Programming a mini shooter

Gaëtan Blaise-Cazalet - Responsable de la section programmation

Start code

- main.py

```
import pygame, os

def main():

    # Load
    pygame.init()
    screen = pygame.display.set_mode((800, 600))
    font = pygame.font.Font(None, 24)
    path = os.path.abspath('.') + '/'
    quit_game = False

    while not(quit_game):
        # Inputs
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                quit_game = True
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    quit_game = True

        # Update

        # Draw
        screen.fill((0, 0, 0))
        pygame.display.update()

if __name__ == "__main__":
    main()
```

Player

Display player

- main.py

```
import pygame, sys

    # Load
    ...
    player = pygame.image.load(path+'player.png').convert_alpha()
    ...
    while not(quit_game):
        ...
        # Draw
        screen.fill((0, 0, 0))
        screen.blit(player, (0, 200))
        pygame.display.update()
```

- pygame.image.load(...) has one argument, which is the path to the image on your hard drive
- path + "player.png" concatenate the content of the path variable and "player.png"
- convert_alpha() function change the pixel format of the image, to give the display surface format

Move player

- main.py

```
import pygame, sys

    # Load
    ...
    key_up = False
    key_down = False
    player_x = 0
    player_y = 200
    ...
    while not(quit_game):
        # Inputs
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                quit_game = True

            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    quit_game = True
                if event.key == pygame.K_UP:
                    key_up = True
                if event.key == pygame.K_DOWN:
                    key_down = True

            if event.type == pygame.KEYUP:
                if event.key == pygame.K_UP:
                    key_up = False
                if event.key == pygame.K_DOWN:
                    key_down = False

        # Update
        if key_up:
            player_y = player_y - 5
        if key_down:
            player_y = player_y + 5

        # Draw
        ...
        screen.blit(player, (player_x, player_y))
        ...
```

Limit player move

- We want the player to stay at position 0 if position become inferior to 0
- Same if position go ever screen height (- player height)

Limit player move

- main.py

```
  # Load
   player_y = 200
   player_height = 120

   screen_height = 720
   screen_width = 1280


...


        # Update
        ...
        if player_y < 0:
            player_y = 0
        if player_y > screen_height - player_height:
            player_y = screen_height - player_height
```

Shots

Make the player shoot

- Player will shot projectiles. We want a function to create shots, and an other to draw shots.
- Projectiles are fired when the player presses Space.
- We will store projectile's variables in a map :

```
bullet = { 'x': 120, 'y': y, 'speed': 5, 'image': pygame.image.load(path+'bullet.png').convert_alpha() }
```

- We should have a list of projectiles

Make the player shoot

- main.py

```
 # Load
  key_space = False
  bullets = []

  def create_bullet(y):
      bullet = { 'x': 120, 'y': y, 'speed': 5, 'image': pygame.image.load(path+'bullet.png').convert_alpha() }
      bullets.append(bullet)

  def draw_bullets():
      for bullet in bullets:
          screen.blit(bullet['image'], (bullet['x'], bullet['y']))


...
        # Inputs
        if event.type == pygame.KEYDOWN:
            ...
            if event.key == pygame.K_SPACE:
                key_space = True

          if event.type == pygame.KEYUP:
            ...
            if event.key == pygame.K_SPACE:
                key_space = False


        # Update
        ...
        if key_space:
            create_bullet(player_y)

        # Draw
         ...
         draw_bullets()
...
```

Make the shots move

- Now we want bullets to move from the left to the right
- We create a function that would move each bullet of the bullet list

Make the shots move

- main.py

```
 # Load
  def move_bullets():
      for bullet in bullets:
          bullet['x'] = bullet['x'] + bullet['speed']


        # Update
        ...
        move_bullets()
...
```

One press one shot

- Our player shoots every frame when space bar is pressed
- We want the player to shoot only once when space bare is hit
- Use a boolean to implement this behaviour

One press one shoot

- main.py

```
# Load
...
already_shoot = False
...
def create_bullet(y):
    nonlocal already_shoot
    bullet = {'x': 120, 'y': y, 'speed': 5, 'image': pygame.image.load(path+'bullet.png').convert_alpha()}
    bullets.append(tir)
    already_shoot = True


...
     # Inputs
     if event.type == pygame.KEYDOWN:
          ...
          if event.key == pygame.K_SPACE:
             key_space = True
             already_shoot = False


     # Update
     ...
     if key_space and not already_shoot:
        create_bullet(player_y + 50)
     move_bullets()
...
```

Erase shots to free memory

- If our code is left like that, we would manage a growing number of shots
- We want to erase shots when they get out the screen
- We cannot erase shots while we are going through them
- So we will register in a list the indexes of the shots we want to erase, then delete the shots with those indexes

Erase unused shots

- main.py

```
# Load
...
bullets_to_erase = []
...
def move_bullets():
    for index, bullet in enumerate(bullets):
        bullet['x'] = bullet['x'] + bullet['speed']
        if bullet['x'] > screen_width:
            bullets_to_erase.append(index)

 def delete_bullets(to_erase):
    for index in to_erase:
        del bullets[index]
    to_erase[:] = []


...
      # Update
      ...
      delete_bullets(bullets_to_erase)
...
```

```
def move_bullets():
    i = 0
    for bullet in bullets:
        bullet['x'] = bullet['x'] + bullet['speed']
        if bullet['x'] > screen_width:
            bullets_to_erase.append(i)
        i = i + 1
```

Enemies

Create, draw, move and erase enemies

- You can use the shot algorithm to create enemies
- Use a counter variable to trigger enemy spawn: increment it by 1 each frame, when it exceeds 500, create an enemy
- Enemies go from right to left

Create, draw, move and erase enemies

- main.py

```
# Load
...
 enemies = []
 enemies_to_erase = []

 def create_enemy(y):
     enemy = {'x': screen_width, 'y': y, 'speed': -3, 'image': pygame.image.load(path+'enemy.png').convert_alpha()}
     enemies.append(enemy)

 def draw_ennemies():
     for enemy in enemies:
         screen.blit(enemy['image'], (enemy['x'], enemy['y']))

 def move_enemies():
     for index, enemy in enumerate(enemies):
         enemy['x'] = enemy['x'] + enemy['speed']
         if enemy['x'] < 0:
             enemies_to_erase.append(index)

 def erase_enemies(to_erase):
     for index in to_erase:
         del to_erase[index]
     to_erase[:] = []

 enemy_counter = 0


...
        # Update
        ...
        # Ennemis
        move_enemies()
        erase_enemies(enemies_to_erase)
        enemy_counter = enemy_counter + 1
        if enemy_counter > 500:
            create_enemy(300)
            enemy_counter = 0
...
```

Create enemy at random position

- Import random package and use random.seed() to generate a random series of number
- random.randint(min, max) give a random integer between min and max
- Make enemies appear at a random position

Create, draw, move and erase enemies

- main.py

```
import random


...
   # Load
   ...
   random.seed()


...
          # Update
          ...
          if enemy_counter > 500:
            create_enemies(random.randint(0, screen_height - 120))
...
```

Detect player colliding with enemy

- Here is a code to detect collisions and trigger game over when there is a collision:

```
game_over = False

def collision_player_enemies():
    nonlocal game_over
    for i_enemy, enemy in enumerate(liste_ennemis):
        x1, y1, w1, h1 = player_x, player_y, player_width, player_height
        x2, y2, w2, h2 = enemy['x'], enemy['y'], enemy['image'].get_width(), enemy['image'].get_height()
        if(not(x1 + w1 < x2 or x2 + w2 < x1 or y1 + h1 < y2 or y2 + h2 < y1)):
            destroy_enemy(i_enemy)
            game_over = True
```

- Display a game over text when game is over

Display game over

- main.py

```
import random


...
   # Update
   ...
   if not game_over:
       ...
       collision_player_enemies()
    else:
        if key_space:
             game_over = False

   # Draw
   if not game_over:
       ...
   else:
       screen.blit(game_over_text, (600, 300))
```

Fixes

- We don't wan't the player to shoot when he or she restarts game
- We want all enemies to be erased after game restart

Fixes

- main.py

```
import random

   # Load
   def erase_all_enemies():
        enemies[:] = []
        enemies_to_erase[:] = []

    def collision_player_enemies():
        ...
                erase_all_enemies()
...
   # Update
   ...
   if not game_over:
       ...
    else:
            if key_space:
                game_over = False
                already_shot = True
```