls" files instead it does accept ".cvs" files which are perfect because Excel allows us to transform a sheet into a CVS file. In short, a CVS file is a file where every cell is delimited by a comma. (CVS stands for Comma-separated values)

Now that we have a CVS file, we have to be able to read it and then create the tool. Reading the file was not a problem. We used REGEX expressions (**Reg**ular **Ex**pression). It was useful to determine which text was a header or a comment and which text we wanted to save. This REGEX just filtered in only expressions written between the particular double quotes I talked about a few lines before. When getting those texts and we put them in the right dictionary: either the English one or the French one. The text value goes in with its according key. For example: (greetings_intro_1, "Hello World"). Then we just have to call the script and ask for the text for the key "greetings_intro_1" and we get "Hello World".



"key"	"en"	"fr"
"Description_Level1"	"Mr. Insatiable is a food critic at "L	"Mr. Insatiable est un critique culir
NOTE: DEBUT SCENARIO		
"Level1_Intro_1_Barman"	"Hello! I assume that you are Mist	" Bonjour ! Je présume que vous êt
"Level1_Intro_2_Barman"	" It is such an honor to be welcomi	"C'est un honneur de recevoir un ci
"Level1_Intro_3_Insatiable"	"Hello! So I'm here to test and tast	"Bonjour ! Je suis ici pour goûter et
CHOIX 1 - Faux choix		
"Level1_Intro_4_Barman_C1"	"I've heard that you have travellec	"Je sais que vous venez de loin pour
"Level1_Intro_4_Barman_C2"	"Say nothing."	"Ne rien dire."
FROM C1&2 |		
"Level1_Intro_5_Insatiable"	"I must say that I've had some issu	"Et bien je dois dire que j'ai eu de n
"Level1_Intro_6_Barman"	"I'm very sorry about that. Would	"Vous m'en voyez désolé. Voudriez
"Level1_Intro_7_Insatiable"	"Why not... Please a TGV to forget	"Pourquoi pas, un petit TGV pour o
"Level1_Intro_8_Barman"	"Of course. It will ready soon."	"Bien sûr, je vous prépare ça à vite
APRES | SIMULTANE avoir servi le verre		
"Level1_Intro_9_Barman"	"Here it is!"	"Voilà pour vous !"
INSATIABLE BOIT		
"Level1_Intro_10_Insatiable"	"It's pretty good! I'm hyped up abc	"C'est plutôt bon. J'ai hâte de voir c
"Level1_Intro_11_Barman"	"May I take your order then?"	"Puis-je prendre votre commande a
"Level1_Intro_12_Insatiable"	"It will be hamburger to begin!"	"Ce sera un hamburger pour comm
"Level1_Intro_13_Insatiable"	"I must warn you, I won't tolerate	" Je vous préviens, je ne tolérerai a
"Level1_Intro_14_Barman"	"Sure, I understand. A waiter will g	"Bien sûr, je comprends tout à fait.
"Level1_Intro_15_Insatiable"	"Thank you."	"Merci."
FIN DE L'INTRO		
Fin de partie : Loose		
"Level1_Lost_1_Insatiable"	"I'm very disappointed. I can't wait	"Je suis très déçu. J'en ai assez d'at
"Level1_Lost_2_Insatiable"	"They don't even reach the standa	"Ils ne sont même pas à la hauteur
"Level1_Lost_3_Barman"	"We are very sorry about that."	"Vous nous voyez désolés."

Figure 1: Excel dialogue sheet

Now that we have a working localization tool, it is time to work on the dialogue system.

The dialogue system

The dialogue system is a simple script that asks for the localized text from a list of keys and displays it. When the player presses the space bar, the system changes the current key to the next one. But we wanted to add more character to our dialogues... More interaction... We thus decided to implement a possibility to influence the dialogue we get. This is what we called "Dialogues options": You get multiple options, and according to which one you chose, you are greeted with a different dialogue. This made this script inoperable. That is when we introduced more scriptable objects. One for the dialogue: a list of keys to display, but also a list of dialogue options. The second scriptable object was the dialogue options: a key to get the dialogue option text and a dialogue to know what dialogue was associated with this option. Finally, we could just fix our script by replacing the list of keys by a simple dialogue scriptable object. And this concludes the dialogue system. All that was left to do is to have a camera to introduce some movement in the cinematic.

Moving cameras

Cameras are a key point for successful cinematics, they are what makes a cinematic look great. Smooth movement, right camera angles, everything is made to deliver the best shot of the game. When I started working on moving cameras, I wanted to create my own script. But the result was good but not enough. The big problem was that my camera's movements were a bit abrupt and sharp which was the opposite of what we wanted to achieve. I used key points, and the camera had to move to the next point while also slowly trying to match the rotation of the next point. However, we wanted something more professional and good looking. With some research on the internet, I found an amazing Unity package: Cinemachine. This package allows us to have a very smooth rotation but also to change the camera during the cinematic to get the best shots. This was perfect. It took a bit of time to get used to this new powerful tool, but in the end I think the results looks great. We have two languages, we have a dialogue system, and we have moving cameras. Now we need to have a script that links everything together.

Cinematic manager

All we need to do now is a script that controls everything. Which means launching the dialogue at the right time, having the cameras match the client's animation, stopping the dialogue during animations (e.g.: when the barman prepares the drink for the client, the dialogue must stop so that the animation plays correctly) and at the end loading the right scene. An interesting fact to know is that each cinematic takes place in a different scene. For example, because the camera in the defeat and victory scene does not move, an entire part of the level is missing because it would never be seen. But that being said we have to load the correct scene. We decided to label them as follows: *Level XX + (KEYWORD)* [XX represents the number of the level and the KEYWORD can either be Intro, Victory, Defeat or nothing for the actual playable scene]. With that, we can just load the right scene with its build name, and this marks the end of the cinematics implementation. Now that we have beautiful cinematics we need to put in between an actual gameplay.

2.1.3 Scripting the gameplay

Another decision we took was to script the gameplay for the singleplayer so that we could tell a story. I will go in details with our first implemented level (Mr. Insatiable) but Jean will tell you all about the plans.
The concept for Mr. Insatiable was to make him (and only him because he is the only client in the room) order infinite amount of food (from a limited list of 5 burgers: the hamburger, the burger steak, the burger steak lettuce, the burger steak tomato, and the burger lettuce tomato). The level has a timer and you need to get as many points as possible in this amount of time. The victory condition is yours to take if you managed to reach the first point threshold (representing the first star but we will go in greater details in a few lines).
Each time you give him his order you get points and time is added to the timer to give you a little more time while making sure it is not infinite (the time given is not the time it took to make the order). But if you gave him the wrong food, the food gets sent back in the kitchen (destroyed in game). This level was interesting regarding technical specifications because it required to rework the order manager and to add a few methods. Even if it was complicated nor time consuming, it was still a very amusing change of pace after coding all the previous more complicated scripts. But overall, I did not change much, and the core gameplay remains the same. Now that the scripting is done let's move on to the end panel.

2.1.4 End panel and score calculation

A small challenge arises. The goal was to display the score and the according stars after the Defeat/Victory cinematic. But theses cinematics were on different scenes. I then had to find a way to information about the total score accumulated and the points thresholds. For that I used a Don'tDestroyOnLoad script that contains those values. When the correct end scene was loaded, a script would try and access those values to display them in the end panel. This panel's goal is to display the total score accumulated and to tell you if you met the thresholds' expectations. That's what stars are for: it is a graphic indicator for how well you did on the level. If your total points surpass the first threshold, the level is considered successfully completed and you are rewarded with the first star. To get the second and the third and last star, you need to have more that than the respectively second and third threshold. Now having those stars displayed is beautiful, but the end goal here is to save these values (amount of star and best score) so that the player can later play again this level to try to beat his highscore.

2.1.5 Saving data

This is the final part of the singleplayer: being able to save you progression. I will not go into detail as it is a pretty simple part, but we created a script to handle the save and the load of the star count and best score for each level. I honestly had very little experience in this field, so I mainly took inspiration from online tutorials and forums. A few lines of code later and the system was correctly working. It basically uses serialization to convert data into text and then saves that file in the game's file. I am pretty sure that with enough knowledge, or a bit of research, anyone would be able to understand the text file and change some values, but we decided to turn a blind eye and avoid any attempt at encoding our data for time reasons. And this wraps up the singleplayer nicely. The base is now created and creating new levels will now only require changing the scripts for the level to tell a different story as well as creating new dialogues and dialogue options.

2.2 Reworks of menus

2.2.1 Main Menu

The main menu was properly finished. It now features credits, settings, singleplayer, and multiplayer. On top of that, we added a simple animation to open the fridge door. The menu panels are then located on the inside of this door.

The credits menu just lists the different tasks all members were assigned to, the settings menu allows the user to change the general volume, music volume, sound effect volume and graphics quality. (cf. **3.1 Audio Management**)

For the singleplayer menu, we created a simple level selector with the name of the level and shot representing the level. We will, for the last defense, add a star and best score display loaded from the save file.

Finally, on this menu you can also choose which language you want to play in. We originally wanted to translate the entire game (menu buttons, headers, credits...) but this would require a huge amount of work because it would mean having to translate all text and then implement the localized text for each text component... We should have started our project with the localization tool in mind, but as we did not, we decided it would be best if we would only translate the cinematics. For the multiplayer menu, it is merely a button connected to the also reworked lobby.

2.2.2 Lobby

We showed a first version of the lobby during the first defense and we said that we would rework it to add more customization. We indeed came back to this menu, and this will not be the last time this menu changes. This time we added a settings menu that has the same functionalities as the settings menu in the main menu. We also added the possibility to choose how many players will play the game. We improved the user interface by making it clearer and more user-friendly as well as fixing a connectivity bug. We would like to add more customization for the last defense. This includes a player color, a skin selector as well as a map selector. This menu concludes the rework of menus. But the menus looked a bit plain...That is why we decided to add post-processing effects.

2.3 Esthetic work

As previously said the menus looked very sad. So, we downloaded a package called Post Processing Effects to put some life into our menus and later our game scenes too.

2.3.1 Main Menu and Lobby

The main color of our game is pink. Two effects were added to the main menu and the lobby:
- A pink vignette (pink on the edges of the screen and transparent in the middle)
- Chromatic aberration (this causes the colors to slightly shift on the edge causing the appearance of layers of red, blue and green on the edge of the screen). This effect was chosen purely for its esthetics but also because we thought it was a subtle nod to the altered vision associated to a heavy consumption of alcohol.

This chromatic aberration was reduced in the lobby because some texts were too close to the edge causing them to be unreadable.

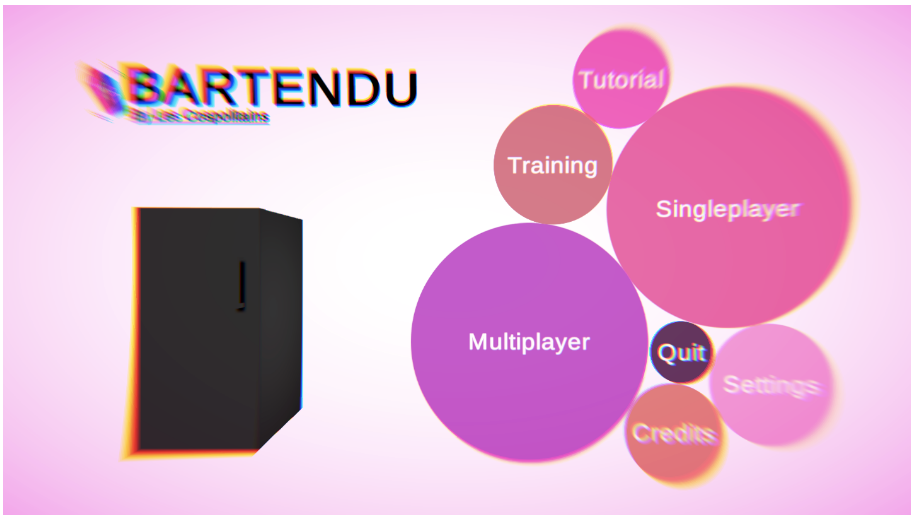
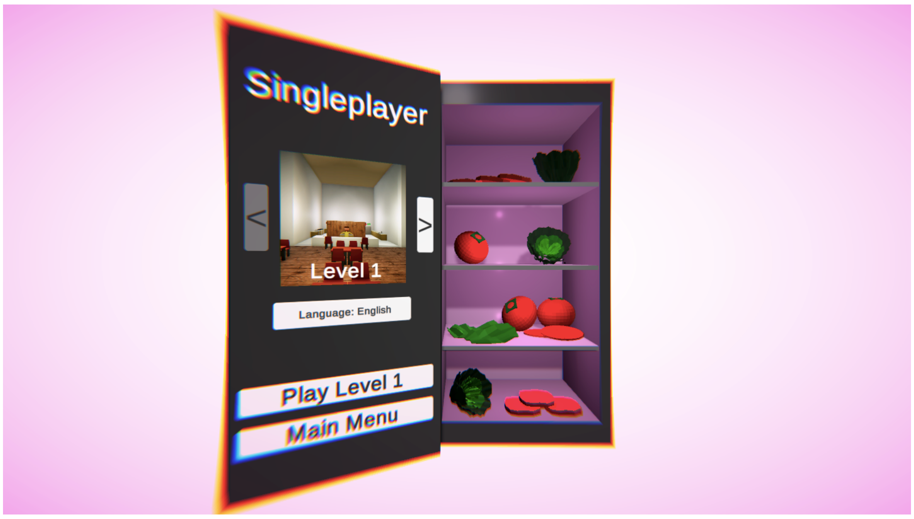
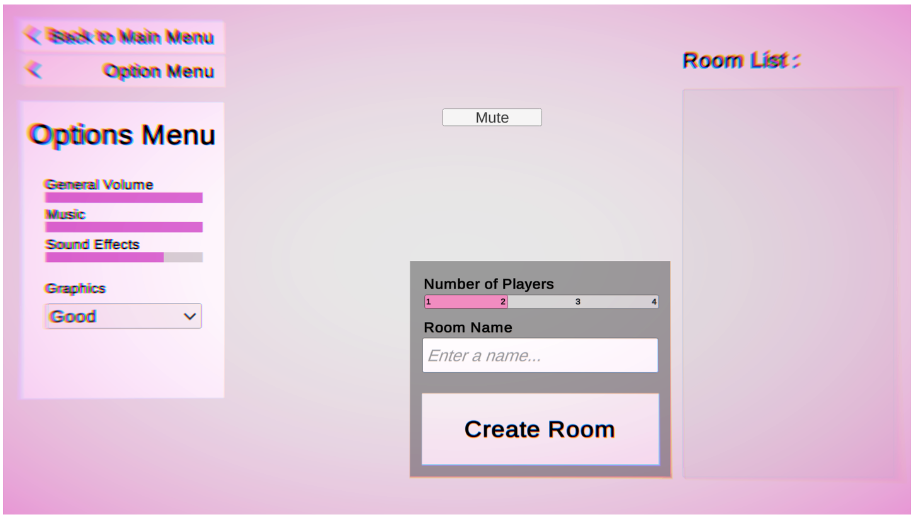


Figure 2: Main Menu with post processing effects



Language: English

Figure 3: Singleplayer level selector in the Main Menu



Graphics
Good

Figure 4: Lobby with post processing effects

2.3.2 In-Game

Being satisfied with the esthetics of our menus, we decided to add those effects in-game as well. Though the effects used are not the same:
- A black vignette (same as the pink one but black this time)
- Shadow enhancing
- Anti-aliasing
- Overall color balance, contrast, exposure
- Color correction and color temperature
All these effects made the game more realistic, good looking and were used in cinematics as well. This esthetic work really helped make our game look more beautiful and professional.

Finally, other minor work was done on the project :
- Correction of multiple bugs concerning the gameplay
- Implementation of all 3D assets created by Lou
- Creation of 2D sprites for new recipes and orders (Fig. 5)
- Client rework and addition of soft drinks acting as bonus points

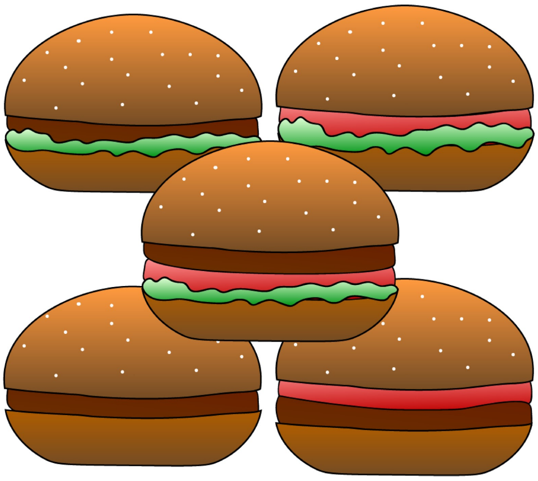


Figure 5: 2D sprites for burgers

I am pretty satisfied with how the game turned out. Everyone worked hard to make the game we are presenting today and almost every feature I wanted to add made it into this version. I insist on the "almost" because as I was caught surprised by the amount of work to transform a multiplayer game into a singleplayer one, I did not have enough time to add the AI. The goal for this second part was to have a working AI and then to train it during the last time slot before the last defense. I can assure that the research work is done, the ideas on how to do it are also very clear in my head. It is just a shame that I did not have enough time to fully implement it. I started creating the base functions (getting the right food, transforming the food (cutting or cooking), putting the food together on a plate, serving the food to the right client. . .) But all I need now is to put everything together and train the AI. This training will have to be done during the last rush. . . But on the bright side the project is going well and we are proud to show this second version.

3 Jean's Achievements

During the few weeks after the first defense, I worked on many tools for our project from music management to level designing and even story writing.

3.1 Audio management

3.1.1 The feature

After reaching our first objective with the creation of 3 pieces of music, the next challenge was to integrate them correctly into our game. Indeed, for the first defense the implementation of music was really raw by using the audio source[1] objects in Unity. We could not manipulate any parameters in game at all. The Audio Management has been created to fill fix this issue. The Volume Management script has been created to change volume parameters while the Music Management script handles the tracks with their parameters (volume, loop, muted or not).

3.1.2 Creating the script

Before the first defense, most of the code in the game had been written by Vincent. Therefore, when I had to implement those features, I discovered a lot about the project on a more technical side. It also required an adaptation time. Overall, it did not take long to adapt because Vincent's code was well commented. Therefore, the code's structure was easy to understand thanks to these comments.

The first important feature to implement was the ability to play new tracks when changing scenes. That includes the creation of a smooth transition. The first one was not hard thanks to Unity Audio Source objects which were easy to operate.
The transition was a little trickier because it involved creating a flexible, and easy to use method that could adapt to future levels and the story mode. Finally, I have decided to create a method that could allow variable timings for transitions but also the creation of 2 modes: one to decrease the sound level and another to increase it. It is a small coroutine which does not last long. So it does not impact the game's performance.
After creating those basic features, most of the difficulties were still up ahead. Indeed, I had to integrate the music management script to the existing code made by Vincent, and adapt to multiplayer constraints. Therefore, music management is manipulated thanks to Photon Pun which will remotely call a method to play the new tracks during a game or even to to make a transition (it is called an RPC).

The last thing to implement properly is the ability for this game object to transition between scenes. This was done easily thanks to the DontDestroyOnLoad function of Unity.

[1] `https://docs.unity3d.com/ScriptReference/AudioSource.html`

3.1.3 Flexibility

During the creation of this script, Vincent improved the UI and implemented new menus. Therefore, he added an essential feature: settings for volume.
To set up the volume, Vincent created some sliders: one for general volume, one for sound effects, and the last one for music volume.
Therefore, I had to get the values from these sliders and change the volume accordingly to them. The only difficulty was my lack of experience which was easily compensated thanks to Vincent's advice.
However, this feature required a new script and some new interactions with the music management script created before. This feature called volume management is destroyed when the scene is changed. But thanks to some new attributes in music management, which is NOT destroyed when loading a new scene, I was able to keep the values of general volume and music volume in memory and re-attach them to a new volume management script and if there were sliders for volume management in the scene we would update the sliders to their old values.
There was another feature implemented in the game manager script of Vincent to synchronize the timer's blink with the beginning of the rush music while making sure that it plays the whole track. This was a small detail that makes a difference in-game.

3.1.4 Some issues

Once the implementation was finished, we tested the music management only to discover some bugs and problems around it.
For example, the transition system which was functional on my build of the project was not on Vincent's side. The volume was not reaching a value above 0 (this is equivalent to a muted sound). So, I had to rework the mathematical model to make it work correctly on all systems. This issue also gave me an opportunity to polish the transition system which now respects the user's settings.
There were also some exceptions which were very easy to fix because they were often beginners mistakes.

To conclude, the Audio Management's creation was not a simple process. It required some knowledge that I did not have before. Thanks to the help provided by Vincent, I was able to produce what the team had in mind for this tool: flexible and functional. This feature may require some polishment before the last defense but for now, we are pretty happy with the result.

3.2 Website

As we were satisfied by the result for the first defense, I have only made a small update on content. More content will be added later with the last defense deadline getting closer. You can find some new documents and a new executable of our project on this website[1].

[1] `https://areas0.github.io/website/download.html`

3.3 Music

After the first batch of sheets, I had to prioritize some other tasks such as story writing and level design to stick to the schedule.
Therefore, I only created one music sheet for this defense. As usual, you will find all the documents and audio I mention on our website.
For this defense, we have started the work on a new level: the boat. So, my task was to create a music that could fit the environment. Usually, when we talk about the sea in music most of the work is done on rhythm to destabilize the audience.
Hence my choice to widely use offbeats alongside a fast tempo.

The melodic line has been chosen to fit with the pirate theme because we aim to create a story mode level based on this theme. The inspiration mainly came from a very known piece of music of the film *Pirates of the Caribbean*[1][2].
This is a preview of the sheet and you can find the PDF file on our website:

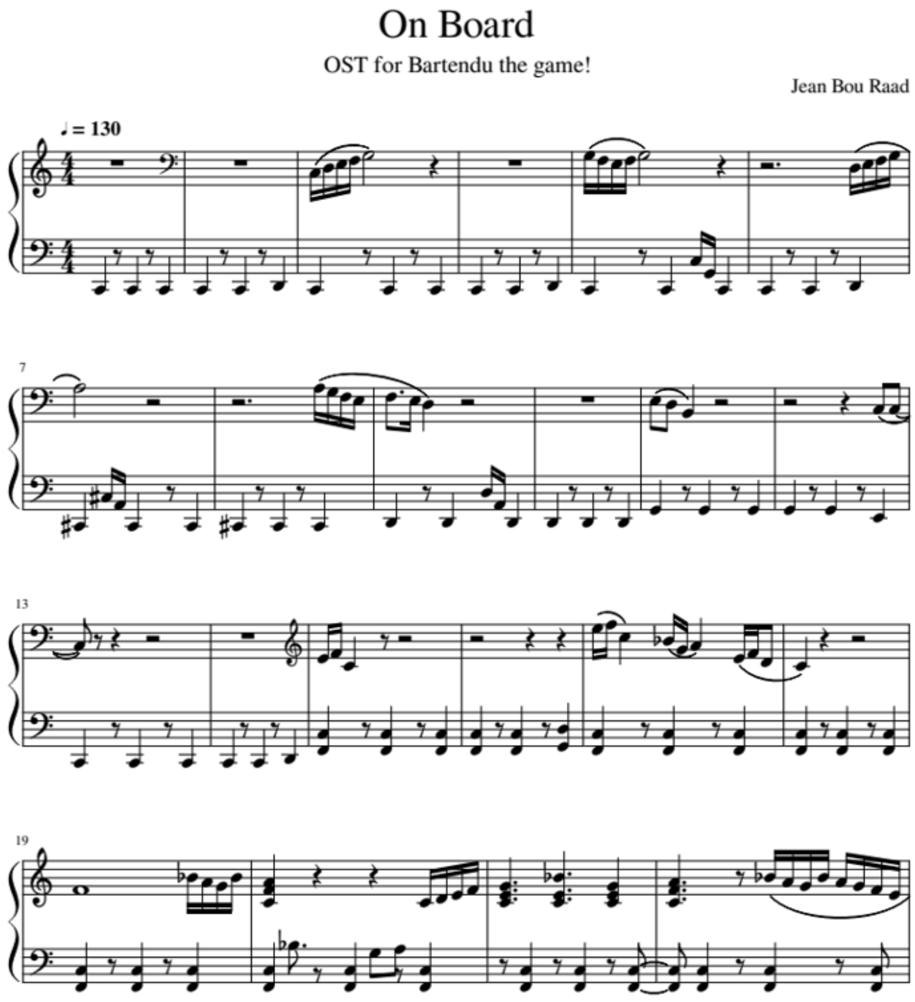


Figure 6: Music sheet for one of the songs of the game *(Musescore3)*

Today, the music is currently on schedule with the book of specifications. For the last defense, we plan to create at least one main theme and two secondary themes.

[1] `https://en.wikipedia.org/wiki/Pirates_of_the_Caribbean_(film_series)`
[2] `https://www.youtube.com/watch?v=27mB8verLK8`

3.4 Story Writing

Vincent has worked a lot after the first defense to come up with a working story mode. Thus, I had to start writing the story that would be featured in this mode.
This task took a lot of thinking to get it started because the whole team had to think of the what's and the how's of this story. For instance, we thought about introducing cutscenes in the story with some writing. This led to reflect on how we would be implementing them. Vincent found an easy solution with cinemachine. That is only an example but it gives an idea of the process behind the pre-production phase of the story. This phase's conclusion is some directives on what I could write and what I could not.

The next phase was story creation. The concept for the story mode is pretty simple: creating a narrative story around a gameplay concept in a unified universe. Each member of the team came up with an idea. Here, you can find each of these concepts:

1. A food-critic comes in the restaurant to evaluate its quality. He is the only client and if there is any delay you will lose many points. (the script was written and implemented)

2. At the grand opening of the boat restaurant, a pirate comes to defy your team. If you lose against the pirate's team you will lose your boat. (the script was written)

3. A student wants to earn some money. To do so, he has to cook in a classroom and avoid its teacher. If he gets caught by the teacher, he will be punished by his parents.

4. The last concept around the rabbit of Alice in Wonderland... More details will be provided for the last report.

Then scriptwriting was the next step. It was not an easy task because I had to integrate jokes and references around specific themes and thematics. This required a lot of brainstorming sessions to produce a “good” script. These scripts are also available on our website[1].

The two last scripts will be written by the team for the last defense and implemented following the schedule.

[1] `https://areas0.github.io/website/download.html`

3.5 Level Design and level implementation

Until the first defense, there was only a test level and another one under construction. Therefore, I have had to make some progress to finish the first real level, and go further to build the next one.

3.5.1 Kitchen basic stage

This level was shown for the first defense as a stage in-progress because some items were missing to make it complete and playable.
The first task was a rework of the whole level. Indeed, the first implementation of the level was raw because some alignments were missing. These alignments were critical and unfixable without creating from scratch the level again. And we knew doing this would not be enough because Unity does not provide enough tools to make great alignments of objects. So, we added a new tool: ProGrids[1]. This tool is powerful because it allows tweaking easily the snap to grid feature implemented in Unity. This tool helped a lot in building a clean and good-looking level. Also, this rework was a golden occasion to make it bigger because we all felt like the kitchen was too small for a game that aims to be played with 4 players. The second task was a bit trickier because it involved working with the whole team. Once this level was built and finished, some core design elements were missing such as camera placement and some instantiators. Concerning the camera placement, we had to decide a viewing angle for the camera together. This is the outcome:

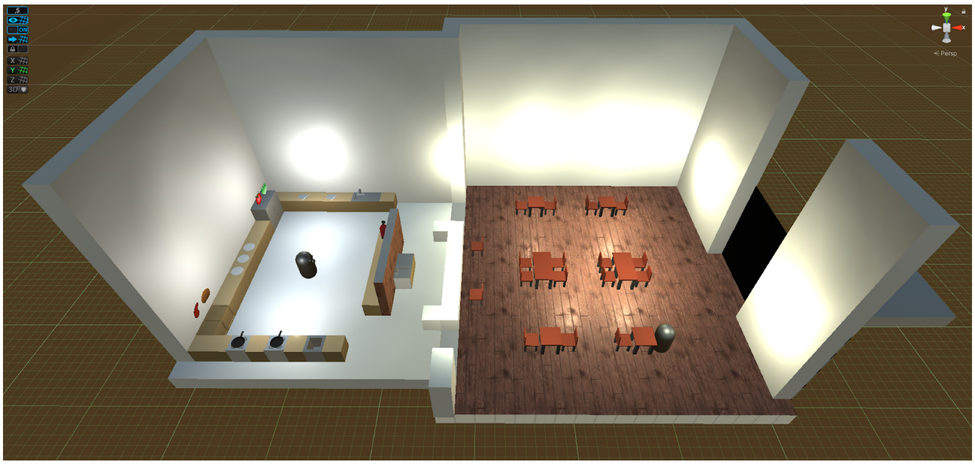


Figure 7: Level 1 in-game view in Unity

To allow such a view, I had to destroy some walls and work with Vincent to create an invisible wall for players. It was mandatory to create it because the restaurant had no exterior and the players should not be allowed to get out of the restaurant.
Also, the lighting had to be done on this level. Due to some issues with the contrast on my screen, I was not able to do correctly my job on lighting. Therefore, Vincent provided some help to do it.
The result on my screen was different from what the others were getting on theirs.
Finally, with the camera angle decided, I had to do a little rework of the level to adapt to the fact that some items were hidden by higher ones.
This concludes my work for the multiplayer version of this level.

[1] https://www.procore3d.com/progrids/

But a solo mode version had to be done due to some technical requirements such as changes on the level's objects to fit the fact that Photon was disabled in this mode.

This third rework was welcome because I would have edited this level anyway to satisfy some script writing requirements concerning layout. Indeed, for the multiplayer variant, the bar is in the kitchen part of this level, while on the other version it is in the restaurant part because It is used in a cutscene for the story mode. Here, you can find a comparison between the two versions:

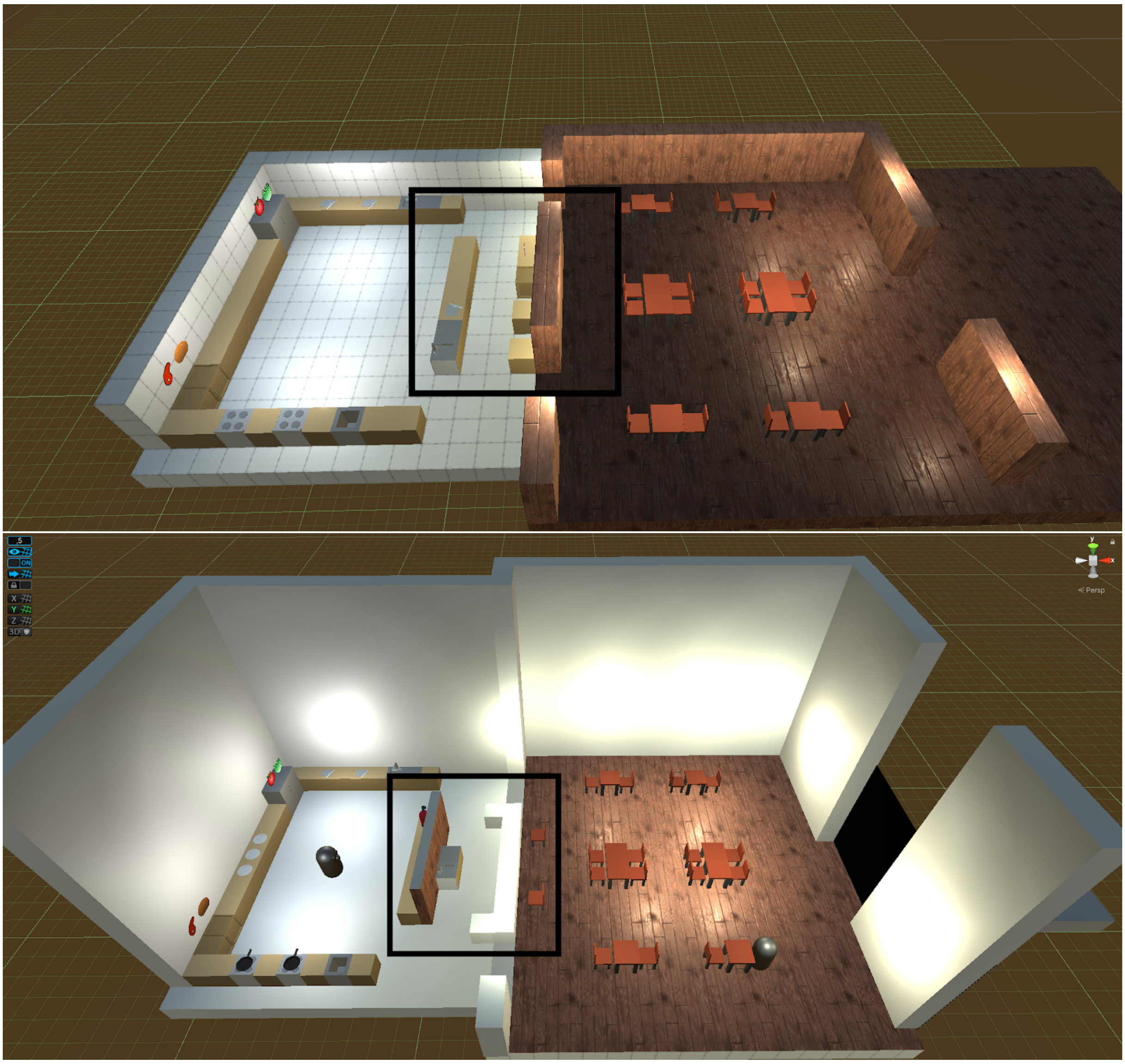


Figure 8: A comparison between the multiplayer version and solo version of level 1

To summarize, this level had a lot of rework. It required a lot of time to finish, but I am still on schedule for level design. I have learned a lot by using new tools and working with various team members which will provide a non-negligible advantage when I will be building the next stage.

3.5.2 The boat

For the first defense, the layout of this level was done. After that, I have worked with various team members to get artistic production started. Indeed, this level requires a new and difficult 3D model of a boat (see Lou's part for more details). Among other details, we have had to work on some which were more difficult. We have had to decide whether we would be doing a model with 2 levels or only the deck of the boat with its exterior. Creating a wedge inside a model would have been way more time consuming for Lou, and it could have jeopardized our schedule.
Now, Lou has finished her work on the 3D level, and I will be finishing the level for the next defense with another concept: the classroom level.

Level design and implementation are following the schedule planned. These tasks are on a good path for completion for the last defense thanks to the work provided by the artistic team.

4 Lou's Achievements

Since the last defense, a lot has been done concerning 3D modeling. First of all, new assets have been made. Then for the needs of some levels a new ragdoll for a special customer has been created as well as a boat and themed assets for another level. Finally characters have been created for the players.

4.1 The assets

New assets have been created concerning the bar part of the game: two cocktails are now available, Mojitos and Cosmopolitans, as well as soft drinks, tools and ingredients to create those cocktails.

But some assets that were previously created in Unity have been replaced by Blender assets.

The assets that have been made are:
— Mojito 's glass (empty and full)
— Cosmopolitan's glass (Fig. 9)
— Shaker (Fig. 10)
— Machine to create soft drinks
— Ingredients (bottle of alcohol, mint, lime, ice, cranberry juice,. . . ..)
— mortar and pestle (to prepare some ingredients)
— Chair (previously made in Unity)
— Table (previously made in Unity)
— . . .

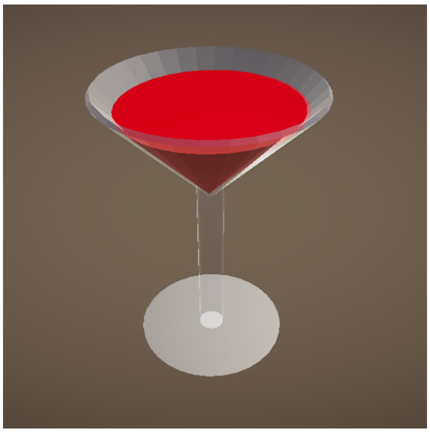


Figure 9: Cosmopolitan's glass

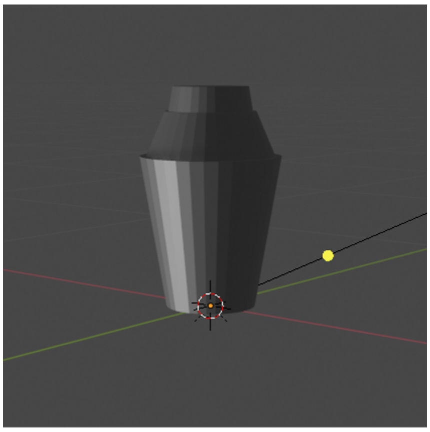


Figure 10: Shaker

For the texture of the glasses and the ice, they have been added directly in Unity. Those assets will be implemented shortly, and new drinks recipes could be added.

4.2 Special assets for some levels

A new ragdoll has been created called Mr. Insatiable (Fig. 11). This one will be bigger and different than the other ragdoll for the need of a level created by Jean (cf. **3.5.2 The boat**).

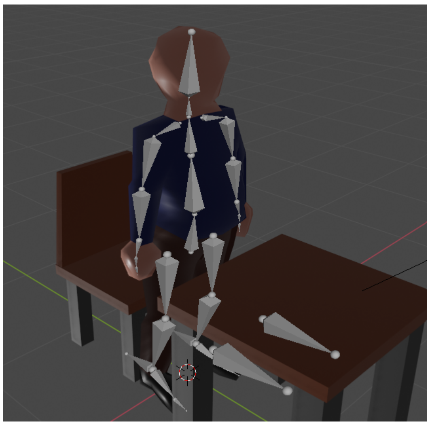


Figure 11: Mr. Insatiable

Then for the level on the boat (Fig. 12), a pirate theme has been decided so assets such as a barrel, and a bridge to go on the boat have been made. In addition, the boat has been created in Blender, and was by far the biggest asset created for this project. We had a few difficulties to make it, so it would look like a pirate ship as much as possible. The main difficulty was to scale it, so the restaurant could be on the boat and so that the proportion compared to the size of a player or a customer would work.

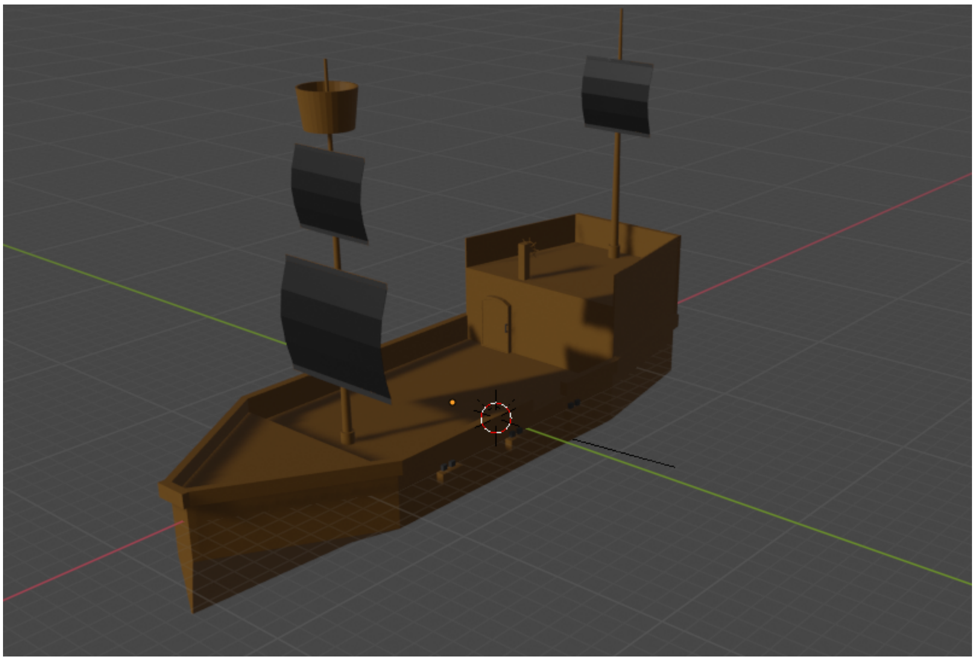


Figure 12: Boat

New assets will be created for other levels like the one we will make in a classroom.

4.3 The characters for the players

Since the design of our game are is “cartoonish” , we had this idea that the characters used by the players could be Funko Pop!’s. So we had this idea to scan in 3D those figurines and polish them in Blender, add bones and weight, animate them and then implement them in Unity.

To do so I first tried to take more than 30 photos of a figurine and then use this software called Mushroom to create a 3D model. After a few tries with different numbers of photos, different lights, and backgrounds, it still did not work, I had nothing and the process stopped before even creating a part of the model.

So instead, after a lot of research, I found different smartphone applications that would allow us to create a 3D model.

The first one was called display. land, and the goal was to film the object while moving around it. After a lot of different tries, it was still not working. This time the process was but the final product was not what we wanted, the model was flat and did not look like the figurine, I still managed to model a stool but it seemed like the model we wanted to model was too small.

The second application was called SCANN3D, and you had to take photos. I tried a lot of different ways to model the figurine, different lights, different backgrounds,. . . .. But it was still not working, the model created was looking weird and it had modeled only the front of the figurine.

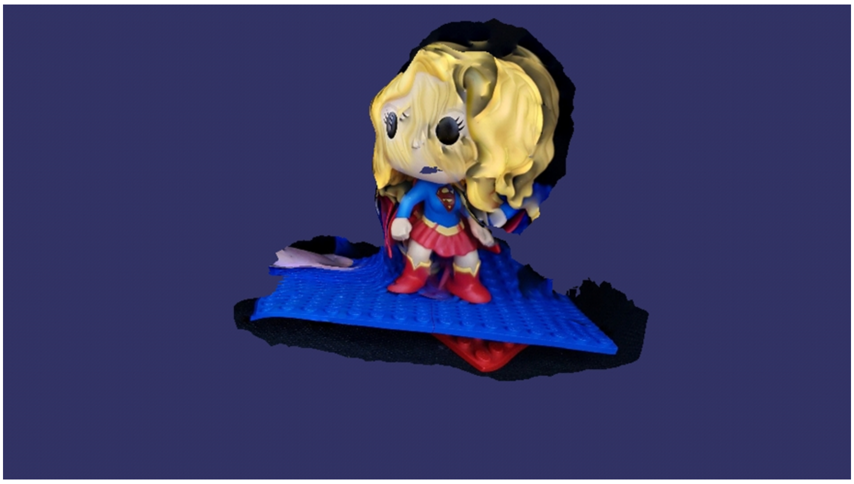


Figure 13: Failed 3D scan of a Funko Pop! with SCANN3D

Finally, we decided not to use 3D Scanning to create the players, we did not have the time nor the equipment to create something that was good looking and working with the environment of the game.

So I created a new character in Low poly using Blender that looks “cartoonish”, the players will have the possibility to choose between different characters (different looks and genders).

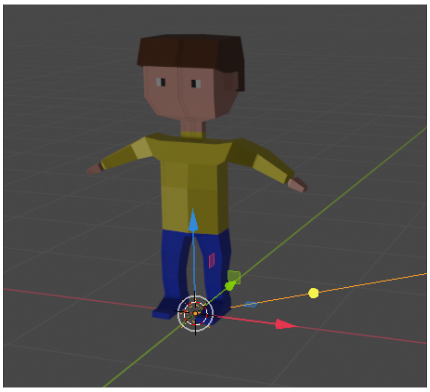


Figure 14: Player character

Yet even though 3D Scanning an object was not working for this project, it allowed us to discover more on other software and tools that could be used in parallel in a project.

To conclude on 3D modeling, the group is currently on schedule and new assets will be created for the last defense. Even though we had a few technical difficulties as mentioned previously, the design of the game looks exactly like we imagined.

5 Emeline's Achievements

5.1 3D animations

In Bartendu, there are 2 different types of characters : customers and cooks. Both of them perform actions throughout the game and therefore require animations.

What we call 3D animations here is the animations of the armatures inside the ragdolls. This is all done in Blender, which allows to set keyframes that basically save the exact location and rotation of the bones at any point in time we may decide. Then Blender is able to make the smooth transitions between every recorded positions – and create an animation. As of now, we have finished the animations that will be needed for the customers. These animations are :

- Idle, for when the customers wait in line outside
- Walking, for when the customers enter the room and go to their seats
- Sitting down, when the customers reach their table
- Ordering, which happens when an order is sent on the board for orders
- Eating, which customers use once they are served

The main difficulties we encountered so far with this task were, firstly, learning how to use the software with its shortcuts, and secondly, the actual making of the animations. Indeed, it is quite hard to figure out how our bones and limbs behave when we move. But since we wanted these movements to look as natural as possible, we had to find a way to understand how our bodies work. We had to do these movements over and over again in real life to see exactly which parts moved and how. And for trickier actions like sitting down, the solution we found was to record ourselves with our phones, doing the actions step by step, which truly facilitated the process for making keyframes.

Some of these animations will be reused for the players, but we will have to create new ones for other specific actions that customers do not perform. We will also have to implement all of them in the project. However, we are on schedule for 3D animations.

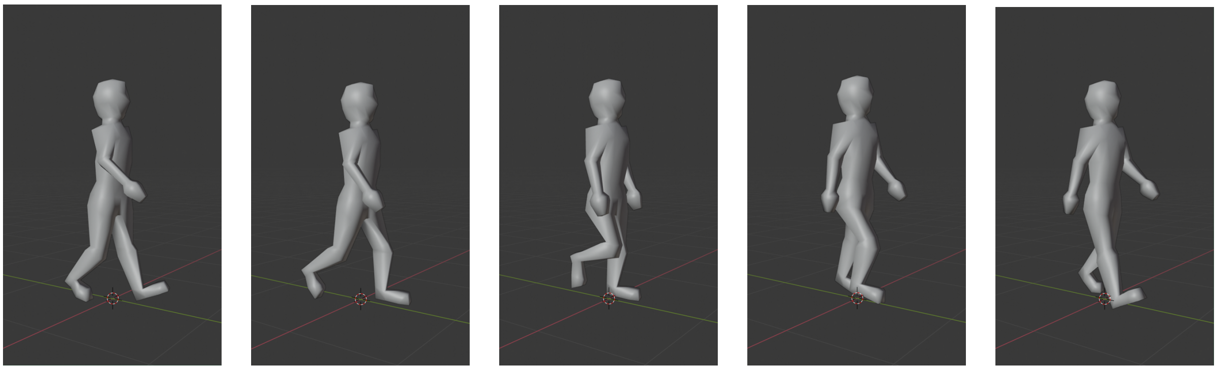


Figure 15: Keyframes for the ”Walking” animation

5.2 Particle effects

In the last report, we mentioned considering using shaders for Particle Effects,as they offer very interesting options. However, we have given up on this idea as after doing more research and getting more used to Unity, we found other ways of making them that still look good. It took a lot of experimenting to reach satisfying results.

Here are the different particle effects that will be included in the game:
— fire effect
— foam effect for when you wash the dishes
— boost effect for when the player jumps forward
— (fire extinguisher effect that was already shown in the last report)

The solutions we found were working with more advanced parameters and creating our own textures for the materials of the Particles. We used Gimp to draw and design some of the materials, because its tools particularly fit our needs. Different brushes and filters especially were useful, such as the moving blur effect. One problem we had was that the particles had to fit the global atmosphere and looks of the game. The fire effect was the hardest to achieve : it had to be not too realistic but not too cartoonish at the same time. The following is the progression made on the fire effect (which looks much better than the one we showed during the first defense, if we may say so ourselves):

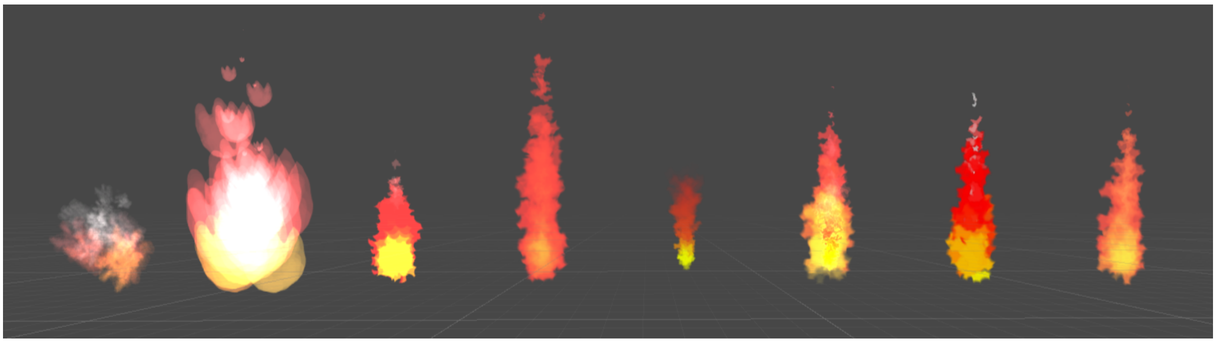


Figure 16: Fire progression

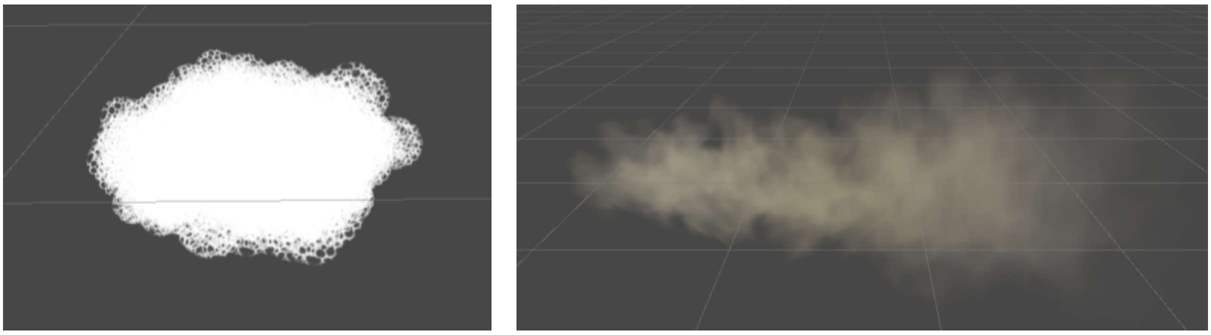


Figure 17: Foam effect and Boost effect

These particles may be slightly changed before the last defense but most of the work has been done.

Conclusion

Despite a few technical but also organizational difficulties because of the COVID-19 crisis, we are on schedule concerning our tasks. Even though we have some delay on sound effects and the implementation of the AI, about two third of the project has been done.

Our website has been updated, new music, assets as well as animations and particle effects have been created thanks to the hard work of the artistic team. The technical team has been working a lot too, and thanks to them, the game has most of its core gameplay mechanics implemented and the first level has been done.

For the last defense we aim to pursue in this direction with the creation of new assets, new music, and new animations as well as new levels. In addition, we will polish the UI, create a tutorial, and make sure to have an operational AI.

Finally, even though there is still a lot of work to come to finish the project, we have all learned a lot. We are very excited to create such a project and we are ready to bring it to its final form.