Institut Saint Jean Berchmans – Sainte Marie

Section Transition Sciences – Informatique

Programmation
Zombie Land

Travail de fin d'études
réalisé par
ALOISIO Alessandro

Année académique 2015 - 2016

Sommaire

Introduction....4
Cahier des charges....4
  Scénario....4
  Analyse....5
    Menu du jeu....5
    Carte....5
    Joueur....5
    Ennemis....6
    Vagues....6
Manuel d'utilisation....7
    But du jeu....7
    Menu principal....7
    En cours de partie....7
    Pause....9
    Fin de partie....9
Développement....11
Problèmes rencontrés....13
Apprentissage personnel....15
Améliorations possibles....15
Conclusion....16
Bibliographie....17
  Ressources Internet....17

Remerciements

Je tiens à remercier Madame Charlier qui m'a aidé pour mon système de collision.
Je tiens également remercier ma famille qui m'a soutenu tout au long de mon projet.
Et pour finir, je remercie mes professeurs d'informatique pour toutes les connaissances qu'ils m'ont transmises.

Introduction

J’ai décidé de faire un jeu car je n'avais pas d'inspiration pour le développement d'une application lambda. J'ai déjà joué à pas mal de jeux, donc j'ai eu plus de facilités pour trouver un projet réalisable à temps.
Zombie Land est un jeu de survie, vous êtes seul dans une île infestée de zombies. Votre but est de survivre le plus longtemps possible aux vagues de zombies qui viennent sur vous.

Chemin :
/media/dartagnan/classe6tb/aloiale/programmation/TFE/main.py

Cahier des charges

Scénario

L'application se lance en affichant un menu permettant de lancer le jeu et de le quitter.
Pour quitter le jeu à tout moment, il suffit d'appuyer sur la touche « Esc » grâce aux touches du clavier.
La partie lancée, le joueur apparaît au milieu de la plate-forme de jeu.
Pour faire avancer le joueur, il suffit d'appuyer sur la touche « X ».
Pour diriger le joueur, il suffit de placer le curseur de la souris vers la direction que vous voulez. Un clic gauche de la souris permet de tirer des balles vers la direction du curseur.
Les ennemis apparaissent aléatoirement, il y a trois lieux d'apparition.
L'apparition des ennemis se fait par vague, les vagues sont de plus en plus dures. Les ennemis se dirigent automatiquement vers le joueur.
La vie du joueur diminue s'il rentre en collision avec un ennemi.
La partie se termine lorsque votre vie est à zéro.
Le but du jeu est de tuer tous les ennemis se trouvant sur la carte sans être tué et d'obtenir le plus gros score possible.

Analyse

Menu du jeu

Le lancement du jeu affiche un menu nous permettant de lancer le jeu « JOUER » ou de le quitter « QUITTER ».
Il est possible à tout moment de mettre le jeu en pause en appuyant sur « p ». Pour reprendre la partie, il suffit d'appuyer sur « p ».
Un menu « Game Over » s'affiche lorsque vous n'avez plus de vie. Ce menu permet de recommencer la partie ou de revenir au menu principal.

Carte

Le jeu est en 2D, avec une vue isométrique donc avec une perspective de 30°, ce qui permet d'avoir un faux rendu 3D. Sur la carte sont disposés des arbres et des rochers pour créer un effet de « ring ». Ceux ci sont des obstacles pour les zombies et le joueur.
La carte se déplace une fois que l'on avance le joueur. Ce qui donne l'impression à l'utilisateur que le joueur avance.

Joueur

Le joueur est placé au centre de la carte. Le joueur avance lorsqu'on appuie sur la touche « x » du clavier. L'orientation du joueur se fait sur huit axes (N,S,O,E,NO,NE,SO,SE), cette orientation se fait selon la position du curseur sur la carte.
Le joueur ne peut pas sortir de la carte et rentrer dans des obstacles.
Le joueur tire des balles, les munitions sont illimitées. Les balles disparaissent lorsqu'elles rentrent en contact avec un ennemi, un obstacle ou si elles sortent de la carte.
Lorsqu'une balle touche un ennemi, la balle et le zombie disparaissent et le joueur gagne 10 points.
Le joueur perd 5 points de vie lorsqu'il est touché par un ennemi.

Ennemis

Les ennemis se dirigent vers le joueur, la vitesse de déplacement du zombie augmente tout au long de la partie. Les ennemis regardent vers le joueur, et si ceux-ci rentrent en collision avec un obstacle, ils essayeront automatiquement de le contourner pour continuer leur chemin vers le joueur. Les ennemis sont bloqués si ils se rentrent dedans. Le temps entre chaque attaque est prédéfinie pour ne pas mourir en quelques secondes. Il y a trois lieux d'apparition pour les ennemis et ils y apparaissent aléatoirement.

Vagues

On commence à la première vague, le nombre de vagues est illimité.
Lorsque le joueur tue tout les ennemis de la vague actuelle, la vague suivante engendre des modifications : le nombre de zombies augmente, le temps d'apparition de chaque zombie diminue de 0,1s et la vitesse de déplacement des ennemis augmente de 0,1%.

Manuel d'utilisation

But du jeu

Le but du jeu est de tuer tous les ennemis présents sans se faire tuer. Votre objectif est d'avoir le plus haut score possible.

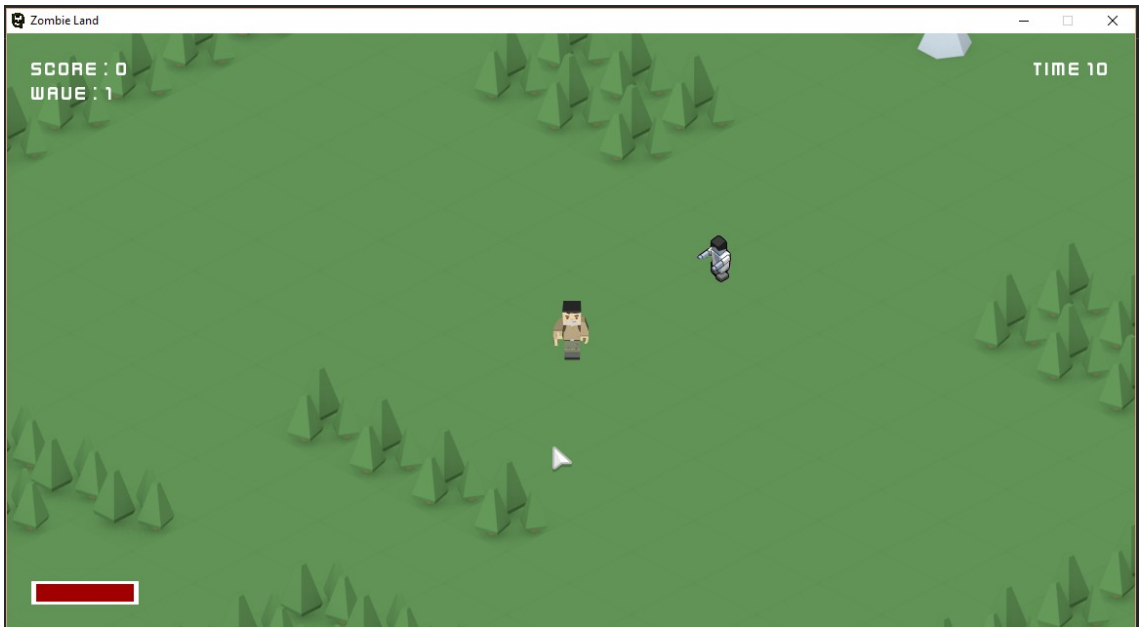
Menu principal

Depuis le menu principal, vous pouvez choisir de jouer ou de quitter le jeu. Vous pouvez également mettre le jeu en plein écran en appuyant sur la touche « f » du clavier.

En cours de partie

Une fois la partie lancée, vous devez appuyer sur la touche « X » pour vous déplacer, le joueur avance vers la direction du curseur donc par exemple si vous voulez monter sur la carte, vous devez mettre le curseur au dessus du joueur. Pour tuer un ennemi, il faut le viser avec le curseur et cliquer sur le bouton gauche de la souris pour tirer. Vous devez éviter de vous faire toucher par un ennemi.
Au début de chaque partie votre vie est au maximum, une fois que vous êtes touché par un ennemi votre vie est diminuée de 5 points. Votre score débute à 0 et augmente de 10 lorsque vous tuez un ennemi.

Au fils du temps, les vagues deviennent de plus en plus dures. La vitesse de génération des ennemis diminue, la vitesse de déplacement des ennemis augmente et le nombre d'ennemis a tuer augmente de 2.
La vague en cours et le score sont affichés dans le coin supérieur gauche. Un chronomètre est affiché dans le coin supérieur droit de votre écran et pour finir une barre de vie dans le coin inférieur gauche.

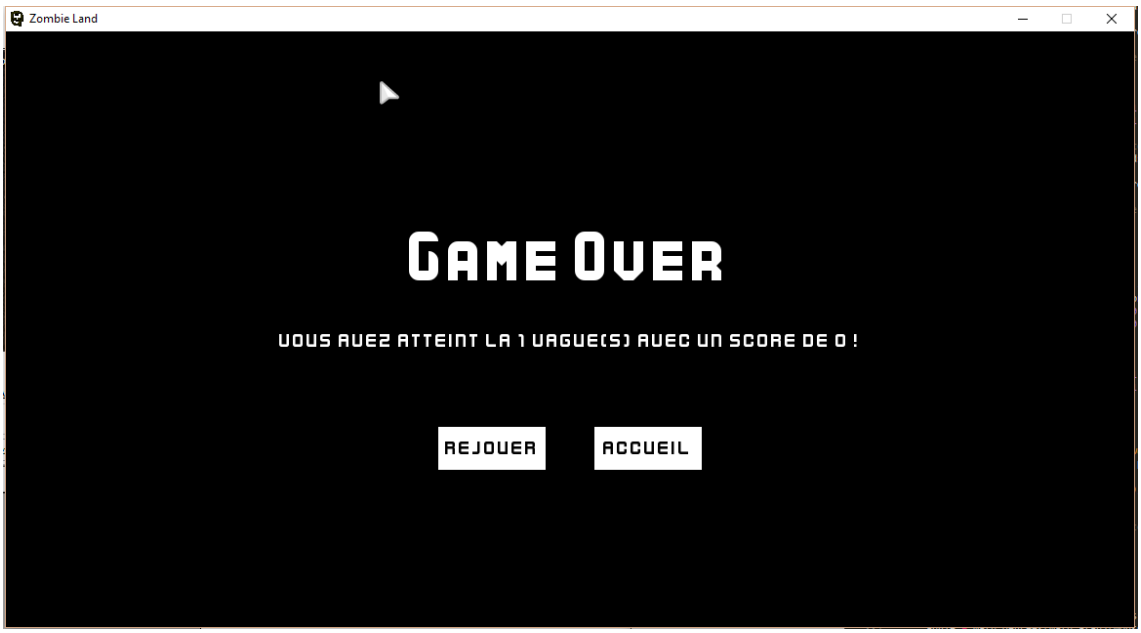


Pause

Il est possible de mettre le jeu en pause en appuyant sur la touche « p » du clavier.

Fin de partie

La partie se termine lorsque votre vie atteint 0.
Une fois la partie finie, un menu affiche votre score et vous indique à quelle vague vous êtes mort. Vous aurez alors la possibilité de recommencer la partie ou de revenir au menu principal.


Zombie Land
GAME OVER
VOUS AVEZ ATTEINT LA 1 VAGUE(S) AVEC UN SCORE DE 0 !
REJOUER
ACCUEIL

Développement

```
# spawn enemy, add enemies with timer
timerEnemy -= (FPSCLOCK.tick_busy_loop(60)/1000)

# Vague terminer
if(enemyKilled == waveNbEnemy):
    print('vague suivante')
    wave += 1
    # on augmente le nombre d'ennemies
    waveNbEnemy += 2

    nbEnemyMap = 0
    enemyKilled = 0
    timerEnemy = 0

    # on diminue le temps d'apparition
    if(delaySpawnEnemy > 0.3):
        delaySpawnEnemy -= 0.1
    # on augmente la vitesse de déplacement
    if(speedEnemy < 5):
        speedEnemy += 0.1

# Ajouter les ennemies avec un delay
if(timerEnemy <= 0):
    timerEnemy = delaySpawnEnemy

    if(waveNbEnemy > nbEnemyMap):

        posEnemy = spawnPos[randrange(0, len(spawnPos))]
        where = lookTo(playerPos, posEnemy)
        enemiesGroup.add(Enemies(enemiesRect, where, posEnemy))
        nbEnemyMap += 1
```

Je vais vous présenter l'algorithme qui permet de gérer les vagues et l'ajout de zombies sur la carte.
Tout d'abord, on initialise deux variables « timerEnemy » et « delaySpawnEnemy » qui stockent le temps entre chaque apparition d'un ennemi sur la carte. La variable « timerEnemy » diminue toutes les millisecondes grâce à la fonction « tick_busy_loop() ». Cela permet de créer un petit timer.
Pour qu'une vague soit terminée, il faut que la variable « enemyKilled » soit égale à la variable « waveNbEnemy ». Car la variable « waveNbEnemy » correspond au nombre d'ennemis qui

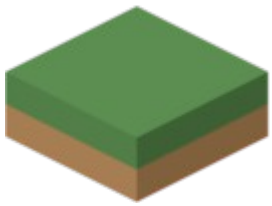
apparaîtront dans la vague et la variable « enemyKilled » correspond au nombre d'ennemis tués lors de la vague.

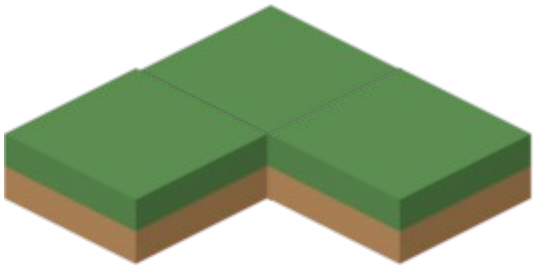
Si nous rentrons dans cette condition alors nous augmentons de un la variable « wave » qui permet de savoir le nombre de vagues atteintes. Nous ajoutons également 2 ennemis a tuer pour la prochaine vague.
On remet à zéro les variables « nbEnemyMap », « enemyKilled » et « timerEnemy » pour permettre de recommencer l'ajout des ennemis sur la carte. Pour finir cette condition et augmenter la difficulté du jeu, nous diminuons la variable « delaySpawnEnemy » pour que les ennemis apparaissent plus rapidement et nous augmentons la vitesse de déplacement des ennemis. Les conditions qui les précèdent permettent d'éviter des bogues et de limiter la difficulté.

Nous allons maintenant rentrer dans la deuxième condition qui lui permet d'ajouter des ennemis dans un sprite.group (enemiesGroup). Pour rentrer dans la condition, il faut que la variable « timerEnemy » soit inférieure ou égale à zéro. Si nous rentrons dans cette condition, la variable « timerEnemy » se remet à la valeur par défaut ce qui permet d'attendre entre chaque intégration. La condition qui suit permet de vérifier que le nombre d'ennemis sur la carte soit inférieur au nombre d'ennemis à ajouter pour que la vague soit terminée. Si nous entrons dans cette condition, nous choisissons aléatoirement un lieu d'apparition. La variable « where » permet de savoir l'axe d'apparition, donc elle permet de faire en sorte que l’ennemi regarde le joueur. La fonction « lookTo » permet de calculer l'angle entre deux points et elle nous retourne un des huit axes. Nous ajoutons maintenant un nouvel ennemi dans le groupe « enemiesGroup », nous initialisons un ennemi grâce à la classe « Enemies » qui prend en paramètre la surface, l'axe et la position initiale. Pour finir, nous incrémentons de un la variable « nbEnemyMap » car l'ennemi est maintenant créé.

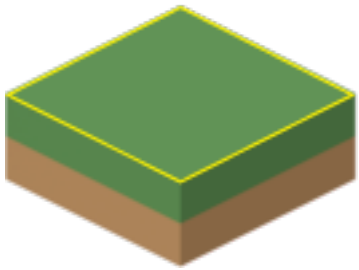
Problèmes rencontrés

Le plus gros problèmes que j'ai rencontrés, sont la gestion des collisions. La carte est une superposition de blocs, chaque bloc dispose d'une hauteur que je n'ai pas besoin.





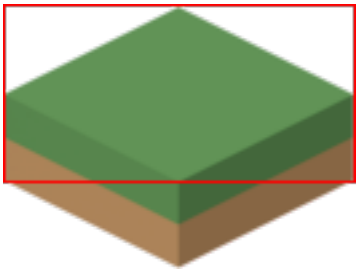
Tout d'abord, nous avons besoin de connaître les coordonnées des quatre sommets de la surface plane.

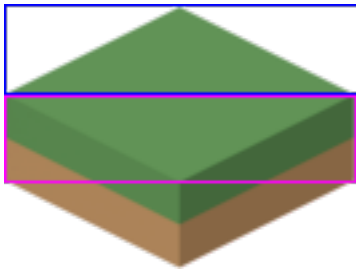


Notre but était de savoir si un point de coordonnées rentrait dans le cadre jaune. Pour ce faire, à l'aide de Madame Charlier, j'ai créé une fonction « checkInsideTile » qui prend deux paramètres, le premier paramètre correspond à la position que nous voulons vérifier et le deuxième paramètre correspond aux quatre sommets de la surface.

```
def checkInsideTile(pos, coords):
    if pos[0] > coords[0][0] and pos[0] < coords[2][0] and pos[1] > coords[1]
[1] and pos[1] < coords[3][1]:
        if pos[0] < coords[0][0] + ((coords[2][0] - coords[0][0]) / 2):
            m = (coords[1][1] - coords[0][1]) / (coords[1][0] - coords[0][0])
            yd = (m * (pos[0] - coords[0][0])) + coords[0][1]
            if pos[1] > yd and pos[1] < (yd + ((coords[0][1] - yd) * 2)):
                return True
        else:
            m = (coords[2][1] - coords[1][1]) / (coords[2][0] - coords[1][0])
            yd = (m * (pos[0] - coords[1][0])) + coords[1][1]
            if pos[1] > yd and pos[1] < (yd + ((coords[2][1] - yd) * 2)):
                return True
    return False
```

La première condition permet de vérifier que le point se trouve au moins dans le rectangle rouge ci-dessous.





Si nous rentrons dans la condition, nous voulons savoir dans quelle partie nous nous trouvons car la formule est différente si nous nous trouvons en bas de la moitié de la surface (partie magenta).
La variable « m » correspond à la pente de la surface. Et la variable « yd » permet de récupérer la valeur de l'y correspondant à la position x recherchée. Donc pour savoir si nous nous trouvons effectivement dans le bloc nous devons simplement vérifier que la valeur y de la position recherchée soit plus grande que la variable « yd ».

Apprentissage personnel

Lors du développement de ce projet, j'ai acquis de nouvelles compétences pour la création de jeux à l'aide de la librairie Pygame. Pygame est une bibliothèque libre multiplate-forme qui facilite le développement de jeux vidéo en temps réel avec le langage de programmation Python.

Améliorations possibles

Pour avoir un meilleur rendu, je pourrais améliorer le système de contournement automatique des ennemis car celui-ci fonctionne mais comporte encore quelques bogues.
Il serait intéressant d'ajouter des bonus :

- augmenter la vie du joueur
- changement d'arme selon la vague
- bombe permettant de faire des dégâts de zone

Conclusion

Je suis satisfait de mon travail car je m'étais fixé beaucoup d'objectifs pour le faire le plus complet possible. Mon objectif principal était de faire un jeu fonctionnel, ne voyant pas l'utilité de passer autant de temps sur un jeu qui ne sera pas jouable à sa présentation.

J'ai malheureusement dû supprimer et revenir sur certains objectifs par manque de temps. En effet, je n'avais pas conscience de la difficulté du travail demandé.

Malgré tout, j'ai su garder en tête que le but principal du TFE était de nous apprendre à nous débrouiller et de nous mettre face à un travail plus important. C'est ce qui m'a permis de revoir mes attentes et avancer à un rythme régulier sans me mettre en danger par rapport à la date limite.

De plus, voulant aller à l'INPRES l'année prochaine, je pense que la réalisation de ce projet nous permet de mieux appréhender la quantité de travail que l'on aura.

Ce projet nous permet également d’engranger un maximum d'expérience quant à l'écriture de programmes plus conséquents. Cela nous permettra certainement de ne plus reproduire des erreurs que l'on aurait pu faire sans avoir fait ce travail.

Je pense réellement qu'il va nous aider pour la suite et va nous permettre d'avancer plus facilement en travaillant avec d'autres langages car on aura déjà acquis une bonne logique de programmation.

Bibliographie

Ressources Internet

PyGame,

- http://www.pygame.org
- http://www.pygame.org/docs/ref/sprite.html
- http://www.pygame.org/docs/ref/event.html
- http://www.pygame.org/docs/ref/mouse.html
- http://www.pygame.org/docs/ref/music.html
- http://www.pygame.org/docs/ref/font.html
- http://www.pygame.org/docs/ref/display.html

Stackoverflow,

- http://stackoverflow.com/
- http://stackoverflow.com/questions/10473930/how-do-i-find-the-angle-between-2-points-in-pygame
- http://stackoverflow.com/questions/18012431/countdown-timer-for-pygame
- http://stackoverflow.com/questions/25851090/id-function-in-python-2-7-is-operator-object-identity-and-user-defined-met
- http://stackoverflow.com/questions/21980395/how-can-i-exit-fullscreen-mode-in-pygame

Kenney,

- http://www.kenney.nl/
- http://kenney.nl/projects/kenney-game-assets
- http://kenney.nl/projects/kenney-game-assets-2