

“Aidez MacGyver à s’échapper !”

Projet n°3

Zoé Belleton • 28/06/2018





Le projet

un application GUI (TKinter)



Documentation de module



Algorithmique / Analyse logique



Conceptualisation / Structure (UML)



Environnement de développement (IDE)



Distribution du script (open source)

(pygame) (pyCharm / pylint) (cx_Freeze)

Plateformes

Linux , Windows ,
Windows CE, BeOS,
MacOS, Mac OS X,
FreeBSD, NetBSD,
OpenBSD, BSD/OS,
Solaris, IRIX, QNX.

Le code contient un
support pour
AmigaOS,
Dreamcast, Atari,
AIX, OSF/Tru64, RISC
OS, Symbian OS, and
OS/2, nokia, game
consoles like gp2x,
the One Laptop Per
Child (OLPC), and the
Orange Pi.

Pygame (www.pygame.org)

Set de modules dédié à la conception d'applications GUI

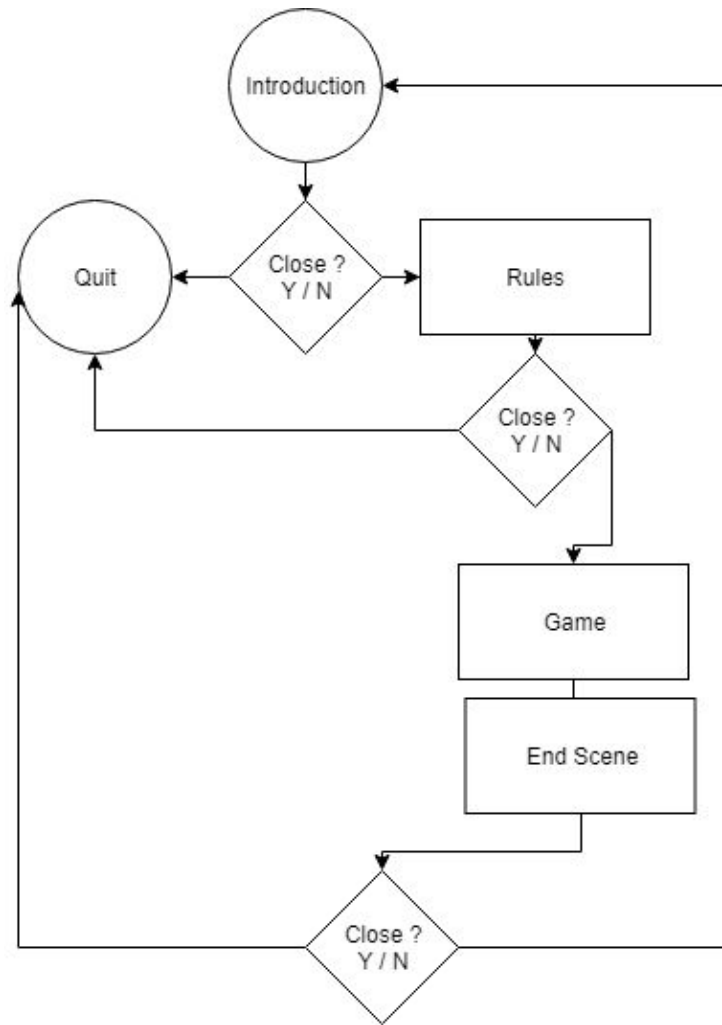


SDL*



- Libre & open-source
(sous licence GNU LGPL)
- Repose sur la SDL | Pas OpenGL
(conçu pour remplacer pySDL)
- Utilise des langages qui multiplie sa
vitesse d'exécution
(C (10 à 20 *) et Assembleur (~ 100*))
- Cross-platform





Game loop

Re démarrage facilité



Un seul niveau (cf : consigne)



Une variable “status”



La logique du niveau dans une classe dédiée



“intro” / “rules” / “game” / “end”



Diffusion du score ! (score.txt)

Statut courant

- Fermeture ?
- Appui sur une touche ?
- Résultat !

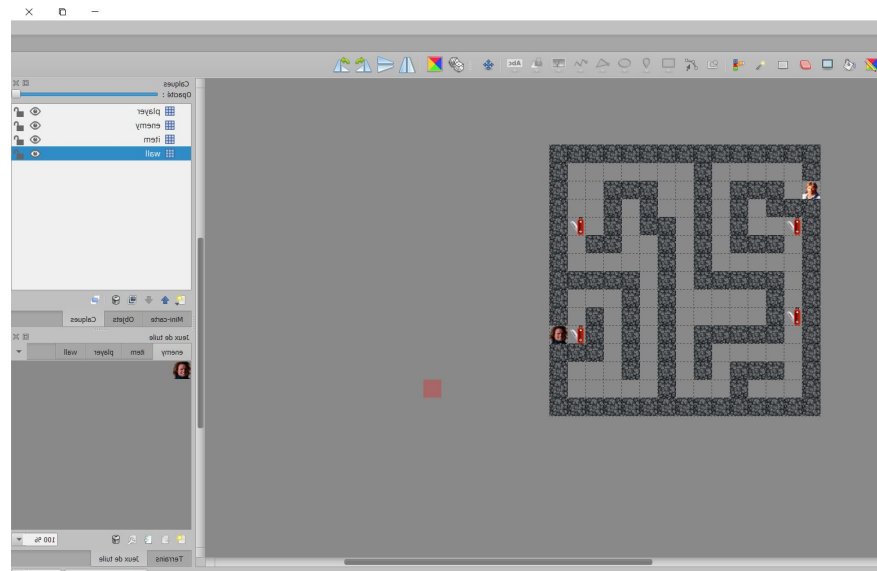
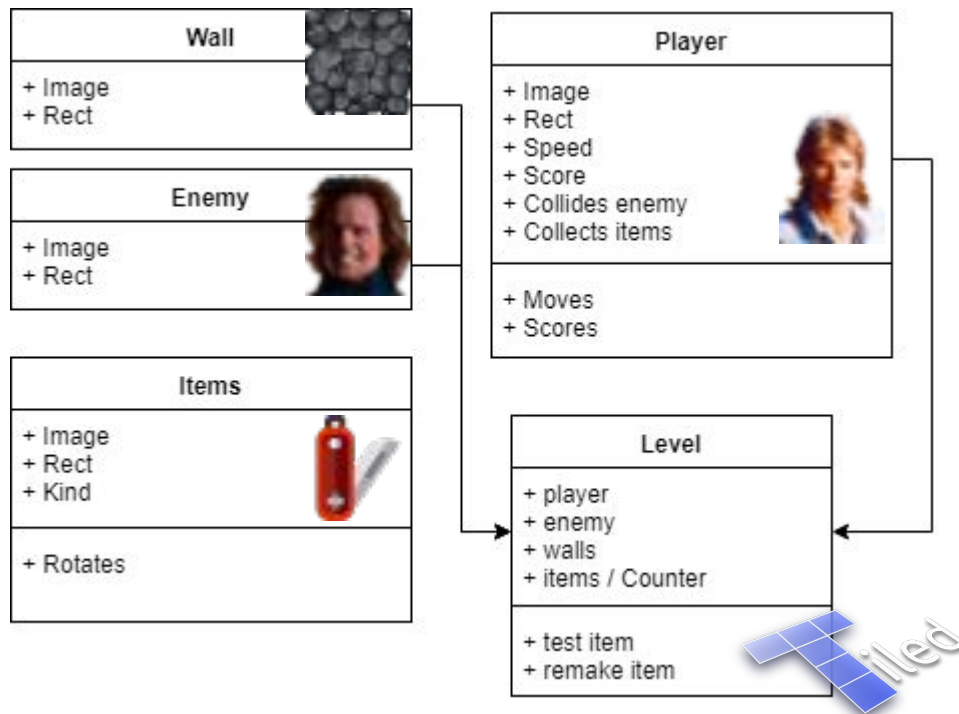



```
def intro(status):  
    """  
    Introduction Scene  
    :param status: A boolean ready to capture User Event  
    :return: Updated Status Param  
    """  
  
    current = Intro()  
    if current.status:  
        status = current.status  
    return status
```

```
def update(self):  
    """  
    Event loop  
    """  
    while self.run:  
        for event in pygame.event.get():  
            if event.type == MOUSEBUTTONDOWN or event.type == KEYDOWN:  
                self.status = 1  
                self.run = False  
            elif event.type == QUIT:  
                self.run = False  
        pygame.time.wait(500)  
        self.draw()  
    return self.status
```

Level design

Une classe pour chaque élément du niveau





```
def walking_in_maze(self, keys):
```

```
    """
```

```
    Player is evolving in the maze
```

```
    :param keys: Pressed keys
```

```
    """
```

```
    # Check if the game is paused
```

```
    if not self.pause:
```

```
        # up
```

```
        if keys[K_UP]:
```

```
            # Check the player position before moves
```

```
            if self.player.rect.y > 0:
```

```
                self.player.move_up()
```

```
            # Prevent player colliding walls
```

```
            for self.wall in self.walls:
```

```
                if self.player.rect.colliderect(self.wall.rect):
```


```
                    # Going back
```

```
                    self.player.move_down()
```

Défi #1

Déplacement dans le labyrinthe

- Pause ?
- Direction ?
- Obstacle ?
- Résultat !




```
# Game loop is running
```

```
while self.run:
```




```
# Capture user events
```

```
for event in pygame.event.get():  
    self.check_for_ending(event)
```



```
# Pressing a key
```


```
keys = pygame.key.get_pressed()  
self.handle_pause(keys)
```



```
self.handle_music(keys)
```


```
self.walking_in_maze(keys)
```

```
self.checking_player_colliding_enemy()
```




```
# Check if player is collecting items
```

```
self.check_items_collecting()
```



```
# Prevent too many recursion errors
```

```
pygame.time.wait(50)
```



```
# Draw the updated game
```

```
self.draw()
```

Défi #2

Mise à jour du jeu

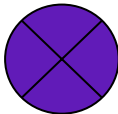
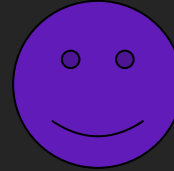
- Mettre en pause ?
- Couper la musique ?
- Se déplacer ?
- Gagner des points ?
- Terminer le jeu ?

Défi #3

La fin du jeu

- Fermeture ?
- Face à l'ennemi ?
- Bien équipé ?
- Résultat !
(dans l'écran de fin)

```
def win_or_fail(self, ending):  
    """  
    Those actions differs when game is terminated  
    :param : ending : The ending status of the game  
    :return : status ( always 0 if only one level )  
    """  
    ending_status = False  
    if ending == "win":  
        self.sound_win.play()  
        ending_status = "Victory ! You made the guard falling asleep !
```



```
# Fading out music  
self.music.fadeout()  
  
# Writing score in text file  
file = open('score.txt', 'w')  
file.write(ending_status)  
file.close()  
  
# Exit game loop  
self.run = False
```

Forces

Tiled / PyTMX

(éditeur de niveau)

(chargeur de carte)



```
class Level:
    """
    Class Level
    """

    def __init__(self):
        level_file = os.path.join(ASSETS_DIR, "gfx", "level.tmx")
        level = pytmx.load_pygame(level_file)
        self.walls = []
        self.items = []
        wall_tiles = 0
        enemy_tiles = 2
        player_tiles = 3
```

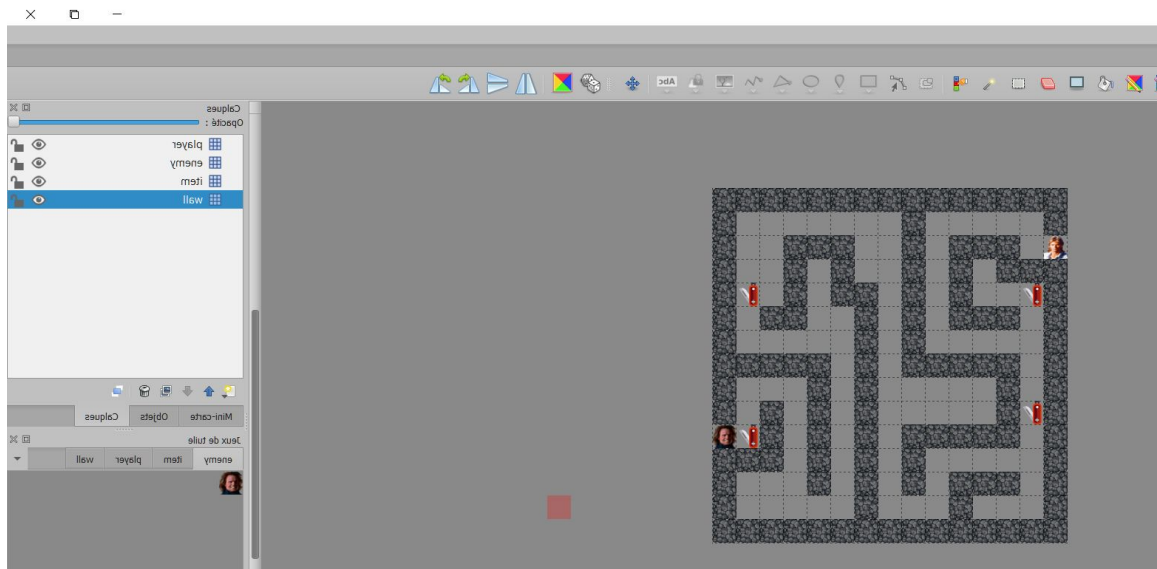
```
for row in range(15):
    for col in range(15):
        # Walls in tmx file
        wall = level.get_tile_image(row, col, wall_tiles)
        if wall is not None:
            self.wall = Wall((row, col))
            self.walls.append(self.wall)
```

Forces

Tiled / PyTMX

(éditeur de niveau)

(chargeur de carte)

[illegible]

Forces

Un exécutable

(cx_Freeze)



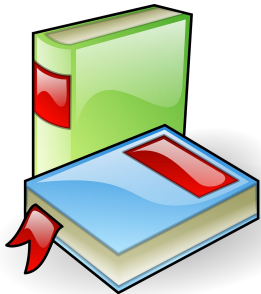
```
from cx_Freeze import *
```

```
target = Executable(  
    script="main.py",  
    base=base,  
    icon='item.ico',  
)  
  
setup(  
    name="Mac Gyver Maze",  
    description="Run a maze with Mac Gyver and collect 3 items",  
    author="Zoe Belleton",  
    options={'build_exe': {'packages': ['pygame', 'pytmx', 'rand  
    executables=[target]  
)
```

Démonstration



Pistes d'améliorations



1. Multi level

(Branche Master du dépôt Git)

2. Inventaire

(Lister les points au lieu de les compter)

3. Vue à la première personne

(Afficher dans un deuxième écran ce qui est visible par le joueur)

Des questions ?

