English project - An2
2A TELECOM Nancy - 2022-2023

Project report

TypeRun

Members of the project team :
Quentin LENFANT & Julien DE TOFFOLI

Teachers :
Muriel DUVAL & Charles DESPRES

Contents

1 Introduction

The english project aimed to create a software tool that enables the user to work on english language or culture. In this report, we will present you our application, called TypeRun. This is a little video game in which the user can work on thematically sorted english words on one hand, and improve his typing skills on the other hand.

The various projects we were involved in through our courses mainly focused on the creation of websites or programming oriented applications. However, with this project, we have an opportunity to try a new aspect of software development which is the video game design. We chose that path and we tried to be creative in order to create an original game that is fun to play.

Through this document, we are going to present you more precisely TypeRun, firstly by showing the user documentation. Then we will give some informations about the technical aspects of the software and the possible improvements.

2 User documentation

2.1 General presentation

TypeRun is a mix between a typing game and a running one. The goal is to let the little fox continue its run by destroying all rocks on its path. To do that, you have to correctly type the word associated to each obstacle. But be careful: the fox runs faster and faster and if a mistake is made while typing a word, you will have to retype it entirely!

You have 3 lives. You lose one each time the fox hits a rock, and the game ends when the last life is lost. Good luck!

2.2 How to launch the game

The game is available on Windows and Linuses distributions. All you have to do is to launch the executable file corresponding to your operating system.

On Windows, you have to double click on the file `TypeRun.exe`.

On Linuses distribution, double click on the file `TYPERUN-Linux_Build.x86_64` or execute it using the terminal. In that case, use the following command in the repertory that contains the executable file:

```
./TYPERUN-Linux_Build.x86_64
```

2.3 Game's user interface

When you start the game, the first screen displayed is the title one. (Figure 1).

Click on "Click to play" to go on next screen.

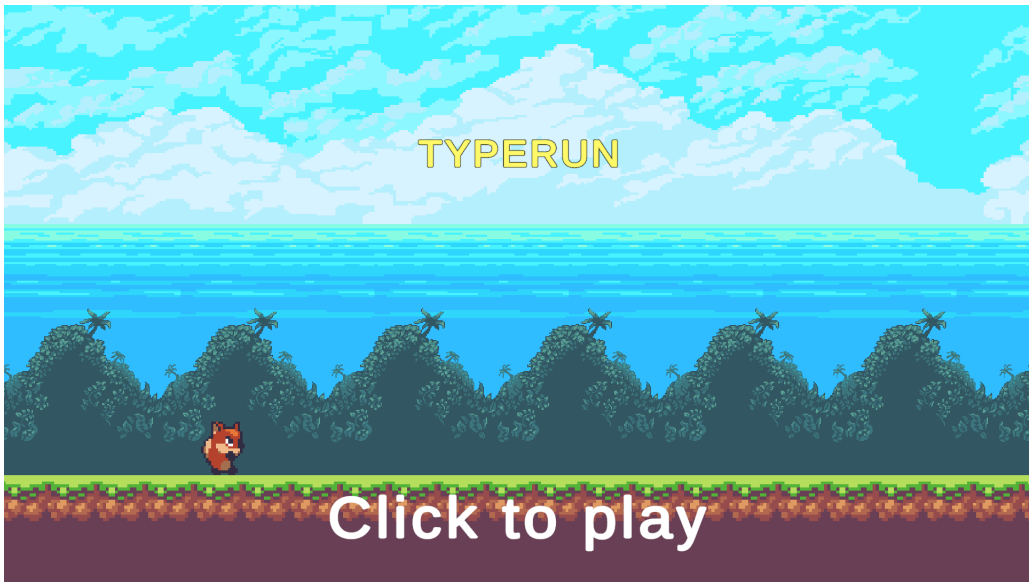

Figure 1: Title screen

The next screen is the game settings one (Figure 2).

To leave the game and return to computer's desktop, click on "Quit game" button on the bottom right corner of the screen.

To start a game, choose one theme among those who are available before clicking on "Run !" button. Notice that it is not clickable if you haven't chosen any theme. In that case, this button is visually darker than the "Quit game" one (see the picture one the left below).

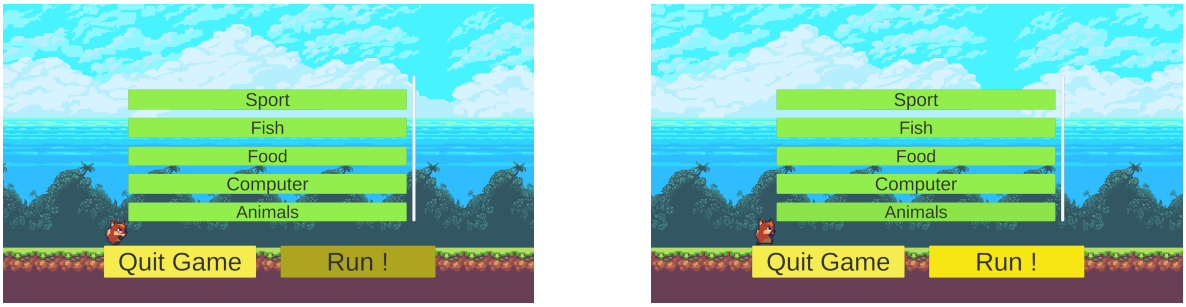

Figure 2: Game settings screens

Moreover, this version of the game provides six themes: Sport, Fish, Food, Computer and Animals. A list of words is associated to each one and each word is semantically linked to the theme it belongs to. Consequently, a word can appear in multiple themes.

The next figures will show you what is displayed on screen when a game has begun. The global interface is described on Figure 3. First, you can notice that the remaining lives are on the top left corner of the screen and the number of words successfully typed on the top right one.

Figure 3: Global interface of game screen

Notice also that during the game:

- the appearance frequency of rocks will increase, from 9 seconds between two rocks to 1 (without counting a bonus time between 0.5 and 2 seconds),
- the speed of the game will increase gradually,
- if you see multiple rocks on the screen, you can only destroy them in the order, from the leftmost to the rightmost one. You can't choose the rock to destroy first.

Figure 4 explains you what is going on while typing a word. To destroy a rock, you have to type the letter of the word in the order. Each correctly typed letter is colored in green and the remaining-to-type letters are in white.
However, if you make a mistake, the entire word is colored in red and has to be retyped from the beginning.

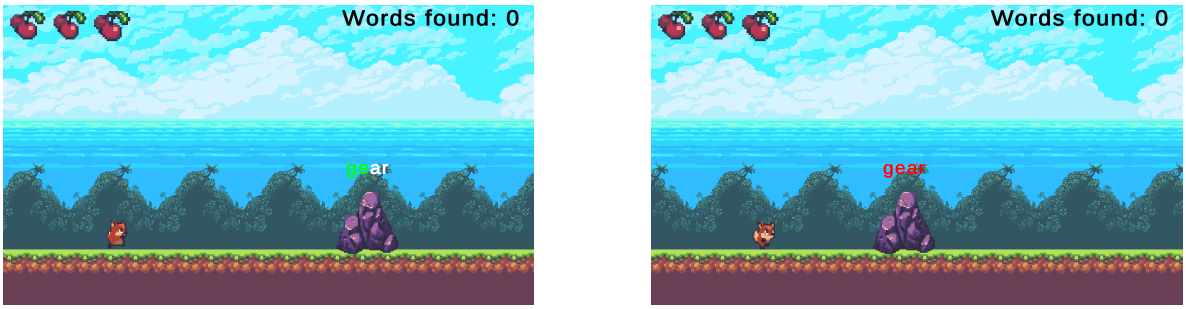

Figure 4: Correctly type vs wrongly typed word

If you succeed in typing the word, the rock will be destroyed and the word found counter will be incremented automatically. (Figure 5)

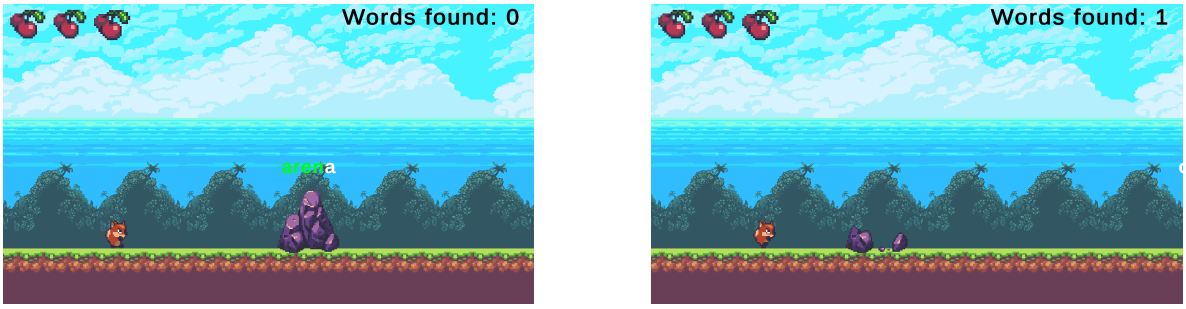

Figure 5: Destruction of the rock when a word is successfully typed

On the contrary, if you don't type the word in time, the rock will also be destroyed but instead of incrementing word found counter, a loss of a life will result. (Figure 6)

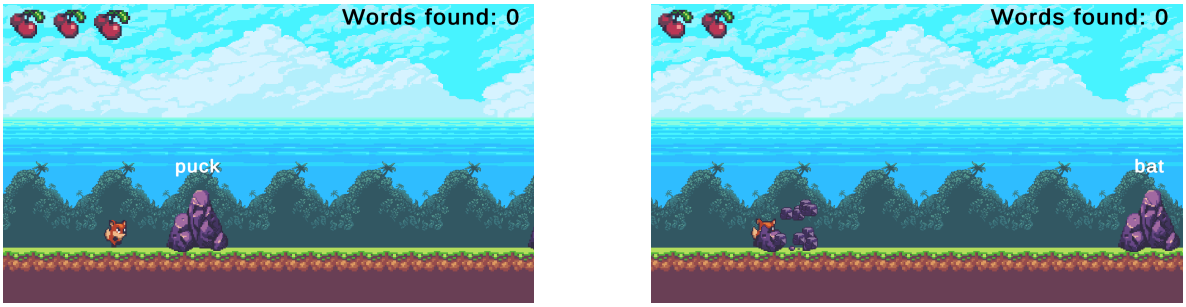

Figure 6: Lost of a life

Finally, if the last life is lost, the scenery stops going by and the little fox disappears. (Figure 7) A game over screen is displayed for a few second then the application automatically go back to title screen.

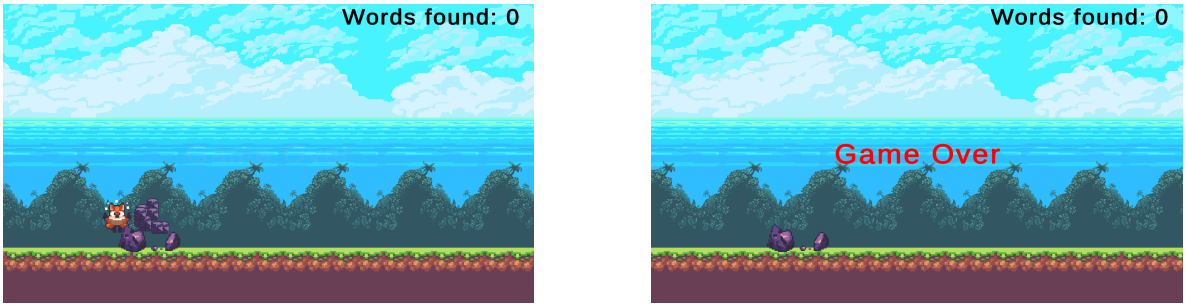

Figure 7: Game over screen

3 Technical specifications

3.1 Mise en place informatique

TypeRun est un jeu vidéo entrant dans la catégorie des runners. Inspirée du jeu du T-rex de Google Chrome [2], cette application est développée en C# *(prononcé "C sharp")* et utilise le moteur graphique Unity.[3] Elle est décomposée en deux interfaces qui interagissent entre elles :

- l'interface de programmation (API) qui gère l'exécution des scripts,
- l'interface utilisateur qui gère l'affichage graphique avec Unity en fonction des résultats de l'exécution des scripts.

La structure de l'API est présentée en Figure 8 et la description du rôle de chaque classe est détaillée ci-après.

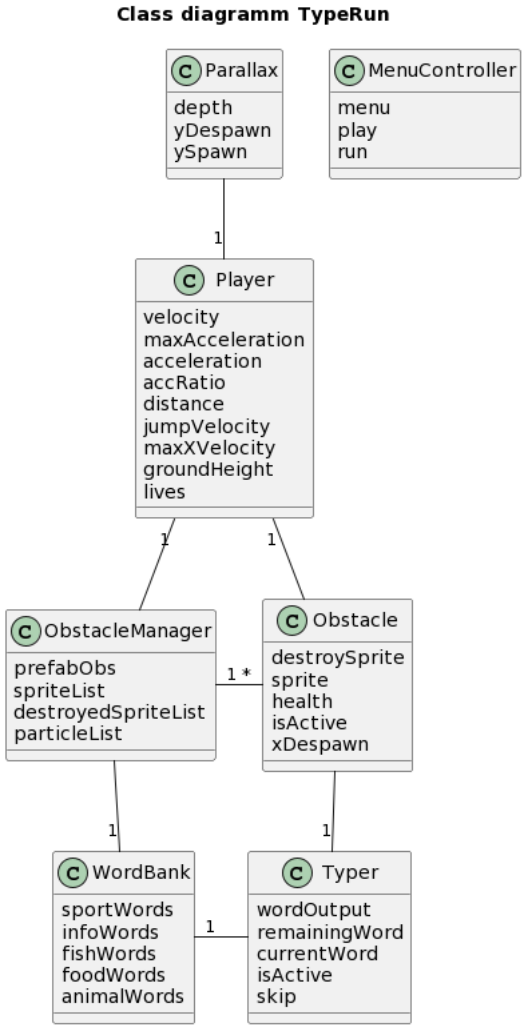

Figure 8: Diagramme de classe de l'API

- La classe `MenuController` gère les scripts liés aux écrans de titre et de paramétrage des parties.
- La classe `Parallax` gère les scripts de mise en mouvement du décor (le personnage reste fixe sur l'écran).
- La classe `Player` gère la vitesse de déplacement du renard, les points de vie et le lancement de l'animation du personnage lors d'un "game over".
- La classe `ObstacleManager` gère la création de nouveaux rochers, leur fréquence d'apparition et de leur mise à jour. Elle s'occupe également du nombre de mots tapés correctement et de l'affichage du texte "Game over".
- La classe `Obstacle` gère les caractéristiques d'un rocher. Elle indique également si le rocher est celui que l'utilisateur doit détruire si plusieurs rochers sont affichés sur l'interface graphique.
- La classe `Typer` gère le mot associé à l'obstacle (coloration des lettres déjà tapées et celles à dactylographier, détection des erreurs, ...).
- La classe `WordBank` gère la base de données des thèmes et des listes de mots qui leur sont associées.

Pour ce qui est de la partie graphique, celle-ci a été construite à l'aide du logiciel gratuit d'Unity. Les décors mis en place sont tirés de l'asset gratuit "Sunny land" [1] et la connexion à l'API est faite par l'association des différentes interactions de l'utilisateur aux actions à effectuer par le programme.

3.2 Difficultés rencontrées

Pour la réalisation de ce projet, nous avons fait face à une difficulté majeure liée aux technologies utilisées. Le choix du moteur graphique Unity et du langage C# est justifié par le fait que l'un des deux membres de l'équipe projet avait une petite expérience dans l'utilisation de ces deux outils. Cependant, l'autre membre de l'équipe n'avait pas le même niveau de maîtrise de ces technologies au départ (même si les paradigmes utilisés par le langage de programmation ont pu être appréhendés en cours). Il a donc fallu commencer le projet par une phase d'apprentissage et de mise à niveau des membres dans le but de gagner du temps sur la phase de développement.

3.3 Améliorations possibles

Nous pourrions améliorer le jeu, tout d'abord en augmentant le nombre de listes de mots ainsi que leur contenu. En effet, les six listes présentées ont été construites manuellement et ne sont constituées que d'une centaine de mots chacune. Apporter plus de diversité dans les thèmes et les mots proposés permettrait donc de réduire la probabilité de retomber sur les mêmes mots en effectuant plusieurs parties d'affilées. Nous pourrions également créer un système de statistiques avec notamment la possibilité pour l'utilisateur de voir le meilleur score qu'il a pu effectuer au cours d'une partie.

4 Conclusion

This project was a very interesting one because of two aspects. On one hand, it enables us to try something different compared to the other projects done within the scope of our academic courses. On the other hand, we have an occasion to work in a field we both like which is the video game design.

Finally, we know that this is the first version and despite the lack or experience and the limited time we have (compared to other video game), we succeed in creating a working application. A lot of improvements can be brought, and other idea can be added with more time to make the game even more amusing.

References

[1] Ansimuz. Sunny land assets. https://assetstore.unity.com/packages/2d/characters/sunny-land-103349?utm_source=google&utm_medium=cpc&utm_campaign=CC_DD_UnityPro_EMEA_EMEA_EN_AW_Display-PMAX_acquisition_PR_2023-03_TOFU_CC3022&utm_content=&utm_term=&gclsrc=aw.ds.

[2] Google Chrome. T-rex runner. https://dino-chrome.com/, 2014.

[3] Unity Technologies. Unity. https://unity.com/.