Carrera: Ingeniería en Computadores
Curso: CE-1102 Taller de Programación, II Semestre 2017
Proyecto programado II, grupo 2
Valor: 100 puntos, 15%
Entrega: 3 de noviembre de 2017, antes de las 23:55
Profesor: Antonio González Torres
Consultas: antonio.gonzalez@tec.ac.cr
Envío: grupo2@gonzalez.cr

1. Objetivo

Implementar un juego de batalla entre naves espaciales utilizando Python, TKinter o Pygame, listas, matrices, archivos JSON y CSV, sonido y programación orientada a objetos con el fin de comprender el proceso de desarrollo de sistemas y los principales conceptos de programación orientada a objetos.

2. Introducción

En el año 1978 Tomoshiro Nishikado creó Space Invaders, un juego clásico que consiste en una batalla entre una nave defensora y un grupo de naves invasoras (http://bit.ly/2zBtSdd). Este juego estuvo inspirado en películas como la Guerra de las Galaxias y La guerra de los mundos y fue uno de los primeros en utilizar disparos para atacar y defenderse de los rivales. En general, el juego se desarrolla de la siguiente forma:

1. Las naves invasoras atacan de forma masiva a la nave defensora.
2. Conforme las naves intrusas disparan, se acercan de forma progresiva a la nave defensora, por lo que esta última debe evitar colisionar con las invasoras.
3. La nave defensora debe evadir los disparos de las invasoras a la vez que lanza tiros explosivos para destruirlas.
4. En el lugar donde se desarrolla la batalla existe un gran número de meteoritos, que deben ser esquivados o destruidos por la nave defensora para evitar una colisión directa.

3. Descripción del proyecto

El juego inicia con un grupo de 30 naves invasoras que se encuentran alineadas en la parte superior de la pantalla y la nave defensora, que se encuentra en la parte inferior. Cuando el juego inicia el bloque de naves intrusas se mueve de forma constante de izquierda a derecha y se comienzan a desprender, una a una, de forma aleatoria, a la vez que varios meteoritos en movimiento se mueven en dirección de la nave defensora.

Las naves que se desprenden del bloque de naves invasoras se mueven en dirección de la nave protectora mientras disparan de forma continua, por lo que esta última debe moverse para evadir los disparos y las colisiones con las atacantes y los meteoritos. Lo

anterior es ilustrado por las figuras 1 y 2, y se recomienda ver el video en el enlace http://bit.ly/2zBtSdd para realizar el diseño y programación del juego.

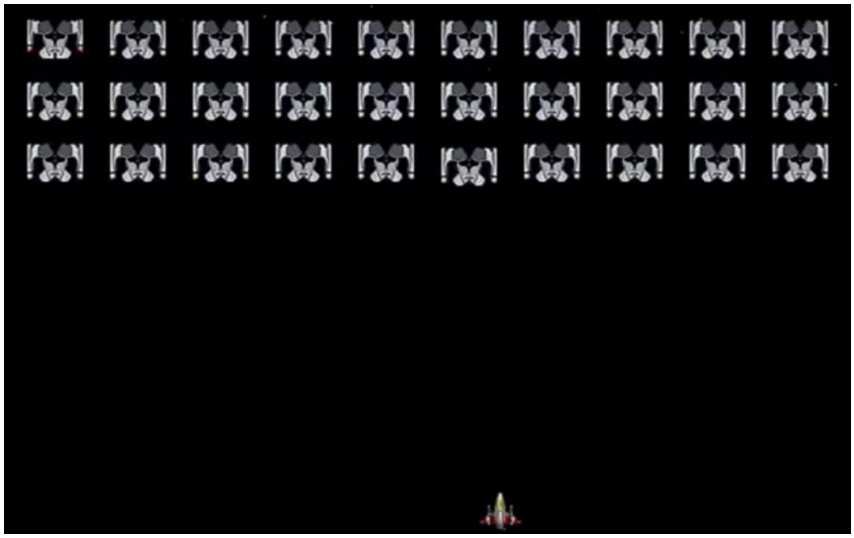

Figura 1. Configuración inicial del juego

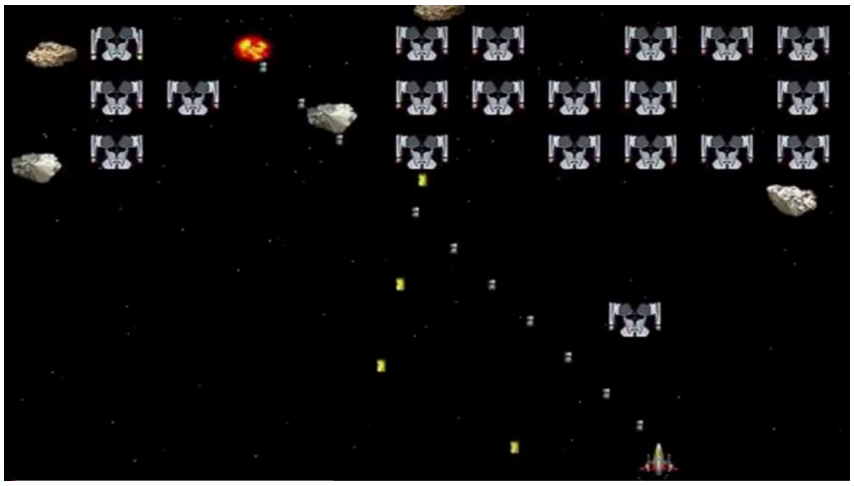

Figura 2. Juego en acción

El juego debe permitir la creación de jugadores y el almacenamiento de los resultados de los jugadores, de forma que cuando el juego no ha comenzado aparecerá una pantalla que mostrará los 5 mejores resultados que han sido obtenidos por los jugadores. En concreto, además de las funciones descritas anteriormente el juego debe permitir:

- → Crear usuarios nuevos o seleccionarlos de una lista cuando estos se encuentren registrados. Los usuarios deben ser almacenados en un archivo CSV.
- → Almacenar el resultado de los 5 mejores jugadores en un archivo JSON.

→ Presentar una pantalla con el resultado de los 5 mejores jugadores, antes de comenzar una partida.

El cálculo del puntaje obtenido por cada jugador se calculará en función del número de naves invasoras y meteoritos que la nave defensora logra derrotar o demoler con disparos (el jugador obtiene un punto por cada una nave o meteorito destruido). Para mantener el registro de las naves intrusas y los meteoritos se recomienda utilizar matrices, de forma que en todo momento sea posible conocer el estado de cada nave y meteorito y el puntaje acumulado por el jugador. Los estados de las naves y meteoritos son activo, destruida o inactiva. El estado activo corresponde a las naves que todavía se encuentran en batalla mientras que la condición de inactivo es asignada a todos objetos que se han movido por debajo de la parte inferior de la pantalla (naves y meteoritos).

La nave defensora tiene solo una vida y en caso de ser destruida por un disparo, el jugador debe comenzar una nueva partida.

4. Investigación

Debe investigar sobre UML, el uso de una biblioteca gráfica como Tkinter o Pygame, así como sobre la utilización de hilos (threads) para gestionar el comportamiento de cada objeto gráfico. Considere investigar sobre funciones para el manejo de archivos JSON y CSV para el almacenamiento de los datos, así como de archivos de sonido para que se escuchen tanto los disparos como las aeronaves (similar al video).

4. Documentación.

La documentación interna se refiere a la inclusión de comentarios en el programa fuente, al menos antes de definir cada función. Estos comentarios explican el detalle sobre lo que realiza la función, las entradas, salidas y restricciones. En el código que define la interfaz gráfica, se debe identificar cada componente utilizado y qué función realiza.

La documentación externa debe incluir:

- Tabla de contenidos o índice
- Introducción
- Descripción del problema.
- Descripción de la solución con un diagrama UML.
- Análisis de resultados. (incluyendo corridas de ejemplo)
- Bitácora de actividades: se deben ir anotando todas las actividades, tipo de actividad, su descripción y duración.
- Estadística de tiempos: un cuadro que muestre un resumen de la Bitácora de Actividades en cuanto las horas **REALES** invertidas como el que se muestra en la tabla 1.

Análisis de requerimientos	xx horas
Diseño de la aplicación y diagrama de clases	xx horas
Investigación de funciones	xx horas
Programación	xx horas
Documentación interna	xx horas
Pruebas	xx horas
Elaboración documento	xx horas
TOTAL	**xx horas**

Tabla 1. Ejemplo de bitácora con el registro de actividades

- Conclusión personal

5. Evaluación.

→ Documentación: 15%
  - ✓ Interna: 5%
  - ✓ Externa: 10%

→ Diagrama de clases y estructura del sistema usando herencia y asociaciones entre clases: 15%

→ Resultados (ejecución, eficiencia, presentación): 70%

→ Puntos **adicionales** por agregar varios niveles de dificultad al juego: 20%

7. Aspectos Administrativos.

La tarea es **individual** y se debe entregar a más tardar el viernes 3 de noviembre de 2017 hasta las 11:55 p.m. en forma electrónica, en un archivo comprimido con el nombre el estudiante, que contenga TODO lo necesario para poder ejecutarla. El archivo comprimido debe tener su nombre en el formato: tp2_xxxx.zip, por ejemplo, en mi caso sería tp2_antoniogonzalez.zip

No se aceptarán tareas después de la fecha y hora indicadas. Debe enviarse un archivo readme.txt con la versión de Python a utilizar para la revisión y alguna otra indicación que se considere importante.

- Se puede utilizar cualquier elemento de la interfaz gráfica de Tkinter o Pygame y **la presentación será un elemento importante dentro de la calificación** (no puede usar ninguna otra biblioteca para la presentación visual).
- No se aceptarán tareas cuyo archivo sobrepase los 2 Mb de espacio.
- Se debe adjuntar la documentación solicitada en formato .doc .odt o .pdf.
- Cualquier falta a los aspectos aquí enunciados implicará pérdida de puntos.

- En caso de probarse algún tipo de fraude en la elaboración de la tarea se aplicarán todas las medidas indicadas al inicio del curso, incluyendo una carta al expediente del estudiante.
- **El profesor se reserva el derecho de calificar forma y fondo de las actividades tomando como referencia la mejor actividad presentada**

8. Bibliografía.

Documentación técnica Python y Tkinter.

8. Consultas.

Puede dirigir cualquier consulta a antonio.gonzalez@tec.ac.cr.