UNIVERSITÉ
CAEN
NORMANDIE

Rapport projet conception logicielle

PyPuzzle

Romuald GAFFE
Yanis COSNEFROY
Nguyen Phuong Vy VU
Alexandre BOURGOIN

2 avril 2020

Table des matières

Première partie

Objectifs du projet

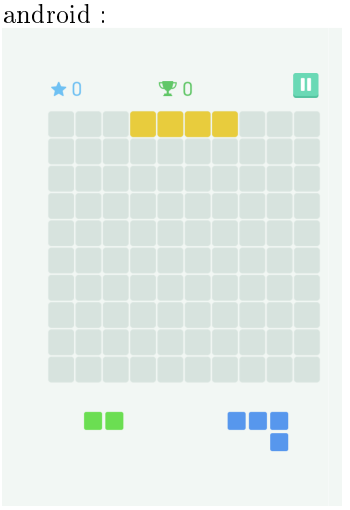
0.1 Présentation du concept

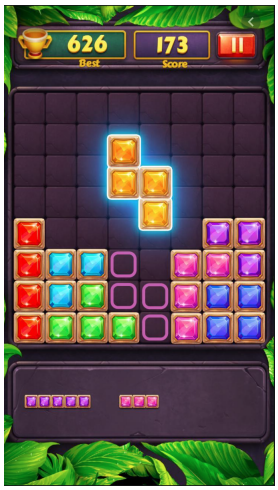
Tout d'abord, nous avons choisi de créer un puzzle block puisque nous avions plus d'affinité et d'idées en rapport avec ce sujet. Nous voulions que ce jeu soit simple d'utilisation afin que tout le monde puisse y jouer. Le jeu que nous avons développé est composé de 4 modes de jeux, le premier étant le mode de jeu solo, il se présente sous forme d'une grille carré de 10 cases de côté. Le joueur obtient 3 pièces à placer sur la grille tirées aléatoirement parmi un total de 30 pièces(toutes orientations inclus). Les pièces qui apparaissent doivent toutes être placés dans la grille avant que les suivantes soit tirés. Si le tirage courant du joueur ne peut pas être placé alors un écran de fin de jeu s'affiche, montrant le score de la personne. Sur ce menu, le joueur aura la possibilité de retourner au menu principal ou de recommencer une partie. Ensuite le jeu se compose d'un mode multijoueur. Le premier étant contre une intelligence artificielle ayant le même mode de fonctionnement que le mode solo. Seul le score actuel de l'intelligence artificielle figure sur l'écran du joueur afin d'éviter d'en copier la stratégie. Le second mode multijoueur est, lui, un mode joueur contre joueur en local sur une même machine. Les grilles des joueurs s'affichent l'une après l'autre afin de limiter, de même, l'imitation de la stratégie adverse. Le dernier mode de jeu est un mode joueur contre joueur également mais en ligne où le but est de faire le plus de points.

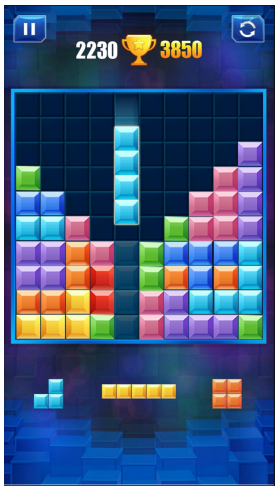
0.2 Cahier des charges

0.3 Exemples du jeux publiés

Voici plusieurs block-puzzle provenant de jeux.fr, et des applications pour android :







On peut y retrouver les mêmes caractéristiques que notre projet. Notre projet a été conçu sur une base classique de block-puzzle qui réunit ces trois images montrées précédemment. Les différences entres ces block-puzzles et le nôtre sont l'aspect graphique ainsi que la sauvegarde du meilleur score établis sur le jeu par le joueur. On peut constater à l'essai de notre block-puzzle uniquement l'ap-

parition des scores qui vont être actifs en fonction du jeu du joueur. Cependant nos règles sont un peu différentes par rapport au jeu déjà publié. On a décidé de ne pas rajouter de bonus ou d'aide à chaque fois qu'une ligne ou une colonne est remplie. C'est un choix volontaire pour provoquer une augmentation de concurrence de jeu avec d'autres joueurs mais aussi pour une adaptation plus développée de la difficulté de jeu.

Deuxième partie

Fonctionnalités implémentés

0.4 Mode solo

Tout d'abord la structure du jeu en mode solo est une structure de jeu plutôt classique comme on peut le voir dans l'image présente ci-dessous :

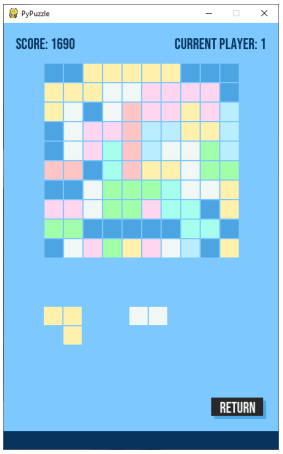


FIGURE 1 – Gameplay solo

Comme on peut le constater à l'image du projet, il y a une grille placée au centre de la fenêtre qui possède une taille fixe. Autour de cette grille, va se trouver des informations essentielles pour le joueur comme son score actuel qui est placé en haut à gauche de la fenêtre. À droite, va se trouver le joueur qui joueur qui joue au jeu. En dessous de la grille, on va retrouver les 3 blocs de départ que le joueur devra placer dans la grille. Enfin, tout en bas à droite, il va y avoir un bouton pour permettre à l'utilisateur de retourner au menu du jeu.

Ensuite, les règles du jeu vont s'appliquer au fur et à mesure du jeu du joueur. Quand un block sera placé dans la grille, il va y avoir un ajout de 30 points à son score. Et dès qu'une ligne ou une colonne sera remplie, il va y avoir une disparition de la ligne ou de la colonne voir même les deux si les conditions sont réunies. Le bonus de point sera alors de 100 points lorsqu'une ligne ou une colonne est construite par l'utilisateur. Le joueur ne peut pas mettre sa partie en pause. Cependant, il n'y a pas de limite de temps. Dès que les trois blocs sont placés dans la grille, trois autres bloques vont réapparaître avec une probabilité de difficultés qui est calculée en fonction du score.

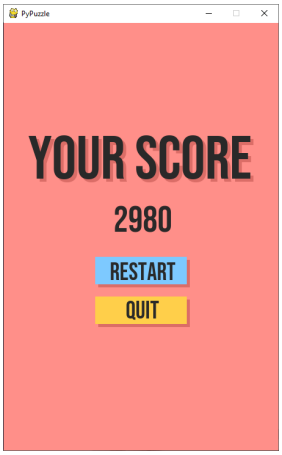


FIGURE 2 – Gameplay solo

Enfin, si je le joueur perd en étant bloqué sur le placement impossible d'une pièce dans la grille, alors le jeu s'arrêtera et affichera un Game Over montré ci dessous avec les information de jeu du joueur :

Comme on peut le voir sur cette photo, le score du joueur est affiché. Afin d'éviter au joueur de repartir dans le menu pour relancer une partie, il a la possibilité de la relancer directement grâce au bouton "restart". Il a aussi la possibilité de quitter le jeu en cliquant sur le bouton "quit". Si le joueur veut retourner au menu, alors il devra d'abord recommencer une partie et cliquer sur le bouton "return" se trouvant en bas à droite de la fenêtre.

0.5 Mode multijoueur

Différents modes de multijoueur sont présents dans ce projet. On peut y retrouver un multijoueur où un joueur réel va jouer contre une intelligence artificielle, un multoujoueur ou deux joueur vont s'affronter. Et enfin un multijoueur en réseau où deux joueur vont pouvoir s'affronter sur deux écrans séparés. En dessous va se trouver une photo ou va être représenté le menu du multijoueur. On va aussi y retrouver la possibilité d'un retour au menu principal dans le menu du multijoueur si le joueur veut jouer seul.

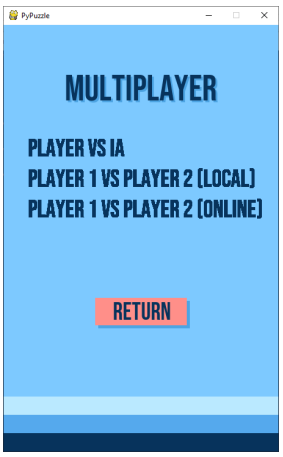


FIGURE 3 – Menu multijoueur

0.5.1 Joueur vs Intelligence artificielle

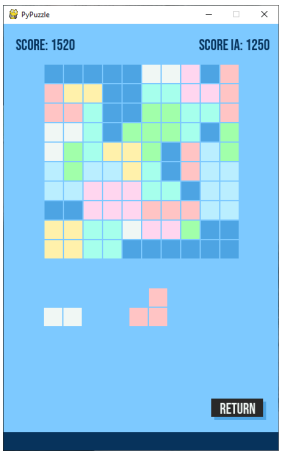


FIGURE 4 – Joueur vs IA

Tout d'abord, il faut savoir que le mode multijoueur avec le joueur contre l'intelligence artificielle se joue comme le mode solo. Le joueur va pouvoir jouer comme il voudra. Il n'aura pas accès à la grille de l'intelligence artificielle pour éviter de la tricherie. Cependant le joueur pourra jouer une fois sur deux. Le premier tour est commencé par l'intelligence artificielle puis est suivie du joueur se trouvant derrière l'écran. L'utilisateur va avoir comme information sa grille, ses points, les points de son adversaire, donc ici de l'intelligence artificielle et de ses blocs. Le joueur et l'intelligence artificielle vont tout les deux avoir les mêmes blocs pour que les difficultés soient égales de chaque côté. Le joueur va pouvoir retourner au menu si il le souhaite, toujours grâce au bouton se trouvant en bas à droite. Ci dessous, va se présenter une photo de la grille du jeu avec les différentes informations énoncées précédemment que le joueur va avoir.

Ensuite les règles du jeu sont les mêmes que pour le solo. Il n'y a pas de mise difficultés pour qui que ce soit car si l'un rempli une ligne ou une colonne, l'autre ne va pas recevoir de pièce qui peut le mettre en difficulté. Ici il y a juste une concurrence au niveau des points qui va justement permettre de départager la victoire entre le joueur et l'intelligence artificielle.

Enfin, si je le joueur perd en étant bloqué sur le placement impossible d'une pièce dans la grille, alors le jeu s'arrêtera et affichera un Game Over montré ci dessous avec les information de jeu du joueur. Si le jeu ne s'arrête pas c'est qu'il y a encore une possibilité de jouer.

Comme on peut le voir sur cette photo, le score du joueur ainsi que celui de l'intelligence artificielle sont affiché. De plus on a décidé de rajouter une ligne pour faire ressortir le gagnant de la partie. Dans cette image, Player1 est le joueur se trouvant derrière l'écran et Player2 est l'intelligence artificielle. Afin d'éviter au joueur de repartir dans le menu pour relancer une partie, il a la possibilité de la relancer directement grâce au bouton "restart". Il a aussi la possibilité de quitter le jeu en cliquant sur le bouton "quit". Si le joueur veut retourner au menu, alors il devra d'abord recommencer une partie et cliquer sur le bouton "return" se trouvant en bas à droite de la fenêtre.

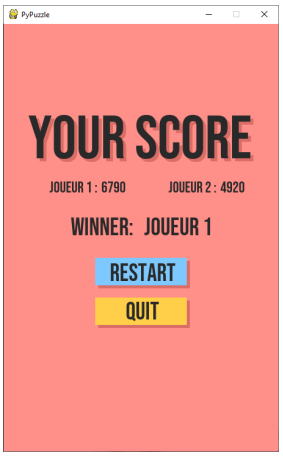


FIGURE 5 – Fin du jeu multi

0.5.2 Joueur vs Joueur en local

0.5.3 Joueur vs Joueur en ligne

Troisième partie

Éléments techniques

0.6 La grille

La grille est mise sous la forme d'une classe,

0.7 Les pièces

0.8 L'aléatoire

0.9 L'intelligence artificielle

0.10 Le réseau

0.11 L'interface graphique

Pour l'interface du jeu, on utilise ***Pygame***, une bibliothèque graphique dédiée aux jeux vidéos. Pygame n'est pas fourni avec Python. Donc, pour jouer, vous devrez télécharger et installer Pygame à la place. Pour appliquer Pygame :

```
import pygame as pg
```

On va utiliser pygame pour faire et design l'interface du jeu, pas utiliser les images déjà faits et les mettre dans le jeu. Et on va appliquer en même méthode pour faire le menu démarrer, le menu multijoueur et le menu de la fin du jeu.

0.11.1 Le menu

D'abord, il faut faire un réglage des tailles des positions. ***pygame.Rect()*** n'est pas dessiner un rectangle, juste déterminé les tailles et les positions d'un rectangle.

```
#tailles de fenetre
SCREENHEIGHT = 700
SCREENWIDTH = 450
screensize = (SCREENWIDTH, SCREENHEIGHT)
#positions de la grille
boardX = 65
boardY = 65
#tailles et positions des boutons
soloButtonRect = pg.Rect(150, 318, 150, 50)
multiButtonRect = pg.Rect(150, 383, 150, 50)
quitButtonRect = pg.Rect(150, 448, 150, 45)
```

Pour commencer le jeu, on doit appeler la fonction ***pygame.init()***, elle doit être appelée en premier pour que de nombreuses fonctions Pygame fonctionnent

```
pg.init()
```

Ensuite, on doit construit un boucle pour tout le menu. Si une boucle est toujours vrai, alors elle fonctionne pour toujours.

```
def menu():
doContinue = True
while doContinue:
    for event in pg.event.get():
        if event.type == pg.QUIT:
            fnc.quitGame() #dans function.py pour quitter
```

Puis, on ajoute des boutons, on utilise ***.collidepoint()*** pour déterminer les positions des boutons.

```
                elif event.type == pg.MOUSEBUTTONDOWN:
                if soloButtonRect.collidepoint(event.pos):
                    solo() #autre menu pour basculer entre eux
                if multiButtonRect.collidepoint(event.pos):
                    multiMenu() #autre menu pour basculer entre eux
                if quitButtonRect.collidepoint(event.pos):
                    fnc.quitGame() #dans function.py pour quitter
        pg.display..flip() #pour mise a jour
```

Ensuite, on construit une interface d'affichage dans une autre fichier appelée **display.py**. Pour les textes, on utilise ***pygame.font*** avec les sous-éléments ***pygame.font.Font()***, ***"nom".render()*** et l'élement ***"screen".blit()***. Pour les rectangles (boutons), on utilise ***pygame.draw.rect()***. Et pour colorer le fond, on utilise ***"screen".fill()***.

```
NAVY = (7, 51, 92)
pg.font.init()
bigFont = pg.font.Font('assets/BebasNeue-Regular.ttf', 100)
pypuzzle = bigFont.render("PyPuzzle", True, NAVY)

def displayMenu(win):
win.fill(BACKGROUNDCOLOR)
pg.draw.rect(win, BOARDCOLOR, (155, 323, 150, 45))
pg.draw.rect(win, BOARDCOLOR, (155, 388, 150, 45))
pg.draw.rect(win, BOARDCOLOR, (155, 453, 150, 45))
#titre
win.blit(pypuzzleShadow, (82, 170))
win.blit(pypuzzle, (78, 165))
#decoration
pg.draw.rect(win, LIGHTBLUE, (0, 610, 450, 30))
pg.draw.rect(win, BLUE, (0, 640, 450, 30))
pg.draw.rect(win, NAVY, (0, 670, 450, 30))
```

Pour les son, on utilise ***pygame.mixer*** avec les sous-éléments ***pygame.mixer.Sound()***, ***"nom".play()*** et ***"nom".stop()***. Type de fichier est .wav (Les formats de fichiers audio pris en charge par Pygame sont MID, WAV et MP3.)

```
pg.mixer.init() #pour fonctioner
soundMenu = pg.mixer.Sound("assets/menu.wav")
soundMenu.play(-1, 0, 0)
```

On juste utilise une animation, c'est le survol quand on entre les boutons. Ici, on utilise ***pygame.mouse*** avec le sous-élement ***pygame.mouse.get_ pos()***. pour déterminer les positions de la souris.

```
# HOVER
pos = pg.mouse.get_pos()
if 340 + 85 > pos[0] > 340 and 615 + 30 > pos[1] > 615:
    pg.draw.rect(screen, dsp.YELLOW, (340, 615, 85, 30))
    screen.blit(dsp.returnMenuText, (354, 617))
else:
    pg.draw.rect(screen, dsp.GRAY, (340, 615, 85, 30))
    screen.blit(dsp.returnMenuText1, (354, 617))
```

0.11.2 Gameplay

Quatrième partie

Architecture du projet

0.12 Diagrammes des modules et des classes

0.13 Cas d'utilisation

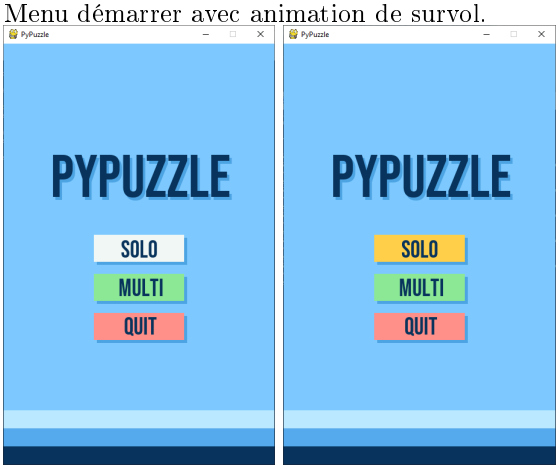
Cinquième partie

Expérimentations et usages

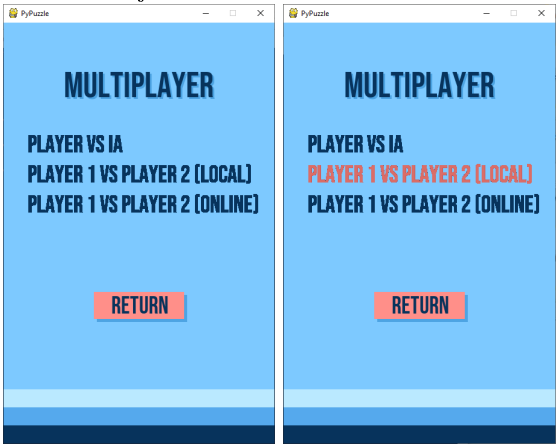
0.14 Capture écrans

Les captures ci-dessous rendront le joueur passionné de jouer au jeu.

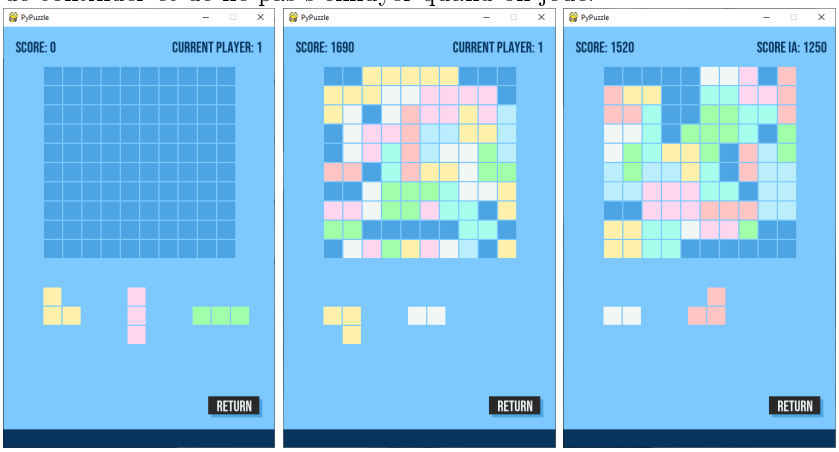
1. Menu démarrer avec animation de survol.



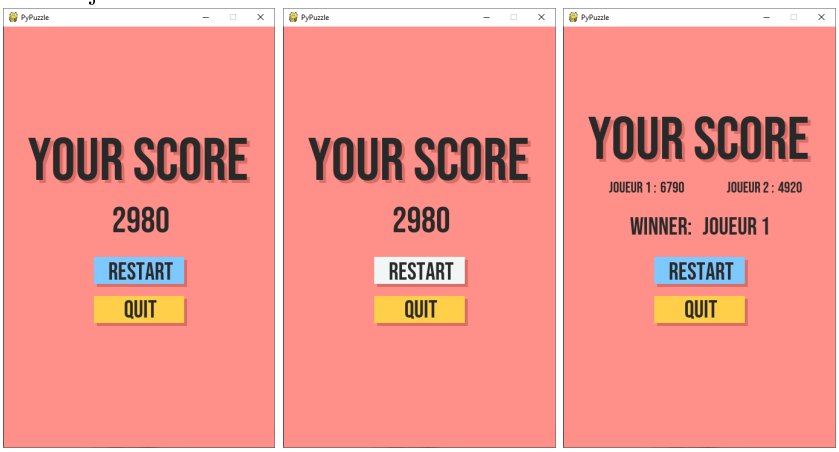
2. Menu multijoueur avec animation de survol.



3. Gameplay (solo et multi). De nombreuses couleurs permettent au joueur de continuer et de ne pas s'ennuyer quand on joue.



4. Fin du jeu avec animation de survol.

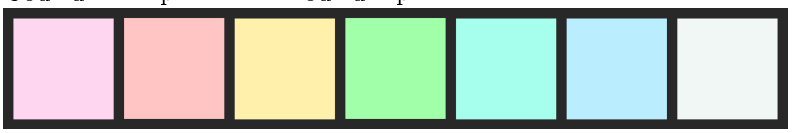


0.15 Mesures de performance

0.15.1 Interfaces graphiques

Quelle est l'interface du jeu pour vous ? Selon nous, après le contenu du jeu, l'interface du jeu est la plus importante pour attirer les joueurs. C'est un jeu pour tout le monde, on choisit le minimalisme avec une palette des couleurs pastels car ce sont les couleurs douces et pas sombre. Elle fait une atmosphère relaxée quand on le joue. Échouer mais pas en colère.

— Font : Bebas Neue Regular (Free Copyright)
— Couleur principale : **bleu(125, 201, 255)**
— Couleurs des pieces : les couleurs pastels



— Couleurs de menu

0.15.2 Effet sonore

On voulait ajouter de la musique à notre jeu, car on pensait que cela attirerait les joueurs. Comme la musique est quelque chose de très populaire à partir de maintenant, on pensait que les joueurs aimeraient que on ajoute de la musique au jeu que on a créé.

Au début du jeu, pas de bruit tonitruant, pas de son nervosité, on souhaite de mettre les conforts et les joyeux en joueant en premier. C'est pourquoi on choisit l'accord de base au piano C(Do) majeur. Ensuite, le son des boutons est la note F(Fa) correspond à la première note de son du menu. Puis, quand on

joue, il n'y a pas de son au fond, il existe que le son des pièces pour que l'on se sent concentrer. Il faut savoir que l'on n'oublie pas que on est entraine de jouer au jeu donc le son du bongo apparait quand on glisse-despose les pièces. Enfin, on choisit l'accord guitare C(Do) mineur, on pense à créer une synchronisation de la musique.

Sixième partie

Conclusion