Instituto Tecnológico de Costa Rica

Área académica de Ingeniería en Computadores

CE 1102 Taller de programación

Segundo Proyecto Programado

Prof. Jeff Schmidt Peralta

Estudiantes:

Gabriel Chacón Alfaro - 2021049454

Jimena León Huertas - 2021016748

I semestre

2021

Tabla de contenidos

Enlace del proyecto en GitHub

https://github.com/GChacon8/Proyecto-2-Taller

Enlace del video explicativo en YouTube

https://youtu.be/B7qGsFGwG4Q

Introducción

Este documento corresponde al segundo proyecto programado del curso taller de programación, en el cual se solicita crear un juego haciendo uso de la Programación Orientada a Objetos en el lenguaje de alto nivel Python. Se definirán algunos conceptos relevantes a continuación.

La P.O.O se originó a raíz de la necesidad de utilizar el pensamiento natural del programador, de forma que la modelación de un sistema complejo se entendiese como la composición de muchos subsistemas. Por ejemplo, en un computador (sistema complejo) se pueden identificar varios subsistemas generales: monitor, mouse, CPU, parlantes, cámara, micrófono y demás periféricos. El enfoque de la POO es tomar cada subsistema como una "clase". Ahora bien, es posible, por ejemplo, contar con varios dispositivos USB; en ese caso, la clase sería el dispositivo USB y cada uno de ellos sería una instancia de dicha clase (Mathew, J. E., 2010).

Cada clase cuenta con ciertas características especiales; a ellas se les conoce como "atributos", de forma que cada instancia creada las tendrá. En palabras de Mathew, J. E., (2010): "los atributos de una clase son aquellos valores cuantificables o distinguibles que pueden ser asignados a muchas instancias de una clase" (pág. 20). Estos atributos también se conocen como variables de instancia.

Además de los atributos, una clase puede contar con "métodos", los cuales son funciones para, utilizando esos atributos u otras variables, realizar alguna acción en particular, propia de la clase. Como se verá en los apartados posteriores, es posible crear una interacción entre dos objetos, por medio de mensajes o envíos de datos.

Reglas de Grupo y Roles

1. La comunicación se realiza por WhatsApp, Discord, Zoom y cualquier otra plataforma conveniente.
2. Programar únicamente cuando ambos miembros estén reunidos virtualmente por videollamada, a excepción de acuerdo previo.
3. Los roles son: programador/encargado y supervisor.
4. Se turnarán los roles en cada sesión. Ejemplo: hoy Gabriel es el programador y Jimena supervisa y da ideas. En la próxima, Jimena programará y Gabriel supervisará.

Descripción del problema

El juego cuenta con un avatar o personaje principal, el cual debe esquivar proyectiles en tres niveles, cada uno con una dificultad mayor que su antecesor, pues los proyectiles aparecen y se mueven más rápido. Al programar el comportamiento de estos se debe crear un objeto para el avatar principal y un objeto para los proyectiles, el cual se deberá instanciar oportunamente. En el nivel 1 los proyectiles aparecen cada 5 segundos y se mueven con baja rapidez, en el nivel 2 aparecen cada 3 segundos y se mueven con rapidez intermedia, mientras que en el nivel 3 aparecen cada segundo y se mueven con una rapidez mayor. Cada nivel tiene una duración de exactamente 60 segundos (un minuto). Si el jugador sobrevive todo ese tiempo, gana el nivel y se le muestra un mensaje en pantalla. Por otro lado, pierde el nivel si colisiona con los proyectiles y su vida llega a cero antes del tiempo establecido.

Al volver a la pantalla de inicio es posible acceder a una pantalla de mejores puntajes, de forma que se aprecian los diez mejores puntajes leídos desde un archivo de texto (.txt) y ordenados mediante el algoritmo quicksort. En caso de necesitar información general sobre los autores y lugar de producción del juego se accede a la pantalla “about” presionando un botón en la pantalla de inicio designado para ello. También se muestra información sobre la universidad y el presente curso.

Se debe escuchar música en cada nivel y en la pantalla de inicio, a menos que el usuario no lo desee. Por ello, se debe dar la opción de detener o empezar la música. Además, en cada nivel debe escucharse un sonido de rebote de cada proyectil con los bordes de la pantalla, de forma que el sonido sea acorde con la temática escogida.

Modelo de objetos: diagramas de clase y diagramas de secuencia

Figura 1. Diagramas de clase.

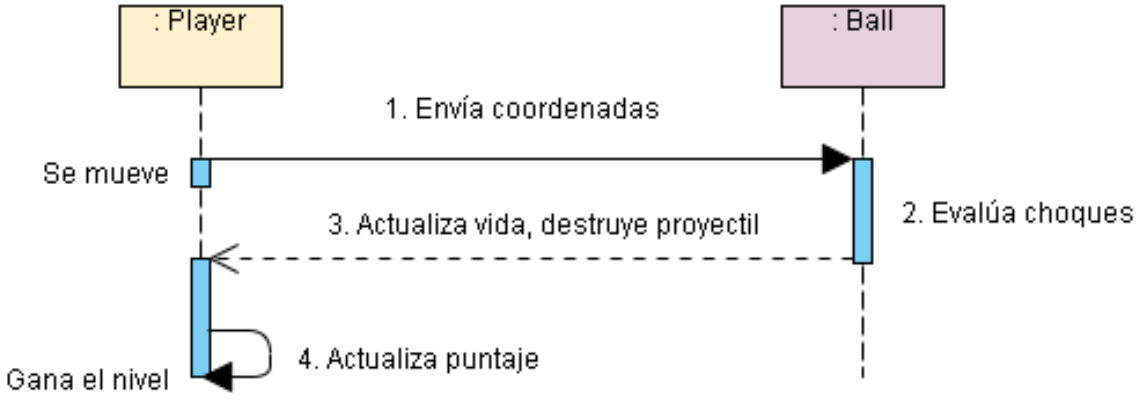

Figura 2. Diagrama de secuencia.

Cabe destacar que el diagrama de secuencia mostrado en la figura 2 también aplica para cuando el Player pierde el juego. Además, en el punto “4” se actualiza el tiempo, de la mano con el puntaje.

Análisis de resultados

La temática escogida es sobre Dodgeball (quemados); los proyectiles son pelotas deportivas y el avatar es un estudiante. La música varía en cada nivel y mientras se juega es posible escuchar un sonido de rebote de los proyectiles con los bordes de la pantalla. Las librerías utilizadas fueron Tkinter para la interfaz gráfica, Winsound y Pygame para la música y sonidos de fondo, Time para establecer tiempos de espera en algunos ciclos iterativos y Random para generar números aleatorios en el método moveBall de los proyectiles. Sus respectivos autores se encuentran en la pantalla de “about”.

En la pantalla de inicio se cuenta con un entry para escribir el nombre del jugador, dos radiobutton para escoger tener música o no, aunque por defecto se muestra la opción de “sin música”; seguidamente, se muestra un combobox para seleccionar el nivel del juego. Finalmente, en la parte inferior se muestran tres botones y cada uno de ellos accede a una pantalla distinta. Para crear y posicionar estos Widgets se tomó como referencia la documentación de Tkinter.

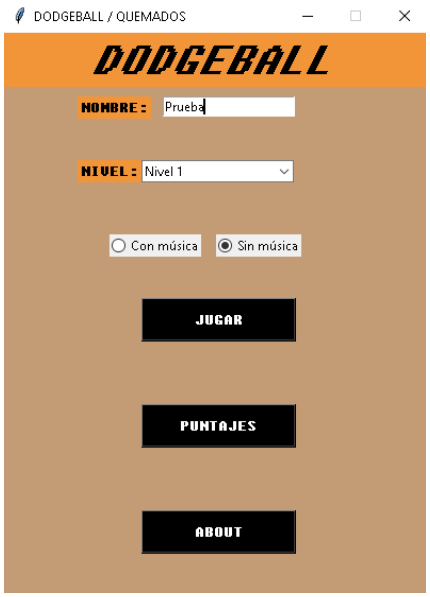

Figura 3. Pantalla de inicio.

Para las otras pantallas -mejores puntajes y about- se definieron funciones independientes, accesibles desde botones en la pantalla de inicio. La pantalla de about contiene labels con

información general de los estudiantes, así como del curso, a modo de portada o presentación. Además, se posicionó un botón para volver a la pantalla de inicio.

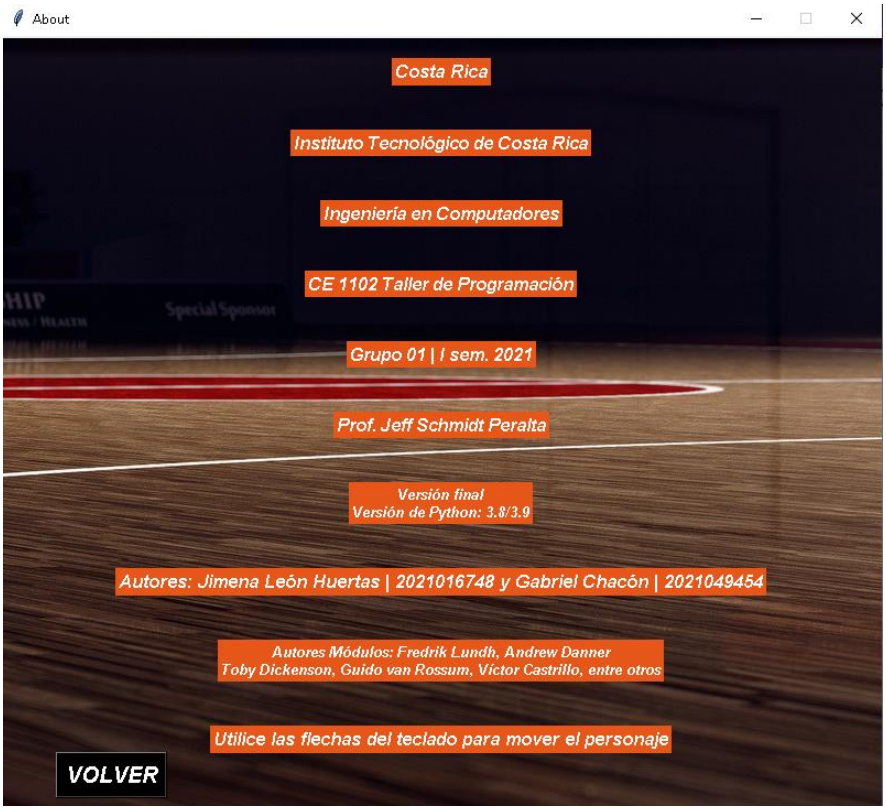

Figura 4. Pantalla del “about”.

Por otro lado, en la pantalla de mejores puntajes es posible observar varios labels mostrando los diez mejores puntajes leídos desde un archivo de texto (.txt). No obstante, si no se ha creado el archivo de texto y por ende no se han registrado puntajes, se muestra un label indicándose lo anterior.

Figura 5. Pantalla de mejores puntajes cuando no hay un .txt creado.

Se podrá apreciar en el código que para la creación del juego se definieron dos objetos: Player y Ball. El primero hace referencia a todo lo relacionado con el avatar principal (el estudiante), mientras que el segundo a todos los proyectiles que aparecen cada cierto tiempo en la pantalla. Para la interfaz de la pantalla de juego se definió una función llamada “Play”, en la cual se crea una instancia del objeto Player y se llama a otra función para crear las instancias del objeto Ball por medio de Threads o hilos.

Al escoger un nivel en la pantalla de inicio y presionar el botón “jugar” se cierra la ventana de inicio y se abre la ventana de juego. Allí, empieza a correr el tiempo y se puede mover la imagen del jugador mediante las flechas del teclado. Los proyectiles aparecen cada 5 segundos en el primer nivel, cada 3 segundos en el segundo nivel y cada segundo en el tercer nivel. Además, en cada nivel la velocidad en X y en Y de los proyectiles es distinta, de forma que en el nivel 3 se mueven más rápido que en el nivel 1. Con ello, se consigue aumentar la dificultad de los niveles. En cada nivel la imagen de fondo es la misma y es estática, mientras que la música sí depende del nivel escogido. Si no se desea música esto se puede seleccionar desde la pantalla de inicio, pero se escuchará el sonido de rebote de las pelotas deportivas con los bordes de la pantalla independientemente de esta selección.

A continuación se muestra la pantalla de juego en la Figura 6.

Figura 6. Pantalla de juego.

Al ganar el nivel se muestra un label indicando lo anterior y si el puntaje del jugador superó alguno de los mejores puntajes también se le comunica por medio de un label. Al perder el nivel sucede algo similar, con la excepción de que el primer label indica que ha perdido el juego.

Figura 7. Pantalla de juego al ganar un nivel.

Finalmente, la pantalla de mejores puntajes se actualiza, mostrando de forma descendente los diez mejores puntajes. Si se ha jugado el juego menos de diez veces, ordena únicamente los puntajes existentes.

Figura 8. Pantalla de mejores puntajes luego de jugar varias veces.

Dificultades encontradas

Thread para el movimiento de los proyectiles: al crear el objeto proyectil (Ball) se necesitó de un thread para que el movimiento se efectuase continuamente y con varias instancias en un mismo programa. Inicialmente, este se posicionó dentro de la función de la interfaz del juego (la función "play"). Al ejecutar se mostraba un error, pues el thread se encontraba dentro del thread principal. Un thread secundario no es capaz de crear otros threads; por lo tanto, se colocó este thread fuera de la función de la interfaz, mediante una función (threadBall) que, desde la función de la interfaz, llama a la función del thread del movimiento de los proyectiles (ballSet). De esta forma se logró crear varios proyectiles en un mismo nivel.

Música y sonido de rebote: inicialmente se pretendía usar únicamente la librería Winsound para la reproducción de la música y el sonido de rebote de los proyectiles. Sin embargo, al probar las instrucciones detalladas en su documentación la música se interrumpió, de forma que al reproducirse el sonido de rebote se detenía la reproducción de la música del nivel. Por ello, se optó

por utilizar una librería adicional. Así, la reproducción del sonido de rebote no interferiría con la reproducción de música de cada nivel.

Bitácora de actividades

Tabla 1. Bitácora de actividades y demás datos relevantes.

Fecha	Actividad	Duración (horas)	Descripción	Programador / encargado	Supervisor
Domingo 06/06/21	Investigación sobre GitHub y Git	2	Ambos estudiantes abrieron una cuenta en GitHub, descargaron Git y GitHub Desktop	Ambos	Ambos
Martes 08/06/21	Creación de ventanas	1	Se creó la interfaz inicial del juego	Ambos	Ambos
Miércoles 09/06/21	Clase juego y movimiento	3	Se definió una clase juego en donde se programó el movimiento del jugador	Gabriel	Jimena
Jueves 10/06/21	Clase Player, clase Ball y movimiento de Ball	4	Se borró la clase juego y se situó su contenido en la función play. Además, se crearon las clases del jugador, del proyectil y se programó el movimiento de ambos	Jimena	Gabriel
Sábado 12/06/21	Niveles, colisiones y proyectiles	5	La sesión consistió en programar la detección de colisiones dentro de la clase del proyectil y algunos aspectos básicos sobre los niveles.	Gabriel	Jimena
Domingo 13/06/21	Variables, puntajes, ganar/perder	6	Se definieron las variables del objeto player: vida, puntaje y tiempo. Con ellas, se establecieron las condiciones para ganar y perder el juego. Estas variables se actualizan	Jimena	Gabriel

			según las colisiones detectadas con los proyectiles. Además, se inició el algoritmo para el manejo de los puntajes en un archivo .txt.		
Lunes 14/06/21	Mejores puntajes (quicksort)	4	Se realizó el algoritmo para que se guarden los puntajes en un archivo .txt y estos se lean para ser ordenados por el algoritmo "quicksort". Seguidamente, estos puntajes ordenados se muestran en la ventana de Mejores puntajes	Gabriel	Jimena
Jueves 17/06/21	Puntajes, sonido y estética de interfaz	4	Se terminó de programar la lógica referente al manejo de los puntajes para después colocar música en cada nivel y pantalla de inicio. Finalmente, se agregaron imágenes de fondo a ciertas pantallas y se cambiaron los tipos de letra para así mejorar la apariencia de la interfaz. Además, Gabriel realizó el archivo sobre el algoritmo quicksort y en Excel determinó el promedio de los tiempos de ejecución.	Jimena	Gabriel
Jueves 17/06/21	Correcciones finales y documentación interna	3	Se detectaron y corrigieron algunos errores relacionados al funcionamiento del juego. Por otra parte, se añadió la documentación interna del código.	Gabriel	Jimena
Lunes 21/06/21	Elaboración y edición del video final	2	Se grabó la explicación del código y una demostración	Gabriel	Jimena

			del juego.		
Lunes 21/06/21	Documentación	3	Se redactó la documentación externa.	Jimena	Gabriel

Estadística de tiempos

Tabla 2. Tiempo empleado por Gabriel en las diferentes actividades.

Rol / Actividad	Duración (horas)	Porcentaje
Programar	20	51, 28 %
Supervisar	17	43, 53 %
Editar video	2	5,30 %
TOTAL	**39**	**100 %**

Tabla 3. Tiempo empleado por Jimena en las diferentes actividades.

Rol / Actividad	Duración (horas)	Porcentaje
Programar	17	42,50 %
Supervisar	20	50,00 %
Redactar documentación	3	7,50 %
TOTAL	**40**	**100 %**

Conclusiones

Con este proyecto se evidenció que la capacidad de un thread secundario es limitada al estar dentro de un thread principal (main thread). Por ello, lo mejor es posicionarlos fuera de dicho thread. Además, aprendimos que la POO funciona para modelar el comportamiento general de juegos con muchos proyectiles, de forma que previene la repetición de código para cada uno de ellos. Finalmente, con este proyecto se fortalecieron habilidades como trabajo en equipo, orientación a los detalles y pensamiento crítico.

Literatura o Fuentes Consultadas

Algoritmo Quicksort:
https://www.youtube.com/watch?v=kFeXwkgnQ9U&ab_channel=DerrickSherrill

Iconify y Withdraw:
https://stackoverflow.com/questions/22834150/difference-between-iconify-and-withdraw-in-python-tkinter

Mathew, J. E. (2010). *Experiencing Object-Oriented Concepts: For Beginners.*AuthorHouse. https://books.google.co.cr/books?id=xkm3X1-H7tsC&printsec=frontcover&dq=oops+concepts&hl=es&sa=X&redir_esc=y#v=onepage&q=oops%20concepts&f=false

Pygame documentación:
https://www.pygame.org/docs/

Tkinter documentación:
https://docs.python.org/3/library/tk.html