Universidad de Concepción
Facultad de Ingeniería
Departamento de Ing. Informática y Cs. De la Computación

Proyecto Semestral:

Space Invaders

Alumnos: Vanessa Arriagada
Carlos Provoste
Profesora: Pamela Guevara
Asignatura: Python Científico
Fecha: 17 de Diciembre, 2018

Introducción

El objetivo del presente informe es mostrar el proyecto semestral para la asignatura Python Científico 2018-2, el cual consiste en el desarrollo del videojuego “Space Invaders”.

Space Invaders es un videojuego creado por Toshihiro Nishikado. Fue lanzado al mercado el año 1978. Se puede considerar un matamarcianos en 2D donde el jugador controla una nave terrestre que puede desplazarse de derecha a izquierda o viceversa, que tiene la habilidad de atacar mediante un botón de disparo a los enemigos. El objetivo es eliminar las 24 naves enemigas antes que estas lleguen a la tierra, derrotándolas mediante disparos. El jugador esquiva las balas enemigas, intentado no ser atacado por los enemigos. Una vez que se eliminan a todas las naves enemigas, se pasa al siguiente nivel, aumentando la dificultad y la velocidad de los enemigos.

Se creó un videojuego, que si bien tiene los puntos elementales del original, tiene implementado nuevos objetos, los cuales pueden, tanto ayudar a nuestro jugador, como eliminarlo.

Con esto se presenta la solución propuesta, la implementación del videojuego, los resultados obtenidos y por imágenes del prototipo.

Solución Propuesta

Se implementó el videojuego con el comportamiento del clásico videojuego “Space Invaders”, con movimientos del jugador de izquierda y derecha, opción de disparar a los enemigos, aparición y eventual obtención de objetos especiales, y un menú de opciones al jugador.

El videojuego fue desarrollado usando el lenguaje de programación Python 2.7 con la ayuda de la librería Pygame para asuntos de carácter gráfico.

Toda la lógica del programa fue implementada desde 0, características como:

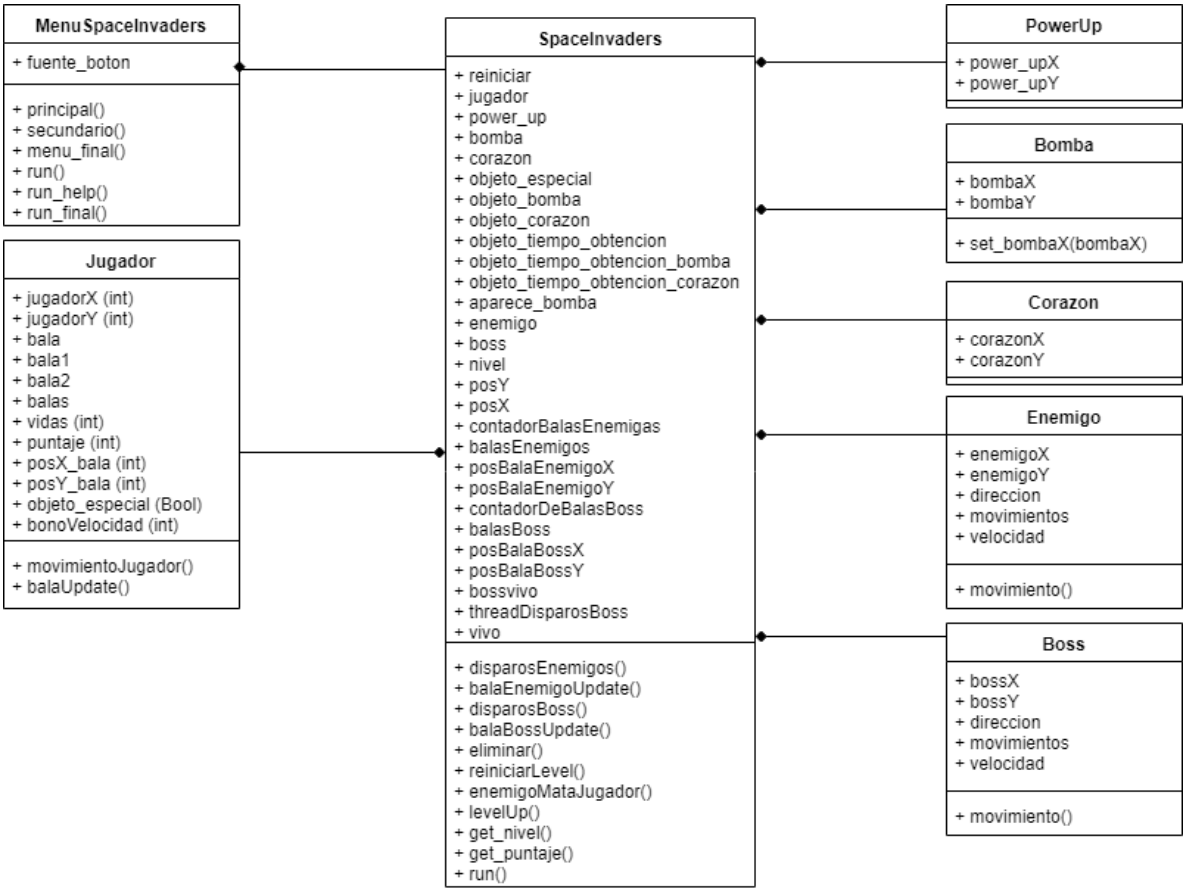
- Comportamiento de los enemigos.
- Movimiento de la nave del jugador.
- Disparo del jugador y enemigos.
- Sistema de colisiones.
- Sistema de puntaje.
- Power Ups.
- Música y efectos según el escenario.
- Etc.

Una vez iniciado el software, se mostrará una pantalla de menú que permitirá al jugador iniciar el juego, ver una pantalla de ayuda o salir del programa.

1. Inicia Juego: Es la función principal, una vez elegida, cambiará a la pantalla de juego donde el jugador debe hacer uso de sus habilidades para eliminar a los enemigos (Más detalles se mostrarán posteriormente en el informe).
2. Ayuda: Muestra una pantalla con los controles del juego.
3. Salir: Cierra el programa.

Al perder el jugador se muestra en pantalla el nivel que alcanzó y el puntaje obtenido, junto con un nuevo menú de opciones para jugar de nuevo o salir.

Implementación



A continuación, se explicará detalladamente la implementación, clase por clase.

1. Jugador: La clase jugador, como se puede deducir, controla ciertas partes de la nave controlable.
    - Atributos:
        i. jugadorX: Maneja la posición X dentro de la pantalla.
        ii. JugadorY: Maneja la posiciónY dentro de la pantalla.
        iii. bala: Objeto que se crea al disparar y con el PowerUp desactivado.
        iv. bala1 y bala2: Funcionan como bala, pero cuando el PowerUp se encuentra activado.
        v. vidas: Contador de vidas del jugador, parte con valor 3 y disminuye en 1 cada vez que el jugador pierde.
        vi. puntaje: Contador del puntaje, parte en 0 y aumenta cada vez que se elimina a un enemigo.

vii. posX_bala: Controla la posición X de la bala en pantalla.
viii. posY_bala: Controla la posición Y de la bala en pantalla.
ix. Objeto_especial: Booleano que determina si el PowerUp se encuentra activo.
x. bonoVelocidad: Posee valor 0 en todo momento excepto cuando el PowerUp se activa, en ese caso, toma el valor = 4, añadiendo más velocidad al jugador.

- Métodos:
  i. movimientoJugador: Controla el movimiento del jugador, cambiando las coordenadas en las que se encuentra la nave cuando el jugador presiona una tecla de movimiento, lo que se ve reflejado en la pantalla.
  ii. balaUpdate: Maneja las posiciones de la bala disparada por el jugador, haciendo que avance en pantalla y también de manera lógica, gracias a este método, se sabe en todo momento donde está la bala que el jugador ha disparado,

2. Enemigo: Esta clase posee lo necesario para programar a los enemigos en pantalla.
   - Atributos:
     i. enemigoX: Atributo que guarda la posición X de un objeto de clase enemigo.
     ii. enemigoY: Guarda la posición Y del enemigo correspondiente.
     iii. dirección: Es un *Entero* que determina la dirección en la que se moverán los enemigos, puede tomar el valor de 1 o -1, si vale 1, los enemigos se mueven de izquierda a derecha, si vale -1, se mueven de derecha a izquierda.
     iv. movimientos: Este atributo guarda la cantidad de movimientos que han ejecutado los enemigos, cuando se llega a cierta cantidad de movimientos, se cambia la dirección del movimiento enemigo.
     v. velocidad: Controla la velocidad a la que se mueven los enemigos, aumenta a medida que se aumenta el nivel.
   - Métodos:
     i. movimiento: Similar al de la clase Jugador, pero acá se controla el movimiento de los enemigos, pudiendo reflejar eso en pantalla.
3. Boss: Esta clase posee lo necesario para programar a la nave enemiga especial que aparece después de algunos niveles dentro del videojuego.
   - Atributos:
     i. bossX: Atributo que guarda la posición X de la clase Boss.
     ii. bossY: Guarda la posición Y de la clase Boss.
     iii. dirección: Mismo funcionamiento de la dirección de enemigos.
     iv. movimientos: Controla el mismo movimiento de los enemigos.

v. velocidad: Controla la velocidad en que se mueve Boss.

- Métodos:
  i. movimiento: Similar al de la clase Enemigo.

4. PowerUp: Clase que controla un Power Up, objeto que cuando se es recogido, otorga ciertas ventajas al jugador durante un tiempo limitado.
   - Atributos:
     i. power_upX: Coordenadas X donde se ubicará el objeto Power Up.
     ii. power_upY: Coordenadas Y donde se ubicará el objeto Power Up.
5. Bomba: Clase que controla un objeto dañino, que al ser tomado por el jugador, éste pierde una vida.
   - Atributos:
     i. bombaX: Coordenadas X donde se ubicará el objeto Bomba.
     ii. bombaY: Coordenadas Y donde se ubicará el objeto Bomba.
   - Métodos:
     i. set_bombaX: Permite cambiar la posición de la bomba. Sirve para que pueda aparecer en cualquier lugar bajo la pantalla.
6. Corazon: Clase que controla un objeto de ayuda, que al ser tomado por el jugador, éste gana una vida.
   - Atributos:
     i. corazonX: Coordenadas X donde se ubicará el objeto Corazon.
     ii. corazonY: Coordenadas Y donde se ubicará el objeto Corazon.
7. SpaceInvaders: Clase principal del juego, contiene 1 objeto de clase Jugador, 24 enemigos, 1 Power Up, 1 Boss cuando se activa, 1 Bomba y 1 Corazón.
   - Atributos:
     i. Reiniciar: Booleano usado para determinar si se ha reiniciado la etapa, permite volver a rellenar con los 24 enemigos, reiniciar contadores, timers, etc.
     ii. objeto_especial: Booleano que permite saber si existe un Power Up en pantalla listo para ser recogido.
     iii. objeto_bomba: Booleano que permite saber si existe una Bomba en pantalla listo para ser recogido.
     iv. objeto_corazon: Booleano que permite saber si existe un Corazon en pantalla listo para ser recogido.
     v. objeto_tiempo_obtencion: Entrega el tiempo exacto en que se ha recogido el Power Up.
     vi. objeto_tiempo_obtencion_bomba: Entrega el tiempo exacto en que se ha recogido la Bomba.
     vii. objeto_tiempo_obtencion: Entrega el tiempo exacto en que se ha recogido el Corazon.

viii. aparece_bomba: Entrega el tiempo exacto en que ha aparecido el Corazon.
ix. enemigo: Lista de todos los enemigos del jugador.
x. nivel: Guarda el nivel en el que se encuentra el jugador.
xi. posX: Se usa para asignar la coordenada X a cada enemigo.
xii. posY: Se usa para asignar la coordenada Y a cada enemigo.
xiii. contadorDeBalasEnemigas: Permite controlar la cantidad máxima de balas enemigas en pantalla.
xiv. BalasEnemigo: Lista que guarda las balas enemigas y sus respectivas coordenadas.
xv. posBalaEnemigosX y posBalaEnemigosY: Coordenadas de cada bala enemiga.
xvi. contadorDeBalasBoss, balasBoss, posBalaBossX, posBalaBossY son atributos que funcionan de forma similar a la de los enemigos.
xvii. bossvivo: Indica si el boss sigue con vida dentro de la pantalla de juego.
xviii. threadDisparosBoss: Controla la hebra para los disparos de boss.

- Métodos:
  i. disparoEnemigos: Maneja la forma en que los enemigos atacarán, es el único método manejado por otra hebra.
  ii. balaEnemigoUpdate: Permite que las balas del enemigo caigan y se remuevan al tocar la parte baja de la pantalla.
  iii. eliminar: Controla la acción de eliminar a una nave extraterrestre cuando una bala del jugador choca con un enemigo.
  iv. reiniciarLevel: Cuando el jugador pierde una vida, este método permite reiniciar todo en pantalla, incluido timers y contadores.
  v. enemigoMataJugador: Cuando una bala enemiga choca con el jugador, se activa este método para hacer todo lo necesario.
  vi. levelUp: Cuando todas las naves enemigas son eliminadas, se aumenta el nivel y se vuelve a llenar todo en pantalla.
  vii. get_nivel: Entrega el nivel actual.
  viii. get_puntaje: Entrega en puntaje actual.
  ix. Run: Método principal de la clase, es un ciclo iterativo que se ejecuta siempre que el jugador tenga vidas disponibles. Contiene a todos los demás métodos en constante ejecución dependiendo de las condiciones de la partida.
  x. disparoBoss y balaBossUpdate manejan de forma similar a los enemigos estos elementos de boss.

8. MenuSpaceInvaders: Clase que maneja los menús del juego, contiene un objeto de clase SpaceInvaders que se activa cuando el jugador inicia la partida.
    - Métodos:
        i. principal: Menú de opciones que permite jugar, obtener ayuda o Salir.
        ii. secundario: Muestra información acerca de los controles y los objetos dentro del juego.
        iii. menú_final: Menú de opciones al final de cada juego. Muestra información del nivel que se alcanzó y su puntuación, junto con entregar la opción de una nueva partida o salir.
        iv. run_help: Ejecuta el menú de ayuda.
        v. run_final: Ejecuta el menú al final de cada juego.
        vi. run: Ejecuta el menú principal.

Si bien no se muestra en los atributos anteriores, se hizo uso de archivos como imágenes y audios para la implementación del fondo, diseño del jugador, los enemigos, los objetos especiales y demás. Con esto se realiza una implementación más completa del videojuego.

Resultados

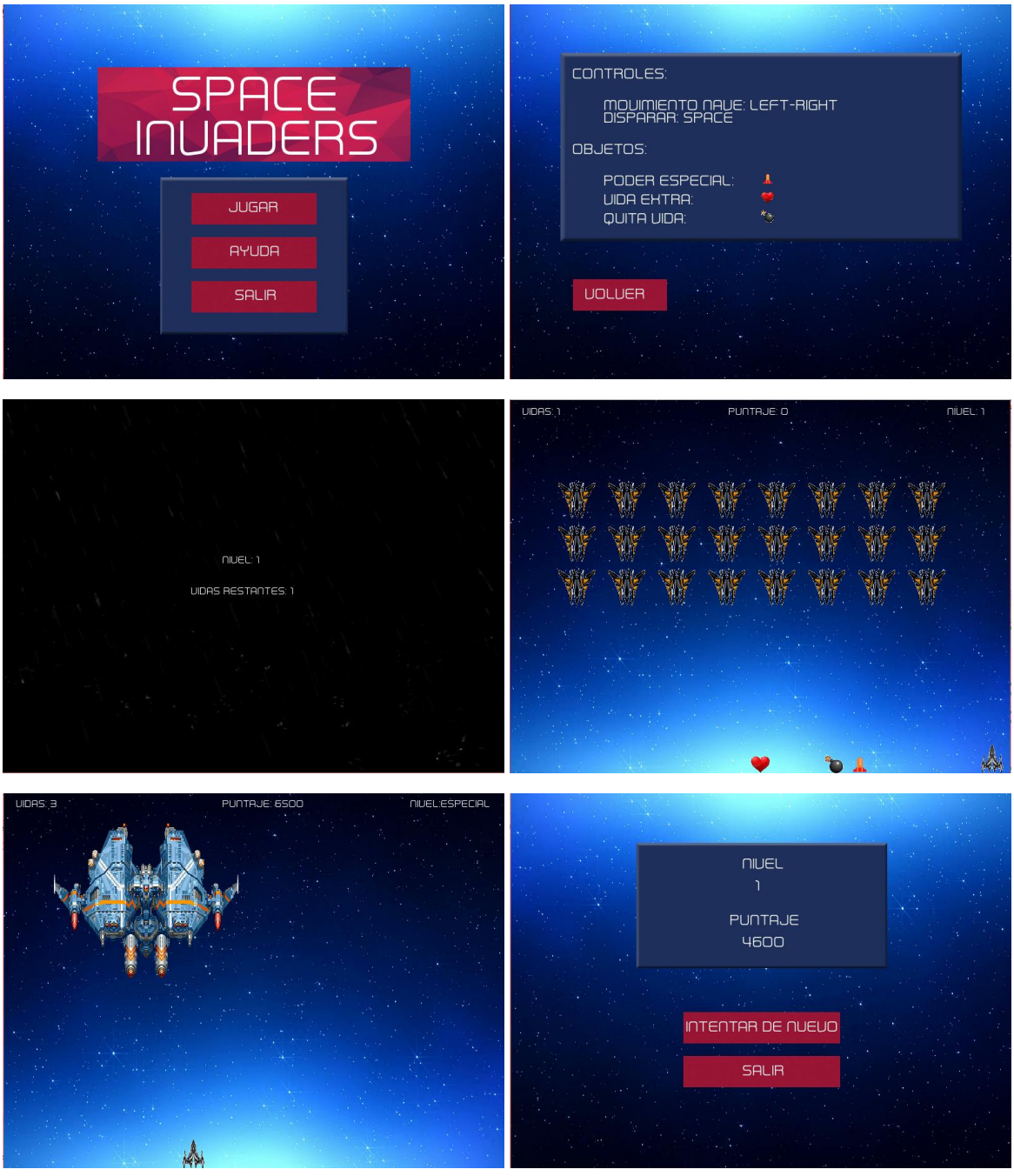
El resultado obtenido es un software completamente funcional, un videojuego totalmente jugable y divertido. Si bien se obtuvieron problemas al momento de implementar cosas nuevas, se lograron realizar a cabalidad los objetivos planteados.

A continuación, una evaluación del software.

- Funcionamiento: 10/10, totalmente funcional.
- Rendimiento: 7/10, si bien rara vez se sufren problemas de rendimiento, se podrían haber usado técnicas de optimización que escapan del curso y requerían más tiempo para implementar.
- Cumplimiento de objetivos: 9/10, casi todo lo planteado inicialmente está incluido en el proyecto final, salvo un modo de 2 jugadores que se decidió quitar por ser algo caótico. Además de añadieron misiones con jefes, que si bien no estaba dentro de la propuesto, se decidió implementar debido a la falencia de multijugadores.
- Originalidad: 7/10, ya que es una réplica de un videojuego ya existente al cual se le añadieron algunos extras como Power Ups, misiones especiales, un diseño renovado y un menú de opciones mejorado.

- Dificultad: 10/10, para los estándares del curso es un proyecto de buena dificultad, implementado desde 0 sin hacer mucho uso de librerías ya existentes.

Capturas de pantalla



Conclusión

Un trabajo que si bien tiene como fin el generar entretención a su usuario, fue a la vez muy divertido de realizar e implementar, además de ser muy útil para aprender el lenguaje de programación foco del curso: Python.

Se adquirieron conocimientos importantes, no solo en el desarrollo de un juego, sino que también en áreas importantes del lenguaje, como Hebras, interfaces gráficas, manejo de clases y objetos, manejo de archivos, etc.

Si bien hubo dificultades al implementar “movimientos”, de forma que no se produzcan errores al momento de realizar alguna acción, se lograron los objetivos generales del proyecto, entregando un videojuego capaz de manejarse de forma sencilla, atendiendo los requerimientos del usuario y haciendo ésta una entretenida experiencia, tanto para niños, como para adultos en busca de recordar viejos momentos.

Como mejoras a futuro se podría considerar equilibrar el manejo de niveles, hacer que la subida de dificultad no sea tan brusca, permitiendo alcanzar una mayor cantidad de niveles para los jugadores, además de introducir mecánicas nuevas al juego, como más variedad de Power Ups.