Space Invaders

David Azofeifa

II Semestre, 2017

Contents

1 Introducción **3**

2 Descripción del problema **3**

2.1 Modelo 4

2.2 Vista 4

2.3 Controlador 4

3 Descripción de la solución con diagrama UML **5**

3.1 Nave 6

3.1.1 Atacante 6

3.1.2 Defensor 6

3.2 Jugador 6

4 Análisis de resultados **7**

4.1 Manejo de Atacanes 7

4.2 Control con Joystick 7

4.3 Dificultad 7

5 Bitácora de actividades **8**

5.1 Creación del diagrama UML 8

5.2 Investigación biblioteca pygame 8

5.3 Esqueleto 8

5.4 Medios visuales 8

5.5 Juego 8

5.6 Menús 9

5.7 Finalización de modelo y controlador 9

5.8 Datos en disco 9

6 Estadística de tiempo **10**

7 Conclusión **10**

7.1 Experiencia de usuario 10

1 Introducción

Space Invaders es el segundo proyecto del curso: CE-1102, Taller de Programación. Consta en crear una versión del juego clásico del mismo nombre, utilizando python como lengua y la biblioteca visual llamada pygame. El proyecto busca ampliar el uso de clases para conformar una lógica con mayor capacidad modular. De modo, que el programador adopto el patrón del programación conocido como: Modelo, vista y controlador. O MVC según sus iniciales.

El juego consiste en un batalla entre naves espaciales. Un grupo de naves conforma el grupo de los atacantes los cuales buscan destruir el otro grupo de naves, el cual está conformado por una sola nave defensora. Los atacantes se mueven constantemente y poseen la habilidad de disparar hacia el el defensor. De la misma manera, el defensor puede moverse en las cuatro direcciones y disparar hacia los atacantes. Esta será la nave con la que el usuario interactua. Cada vez que el defensor elimina un atacante, aumenta el puntaje del jugador. La meta es recolectar la mayor cantidad posible de puntaje. Cada jugador tiene su propio record, por lo que compite con los demás directamente para alcanzar el mayor.

2 Descripción del problema

A pesar de este no ser el caso, para un más fácil entendimiento de la funcionalidad individual en MVC, se puede pensar en ello de la siguiente manera. El modelo representa la base de datos encargada de guardar y escribir datos necesarios a largo plazo. La vista, es la interfaz local con la cual el usuario interactua con el programa. Y el controlador es el servidor encargado de comunicar la base de datos y la interfaz. Así como realizar lógica.

Lo siguiente es una explicación más detallada de lo que debe realizar cada parte de MVC dentro del proyecto, la cual dará a entender la problemática a resolver:

2.1 Modelo

El modelo tiene la meta de proveer al programa con los datos necesarios que ya han sido guardados en disco. Así como la capacidad de sobre escribirlos. Los datos guardados en disco que se acceden se encuentran en los archivos jugadores.csv, el cual se encarga de mantener una recolección de los nombres de cada jugador. Y puntajes.json, el cual guarda el puntaje de cada jugador, así como su posición en la tabla de liderato.

El modelo debe tener la capacidad de interpretar la información dada. Por ejemplo, al guardar un puntaje, debe tomar un objeto Jugador y trasformarlo en un diccionario de manera que se le añade a puntajes.json. Por lo tanto, se trabaja con una conversión de tipos dinámica según el caso requerido.

2.2 Vista

La vista conforma la parte visual del programa. Representa las partes con las cuales el usuario tiene la capacidad de comunicarse. En este caso, se utiliza pygame como biblioteca visual.

Para este programa, la vista conforma todos los menús y el juego *(aquí se encuentra el aspecto visual de Space Invaders)*. Por lo tanto estos manejan la lógica que reglamenta el posicionamiento de diversas superficies en la pantalla.

La vista también maneja las entradas dadas por el jugador, aunque tal vez invoque al controlador para interpretar la información recibida, vista la recibe en primer lugar. Por lo tanto, sabe que hacer con ella una vez ingresada.

2.3 Controlador

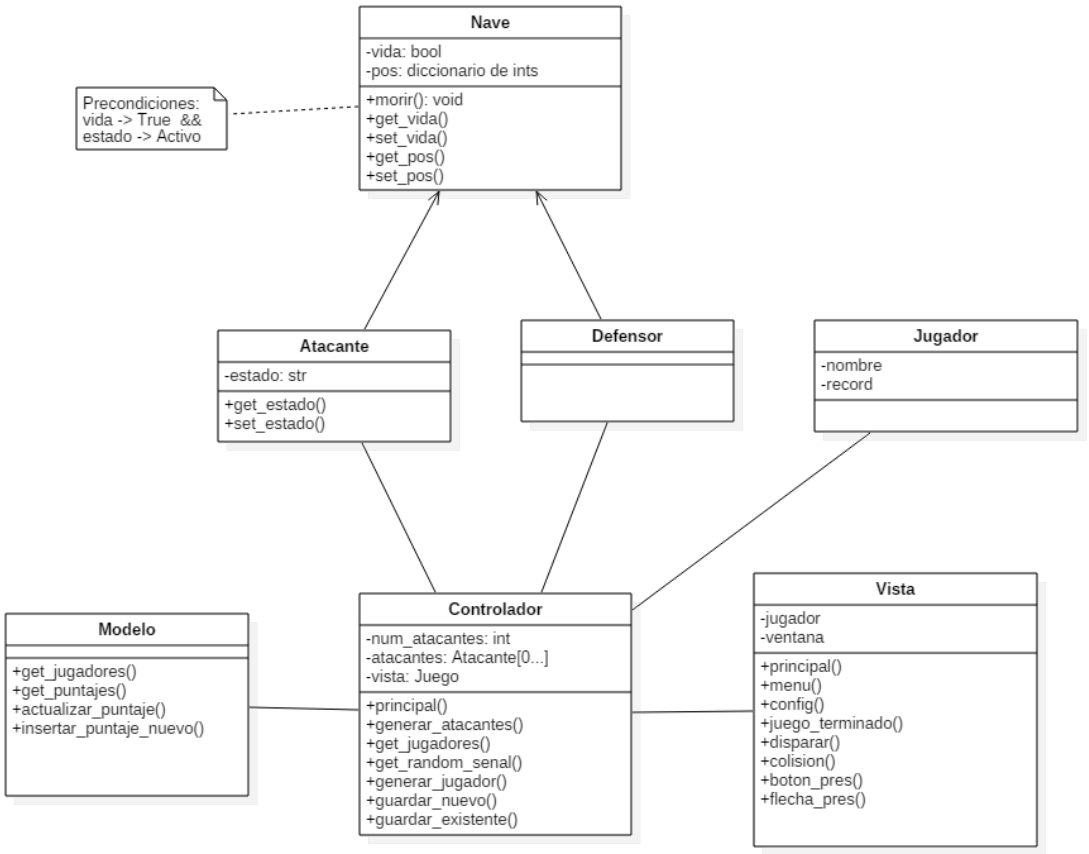
El controlador es el intermediario entra el modelo y la vista. Ninguno de los últimos dos anteriores conocen sobre existencia del otro. Pero una instancia de

controlador se encarga de activar, administrar y comunicar las dos según sus necesidades.

Controlador también se encarga de realizar cualquier tipo de lógica necesaria, fuera del alcance de modelo y vista. Tal como generar las instancias de naves y jugador.

3 Descripción de la solución con diagrama UML

A continuación se encuentra el diagrama UML utilizado en el proyecto:



3.1 Nave

Como se observa en el diagrama, se crea una clase diseñada para ser heredada por Atacante y Defensor llamada Nave. Esta clase es un plano de los requerimientos que debe seguir cada Nave independiente del bando donde se encuentre. Esta posee atributos de vida y poción dentro de la ventana donde se mostrara la vista.

3.1.1 Atacante

Objeto de nave atacante. Aparte de los atributos encontrados en su subclase Nave, esta requiere un atributo de estado. Estado indica cual es la condición del atacante dentro de un estado particular del juego. "activo" significa que la nave posee vida y está dentro de las posiciones que el usuario puede visualizar. Mientras que "inactivo" no significa que necesariamente la nave es eliminada, más bien indica que se salió del rango de la pantalla. Por lo tanto, no se puede interactuar con ella.

3.1.2 Defensor

Objeto de nave defensora. Dado que esta nave no requiere con atributos o funciones a las ya poseídas en su subclase Nave, basta con inicializar la subclase.

3.2 Jugador

Objeto Jugador que contiene como atributo el nombre y puntaje de un jugador particular. Permite tener una recolección del jugador durante la partida. Una vez terminada este se traduce en un diccionario el cual se guarda en puntajes.json y el atributo nombre en un string guardada en jugadores.csv.

4 Análisis de resultados

4.1 Manejo de Atacanes

Durante el desarrollo de vista, la matriz mostraba problemas con el algoritmo utilizado para atravesarla y así mover los atacantes. Para su resolución se debió invertir su recorrido, atravesando primero columnas y luego filas. Así como pulir el algoritmo hasta comprobar que cada uno de los atacantes fuera analizado según se debe.

4.2 Control con Joystick

Dado que es un módulo poco utilizado por la comunidad, el aspecto más tedioso que enfrento el programador recayó en la investigación. Una vez encontrada la información requerida, se experimentó con los botones que conforman el movimiento de la nave. En finalidad, se debió utilizar un algoritmo que considere todas las posibilidades posibles de movimiento con flechas(incluyendo movimiento diagonal presionando dos flechas al mismo tiempo), para garantizar una experiencia placentera al jugar con el joystick.

Por su naturaleza, pygame maneja los eventos generados por un joystick de diferente manera a los generados por un teclado, por lo tanto, se debió implementar una solución adicional que permite generar todas las acciones que son permitidas con teclado también en con joystick.

4.3 Dificultad

Una vez cumplidos los requerimientos necesarios del juego, se experimento con su dificultad. Como consecuencia, se modificaron los atributos para ofrecer una experiencia que presente un desafío al jugador, sin ser injusto.

5 Bitácora de actividades

5.1 Creación del diagrama UML

Como primer paso creó un primer borrador del diagrama UML cual detalla la estructuración del proyecto. El diagrama fue evolucionando de forma dinámica durante la realización del mismo. Dado que se dio conocer aspectos que se ocupaban añadir, reemplazar o eliminar.

5.2 Investigación biblioteca pygame

Previo a comenzar a programar, el programador investigó sobre la biblioteca pygame. Para así aprender a utilizar la herramienta de manera que el proyecto se pueda realizar de forma más fluida una vez se empiecen a escribir lineas de código.

5.3 Esqueleto

Se creo un esqueleto temporal de lo que estructura el programa según el patrón ya expuesto MVC.

5.4 Medios visuales

Se recolectaron las imágenes y sonidos necesarios para implementarlos en la interfaz del juego.

5.5 Juego

Parte más extensiva del programa. Se crea la interfaz que presenta el juego Space Invaders. Así como la habilidad de controlarlo con teclado o control joystick **(cualquiera compatible con windows y que tenga por lo menos tres botones!).**

En esta actividad se creo el manejo de matriz que contienen instancias de Atacante. La creación y manejo de disparos para ambos bandos. El movimiento de los atacantes y él defensor. Así como todo el texto mostrado en pantalla.

5.6 Menús

Creación de los menús dentro de la interfaz de diversos propósitos: Bienvenida, configuración de partida y juego terminado. Estos menús tienen sus propios ciclos ya que se manejan de la misma manera que el juego principal por pygame.

5.7 Finalización de modelo y controlador

Una vez finalizada la creación de vista y el diagrama UML, se determino las operaciones faltantes según las necesidades no cumplidas. Estas fueron añadidas al código de modelo y controlador.

5.8 Datos en disco

Se implementó el algoritmo utilizado para lograr guardar los datos que se desean de los jugadores a largo plazo. Así como lograr una operación sin fallo entre este y los parámetros que puede recibir.

6 Estadística de tiempo

FUNCION	**HORAS**
Análisis de requerimientos	4
Diseño de la aplicación	14
Investigación de funciones	8
Programación	40
Documentación Interna	2
Pruebas	8
Elaboración del Documento	4
Total	60*

***** *Varias horas fueron usadas intersecando con el tiempo utilizado para otras tareas, por ende el total no es la suma de las mismas.*

7 Conclusión

MVC Utilizar objetos en conjunto con un patrón de diseño, permite una solución mucho más organizada del código implementado. Al dividir en módulos las especificaciones, se crea una visualización más clara de la funcionalidad. También permite repartir el proyecto en tareas pequeñas menos abstractas y realizables.

7.1 Experiencia de usuario

Al cumplir los requerimientos mínimos fue claro que entregarle una experiencia atractiva al usuario era la siguiente prioridad. Para el juego cumplir su misión, no bastan con definir el objetivo del juego de manera que baste llegar el punto B del A. El camino que recorre el jugador es lo más importante, ya que si este es desafiante, este sentirá una sensación de logro.

El programador notó que al permitir que personas externas al proyecto utilizar y probaran libremente el juego, estos proveyeron retroalimentación que permitió mejorar la experiencia. El cual permitió la internacionalización de la importancia que conforma la prueba del programa, tanto por parte del desarrollador, como de externos.