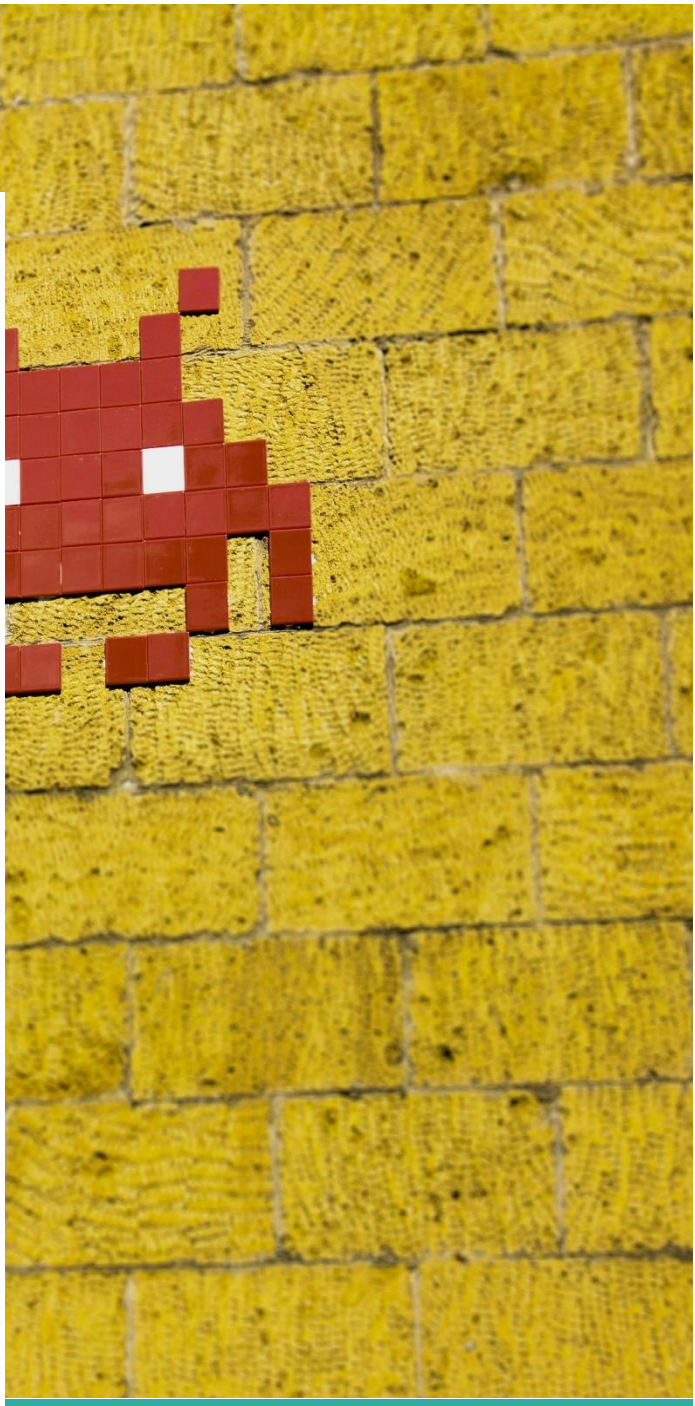
Unidad 2
Space Invaders



14/ 5/ 24

Estructuras Datos Y Algoritmos
Faith Goldsberry

SANTO TOMÁS

DESARROLLO DE ACTIVIDADES UNIDAD 2

Actividad 2.2

Se incorporan:

- Declaración de variables.
- Tipos de datos.
- Operaciones de asignación.

Estos requerimientos se reflejan en varios archivos del juego, pero podemos tomar como ejemplo los atributos del objeto Alien dentro de su respectivo archivo.

Actividad 2.4

Se incorporan:
- Tipos de Operadores
- Literales
- Expresiones

Del punto anterior, se sugiere llevar a cabo el requerimiento N° 2: Alienígenas:
- Cada alienígena debe contar con su propia velocidad y movimiento lateral.
- Los alienígenas deberán descender hacia la nave espacial con el tiempo, aumentando la dificultad
del juego.

Se define la clase Alien en Alien.py, que incluye el uso de distintos tipos de operadores, literales, y expresiones, para dar cumplimiento al requerimiento N° 2.

Game.py | Alien.py | SpaceShip.py

```python
from MiniGameEngine import GameObject

class Alien(GameObject):
    paso_lateral = 100
    paso_vertical = 25

    # inicializamos el Alien
    def __init__(self, x, y):
        super().__init__(x, y, "Recursos/Alien.png", "Alien")
        self.desplazamiento = 0
        self.direccion = "derecha"
        self.orientacion = "adelante"
        self.contador = 0
        self.nivel = 0
        self.y_original = y

    # manejamos las colisiones
    def onCollision(self, dt, gobj):
        if gobj.getTipo() == "Bullet":
            self.destroy()
            print("Alien:me dieron")
        elif gobj.getTipo() == "SpaceShip":
            self._gw.exitGame()
            print("Cuidado con la Nave Espacial")

    # actualizamos el estado del Alien en cada frame
    def onUpdate(self, dt):
        ww = self.getWorldWidth()
        w = self.getWidth()
        hh = self.getWorldHeight()
        h = self.getHeight()
        x = self.getX()
        y = self.getY()

        match self.direccion:
            case "derecha":
                x += 2
```

```
                self.desplazamiento += 2
                if self.desplazamiento >= self.paso_lateral:
                    self.direccion = "abajo"
                    self.desplazamiento = 0
            case "abajo":
                y += 4
                self.desplazamiento += 4
                if self.desplazamiento >= self.paso_vertical:
                    if self.orientacion == "adelante":
                        self.orientacion = "detras"
                        self.direccion = "izquierda"
                    else:
                        self.orientacion = "adelante"
                        self.direccion = "derecha"
                    self.desplazamiento = 0
            case "izquierda":
                x -= 2
                self.desplazamiento += 2
                if self.desplazamiento >= self.paso_lateral:
                    self.direccion = "abajo"
                    self.desplazamiento = 0

        if self.contador == 700:
            y= self.y_original
            self.contador = 0
            self.nivel += 1
        else:
            self.contador +=1
        self.setPosition(x, y)
        for i in range(self.nivel):
            self.desplazamiento +=1

```

Explicación: Cada alienígena cuenta con su propia velocidad y movimiento lateral haciendo uso del Match Case. Se controla cada alienígena y su respectiva velocidad y posición a través de la variable desplazamiento. Así, la velocidad del conjunto y de cada alienígena son iguales. Además, el movimiento lateral se ejecuta de forma parecida. Esto hace que el bloque de Aliens se mueve en conjunto a la misma velocidad y hacia la misma dirección característica del juego original: el zigzag. Mientras se ajusta el valor de la variable desplazamiento, se incrementa la distancia que recorre el conjunto de Aliens en un mismo lapso de tiempo. Todo esto incorpora varios operadores, literales, y expresiones. Para terminar, cada 700 frames, los Aliens van descendiendo cada vez más rápido, a medida que el nivel se incrementa, aumentando así la dificultad del juego.

Actividad 2.6

Se incorporan sentencias de decisión:
- if/elif/else
- if/else
Del punto anterior, se sugiere llevar a cabo los requerimientos:
- Naves Espaciales: Utilizar sentencias if para detectar la entrada del jugador y mover la nave espacial hacia la izquierda o la derecha.
- Colisiones: Implementar sentencias if para detectar colisiones entre los láseres de la nave y los alienígenas, así como entre los alienígenas y la nave espacial.
- Finalización del Juego: Utilizar sentencias if para verificar si los alienígenas alcanzan la nave espacial y terminar el juego en consecuencia.

Se utiliza el archivo SpaceShip.py para mover la nave espacial tanto hacia la izquierda o derecha, como hacia arriba o abajo.

Game.py | Alien.py | SpaceShip.py

```python
import time
from MiniGameEngine import GameObject
from Bullet import Bullet


class SpaceShip(GameObject):
    # inicializamos la Nave Espacial
    def __init__(self, x, y):
        super().__init__(x, y, "Recursos/SpaceShip.png", "SpaceShip")
        self.lastBullet = time.time()

    # actualizamos el estado de la Nave Espacial en cada frame
    def onUpdate(self, dt):
        ww = self.getWorldWidth()
        w = self.getWidth()
        hh = self.getWorldHeight()
        h = self.getHeight()
        x = self.getX()
        y = self.getY()

        # movimiento lateral
        if self.isPressed("a"):
            x = x - 4
            if x - w / 2 < 0:
                x = w / 2
        elif self.isPressed("d"):
            x = x + 4
            if x > ww - w / 2:
                x = ww - w / 2
        elif self.isPressed("w"):
            y = y - 4
            if y - h / 2 < 0:
                y = h / 2
        elif self.isPressed("s"):
            y = y + 4
            if y > hh - h / 2:
                y = hh - h / 2
        self.setPosition(x, y)

        # disparamos una bala
        if self.isPressed("space"):
            if time.time() - self.lastBullet > 0.3:
                Bullet(x, y - 30)
                self.lastBullet = time.time()

```

Explicación: Para dar movimiento lateral y horizontal, el método onUpdate() contiene sentencias de decisión que determinen la posición de la nave en base a las entradas de las teclas a, d, w, o s, moviéndose hacia la izquierda, la derecha, arriba, o abajo, respectivamente.

Se manejan las colisiones a través del objeto Alien dentro del respectivo archivo.

Game.py | Alien.py | SpaceShip.py

```python
from MiniGameEngine import GameObject


class Alien(GameObject):
    paso_lateral = 100
    paso_vertical = 25

    # inicializamos el Alien
    def __init__(self, x, y):
        super().__init__(x, y, "Recursos/Alien.png", "Alien")
        self.desplazamiento = 0
        self.direccion = "derecha"
        self.orientacion = "adelante"
        self.contador = 0
        self.nivel = 0
        self.y_original = y

    # manejamos las colisiones
    def onCollision(self, dt, gobj):
        if gobj.getTipo() == "Bullet":
            self.destroy()
            print("Alien:me dieron")
        elif gobj.getTipo() == "SpaceShip":
            self._gw.exitGame()
            print("Cuidado con la Nave Espacial")

    # actualizamos el estado del Alien en cada frame
    def onUpdate(self, dt):
        ww = self.getWorldWidth()
        w = self.getWidth()
        hh = self.getWorldHeight()
        h = self.getHeight()
        x = self.getX()
        y = self.getY()

        match self.direccion:
            case "derecha":
                x += 2
```

Explicación: Se implementan sentencias if para detectar colisiones entre los láseres de la nave y los alienígenas, así como entre los alienígenas y la nave espacial. Además, verifican si los alienígenas alcanzan la nave espacial, y, de ser así, se ejecuta exitGame()

(de la instancia actual de GameWorld) para terminar el juego.

Actividad 2.8

Se incorporan sentencias de iteración:
- While
- For

Del punto anterior, se sugiere llevar a cabo los requerimientos:
- Alienígenas: Utilizar bucles for o while para mover a los alienígenas horizontalmente y hacia abajo en la pantalla.
- Nivel: Implementar iteración para aumentar la velocidad y dificultad del juego a medida que el jugador avanza en el nivel.

El movimiento de los Aliens, junto con su velocidad y la dificultad del juego, se controlan en el archivo Aliens.py.

Game.py | Alien.py | SpaceShip.py

```python
from MiniGameEngine import GameObject

class Alien(GameObject):
    paso_lateral = 100
    paso_vertical = 25

    # inicializamos el Alien
    def __init__(self, x, y):
        super().__init__(x, y, "Recursos/Alien.png", "Alien")
        self.desplazamiento = 0
        self.direccion = "derecha"
        self.orientacion = "adelante"
        self.contador = 0
        self.nivel = 0
        self.y_original = y

    # manejamos las colisiones
    def onCollision(self, dt, gobj):
        if gobj.getTipo() == "Bullet":
            self.destroy()
            print("Alien:me dieron")
        elif gobj.getTipo() == "SpaceShip":
            self._gw.exitGame()
            print("Cuidado con la Nave Espacial")

    # actualizamos el estado del Alien en cada frame
    def onUpdate(self, dt):
        ww = self.getWorldWidth()
        w = self.getWidth()
        hh = self.getWorldHeight()
        h = self.getHeight()
        x = self.getX()
        y = self.getY()

        match self.direccion:
            case "derecha":
                x += 2
```

```
                self.desplazamiento += 2
                if self.desplazamiento >= self.paso_lateral:
                    self.direccion = "abajo"
                    self.desplazamiento = 0
            case "abajo":
                y += 4
                self.desplazamiento += 4
                if self.desplazamiento >= self.paso_vertical:
                    if self.orientacion == "adelante":
                        self.orientacion = "detras"
                        self.direccion = "izquierda"
                    else:
                        self.orientacion = "adelante"
                        self.direccion = "derecha"
                    self.desplazamiento = 0
            case "izquierda":
                x -= 2
                self.desplazamiento += 2
                if self.desplazamiento >= self.paso_lateral:
                    self.direccion = "abajo"
                    self.desplazamiento = 0

        if self.contador == 700:
            y= self.y_original
            self.contador = 0
            self.nivel += 1
        else:
            self.contador +=1
        self.setPosition(x, y)
        for i in range(self.nivel):
            self.desplazamiento +=1

```

Explicación: Para dar movimiento lateral y horizontal, el Alien tiene un Match Case. Éste le da al conjunto de Aliens movimiento zigzag, como en el juego original. Cuando el conjunto llega a la altura de la Nave Espacial, se sube el nivel y, por consecuencia, la dificultad. Esto quiere decir que se aumenta la velocidad de cada Alien cada vez que el conjunto llegue a la parte de abajo. Se logra el aumento con la

variable desplazamiento, ajustándose para incrementar la distancia que recorre el conjunto de Aliens en un mismo lapso de tiempo. Todo esto se regula a través de un bucle for que se encuentra anidado dentro del método onUpdate().