Améliorez un projet

Projet 11

Modules

Pygame

Random

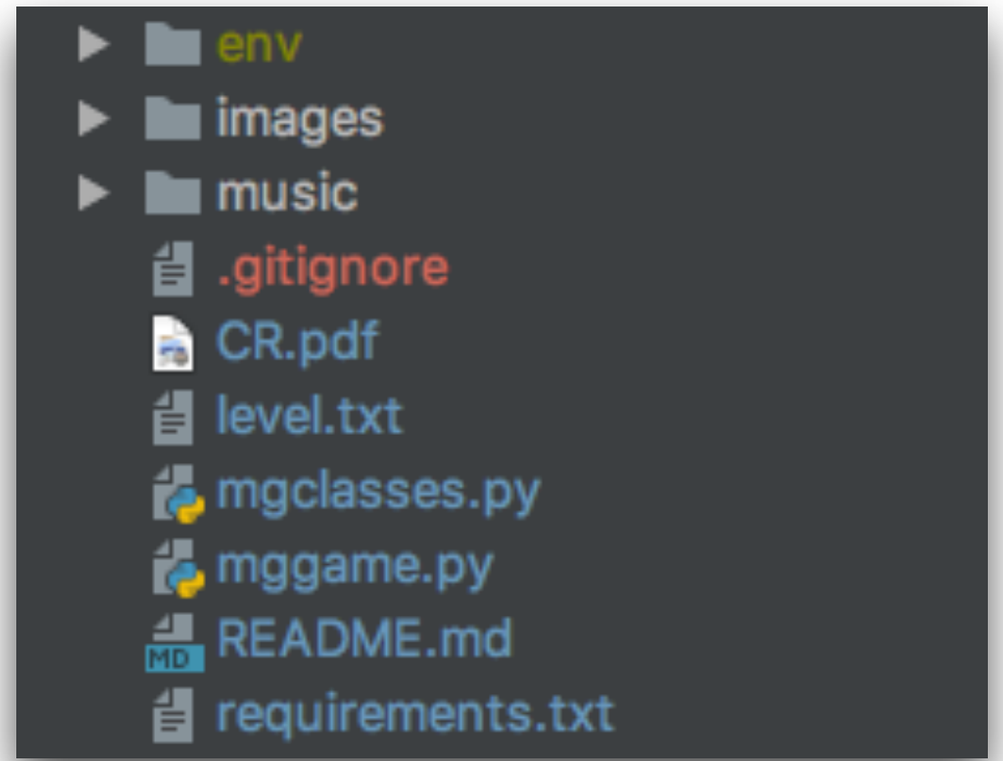
Sys

Pygame.math

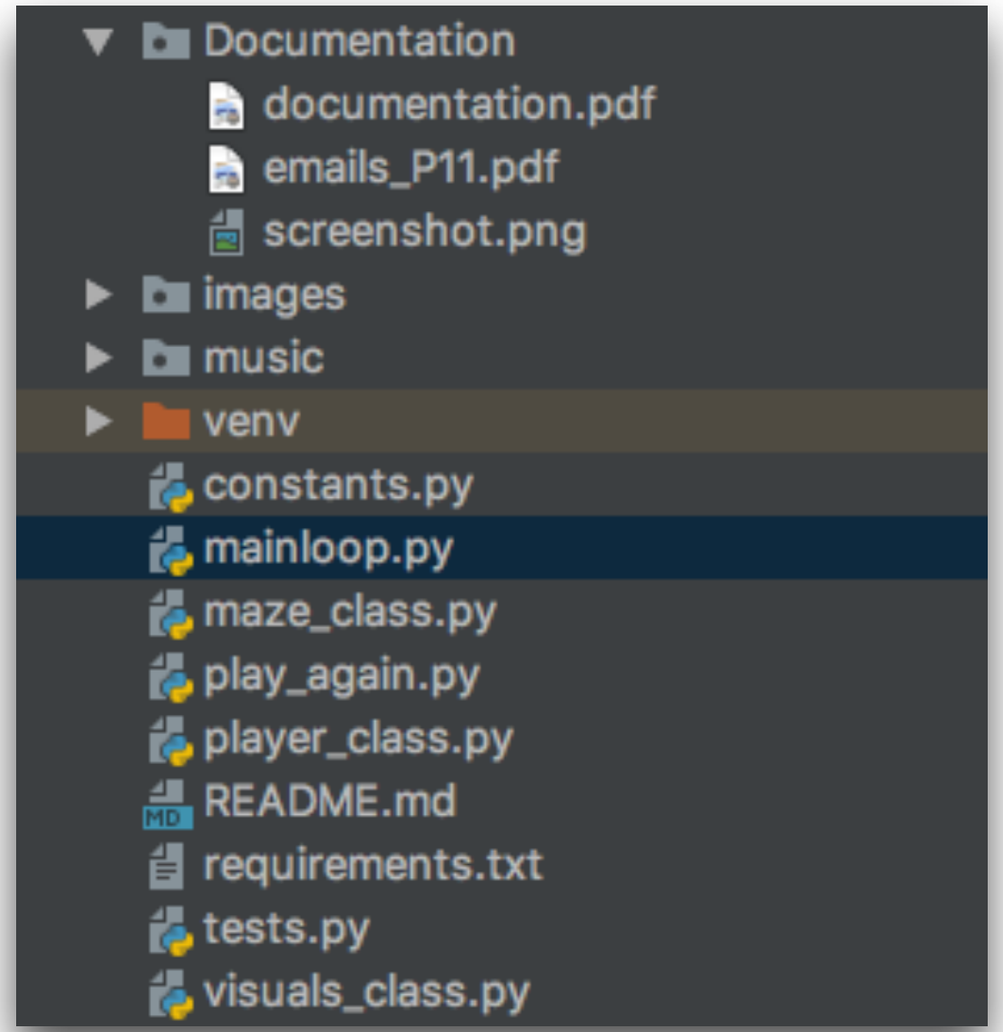
L’app

Changements de l’architecture de l’application

V1



V2



Main Loop

- Utilisation des Groupes de sprites générés par Pygame

V1

```
SCREEN.blit(BACKGROUND, (0, 0))
# display background on screen
labyrinthe.displaying_level(SCREEN)
# display the maze on screen
player.make_move_macgyver(SCREEN)
# display movements of Macgyver
SCREEN.blit(player.character_direction,
            (player.position_character_in_pixel_y,
             player.position_character_in_pixel_x))
             # display position of Macgyver

if potion_unpicked:
# Loop to pick the Potion
    SCREEN.blit(objects.potion, (objects.x_potion * SIZE_OF_SPRITE,
                                 objects.y_potion * SIZE_OF_SPRITE))
                                 # display POTION and converts it into sprite
if (player.position_character_in_pixel_y,
        player.position_character_in_pixel_x) == (objects.x_potion * SIZE_OF_SPRITE,
                                                  objects.y_potion * SIZE_OF_SPRITE):
    potion_unpicked = False
    # remove POTION image
    SCREEN.blit(objects.potion, (180, 450))
    # on the bottombar of the window's game
    lg.debug('potion picked')

if coffre_unpicked:
# Loop to pick the Coffre
    SCREEN.blit(objects.coffre, (...))
                                 # display COFFRE and converts it into sprite
if (player.position_character_in_pixel_y,
        player.position_character_in_pixel_x) == (objects.x_coffre * SIZE_OF_SPRITE,
                                                  objects.y_coffre * SIZE_OF_SPRITE):
    coffre_unpicked = False
    SCREEN.blit(objects.coffre, (210, 451))
    lg.debug('coffre picked')

if monnaie_unpicked:
# Loop to pick the monnaie
    SCREEN.blit(objects.monnaie, (objects.x_monnaie * SIZE_OF_SPRITE,
                                  objects.y_monnaie * SIZE_OF_SPRITE))
                                  # display MONNAIE and converts it into sprite
if (player.position_character_in_pixel_y,
        player.position_character_in_pixel_x) == (objects.x_monnaie * SIZE_OF_SPRITE,
                                                  objects.y_monnaie * SIZE_OF_SPRITE):
    monnaie_unpicked = False
    SCREEN.blit(objects.monnaie, (240, 450))
    lg.debug('monnaie picked')
```

V2

```
def main():
    """Main Loop of the game"""
    pygame.mixer.music.play(-1)
    pygame.mixer.music.set_volume(0.1)
    win = pygame.display.set_mode((winWidth, winHeight))
    score = 0
    pygame.display.set_caption('Macgyver')
    clock = pygame.time.Clock()
    font = pygame.font.SysFont('helvetica', 30)

    all_sprites = pygame.sprite.Group()
    walls_sprites = pygame.sprite.Group()
    floor_sprites = pygame.sprite.Group()
    enemy_sprites = pygame.sprite.Group()
    pick_sprites = pygame.sprite.Group()

    maze = MazeCreation(all_sprites, walls_sprites, floor_sprites, enemy_sprites, pick_sprites)
    maze.draw()

    player = Macgyver(45, 45, walls_sprites)
    all_sprites.add(player)

    game = False
    while not game:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()

        keys = pygame.key.get_pressed()
        if keys[pygame.K_UP]:
            player.macgyverVelocity.y = -3
        elif keys[pygame.K_DOWN]:
            player.macgyverVelocity.y = 3
        else:
            player.macgyverVelocity.y = 0

        if keys[pygame.K_LEFT]:
            player.macgyverVelocity.x = -3
        elif keys[pygame.K_RIGHT]:
            player.macgyverVelocity.x = 3
        else:
            player.macgyverVelocity.x = 0

        pick_objects = pygame.sprite.spritecollide(player, pick_sprites, True)

        if pick_objects:
            pygame.mixer.Sound.play(pickingMusic).set_volume(0.2)
            pick_sprites.clear(win, back)
            score += 1

        if score != 3 and pygame.sprite.spritecollideany(player, enemy_sprites):
            pygame.sprite.spritecollide(player, enemy_sprites, False)
            pygame.mixer.music.stop()
            pygame.mixer.Sound.play(loosingMusic).set_volume(0.5)
            text = font.render('Looser ! Play again ?', 13, (0, 0, 0))
            play_again.restart(win, winWidth, winHeight, text)
        elif score == 3 and pygame.sprite.spritecollide(player, enemy_sprites, True):
            pygame.mixer.music.stop()
            pygame.mixer.Sound.play(winningMusic).set_volume(0.5)
            text = font.render('Winner ! Play again ?', 13, (0, 0, 0))
            play_again.restart(win, winWidth, winHeight, text)

        win.blit(back, (0, 0))
        text = font.render('Score : ' + str(score) + ' / 3', 1, (255, 255, 255))
        win.blit(text, (winWidth / 2 - text.get_width() / 2, 520))
        all_sprites.update()
        all_sprites.draw(win)

        pygame.display.flip()
        clock.tick(27)
```

Class Macgyver

- Calcul de la position du personnage

V1

- Vérification du Sprite suivant

```python
class Macgyver:
    """CLASS MACGYVER : MAIN CHARACTER OF THE GAME.
    THIS CLASS CREATE THE CHARACTER AND ALLOW HIM TO MOVE IN THE MAZE """

    def __init__(self, character_place):
        """ constructor of class Macgyver """
        self.character = pygame.image.load("images/macgyver.png").convert_alpha()
        self.position_character_in_sprite_x = 0 # Character's position in sprite x default
        self.position_character_in_sprite_y = 0 # Character's position in sprite y default
        self.position_character_in_pixel_x = 0 # Character's position in pixel y default
        self.position_character_in_pixel_y = 0 # Character's position in pixel y default
        self.character_direction = self.character # Macgyver looks like this
        self.character_place = character_place # where is Macgyver

    def make_move_macgyver(self, character_direction):
        """ method to make the character move in the Maze """

        if character_direction == 'right': # if going right
            if self.position_character_in_sprite_y < (NUMBER_OF_SPRITE - 1):
            # Character doesn't get out of the window's game
                if self.character_place.level_structure[
                        self.position_character_in_sprite_x][
                            self.position_character_in_sprite_y+ 1] != "m":
                            # checking the destition is empty (no wall)
                            #by reading level structure
                    self.position_character_in_sprite_y += 1
                    # Make move the character by one sprite
                    # change his position_character_in_pixel_y and position_character_in_sprite_x
                    self.position_character_in_pixel_y = self.position_character_in_sprite_y * 30
            lg.debug('Character is going to the right')

        if character_direction == 'down': # if going down
            if self.position_character_in_sprite_x < (NUMBER_OF_SPRITE - 1):...
            lg.debug('Character is going down')

        if character_direction == 'left': # if going left
            if self.position_character_in_sprite_y > 0:...
            lg.debug('Character is going to the left')

        if character_direction == 'up': #if going up
            if self.position_character_in_sprite_x > 0:...
            lg.debug('Character is going up')
```

- Utilisation de Pygame.Sprite

- Gestion des collisions avec les murs

V2

```python
class Macgyver(pygame.sprite.Sprite):
    """Macgyver Class"""
    def __init__(self, x, y, wallsMaze):
        """x, y = Player positions
            wallsMaze = group of sprites"""
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load("images/macgyver.png")
        self.rect = self.image.get_rect(topleft=(x, y))
        self.macgyverPosition = Vector2(x, y)
        self.macgyverVelocity = Vector2(0, 0)
        self.walls = wallsMaze

    def update(self):
        """Function to update player's position by adding velocity
            and check collisions"""
        self.macgyverPosition += self.macgyverVelocity
        self.playerCollisions()

    def playerCollisions(self):
        """Function to check if player collides walls"""

        self.rect.centerx = self.macgyverPosition.x
        for wall in pygame.sprite.spritecollide(self, self.walls, False):
            if self.macgyverVelocity.x > 0:
                self.rect.right = wall.rect.left
            elif self.macgyverVelocity.x < 0:
                self.rect.left = wall.rect.right
            self.macgyverPosition.x = self.rect.centerx

        self.rect.centery = self.macgyverPosition.y
        for wall in pygame.sprite.spritecollide(self, self.walls, False):
            if self.macgyverVelocity.y > 0:
                self.rect.bottom = wall.rect.top
            elif self.macgyverVelocity.y < 0:
                self.rect.top = wall.rect.bottom
            self.macgyverPosition.y = self.rect.centery
```

Class MazeCreation

- Fichier texte

V1

- Screen.blit() abusif

```python
class Maze:
    """ CLASS MAZE IS CREATING AND DISPLAYING
    THE MAZE IN WICH THE CHARACTER (MACGYVER) WILL MOVE """

    level_structure = []

    def __init__(self):
        """ constructor of class Maze """
        self.level = 'level.txt'
        # calling file level.txt, it contains the structure of the maze
        self.level_structure = 0
        # create list of the structure's level


    def creating_level(self):
        """ method to create the structure of the level """
        with open(self.level, "r") as level:
            create_structure_level = []
            # create a list of the file level.txt
            for line in level:...
            self.level_structure = create_structure_level
            # save the structure in create_structure_level

            #  LEVEL_STRUCTURE IS AN ARRAY
            # y represents lines
            # x represents columns

    def displaying_level(self, SCREEN):
        """ method to display the level on screen """
        walls = pygame.image.load('images/walls.png').convert_alpha()
        departure = pygame.image.load('images/depart.png').convert_alpha()
        murdoc = pygame.image.load('images/murdoc.png').convert_alpha()
        floor = pygame.image.load('images/floor.png').convert_alpha()


        line_of_level_default = 0 # default position of a line
        for line in self.level_structure: # read each line in level_structure
            column_of_level_default = 0 # default position of a case
            for sprite in line: # read each sprite in line_of_level_default
                sprite_position_x = column_of_level_default * SIZE_OF_SPRITE
                # default position in pixel on x
                sprite_position_y = line_of_level_default * SIZE_OF_SPRITE
                # default position in pixel on y
                # add corresponding image to each sprite
                #depending of it's type (wall, departure, arrival)
                if sprite == "m":...
                elif sprite == "d":...
                elif sprite == "a":...
                elif sprite == "0":
                    SCREEN.blit(floor, ((sprite_position_x, sprite_position_y)))
                column_of_level_default += 1 # add it to column_of_level_default
            line_of_level_default += 1# add it to line_of_level_default
```

- Utilisation de Pygame.Sprite

V2

- Utilisation de classes créants les éléments graphiques

```python
class MazeCreation(pygame.sprite.Sprite):
    """Class to create a maze and display it on screen"""
    def __init__(self, allSprites, wallsSprites, floorSprites, enemySprites, pickSprites):
        pygame.sprite.Sprite.__init__(self)
        self.mazeList = [maze_1, maze_2]
        self.allSprites = allSprites
        self.wallsSprites = wallsSprites
        self.floorSprites = floorSprites
        self.enemySprites = enemySprites
        self.pickSprites = pickSprites

    def draw(self):
        maze_choice = random.choice(self.mazeList)
        mazing = []
        for line in maze_choice.split('\n'):
            line = list(line)
            mazing.append(line)

        y = 0
        objects_to_display = 0
        counting_sprites = 0
        for line in mazing:
            x = 0
            for sprite in line:
                if sprite == '0':
                    aSprite = Floor(x * spriteSize, y * spriteSize)
                    self.floorSprites.add(aSprite)
                    self.allSprites.add(aSprite)

                    if counting_sprites > 60 and objects_to_display <= 2:
                        x_pick = x * spriteSize
                        y_pick = y * spriteSize
                        list_toBePicked = [PickPotion(x_pick, y_pick),
                                           PickCoin(x_pick, y_pick),
                                           PickSafe(x_pick, y_pick)]
                        choose_toBePicked = random.choice(list_toBePicked)
                        self.pickSprites.add(choose_toBePicked)
                        self.allSprites.add(choose_toBePicked)
                        objects_to_display += 1
                    counting_sprites += 1

                elif sprite == 'm':
                    aSprite = Walls(x * spriteSize, y * spriteSize)
                    self.wallsSprites.add(aSprite)
                    self.allSprites.add(aSprite)
                elif sprite == 'a':
                    aSprite = End(x * spriteSize, y * spriteSize)
                    backgroundSprite = Floor(x * spriteSize, y * spriteSize)
                    self.enemySprites.add(aSprite)
                    self.allSprites.add(backgroundSprite, aSprite)
                x += 1
            y += 1
```

Class Floor, Walls, etc.

V1

- Une seule classe

```python
class Pickittowin:
    """ OBJECTS TO PICK TO MAKE FALL ASLEEP MURDOC THE GUARDIAN
    AND WIN THE GAME ! """

    level_structure = []

    def __init__(self):
        """ constructor of class Pick_it_to_win """
        self.level = 'level.txt' # calling level.txt, it contains the structure of the maze
        self.level_structure = 0 # create list of the structure's level
        self.x_potion = 0 # init x position for potion
        self.x_coffre = 0 # init x position for coffre
        self.x_monnaie = 0 # init x position for monnaie
        self.y_potion = 0 # init y position for potion
        self.y_coffre = 0 # init y position for coffre
        self.y_monnaie = 0 # init y position for monnaie

        # y represents a line on the level structure (array)
        # x represents a column on the level structure (array)

    def display_object(self, SCREEN):
        """ Function to display on screen objects to pick """
        self.potion = pygame.image.load('images/potion.png').convert_alpha() # object to picked
        self.coffre = pygame.image.load('images/coffre.png').convert_alpha() # object to picked
        self.monnaie = pygame.image.load('images/monnaie.png').convert_alpha() # object to picked

        # Reading structure of the Maze Level_structure
        with open(self.level, "r") as level:
            create_structure_level = [] # create a list of the file level.txt
            for line in level:...
            self.level_structure = create_structure_level

        # Generating random position lines for each items to display
        self.y_potion = random.randint(0, 14) # choice a random position line for potion
        self.y_coffre = random.randint(0, 14) # choice a random position line for coffre
        self.y_monnaie = random.randint(0, 14) # choice a random position line for coffre

        # Loop to define position of POTION
        while self.x_potion == 0:...

        # Loop to define position of COFFRE
        while self.x_coffre == 0:...

        # Loop to define position of MONNAIE
        while self.x_monnaie == 0:...
```

V2

- Utilisation de Pygame.Sprite
- Une classe par item graphique

```python
class Walls(pygame.sprite.Sprite):
    """Class to create Walls' surface"""
    def __init__(self, x, y):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load('images/walls.png')
        self.rect = self.image.get_rect(topleft=(x, y))


class Floor(pygame.sprite.Sprite):
    """Class to create Floor's surface"""
    def __init__(self, x, y):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load('images/floor.png')
        self.rect = self.image.get_rect(topleft=(x, y))


class End(pygame.sprite.Sprite):
    """Class to create End's surface"""
    def __init__(self, x, y):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load('images/murdoc.png')
        self.rect = self.image.get_rect(topleft=(x, y))


class PickPotion(pygame.sprite.Sprite):
    """Class to create PickPotion's surface"""
    def __init__(self, x, y):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load('images/potion.png')
        self.rect = self.image.get_rect(topleft=(x, y))


class PickCoin(pygame.sprite.Sprite):
    """Class to create PickCoin's surface"""
    def __init__(self, x, y):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load('images/monnaie.png')
        self.rect = self.image.get_rect(topleft=(x, y))


class PickSafe(pygame.sprite.Sprite):
    """Class to create PickSafe's surface"""
    def __init__(self, x, y):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load('images/coffre.png')
        self.rect = self.image.get_rect(topleft=(x, y))
```

Encore une partie ?

```python
def restart(win, winWidth, winHeight, text_to_display):
    """Function to restart the game and display Winner or Looser on the screen"""
    text = text_to_display
    xTextPosition = winWidth / 2 - text.get_width() / 2
    yTextPosition = winHeight / 2 - text.get_height() / 2
    widthText = text.get_width()
    heightText = text.get_height()
    pygame.draw.rect(win, (255, 255, 255), ((xTextPosition, 520 - 5),
                                            (widthText + 10, heightText + 10)))

    win.blit(text, (xTextPosition + 5, 520))

    pygame.display.flip()
    play_again = True
    while play_again:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if event.type == pygame.MOUSEBUTTONDOWN:
                main()
                if xTextPosition - 5 <= xTextPosition + widthText + 5:
                    if yTextPosition - 5 <= yTextPosition + heightText + 5:
                        play_again = False
                        break
```

- Ajout d’une fonctionnalité
- Fonction restart()
- Affichage d’un bouton permettant de relancer une partie

Nouvelles constantes

- Deux labyrinthes
- Une musique de fond
- Des sons liés aux actions du joueur

```
pygame.init()
spriteSize = 30
winWidth, winHeight = 510, 550
back = pygame.Surface((winWidth, winHeight))
backgroundMusic = pygame.mixer.music.load("music/soundbay_Epic_Movie.wav")
winningMusic = pygame.mixer.Sound("music/Short_triumphal_fanfare-John_Stracke-815794903.wav")
pickingMusic = pygame.mixer.Sound("music/Ta Da-SoundBible.com-1884170640.wav")
loosingMusic = pygame.mixer.Sound("music/Hl2_Rebel-Ragdoll485-573931361.wav")

maze_1 = ("...")

maze_2 = ("...")
```

Tests unitaires

```
    def test_type_Mazes(self):
        """Test if maze_1 and maze_2 are str"""
        self.assertTrue(maze_1, str)
        self.assertTrue(maze_2, str)

    def test_choose_Maze(self):
        """Test if a maze is randomly chosen"""
        self.assertTrue(self.choice)

    def test_splitting_file(self):
        """Test if a randomly chosen maze is split"""
        for line in self.choice.split('\n'):
            line = list(line)
            self.a_level.append(line)

        self.assertNotIn('\n', self.a_level)


class TestSurfaceCreation(unittest.TestCase):

    def setUp(self):
        self.position = Vector2(45, 45)
        self.Group = pygame.sprite.Group()
        self.surface = Floor(self.position.x * spriteSize,
                             self.position.y * spriteSize)

    def test_sprite_got_rect(self):
        """Test if a surface created got a rect"""
        self.assertTrue(self.surface.rect)

    def test_sprite_creation_Floor(self):
        """Test if a sprite is correctly created as a Floor instance"""
        create_surface = Floor(self.position.x * spriteSize,
                               self.position.y * spriteSize)
        self.Group.add(create_surface)
        self.assertTrue(create_surface)
```

- Ajout de tests unitaires
- Vérification de l'intégrité des fichiers permettant de créer le labyrinthe
- Vérification de l'intégrité des classes permettant de créer les sprites (éléments graphiques visibles par l'utilisateur)