PROJET PYTHON

SPACE SLUG

LE FLOHIC THOMAS
MARTIN Christopher

Python & Pygame

- Python
  - Programmation Orienté Objet
  - Indentation
- Pygame
  - Entrée utilisateur (souris, clavier, joystick)
  - Intégration de fichiers externes (images, sons)
  - Classe facilitant la création d’un jeu (Sprite)

Sprites

- Image, position et hitbox
- Appartenance à un Groupe
- Possible collisions

```
157     #Classe du missile
158     #Class mère : Sprite
159 #-------------------------------------------------------------
160 class newShipMissile(imports.pygame.sprite.Sprite):
161 #Constructeur du missile, on hérite de la classe Sprite pour fac
162     def __init__(self, ship):
163         #Constructeur de la classe Sprite
164         imports.pygame.sprite.Sprite.__init__(self)
165         #On définit l'image, le rectangle et les coordonnées
166         self.image = loader.imageShipMissile
167         self.rect = self.image.get_rect()
168         #On créé le missile centré sur le vaisseau du joueur
169         self.rect.x = ship.rect.centerx - 5
170         self.rect.y = ship.rect.top + 5
171 #On appelle cette methode dans la boucle principale, on déplace
172     def update(self):
173         self.rect.y -= constantes.speedMissile
```

Collisions

- sprite.collide_rect()
  - Groupe de Sprites
  - De pair avec image.get_rect()
  - Valeur de retour de type booléen

```
9   groupShip = imports.pygame.sprite.Group()
10  groupEnemy = imports.pygame.sprite.Group()
11  groupBoss = imports.pygame.sprite.Group()
12
13  groupShipMissile = imports.pygame.sprite.Group()
14  groupEnemyMissile = imports.pygame.sprite.Group()
```

```
83  #-------------------------------------------------------------------------------
84      #On gere les collisions entre notre missile et l'ennemi
85  #-------------------------------------------------------------------------------
86      for shipMissile in spriteManager.groupShipMissile:
87          #Si un missile dépasse l'ecran, on detruit l'objet
88          if shipMissile.rect.y < 0:
89              spriteManager.groupShipMissile.remove(shipMissile)
90          for enemy in spriteManager.groupEnemy:
91              #Si un ennemi et un missile entrent en collision, on détruit les deux objets
92              if imports.pygame.sprite.collide_rect(enemy, shipMissile):
93                  explo = GIF.explosionAnimation(enemy, shipMissile)
94                  spriteManager.explosionGroup.add(explo)
95                  imports.pygame.mixer.music.load('SFX/explosion.wav')
96                  imports.pygame.mixer.music.play(0)
97
98                  spriteManager.groupShipMissile.remove(shipMissile)
99
100                 enemy.life -=1
101                 if enemy.life <= 0:
102                     #Quand on touche un ennemi on incremente le score de 1
103                     constantes.score += 100
104                     spriteManager.groupEnemy.remove(enemy)
105                 #Si pas assez d'ennemis, on en créé d'autre pour plus de fun
106                 if len(spriteManager.groupEnemy) <= 3:
107                     for i in range(imports.random.randint(3, 6)):
108                         enemy = Objects.newEnemy()
109                         spriteManager.groupEnemy.add(enemy)
```

Randomiser l’apparition des Sprites

```
58 enemyImageList.append(imageEnemy)
59 enemyImageList.append(imageEnemy2)
60 enemyImageList.append(imageEnemy3)
61 enemyImageList.append(imageEnemy4)
```

- Choix parmis plusieurs images
  - Liste contenant les images
  - random.choice(listeImage)

```
123 class newEnemy(imports.pygame.sprite.Sprite):
124 #Constructeur de l'asteroide, on hérite de la classe Sprite pour f
125     def __init__(self):
126         #Constructeur de la classe Sprite
127         imports.pygame.sprite.Sprite.__init__(self)
128         #On définit l'image et le rectangle
129         self.image = imports.random.choice(loader.enemyImageList)
130         self.rect = self.image.get_rect()
```

Déplacements des ennemis

- Limite : taille de la fenêtre
- Réutilisation des objets
  - Changement de position

```
        self.life = constantes.lifeEnemy
#Si l'asteroide depasse de l'ecran verticalement on le ramene en haut a une position horizontale aleatoire
    def resetPositionEnemyY(self):
        self.speedY = randSpeed(constantes.speedEnemy, constantes.speedEnemy + 3)
        self.rect.y = imports.random.randrange(-300, -20)
        self.rect.x = imports.random.randrange(constantes.screenWidth/2 - 350, constantes.screenWidth/2 + 350)
#Si l'asteroide depasse de l'ecran, on appelle ResetPosition()
    def update(self):
        self.rect.y += self.speedY
        self.rect.x += self.speedX
        if self.rect.y > constantes.screenHeight - 35:
            self.resetPositionEnemyY()
        #On fait fait passer l'asteroides de gauche a droite et inversement quand il depasse
        if self.rect.x > constantes.screenWidth + 10:
            self.rect.x = -10
        if self.rect.x < -10:
            self.rect.x = constantes.screenWidth + 10
#-----------------------------------------------------------------------------------------------------
```

Déplacements du joueur

- Limite : taille de la fenêtre
- Déplacements XY
  - Flèches directionnelles

```
    def moveUp(self):
        self.rect.y -= constantes.speedShip
    def moveDown(self):
        self.rect.y += constantes.speedShip
    def moveLeft(self):
        self.rect.x -= constantes.speedShip
    def moveRight(self):
        self.rect.x += constantes.speedShip

    def movementShip(self):
        if (imports.pygame.key.get_pressed()[imports.pygame.K_LEFT] == True):
            self.image = loader.imageShipLeft
            self.moveLeft()
        elif (imports.pygame.key.get_pressed()[imports.pygame.K_RIGHT] == True):
            self.image = loader.imageShipRight
            self.moveRight()
        elif (imports.pygame.key.get_pressed()[imports.pygame.K_UP] == True):
            self.moveUp()
        elif (imports.pygame.key.get_pressed()[imports.pygame.K_DOWN] == True):
            self.moveDown()

        if self.rect.x < 0:
            self.rect.x = 0
        elif self.rect.x > constantes.screenWidth - self.rect.width:
            self.rect.x = constantes.screenWidth - self.rect.width
        elif self.rect.y > constantes.screenHeight - self.rect.height - 50:
            self.rect.y = constantes.screenHeight - self.rect.height - 50
    def update(self):
        self.movementShip()
```

Affichage à l’écran

- Limite : taille de la fenêtre
- Déplacements XY
  - Flèches directionnelles

```
250     #On appelle les méthodes update() de chaque objet et
251 #-------------------------------------------------------------
252     spriteManager.groupShip.update()
253     spriteManager.groupEnemy.update()
254     spriteManager.groupBoss.update()
255 
256     spriteManager.groupEnemyMissile.update()
257     spriteManager.groupShipMissile.update()
258 
259     spriteManager.groupLifeBar.update(ship)
260     spriteManager.groupeBossLifeBar.update(boss)
261 
262     spriteManager.explosionGroup.update()
263 
264     spriteManager.groupPowerUp.update()
265 #-------------------------------------------------------------
266 #-------------------------------------------------------------
267     #On dessine nos objets avec les attributs image et posi
268 #-------------------------------------------------------------
269     spriteManager.groupBackground.draw(loader.screen)
270 
271     spriteManager.groupShip.draw(loader.screen)
272     spriteManager.groupEnemy.draw(loader.screen)
273     spriteManager.groupBoss.draw(loader.screen)
274     spriteManager.groupEnemyMissile.draw(loader.screen)
275     spriteManager.groupShipMissile.draw(loader.screen)
276 
277     spriteManager.groupLifeBar.draw(loader.screen)
278     spriteManager.groupeBossLifeBar.draw(loader.screen)
279 
280     spriteManager.explosionGroup.draw(loader.screen)
281 
282     spriteManager.groupPowerUp.draw(loader.screen)
283 
284     menu.displayStats()
```


SPACE SLUG
LE FLOHIC Thomas
MARTIN Christopher
INGE 1A
11 - 2015
CREDIT 99