Educational Game Design for an Interactive Experience

Tim Nguyen
Supervisor: Catharine Oertel
École Polytechnique Fédérale de Lausanne, Switzerland, January 2019

Abstract

The main goal of this project is to create a product that will allow young students to learn how to interact with each other or with other external persons in the context of their apprenticeship. While the environment can vary depending on where the students are learning, the objective is to create a platform that could adapt to any possible situations and contexts. In this educational game, the user is immersed in a story and meet multiple different fictional characters. Depending on the choices the player has decided to take will derive different scenarios. The goal is to learn how people would react to what the user has decided to say or do and to see the multiple possible consequences in order to practice this interaction in real life. This paper relates a step-by-step process to build such a project using Unity[1].

1 Introduction

Interactions with other persons can sometimes be a challenge for some people. Many situations can be possible: how to talk politely to a new client in a shop, how to approach other colleagues, how to welcome someone new that just arrived, and so on. For young students, being successful in an apprenticeship and in the future by building a prosperous career might be facilitated by having a large network of peers.

All these everyday applications may seem easy to handle for a few people, but for others who have more difficulties with communicating or who are just beginners in the domain they will work in, creating an interactive experience where they would have the opportunity to train can be a great solution! Having the possibility to make mistakes and exercise social interactions with non-player characters through an educative platform may help to improve a person's self-confidence and experience over time. Furthermore, a "gamification" of this experience could be a way to arouse the interest of the users and to encourage them to pursuit their training through a playful activity.

This project is using Unity as a game engine and is available for further improvements on GitHub[2]. This paper explains the current state of the project and its future goals.

2 Aims

2.1 General goals

This project aims not only to build an interactive experience as an educative game, but also to make it "adaptive", meaning that anyone who wants to create such an experience in a specific environment, with specific characters and dialogues, should be able to perform it easily using this project. Hence, the main challenges will be to find a way to handle varying resources in a structured and logical way, so any experience can be created from the data received, and to explain in this paper each component of the project so any user who wants to create a customized experience can make it conveniently.

2.2 Description of the demo

As an example, this process of building such a project has been made with respect to a *hospital* context. The apprentices will be nurses who want to practice their social interaction with their patients and colleagues. Hence, the section explaining the design of the game should not be considered in the case another wants to create a story in another specific context.

3 Inspiration

Similar experiences already exist in the industry of gaming. A lot of people have been reading *Choose Your Path Story Books* since their childhood and now this concept has been transferred by many developers into video games. What makes this kind of entertainment so popular is the possibility to test different scenarios and see their consequences, which is not possible in real life. This project has been inspired by this kind of game play, where the user is immersed in an adventure where she or he is master of what happens throughout the story.

Combining with an educative purpose, the player will be able to practice her or his social interaction ability while enjoying testing different scenarios available in the experience.

3.1 Visual Novels

A more specific type of game that fully inspired this project is the *Visual Novel* type, which is an interactive game genre, originated in Japan. It consists of a text-based story with narrative style of literature and interactivity aided by visual components[3].
Visual Novels are distinguished from other game types by their **typical minimal game play**. Generally, the majority of player actions is limited to clicking to read the narrative text. At some points, the user has the possibility to choose between different answers which might or might not influence the rest of the story (Figure 1).

Figure 1: *Always Remember Me*, a Visual Novel from Winter Wolves[4]

Figure 2: *Ripples*, a free Visual Novel from SakeVisual[5]

A classic interface for such a game is a background representing the environment where the story takes place, a dialogue box containing the narrative texts, and a sprite of a character with whom the user interacts (Figure 2). Generally, the name of the character who is currently talking is displayed above the dialogue, and the story is is told from the player's point of view (first person).

Even with a simplistic game play, this kind of entertainment has met a lot of success, and despite the huge progress in the industry of gaming, the popularity of this format has been preserved. Some *Visual Novels* has been improved with new features for the player represented by a system of score, a notion of elapsed time with the restriction to do a certain amount of activities within a day, and many other possible actions (Figure 1). This shows that this kind of basic game play has a great potential to be developed into more complex features.

3.2 Episodic Adventure Games

With the actual progress of computer graphics, this type of entertainment has been derived into episodic graphic adventure games, which extend the user actions, giving now the possibility explore a 3D world, and give access to animations of better quality. Being able to walk in a digital environment leads to new abilities, like analyzing the elements around the player and choosing to which character to talk (Figure 3).

At the end of some of these games, the user has access to an overview of all her or his choices made since the beginning of the story (Figure 4).
While the main interactions are still limited to make choices throughout a story, the success of this type of game remains not less than intact, but even more popular. This indeed points out the entertaining power of a *Multiple Paths Story Game*.

Figure 3: *Life is Strange,* an episodic graphic adventure video game, by Dontnod Entertainment[6]

Figure 4: Overview of all the choices made by a player in *Life is Strange*

Hence, all those kinds of storyboard games are very practical in the sense that having a basic game play where choosing one answer among others can be enough to be playful, but if we want to make the experience more complex, it would be totally possible by adding other features like the ones mentioned before.

4 Design

Inspired by many popular interactive games, a first sketch of the design of the game was made, following the example of the hospital environment. Hence, the following choices and thoughts made in this paper have been based on this. The following sections explain the decisions made to design the game for this particular example case. However, they can totally be changed to adapt to any kind of story.

4.1 Environment

4.1.1 Backgrounds

The backgrounds were found only on websites sharing photographs with free license[7]. In this case, real photographs were used and converted into a more "cartoonish" style using a free online converter[8] to fit with the overall components. Many programs are also available to convert photos into images with a specific style.

Here are a few examples of the filter:

On the left: Photos without filtering[9] **On the right:** Photos with filtering

If we had more time at our disposal, another idea would have been to draw by hand the backgrounds by ourselves. However, it is a task that takes a certain amount of time, and since this project is only a demo, it might be more relevant to use this method for more efficiency.

4.1.2 Choice of colors

The majority of visual components in this game were modified using Krita[10], an open source painting program and other photo editing software. Since we are in a hospital environment, the colors were chosen based on the general colors this context inspires the most people.

We ended up using a palette of colors such as below[11]:

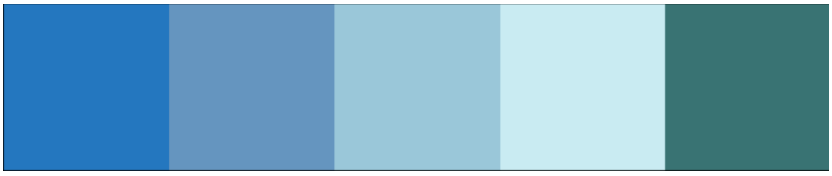

Figure 5: A hospital color-themed palette

Thus, most of the components of the game, such as the *Dialogue Box* and the *Characters*, were drawn in order to follow this range of colors.

4.2 Characters

The characters were fully hand-drawn and colorized using Krita[10]. Since this version of the game will only be a demo, only a few characters were drawn (Figure 7 and Figure 9). Some of them were inspired by models (Figure 6 and Figure 8), while others came from imagination.

Figure 6: Example of a nurse[12]

Figure 7: Resulting drawing

Figure 8: Nurse and the Angry Patient[13]

Figure 9: Resulting drawing

5 Implementation

5.1 Tools & Resources

5.1.1 Game Engine

This project was mainly created using Unity combined with C#, which is a powerful game engine to create efficiently video games. One noticeably advantage of this game engine is that it provides both visual components to work with as well as scripts to code entirely the game if we desire to perform this way. Tutorials given on the official website of Unity were pretty helpful and allowed to acquire some initial knowledge to create our experience.

5.1.2 Centralized data

The main resource in this project is a set of dialogues that will be displayed in the game. The text elements have been stored in a CSV file, since it is a format that is quite easy to handle and also because it gives a nice visualization of the dialogues and all the other complementary information. The idea is to have all the data defined **in one file**, so that when anything is changed in that file, the changes will automatically apply when running the game.

The format of the file is thus cells containing all the data of the game. More precisely, **each row of the CSV file corresponds to one dialogue extract of one character**. Then, **each column gives information about that dialogue extract**, as for instance the name of the character who is actually saying this extract, the scene where the dialogue is taking place, the name of the background that should be displayed at this moment, and so on.

The number of columns and their names is *predefined*. However, it is still possible to modify them in a quite easy way. For our demo, we have defined a lot of columns, but here is a basic essential structure we have started with and which would be recommended:

- sceneID: Identifies a scene where the dialogue takes place
- character: The character's name who is currently speaking
- dialogue: The text dialogue extract that the user will read
- answer_good: The “good” answer the user can give, if there is
- answer_neutral: The default “neutral” answer, if there is
- answer_bad: The “wrong” possible answer, if there is
- next_good: The ID of the next scene if the user chose the “good” answer, if there is
- next_neutral: The ID of the next scene if the user chose the “neutral” answer, if there is
- next_bad: The ID of the next scene if the user choose the “bad” answer, if there is
- background_img: The file name of the image that will be displayed as background

This gives an initial 10 columns. Here is an example of how the table looks like (the columns "answer_neutral" and "next_neutral" are unfortunately not displayed because of the space available, but they work similarly than the two others):

sceneID	character	dialogue	answer_good	answer_bad	...	next_good	next_bad	...	background_img
intro_0	Me	(Oh, okay?)	NA	NA	...	NA	NA	...	reception
intro_0	Doctor	I'm very pleased to have you here. How do you feel?	Well, I feel great, but...	I've never wanted to be here! Let me leave!	...	intro_0A	intro_0B	...	reception
intro_0A	Doctor	This is wonderful! I hope you're not too afraid of what will happen next.	NA	NA	...	intro_1	intro_1	...	reception
intro_0B	Doctor	Oh, don't leave yet, the fun hasn't began yet!	NA	NA	...	intro_1	intro_1	...	reception
intro_0C	Doctor	I know, I know, you will understand a bit more later.	NA	NA	...	intro_1	intro_1	...	reception
intro_1	Me	(What does he means? Now I'm afraid!!)	NA	NA	...	NA	NA	...	reception

This structure allows a complete view of all the possible dialogues and gives information about the flow of the story. The "answers" columns contain NA values, because not all dialogue extracts require the user to choose between 3 possible answers. On the other side, the "next" columns might contain NA in the case the user has to make a choice. After the player has made one, the "next" columns will **redirect her/him to the main plot**.
We are aware that using this structure will limit the user to choose only between 3 possible answers at each time, but based on personal experience, this number seems to be the ideal quantity of answers the user should be allowed to choose.

5.1.3 CSV2Table

Since we are working with Unity, the next thing we needed was to write a parser that would read the CSV file and store everything in a nice and clean table, readable by Unity. The free **CSV2Table** Asset[14] on the *Asset Store* of Unity allows to generate a C# code for a CSV parser. The program takes as input the CSV file that is displayed on the left panel, then generates the corresponding code to parse its content on the right panel (Figure 10). The CSV data is stored into a structured table, and has functions to access to this data. In addition to that, it also has functions to find rows that matches to a specific query.

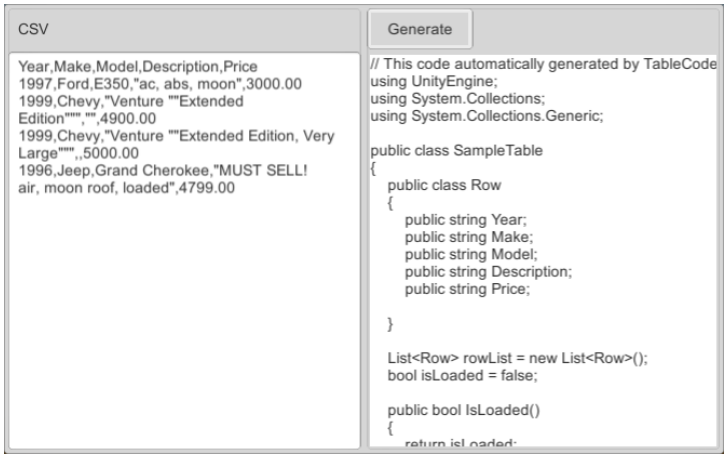

Figure 10: A view of the CSV parser generator

Here is an example of two typical queries' structure:

```
1     public Row Find_sceneID(string find)
2     {
3           return rowList.Find(x => x.sceneID == find);
4     }
5
6     public List<Row> FindAll_sceneID(string find)
7     {
8           return rowList.FindAll(x => x.sceneID == find);
9     }
```

The first type of function allows to find **the first row** of data where the ID of the scene matches the one given as argument, while the second function returns **a list of all rows that match the query**.

5.1.4 The *DialoguesTable* Script

When using the CSV2Table generator, it will create a script that will contains both types of functions **for each column feature in the CSV file**. Hence, if another wants to add more columns to the file (for example to add more complexity to the game), the task of modifying the *parser script* will be easily handled using this Asset. Only a few changes will be necessary to adapt the new script to Unity, like renaming the script (we named ours **DialoguesTable**) and copying the additional non-automatically generated functions.

As an example, for our demo, we have added a few additional functions for practicality reasons, thus it is important to consider to re-write these functions again if we plan to generate a whole new script using CSV2Table.

5.2 User Interface

As we have seen before, our project will essentially be based on the game play of *Visual Novels*. We have already touched on the subject of backgrounds and characters in terms of design, thus remain the other components of the game.

5.2.1 The Dialogue Manager

The *Dialogue Box* is the essential component of the game. It is the element that will display all the characters' lines.

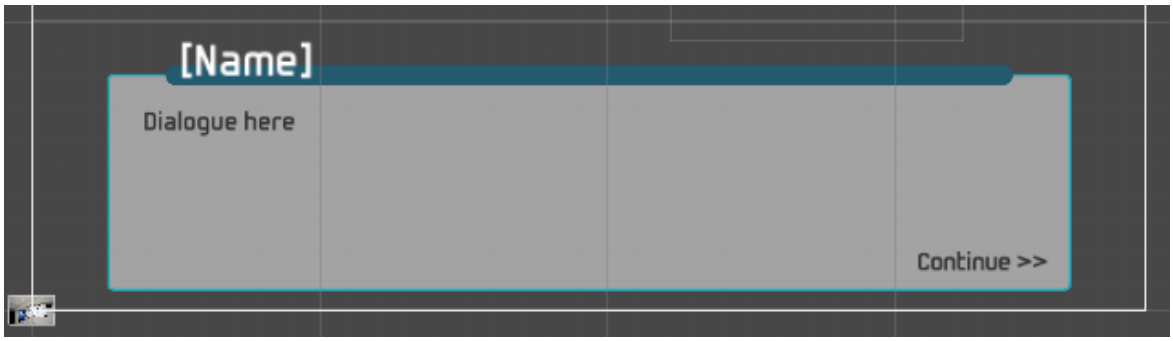

Figure 11: The Dialogue Box component

A *Dialogue Box* basically has the following elements :

- The name of the character who is currently talking
- The speech itself
- A button ***continue*** to jump to the next dialogue extract

To handle at which moment the dialogues are correctly displayed, a script for a **Dialogue Manager** was implemented. This script has access to the **Dialogues Table** and has a few functions to handle the dialogues.
This script has the main task to ensure that the dialogues are displayed correctly, and will also decide at which moment it should display the possible answers to the player if there are.
At the end of the game, **all the answers are saved in a file**, so the user can see the choices made during the experience.

5.2.2 The Choices Panel

The Choices Panel is the component that shows the different possible answers the user can make. By default, it has 3 buttons, and each of them is linked to a « good », « bad » or « neutral » answer. To avoid them to be in the same order when they are displaying, all the buttons positions are randomly shuffled before being assigned an answer.

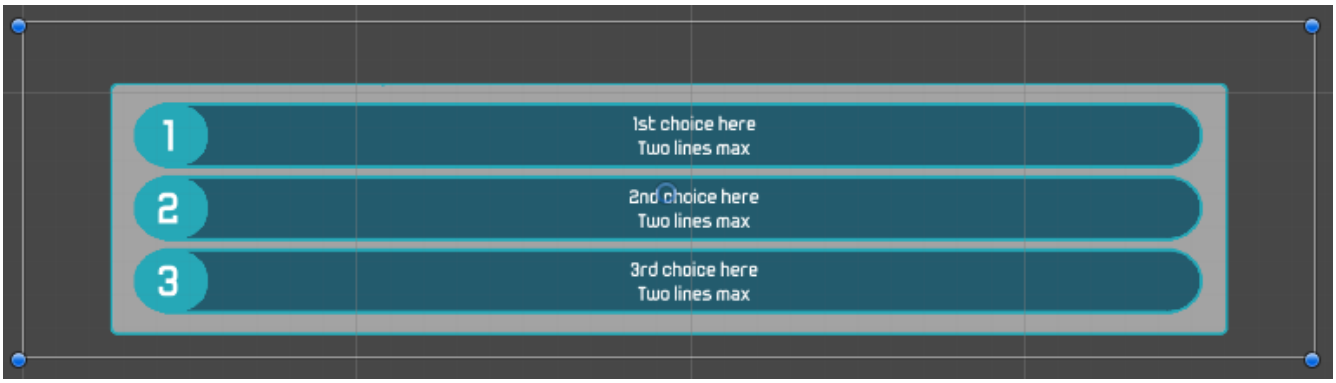

Figure 12: The Choices Panel component

5.2.3 Menu, Animations & Sounds

For a more pleasant visual experience, some animations and sounds were added to the game. Following some good tutorials on Unity[15], animations on the **Dialogue Box** and the **Choices Panel** were applied. For instance, making the text displaying letter by letter is an animation specific to *Visual Novels*.

All the sounds and music were found on free license websites[16]. A **Settings Menu** was added in order to manage the volume of the game.

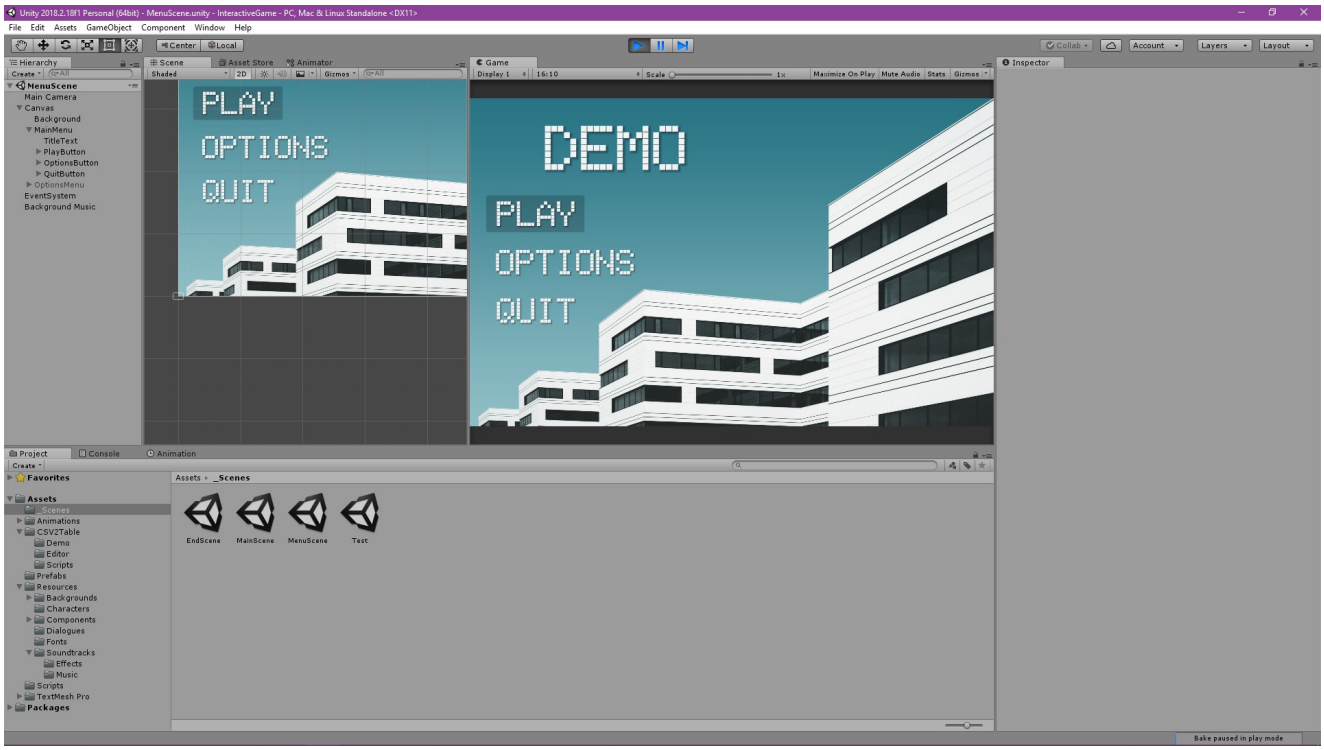

Figure 12: Screenshot of the Start Menu

Finally, a Start Menu has been added to make the experience appear more like a game.

5.3 Game Play

Once having an appropriate game interface and data management, adding more "gamification" features is a way to make the experience more playful for the apprentices and increases the chances to make them more practicing. As we have seen before, this kind of project has the potential to be extended according to one's desire, once the most important elements are implemented, meaning the **User Interface** and the **Data Manager**.

Hence, for this demo, a system of **score** and **feedback** was implemented.

5.3.1 The Score Manager

To enhance the game play of the experience, a few more columns were added to define a scoring system. Since in our example project the students are nurses, two kind of scores were distinguished: an *Empathy Score* and a *Skill Score*.
Thus, the following columns were added to the data file:

- score_emp_good, score_skill_good
- score_emp_bad, score_skill_bad
- score_emp_neutral, score_skill_neutral

Here, the terms "good", "bad" and "neutral" have no significant meanings, they are rather named this way to distinguish with which answer the score is associated.

To take these features into account, an instance of a **Character** was created. Each character will have an *Empathy Score* and *Skill Score* that will change with respect to the user's choices.

The role of the **Score Manager** is to update properly these scores for each character. More precisely, when the user chooses an answer, it will influence the scores she or he has with this character. At the end of the game, the scores the player has made with each character will be displayed, so the user can have an overview of them.

Figure 13: An example of scores display

5.3.2 Feedback & the Character Manager

In addition to the scoring system, a feedback system was also implemented. The goal of this feature is to give to the player a visual feedback of her or his choices.

Using the scoring system we have seen before, a first visual feedback could be to display the gain or loss of points just after the user has made a choice. This was implemented in the **Score Manager**. Some animations were also added for this kind of event.

Another way of giving a feedback is to change the face of the character with who the player is currently speaking. To handle this, the **Character Manager** was implemented. This script manages every event that is related to the characters, which is in other words, to update properly the character sprite according to the current state of the game.

Hence, the two possible events where the Character Manager has to operate are:

- When the scene is switching to a new character
- After the user has made a choice

A function to **randomly** display feedback was also implemented, meaning that when the user has made a choice, the feedback can be visually randomly displayed or not. This could be useful to analyze if showing a visual feedback through a character's change of expression or not can influence the player's performance.

Figure 14: Sprite of Happy Little Girl (Good Answer)

Figure 15: Sprite of Angry Little Girl (Bad Answer)

Figure 16: Sprite of Normal Little Girl (Neutral Answer)

Conclusion & Future works

The potential of storyboard games in an educative perspective can be highly exploited in order to help people to improve their social interaction with others. While a scoring and feedback systems are good ways to display the progress of a student, having an interactive experience through an immersion in a fictional adventure is the solution to practice communication without real life consequences.

This project is a start for an educative experience through a playful activity. Its implementation can still be greatly improved in the future with many features: adding a dynamic way of displaying the answers, implementing relation states with the characters, adding a notion of time in the game, and many others.

However, besides those features, we should consider the main aspect this project has, which is the ability to adapt to any kind of experience one can desire to create, so any student coming from any context can exercise her or his social skill using this platform.

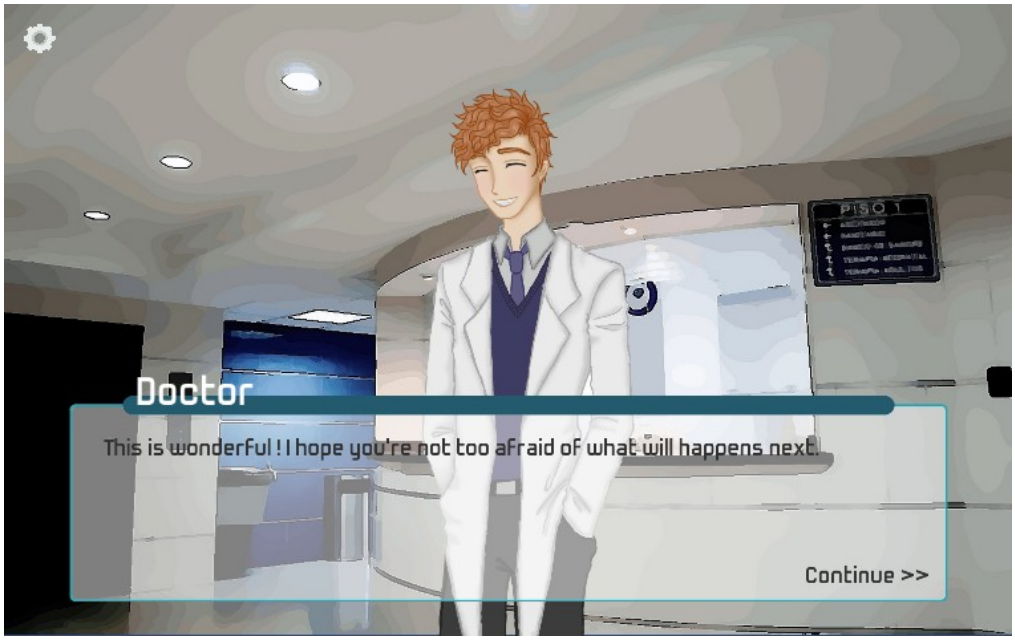

Figure 17: A final screenshot of the project

References

[1] Unity, a cross-platform game engine : https://unity3d.com/

[2] Project repo: https://github.com/Akynna/InteractiveGame

[3] Description of Visual Novels: https://en.wikipedia.org/wiki/Visual_novel

[4] *Always Remember Me,* Winter Wolves: https://winter-wolves.itch.io/always-remember-me

[5] *Ripples*, SakeVisual: http://www.sakevisual.com/ripples.html

[6] *Life is Strange*, Dontnod Entertainment: https://lifeisstrange.square-enix-games.com/en-us/games/life-is-strange

[7] Unsplash, a website sharing free photographs: https://unsplash.com/

[8] Online image to cartoon converter: http://www.cartoonize.net/

[9] Sources of the images: https://unsplash.com/photos/ShJUYkshceY

https://unsplash.com/photos/g0PTp89dumc

[10] Krita, an open-source painting program: https://krita.org/en/

[11] Hospital color theme by lawand haji: https://color.adobe.com/Hospital-color-theme-6977641/

[12] Source of the image: https://de.123rf.com/

[13] Source of the image: https://youtu.be/f4qCP_NEYYU

[14] CSV2Table Asset: https://assetstore.unity.com/packages/tools/utilities/csv2table-36443

[15] Unity Tutorials: https://www.youtube.com/channel/UCYbK_tjZ2OrIZFBvU6CCMiA

[16] Purple Planet, a website sharing free music: https://www.purple-planet.com/
SoundImage.org: https://soundimage.org/city-urban/