

---

## Rapport sur le projet de Labyrithe

---

Réalisés par  
Jurie Eleonore, Raffin Jonathan,  
Hmida Taha Yassine et Renaudin Valentin

Dans le cadre de  
Conception logiciel

Sujet choisi :  
Le Labyrinthe



Université de caen de normandie, Licence 1,informatique  
Janvier à Avril 2021

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Présentation et choix du sujet . . . . .	2
1.2	Description du projet . . . . .	2
<b>2</b>	<b>Réalisation du projet :</b>	<b>3</b>
2.1	Développement des idées . . . . .	3
2.2	Répartition du travail . . . . .	3
<b>3</b>	<b>Fonctionnalités :</b>	<b>4</b>
3.1	Choix des niveaux, menus . . . . .	4
3.2	Sauvegarde du temps, tableau des scores . . . . .	4
3.3	Récupération et placement de l'objet . . . . .	5
3.4	Autre . . . . .	5
<b>4</b>	<b>Élément Techniques :</b>	<b>6</b>
4.1	Bibliothèques et utilisations : . . . . .	6
4.1.1	Date time . . . . .	6
4.1.2	Pygame et Math . . . . .	6
4.2	Les Algorithmes . . . . .	7
4.2.1	Generation du Labyrinthe . . . . .	7
4.2.2	Rotation par degrés de la grille . . . . .	8
4.2.3	Deplacement de la balle . . . . .	9
4.3	Problèmes rencontrés . . . . .	11
<b>5</b>	<b>Architecture :</b>	<b>12</b>
5.1	Organisation du code (répartition des fichiers) . . . . .	12
5.2	Diagramme de classes + explications . . . . .	13
<b>6</b>	<b>Fonctionnement du jeu</b>	<b>14</b>
6.1	Manuel d'utilisation . . . . .	14
6.2	Captures d'écrans du jeu . . . . .	14
<b>7</b>	<b>Conclusion</b>	<b>16</b>
7.1	Resumé du projet . . . . .	16
7.2	Améliorations . . . . .	16

# **1 Introduction**

## **1.1 Présentation et choix du sujet**

Au premier TP du cours de conception logiciel, nous devions choisir un sujet. Nous avons hésité entre plusieurs comme le Puzzle-Quest, le Sokoban et le Labyrinthe. Nous avons donc choisi le Labyrinthe un peu comme un challenge car en lisant le sujet ce projet à l'air complexe.

Ce projet est donc la création d'un labyrinthe à gravité.

Dans ce rapport vous trouverez : la façon dont nous avons réalisé ce projet, une description complète des fonctionnalités, une explication sur les points techniques du code, des schémas de l'organisation du code, et pour finir un manuel d'utilisation de l'application.

## **1.2 Description du projet**

Le but de ce Labyrinthe est de le faire tourner sur lui-même. Dans celui-ci, se trouve une balle à faire déplacer jusqu'à l'arrivée via la gravité soumise à celle-ci. Les étapes clés dans la réalisation du jeu ont été : la génération du labyrinthe, la gestion de la rotation, puis l'affichage graphique du tout. Nous avons rajouté une petite difficulté à ce jeu, car avant que la balle finisse le jeu, elle doit récupérer un objet.

En travaillant sur ce projet, nous avons cherché un thème pour le Labyrinthe. Nous sommes partis sur le thème de l'Espace. Comme l'attaque des extra-terrestres sur Terre. Le joueur incarne le martien qui débarque sur Terre. Au passage, il doit récupérer un bidon d'essence pour pouvoir rentrer chez lui.

Par défaut la balle va toujours partir de en haut à droite pour arriver en bas à gauche. Il serait totalement possible de les déplacer à un autre endroit.

## **2 Réalisation du projet :**

### **2.1 Développement des idées**

Lors du début de ce projet, nous n'avons pas eu trop d'idée. En effet, nous n'avons trouvé aucun jeu de labyrinthe en ligne avec ce fonctionnement. Notre objectif était d'avoir un Labyrinthe qui tourne sur lui-même avec une balle soumise à la gravité. Puis au bout de quelques semaines, nous avons réussi cette objectif. Donc nous avons amélioré le jeu pour qu'il ne tourne pas directement à 90°, mais en plusieurs fois comme désormais, de l'angle que l'on veut, tant qu'il est un diviseur de 90 (45,30,15...). Nous avons fait ensuite, l'animation de la balle pour la voir descendre. Et fait l'ajout de l'objet à récupérer pour finir le niveau.

### **2.2 Répartition du travail**

Avant de commencer la conception du projet, nous nous sommes répartis les tâches. Éléonore à travailler sur la génération du Labyrinthe, de la gestion de la rotation, sur la balle et sur l'objet à récupérer. Jonathan à travailler sur tout ce qui est graphisme avec Pygame (affichage de la grille, gestion du menu, et des scores). Taha Yassine à créé le menu pour pouvoir jouer et Valentin à travailler sur ce rapport et les images du jeu.

Globalement nous avons travaillé chacun de notre côté, à notre rythme, en se tenant au courant de nos avancées. Nous nous sommes cependant aidés sur plusieurs éléments comme la rotation par degrés avec pygame, qui nous a pris beaucoup de temps.

L'utilisation, et la mise à jour régulière de notre travail sur la forge nous a permis, de mieux nous organiser et communiquer sur ce que l'on faisait.

## 3 Fonctionnalités :

### 3.1 Choix des niveaux, menus

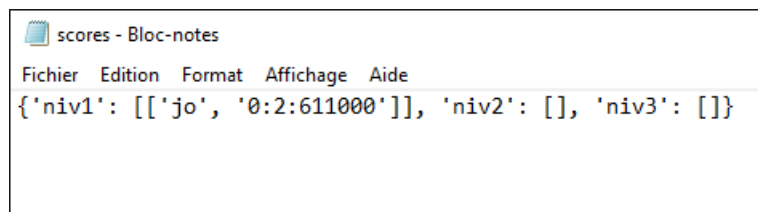
Plusieurs niveaux sont à votre disposition. Le premier niveau est le plus simple, puis plus on avance dans les niveaux, plus ils sont difficiles. Les différences entre les niveaux c'est que la taille du labyrinthe change donc le 3ème niveau va être beaucoup plus long que le 1er.

### 3.2 Sauvegarde du temps, tableau des scores

Après avoir fini un niveau, vous avez le choix de choisir si vous voulez sauvegarder votre score :

- Si vous choisissez de ne pas sauvegarder votre score, rien n'apparaît sur le tableau de score.
- Si vous choisissez de sauvegarder votre score, il apparaît sur le tableau de scores.

Les scores sont sauvegardés dans un fichier 'scores.txt'. Avant de sauvegarder un score, le programme regarde si le nom d'un joueur est déjà existant. Si ce nom existe alors il va comparer les 2 temps, l'ancien et le nouveau pour ensuite garder que le meilleur. Les scores sont enregistrés dans un dictionnaire, avec 3 clés pour les 3 niveaux. La valeur de chaque clé est une liste contenant autant de listes que de temps sauvegardé. Dans chaque "petite" liste, on retrouve le nom du joueur et le temps qu'il a mis. Au début le fichier est lu pour récupérer les scores et à la fin le programme enregistre le dictionnaire avec tous les scores dans le fichier.



```
Fichier Edition Format Affichage Aide
{'niv1': [['jo', '0:2:611000']], 'niv2': [], 'niv3': []}
```

FIGURE 1 – Sauvegarde dans le fichier scores.txt.

### 3.3 Récupération et placement de l'objet

Une fonction *poseObjet* place l'objet dans la grille en fonction de 2 nombres choisis aléatoirement dans une liste de nombre impaire. Ensuite, il y a la fonction *surObj* qui regarde si la base est à la même position que l'objet, si oui alors la variable *recupO* = 1.

Le joueur ne peut pas sortir du labyrinthe tant que l'objet n'a pas été recuperé.

### 3.4 Autre

D'autre fonctionatités on également été implementées :

- un timer, qui calcul le temps passé sur le niveau
- la rotation à 90 de la grille - le moteur physique (chute droite , ou relement a gauche ou à droite)

## 4 Élément Techniques :

### 4.1 Bibliothèques et utilisations :

#### 4.1.1 Date time

Nous avons utilisé la bibliothèque de pygame Date time, pour la gestion du chrono, grace à ces deux fonctionnalités :

```
self.chrono =datetime.combine(date.today(), time(0, 0))
self.label=Jeu.FONT.render(self.chrono.strftime("%M : %S : %f "),True, Jeu.BLANC)
|
def update_chrono(self):
    """Mise à jour du temps écoulé.
    dt est le nombre de millisecondes"""

    old_chrono = self.chrono
    self.chrono += timedelta(milliseconds=self.dt)
    if old_chrono != self.chrono:
        self.label = Jeu.FONT.render(self.chrono.strftime("%M : %S : %f "),
                                     True, Jeu.BLANC)
```

FIGURE 2 – Fonctions du gestion du chrono

avec lesquelles on peut obtenir très facilement un chronometre ( ici nous avons décidé d'afficher min ,sec mili , mais il est également possible d'ajouter les heures, jours, ans...)

chrono inspiré d'un code trouvé sur ce forum :

<https://openclassrooms.com/forum/sujet/chronometre-sous-pygame>

#### 4.1.2 Pygame et Math

Pour la partie graphique du Labyrinthe, nous avons utilisé le module Pygame. C'est un des deux modules de python les plus connus pour la graphisme. Pour mieux gérer l'affichage en pygame, nous avons créé une Class qui prend en argument une grille. Nous avons une première fonction qui permet juste d'afficher une grille. Cette fonction va parcourir toute la grille en lisant les différents symboles de celle-ci. Ensuite pour chaque symbole, la fonction va lui attribuer une image différente comme par exemple un mur ou la balle.

Le but de ce Labyrinthe est de le tourner sur lui-même pour pouvoir faire bouger la balle avec gravité. Il nous faut donc une fonction capable de tourner celui-ci. Cette fonction est appelée via les flèches du clavier et donc elle prend en argument une direction, soit gauche ou droite. Mais pour que ce Labyrinthe puisse tourner correctement il fallait calculer, pour chaque image, la rotation de celle-ci par rapport au centre de la fenêtre. Et sur cette dernière fonction qui

nous a longuement posé problème. Le problème est détaillé dans les prochaines parties. En plus de la rotation dans la fenêtre, les images doivent tourner sur eux-mêmes (pour ne pas faire comme une sorte d'escalier quand le Laby sera en diagonale). Pour ce faire nous avons utilisé la commande déjà prédéfini par pygame : `pygame.transform.rotate`. La commande précédente prend une image ou un objet à tourner et le degré d'inclinaison souhaité. Mais nous avons aussi rencontré un souci sur cette fonction.

En combinant ces différentes fonctions dans une boucle de jeu, le Labyrinthe s'affiche et tourne correctement.

## 4.2 Les Algorithmes

### 4.2.1 Generation du Labyrinthe

L'algorithme de generation d'un grille de labyrinthe une une des partie clé du projet. Pour le realiser nous nous somme basée sur la methode de "Recursive Backtracking" qui utilise une fonction recursive. (algorithme trouvé ici : <http://weblog.jamisbuck.org/2010/12/27/maze-generation-recursive-backtracking>) Pour realisé cet algorithme nous avons eu besoin de plusieurs éléments. Premièrement, une grille avec un quadrillage :

```

X X X X X X X X X X
X p p p p p p p p X
X p # p # p # p # p X
X p p p p p p p p X
X p # p # p # p # p X
X p p p p p p p p X
X p # p # p # p # p X
X p p p p p p p p X
X p # p # p # p # p X
X p p p p p p p p X
X X X X X X X X X X

```

FIGURE 3 – Exemple grille quadrillée.

Deuxiemement, une fonction qui nous donne une liste de 4 directions dans un ordre aleatoire :

Et enfin de la fonction qui trace le labyrinthe. Cette fonction utilise le principe suivant : prend une position x,y

- transforme en case (x,y) en case vide
- appelle la fonction pour les coordonnées
- pour chaque direction dans la fonction verifie si la case à deja été visitée ( non vide) et si il est toujours dans la grille , dans ce cas, x,y changent et la fonction est reappelée.



```

X X X X X X X X X X
X p p p p p p p p X
X p # p # p # p # p X
X p p p p p p p p X
X p # p # p # p # p X
X p p p p p p p p X
X p # p # p # p # p X
X p p p p p p p p X
X p # p # p # p # p X
X p p p p p p p p X
X X X X X X X X X X

```

FIGURE 4 – Fonction direction

```

X X X X X X X X X X X X X
X p p p p p p p p p p p X
X      p      p      p X
X p p p      p p p      p p X
X p      p      p      p      p X
X p      p p p      p p p      p X
X p      p      p      p      p X
X p      p      p      p      p X
X p      p      p      p      p X
X p      p      p      p      p X
X p      p      p      p      p X
X p      p      p      p      p X
X p      p      p      p      p X
X p      p      p      p      p X
X p      p      p      p      p X
X p      p      p      p      p X
X X X X X X X X X X X X X

```

FIGURE 5 – Grille après tracage du labyrinthe.

#### 4.2.2 Rotation par degrés de la grille

La rotation de la grille pour l’affichage pygame s’effectue par la fonction rotation qui prend en argument une direction. Puis modifie le position initiale de chaque image en fonction d’un angle . Nous avons donc créé un fonction qui prend en argument la position initiale et l’angle d’inclinaison (en radiant) et retourne ses nouvelles coordonnées.

```

def calcrotate(self,a,b,angle):
    angle=math.radians(angle)
    px = a
    py = b
    x= self.ox + math.cos(angle) * (px - self.ox) - math.sin(angle) * (py - self.oy)
    y= self.oy + math.sin(angle) * (px - self.ox) + math.cos(angle) * (py - self.oy)
    return x,y

```

FIGURE 6 – Calculs des nouvelles coordonnées de chaque images.

L’angle d’inclinaison est modifié à chaque fois que la fonction est appelée, il est incrémenté de plus ou moins un certain degré (definit en attribut de classe) La

fonction modifie alors chaque image (contenues dans un dictionnaires) et actualise la fenetre a l'aide d'un *display flip* :

### 4.2.3 Déplacement de la balle

La balle est gerée differement de la grille, c'est un element de type Sprite.

#### Déplacement dans la grille :

La balle (ou vaisseau avec les images) est definie par 2 variables ( la position en x et en y) Ces variables vont être modifiée lors des déplacement, à chaque fois que la position change.

La balle se deplace dans la grille a l'aide de deux fonctions : *roule(direc)* et *gravite(direc)* La première déplace la bille vers la droite ou la gauche, jusqu'à un mur. Pendant son déplacement la deuxième et appelée a chaque fois et regarde si une case en dessous est vide. Ces deux fonctions créent ainsi une boucle qui emène la balle le plus loin possible.

Le parcours de la bille est lui meme enregistré dans une liste `[[x,y][x,y]..]` pour être utilisé plus tard dans l'affichage

Cela peut alors créer des motifs de ce type lors par exemple d'un mouvement à droite (en majuscule le debut et l'arrivée :

```
B
b b
  b b b b B
```

#### Representation du déplacement dans pygame :

Contrairement à aux autres élément de la grille, la position de la balle n'est pas fixe, c'est pourquoi nous avons privilégié l'utilisation de *Sprite*. Nous avons donc une classe *SpriteBall* avec 2 methodes : *update* et *calcrotate* Lors du 1er affichage de la grille, la classe Sprite est initialisée avec le parametre de la bille ( image, emplacement, et centre de rotation) Puis ce sprite est ajouté à un *sprite.Group()*. Ensuite lors de l'appel de la fonction rotation, la bille va être déplacée grace a la methode *update(px,py,angle)* qui calcule la nouvelle position avec le meme principe que pour la grille. La grande difference entre les deux et que *self.rect.move(x,y)* fonctionne differement avec les Sprites. X,Y ne sont plus les nouvelles positions, mais le decalage par rapport a la position d'avant. Il faut donc penser à soustraire les anciennes coordonnées.

L'animation se fait en 3 etapes : Pour chaque coordonnées contenues dans la liste de chemin de la balle :

1. Affiche la balle au bon endroit avec les methodes *.update* et *.draw* de Sprite et *display.flip*

```

def update(self,px,py,angle):
    self.calcrotate(px,py,angle)
    self.rect=self.rect.move(self.dx,self.dy)

def calcrotate(self,x,y,angle):
    angle=math.radians(angle)
    self.dx=(self.cx + math.cos(angle) * (x - self.cx) - math.sin(angle) * (y - self.cy))-self.rect.x
    self.dy=(self.cy + math.sin(angle) * (x - self.cx) + math.cos(angle) * (y - self.cy))-self.rect.y

```

FIGURE 7 – Calculs des nouvelles coordonnées de la balle.

2. pour permettre de voir , le mouvement, il faut le ralentir, pour se faire nous utilisons : *py.time.delay(int)*
3. Pour finir il faut recouvrir l'image d'une autre pour ainsi ne pas avoir une ligne de vaisseau.



FIGURE 8 – Resultat de l'animation.

### 4.3 Problèmes rencontrés

Dans la partie Pygame, nous parlons d'un autre problème sur la rotation des images mais sur elles-mêmes. Au début du projet, nous avons cherché l'image d'un mur sur internet. Puis ne trouvant pas d'autres images qui nous plaisait pour le projet, on avait décidé de créer nos propres images. Ces images étant faites sur Paint était en format jpeg (.jpg) sauf qu'avec la fonction de Pygame : `pygame.transform.rotate(img, angle)`, les images en .jpg ne tournaient pas sur elles-mêmes. Alors nous avons exporté ces images en .png puis ceci à fonctionner.

Puis nous avons eu un souci de dernière minute. Pour l'animation de la balle nous voulons une animation pour qu'on la voit descendre mais on voulait quelque chose de discret comme de la fumée pour derrière la vaisseau. Sauf que nous voyons un petit peu le vaisseau en dessous de la fumée. Ce problème est présent car la méthode `pygame.transform.rotate(img, angle)` ne fonctionne pas avec les sprites, l'image du vaisseau ne tourne donc pas sur lui-même, ce qui laisse apparaître un bout sur le côté. Nous obtenons l'erreur suivante :

```
self.image = py.transform.rotate(self.image, int(-self.angle))
AttributeError : 'float' object has no attribute 'transform'
```

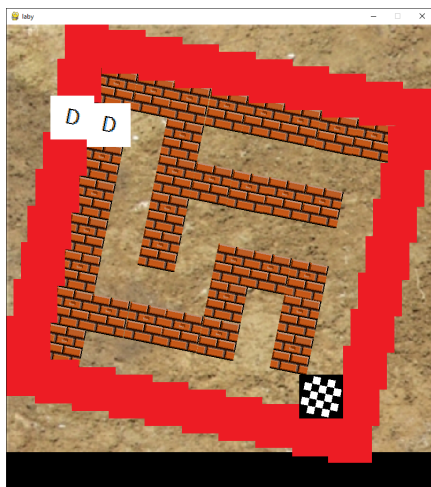


FIGURE 9 – Erreur aux niveaux des formats des images.



FIGURE 10 – Erreur problème d'affichage sur l'animation

## 5 Architecture :

### 5.1 Organisation du code (répartition des fichiers)

Nous avons organisé ce projet en différents fichiers avec comme premier fichier, celui qui va générer une grille pour ensuite générer un Labyrinthe. Le deuxième fichier nous permet de mettre celui-ci en pygame, donc tous ce qui va permettre l’affichage dans une fenêtre. Puis le dernier fichier avec la boucle principale du jeu. Dans ce fichier se trouve toutes les fonctions qui permettent de faire fonctionner le menu, les sauvegardes des scores, et le lancement des différents niveaux.

Grille	class_pygame	Jeu
import random	import pygame as py import grille_1 as gr import math	import pygame Import class_pygame import grille_1 from datetime import timedelta datetime, date, time
Grille Laby	SpBall Labypygame	Jeu main

FIGURE 11 – Shema des fichiers.

En haut se trouve le nom des fichiers, en dessous, les bibliotheques ou les autres fichiers importés et enfin juste en dessous en rouge les classes et en bleu le *main*

## 5.2 Diagramme de classes + explications

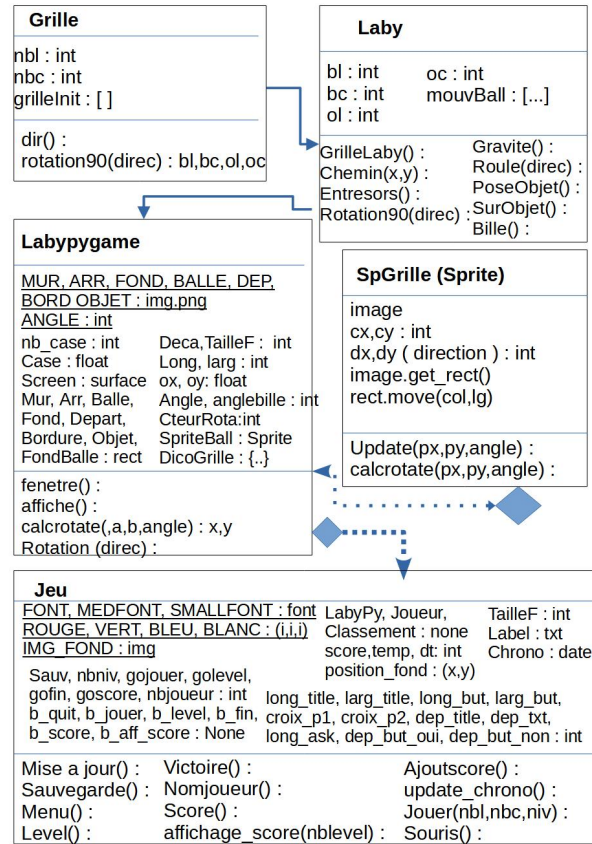


FIGURE 12 – Diagramme de classes.

Les variables en majuscules et soulignées correspondent aux attributs de classes, Les flèches simples aux héritages, et les flèches losanges pointillées à l'utilisation de la classe (coté losange) par une autre (flèche)

## 6 Fonctionnement du jeu

### 6.1 Manuel d'utilisation

Comment lancer un niveau ? Comment sauvegarder et visualiser ses scores ? Voici un petit résumé de comment utiliser et se repérer dans le menu.

1. Lancer le jeu (fichier "jeu"), et cliquer sur le bouton "Jouer" :  
En cliquant sur celui-ci, vous aurez accès à 3 niveaux. Le premier niveau est très simple, puis les 2 autres niveaux sont un peu plus complexes.
2. Se déplacer dans le labyrinthe à l'aide des fleches droite et gauche. :  
Appuyer sur la fleche de gauche pour faire pencher le labyrinthe sur la gauche et ainsi faire rouler la bille et sur la fleche de droite pour le faire pencher de l'autre coté.
3. Récupérer l'objet ( le bidon d'huile)
4. Arriver jusqu'à l'humain pour valider le niveau
5. Sauvegarder votre score :  
soit vous ne souhaitez pas le sauvegarder et vous retournez directement au menu. Ou soit vous sauvegardez votre temps, et vous pourrez ainsi taper votre prénom, puis appuyer sur Entrée pour valider.
6. Afficher les scores :  
en dessous des 3 boutons pour les niveaux, se trouve le bouton "Scores". Vous pourrez voir vos scores classés par niveaux.
7. Quitter le jeu :  
pour quitter le jeu cliquez sur la croix rouge en haut à droite

### 6.2 Captures d'écrans du jeu

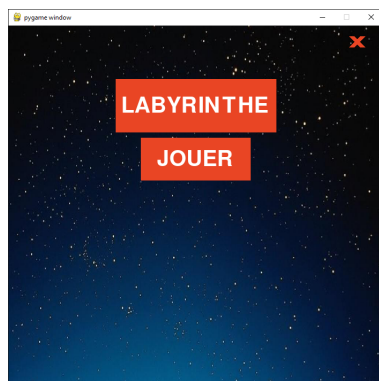


FIGURE 13 – Menu du jeu.

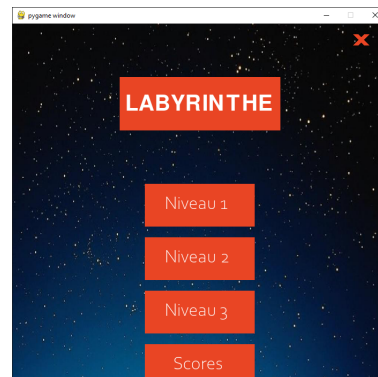


FIGURE 14 – Menu pour accéder aux niveaux.

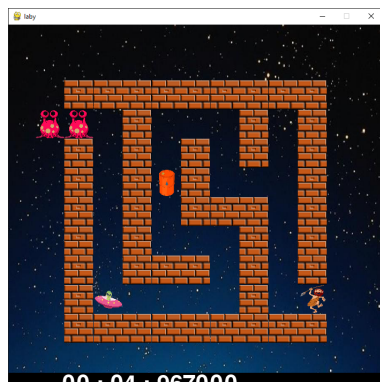


FIGURE 15 – Affichage du niveaux.

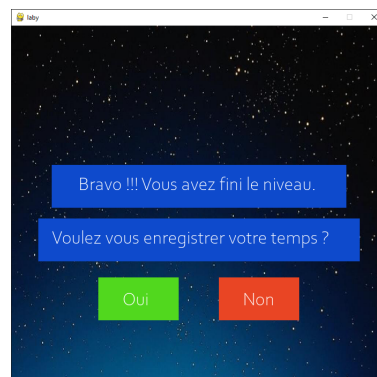


FIGURE 16 – Demande de sauvegarde de votre temps.

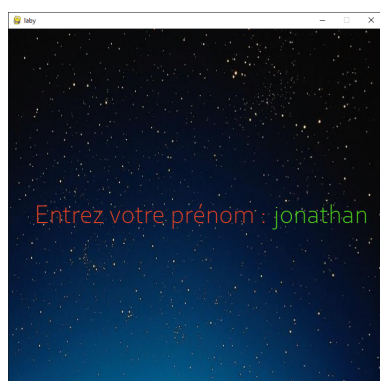


FIGURE 17 – Demande de rentrer votre nom.

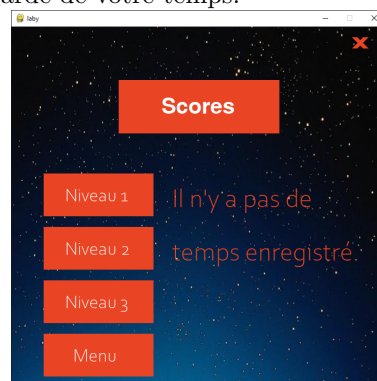


FIGURE 18 – Affichage des scores, si il n'y en as pas.

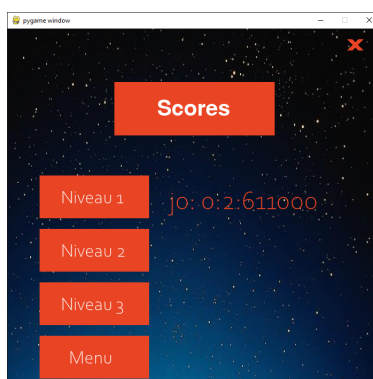


FIGURE 19 – Affichage des scores, si il y en a.



## 7 Conclusion

### 7.1 Résumé du projet

Pour résumer ce projet, on peut dire qu'il aura été difficile à cause des plusieurs problèmes rencontrés. Mais nous avons réussi à les résoudre et de voir notre projet enfin fini nous rend plutôt fier de ce travail.

### 7.2 Améliorations

Nous avons pris beaucoup de plaisir à programmer ce labyrinthe, malheureusement par manque de temps nous n'avons pas pu améliorer le jeu comme on le souhaitait. Nous aurions bien voulu avoir un peu plus de niveau, pour plus de difficulté. Nous aurons bien que dans le jeu, on voit la balle descendre correctement avec la gravité. Nous aurions souhaité voir faire tourner le Labyrinthe degrés par degrés et non par 45 degrés. Nous avons eu la possibilité de faire cette option mais de l'autre côté, ceci générerait beaucoup de problèmes pour la balle. Sachant que nous avons dû finir vite pour rendre ce projet, nous avons joué sur la simplicité plutôt que de se compliquer et raté ce projet. """Update le 13/04""" Finalement nous avons pu réaliser certaines des fonctions précédemment citées, Les améliorations suivantes, seraient alors : de créer une rubrique avec les règles de jeu, avoir des challenges avec les temps ( le laby devient noir pendant qq secondes, ou alors toutes les X secondes une tempête intervient et tout bouge de place ...)