Documentación Externa

Valley Jump

Daniela Brenes Otárola

Taller de Programación

Tabla de Contenidos

Introducción

En el presente documento se explica detalladamente la creación del juego de plataformas “Valley Jump”, que consiste de tres niveles de diferente dificultad, es decir, con más plataformas que superar. El objetivo del juego se basa en llegar hasta la plataforma más alta, esquivando a los enemigos del personaje principal, para conseguir el botín del cofre.

En las siguientes páginas se podrán encontrar explicaciones en detalle de la confección del juego, múltiples dificultades encontradas en el proceso, imágenes representativas de funcionalidades implementadas, una bitácora verídica del tiempo de trabajo en el proyecto, y por último, una conclusión personal.

Descripción del problema

El proyecto consiste en un juego de plataformas, en el que el jugador se mueve a los lados, sube de plataforma por medio de escaleras, salta o esquiva enemigos hasta llegar a la parte más alta donde hay un objetivo que alcanzar, por lo que la configuración de colisiones, gravedad y controles del jugador era esencial. El juego debía de componerse de tres niveles, cada uno con más plataformas, y estos niveles debían de generarse de una manera parcialmente automática, al reciclar código. Una opción para pasar una pantalla era indispensable, ya que un menú, las opciones, la pantalla del juego en sí y la tabla de puntuaciones tenían que mostrarse. Para las puntuaciones se debían de utilizar archivos de texto, para mantener guardadas dichos puntajes. También se debía de incorporar una pantalla animada al inicio de cinco segundos. Las animaciones de los personajes con sprites era algo importante, al igual que el sonido del juego. La cantidad de vidas y el nivel tenían que tener la capacidad de ser escogidas por el usuario.

Se utilizaron clases para crear al jugador y al enemigo, y también a cada pantalla del juego. Esto facilitaba la referencia de características de cada uno, por ejemplo, las vidas del jugador. La aparición de cada pantalla se controla con un diccionario formado por las flags de cada pantalla y su valor de verdad. Los botones en el juego, ya sea para cambiar de una pantalla a otra o para seleccionar las vidas y la dificultad, fueron hechos por medio de una función que utiliza las coordenadas y acciones del mouse para verificar que se esté clickeando la imagen del botón.

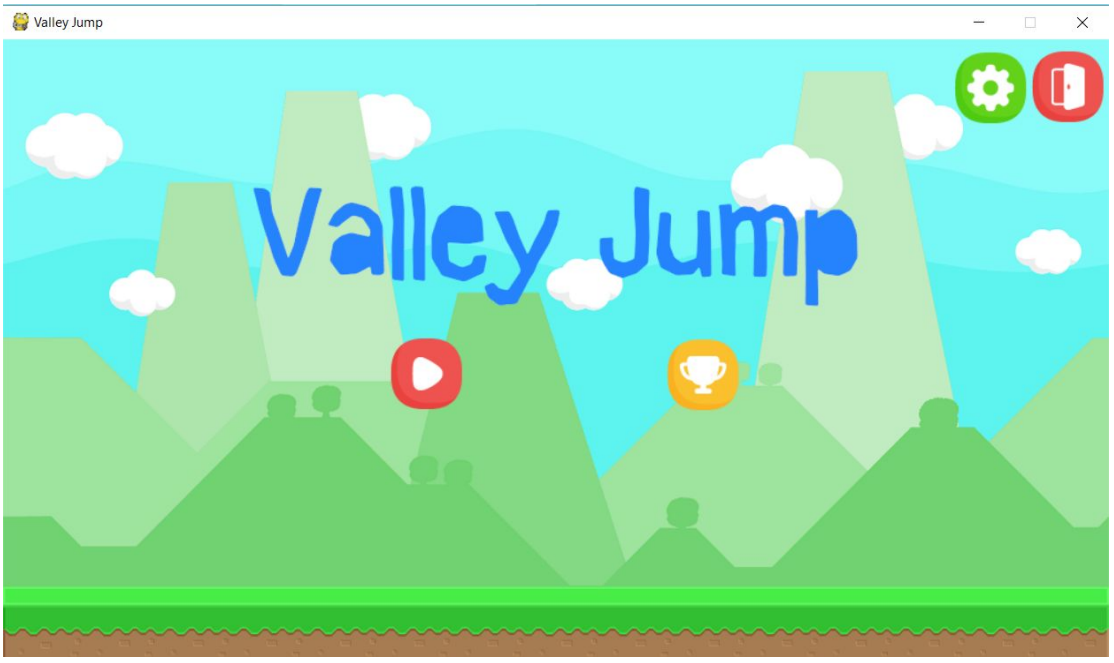
El movimiento del jugador y de los enemigos fue primero controlado por rectángulos, y una vez que la mecánica del juego estuvo lista, se cambiaron estos rectángulos por imágenes animadas.

Los puntajes son guardados en un archivo de texto .txt, y son editados al principio del programa y al final. Al inicio se leen y se guardan en una lista, y al final se actualiza acorde con el desempeño del jugador. Se utilizaron conversiones de strings a enteros, al igual que el acomodo de listas.

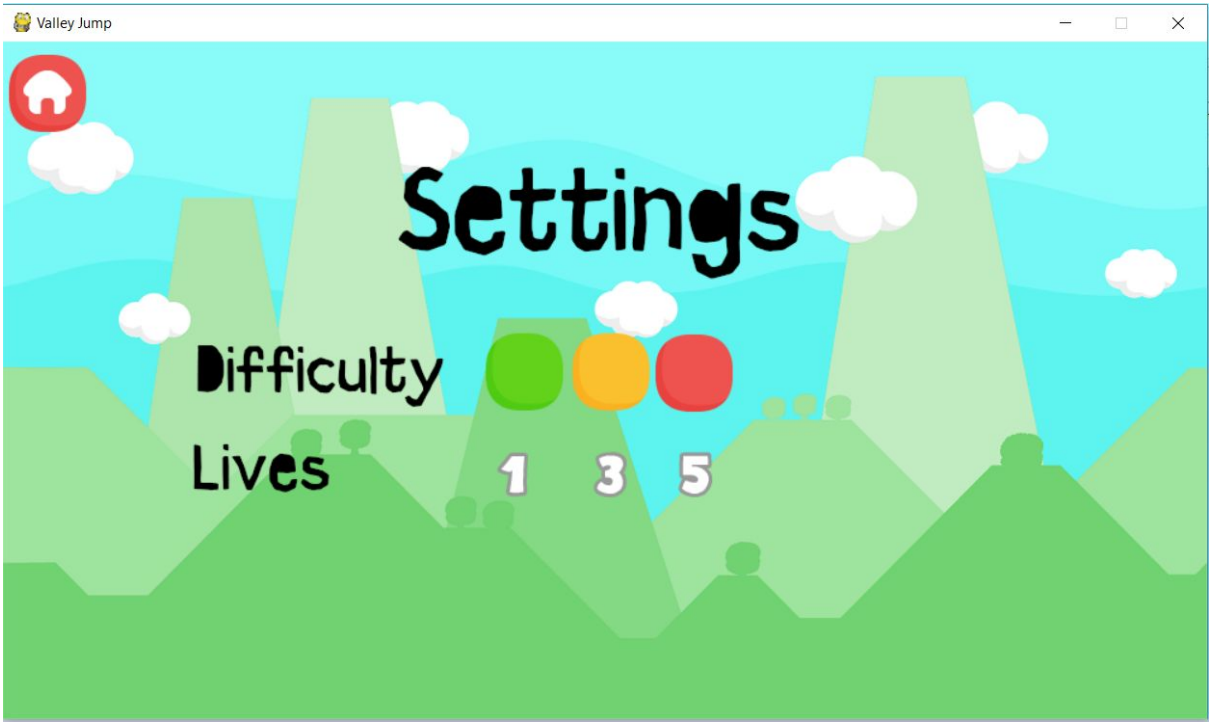
Análisis de Resultados

La introducción del juego consta de una imagen completamente negra, con el título del juego y el nombre del desarrollador, y el personaje principal caminando por la pantalla durante cinco segundos.

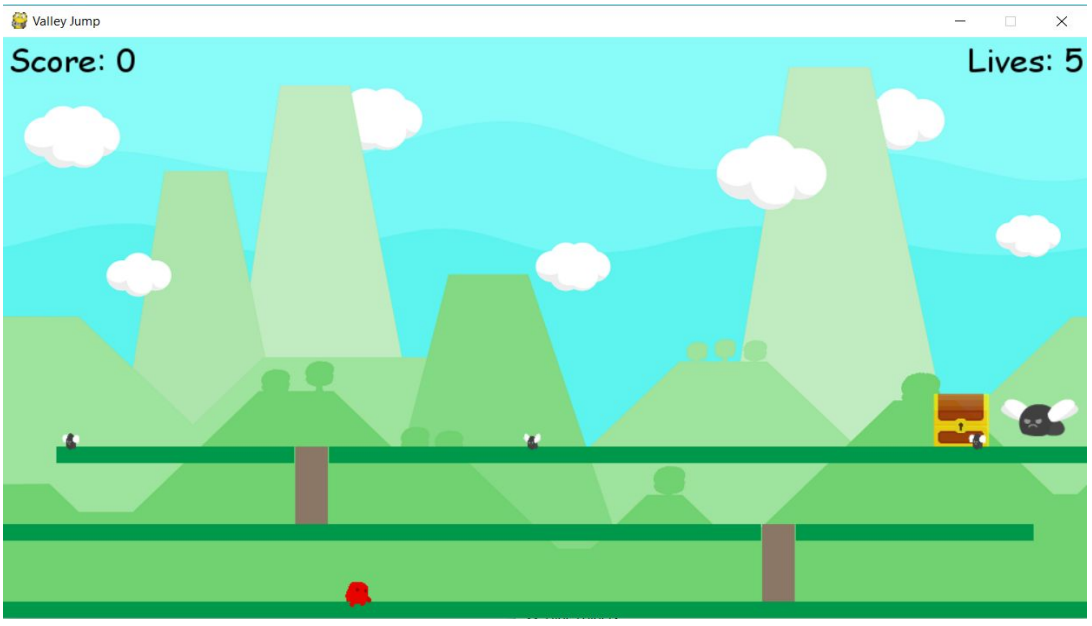
La pantalla del menú está constituida por una imagen de fondo, con el título del juego, y por cuatro botones: en la esquina superior derecha, el botón para ir a la pantalla Settings (opciones) y para salir del juego. Y en el centro de la ventana, el botón para jugar y un último botón que lleva hacia la tabla de puntuaciones.



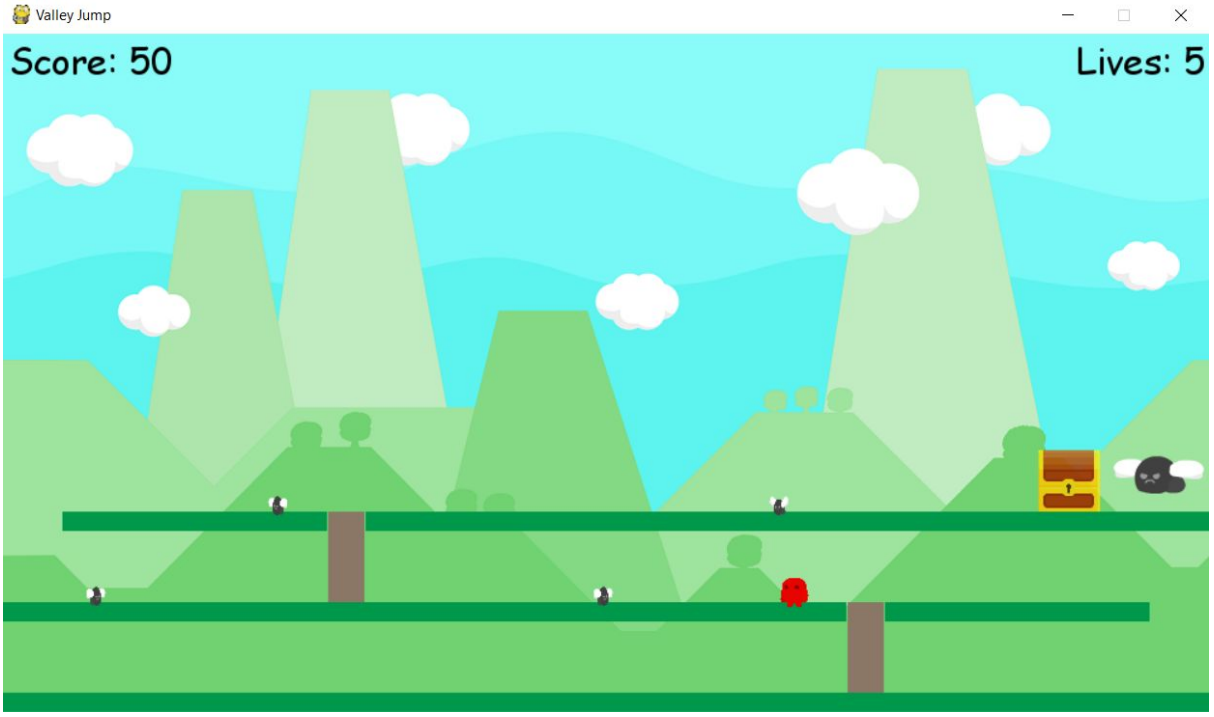
Al clickear el botón de opciones, la pantalla Settings se hace presente con seis botones, los primeros tres para elegir la dificultad del juego, siendo el verde la más fácil, el amarillo la intermedia y el rojo la más difícil. Luego, los botones “1”, “3” y “5” configuran la cantidad de vidas del personaje principal. Por último, un botón en la esquina superior izquierda para regresar al menú.



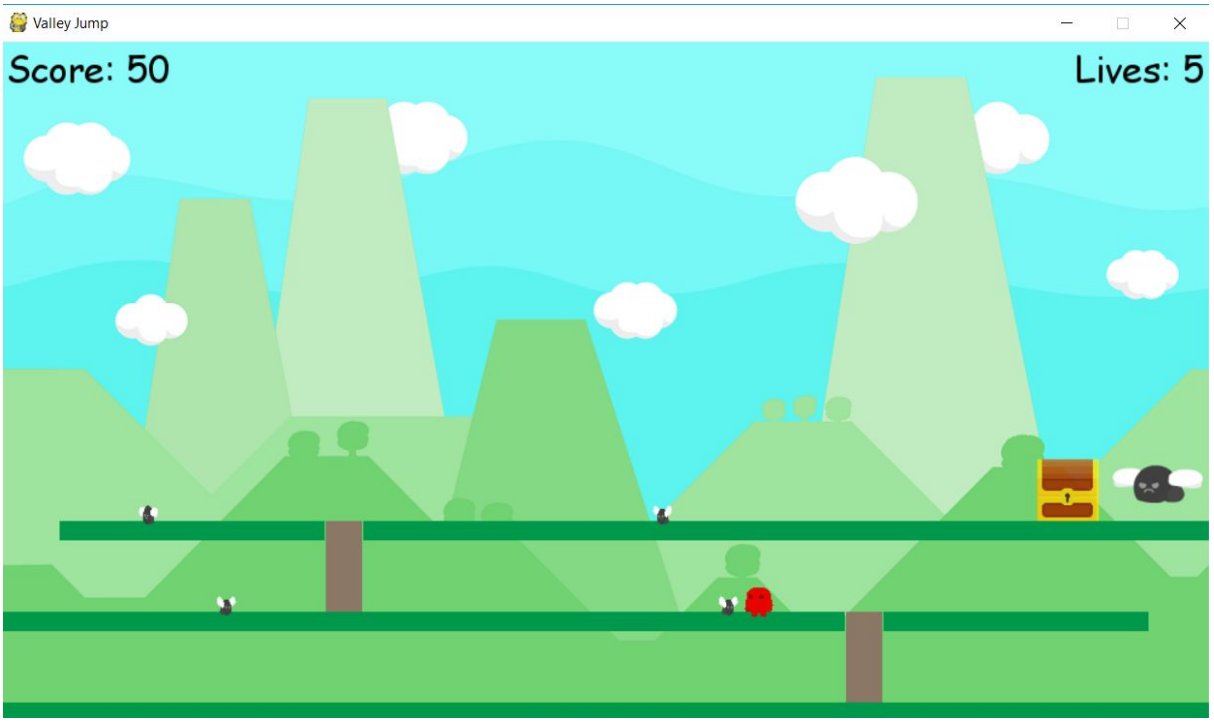
A continuación se muestra el primer nivel con 5 vidas, aunque esto es totalmente opcional y queda a gusto del usuario. El nivel consta de tres plataformas, que de hecho siempre están presentes en cualquier nivel, las que varían son las que se colocan aún más arriba en niveles avanzados. Se observa al personaje principal con su animación de caminar, en la parte más alta el cofre (que al ser tocado, se gana el juego) y un “boss”, desde donde se generan los enemigos (3 en pantalla).

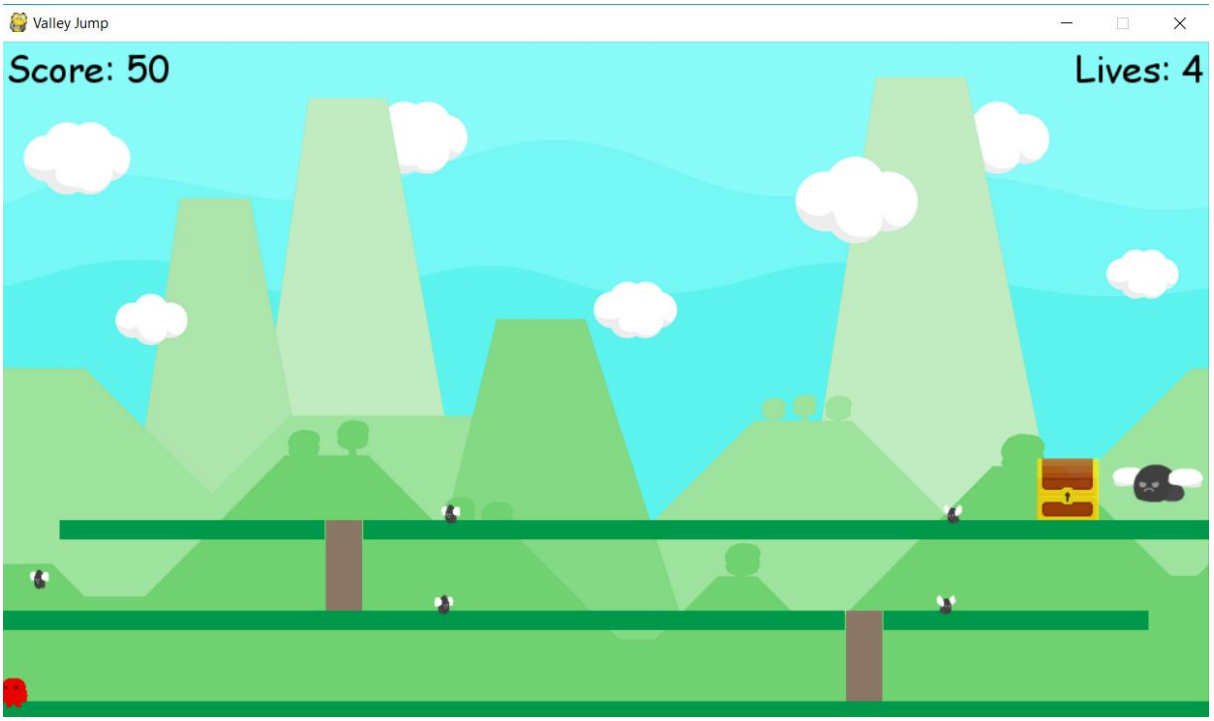


Al alcanzar una nueva plataforma, se suman 50 puntos al puntaje, como se observa a continuación. También cuentas las plataformas del otro lado de la escalera que el jugador suele dejar sin visitar.

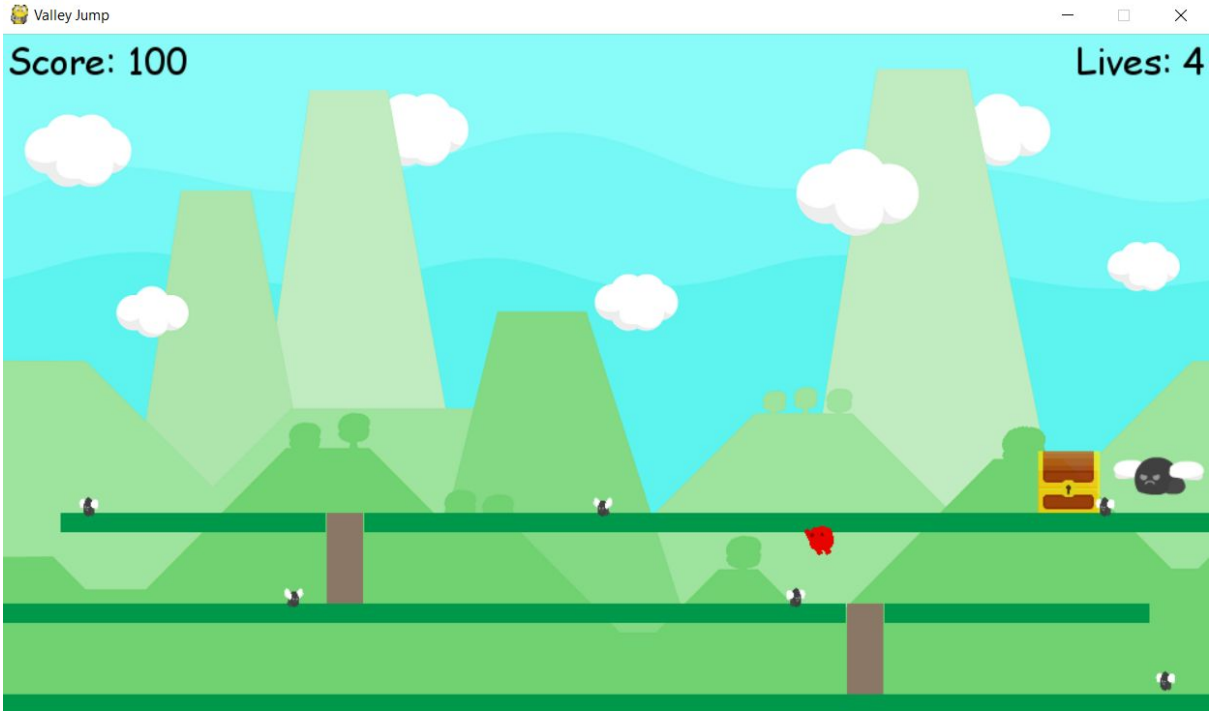


Al ser tocado por enemigo, el jugador pierde una vida y es devuelto a su punto de origen.

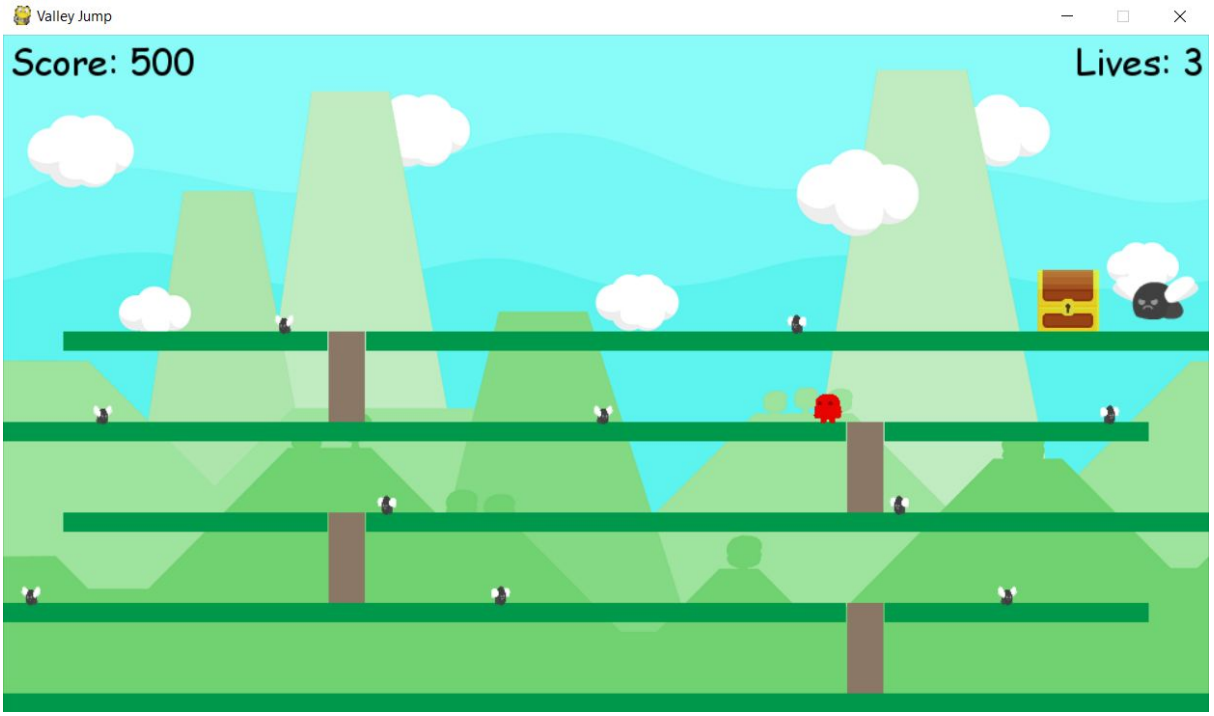


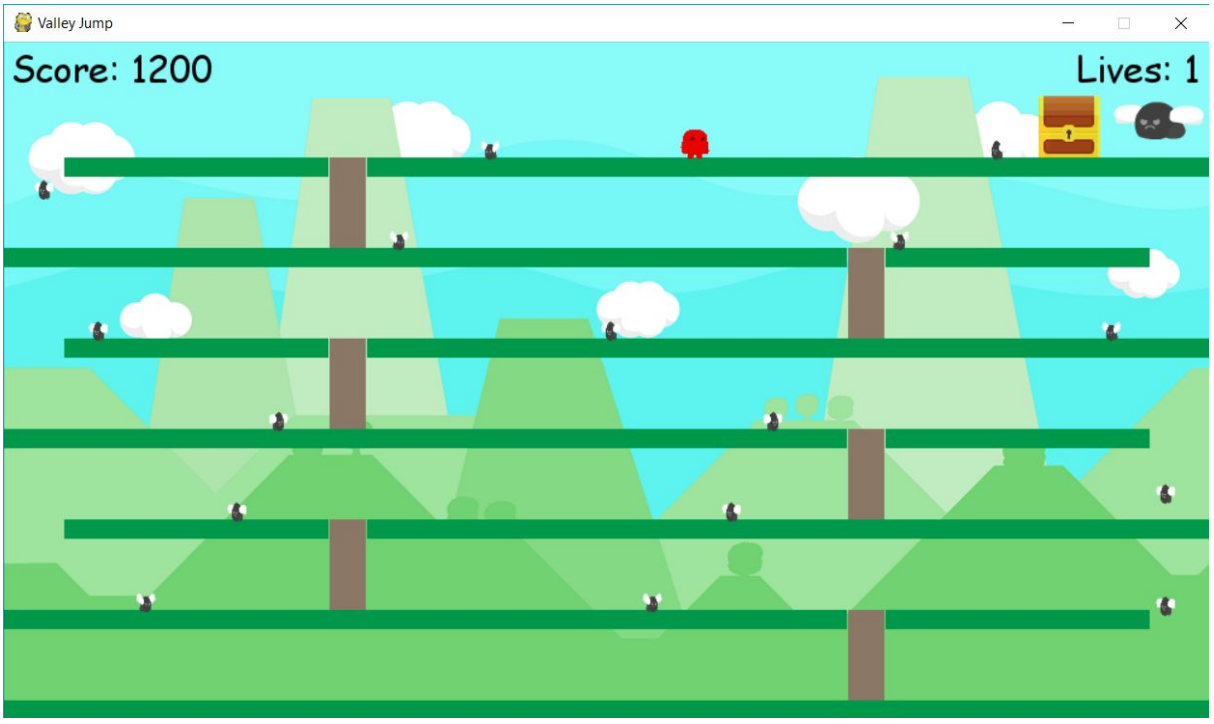


Al saltar un enemigo, se suman otros 50 puntos.

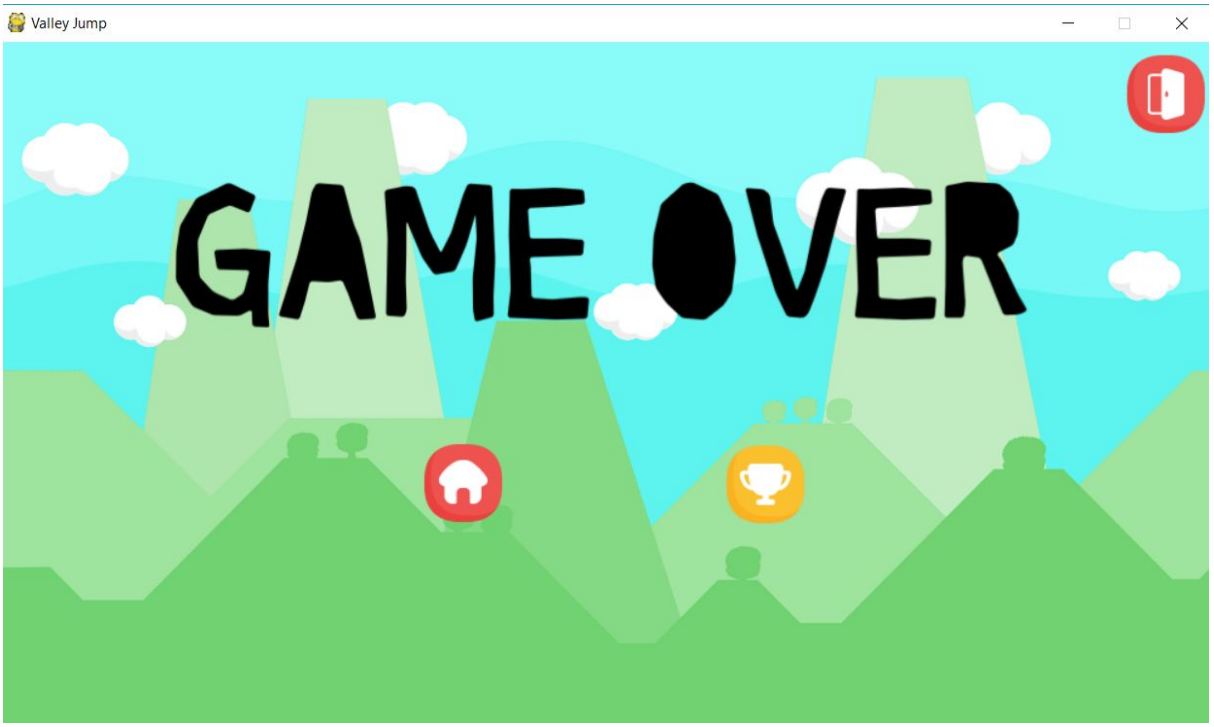


A continuación, respectivamente, se muestran el nivel intermedio y el nivel difícil.

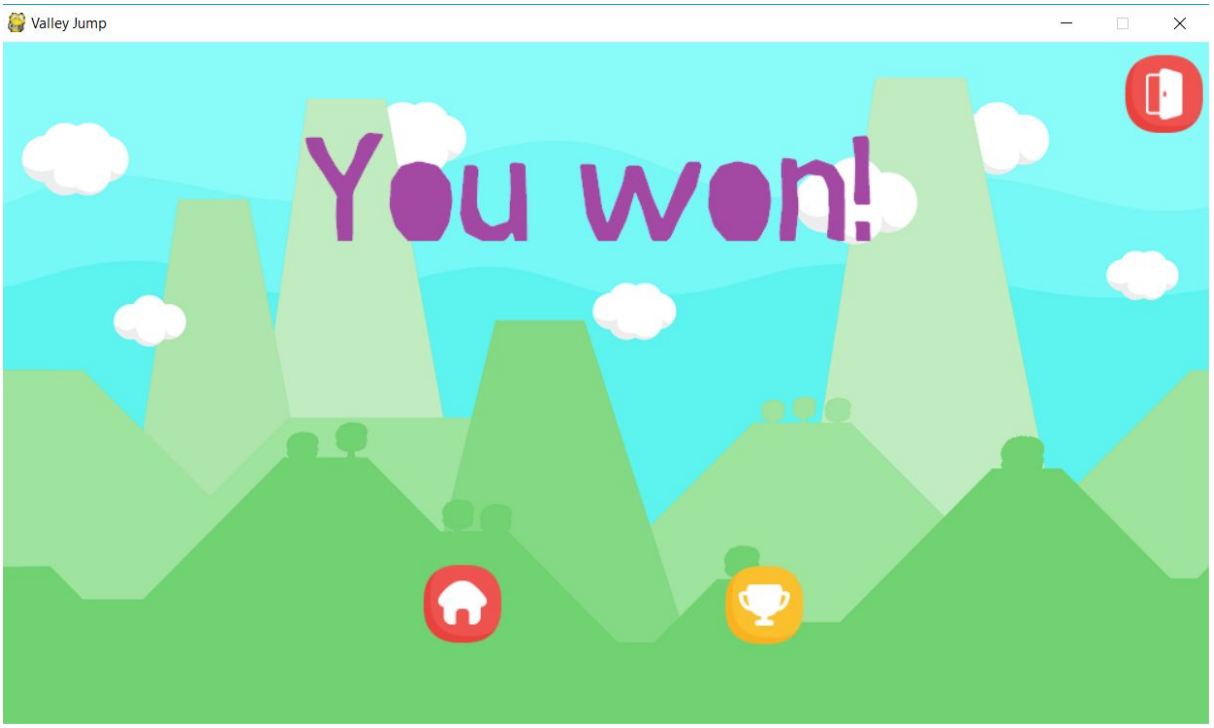




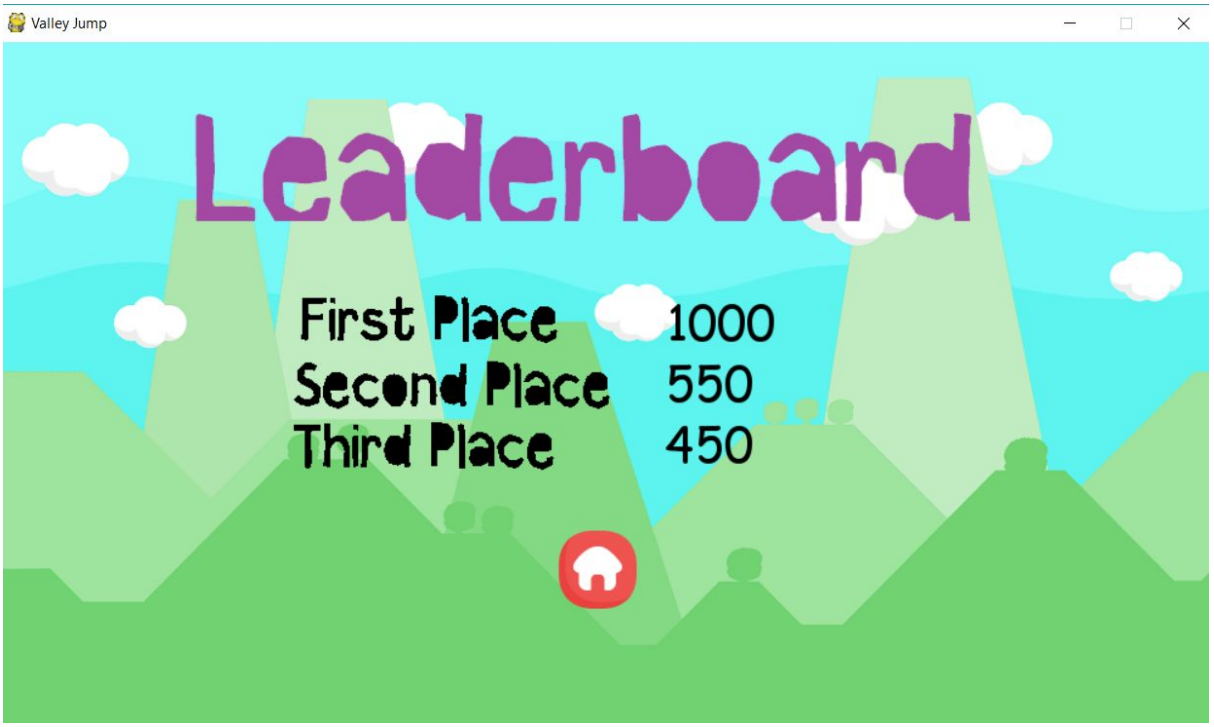
Ahora, se enseñará lo que sucede al perder todas la vidas. Se enseña la pantalla de Game Over (fin del juego), con botones para salir, ver las puntuaciones y regresar al menú.



Ahora, se muestra lo que sucede al tocar el cofre, es decir, ganar el juego. Una pantalla con los mismo botones que la pantalla de perder.



Por último, se enseñará la pantalla con las mejores tres puntuaciones de todo el juego. La puntuación de un jugador se guarda solo si gana el juego.
Cabe destacar que en todo el transcurso del juego, se reproduce una canción.


Valley Jump
Leaderboard
First Place 1000
Second Place 550
Third Place 450

Dificultades Encontradas

- Al crear el primer botón, para pasar de una pantalla a otra, se creó una clase Button, con sus argumentos siendo las instancias de las pantallas que se desean activar y desactivar. Por ejemplo, un botón para ir del menú al juego llevaba como argumentos a la instancia de la clase de menú y a la instancia de la clase juego. En la clase botón, lo que se hacía era acceder al atributo .flag de la pantalla que se quería desactivar y se le daba el valor False, de este misma manera con la otra .flag, pero con True. El problema surgió al crear un instancia de la clase botón en cierta pantalla, por ejemplo, la del menú. La clase botón debía de llevar como argumento a la misma pantalla donde se estaba trabajando, MenuScreen. Al ingresar esta misma clase como argumento, el botón no funcionaba. En cambio, al ingresar self como argumento, funcionaba. Este concepto está relacionado con la memoria y en dónde se guardan ciertos datos, y corresponde a cursos más avanzados.

- En el inicio del juego, el Player era de 32 por 32 píxeles al igual que las plataformas, lo que traía problemas ya que el salto del personaje debía ser lo suficientemente alto para esquivar enemigos, y de esta manera el personaje al saltar atravesaba la plataformas e incluso se quedaba pegado a la plataforma desde arriba. Se pudo arreglar el que el personaje ya no fuera atraído hacia la otra plataforma, pero para deshacerse del problema de atravesarla, se cambió el tamaño del jugador a uno más pequeño, y se disminuyó su salto.

- Al principio, el flag de cada pantalla era un atributo de su propia clase, lo que funcionaba bien, hasta que se crearon más botones y pantallas, causando problemas al referenciar variables. Por ejemplo, la clase MenuScreen necesitaba tener en sus argumentos a la clase SettingsScreen para crear un botón que dirigiera al usuario a la pantalla de opciones. En esta caso, se crea primero la instancia de la clase SettingsScreen y luego se crea MenuScreen para poder referenciar a la anterior. Pero a la hora de crear un botón para ir de SettingsScreen a MenuScreen, se necesita que MenuScreen ya esté definida antes, y todo esto causó un sinfín de problemas con las variables. La solución fue crear un diccionario con strings a los que se les asignó un valor de verdad, y las flags de cada pantalla estaban dadas por ese diccionario.

Bitácora de Actividades

➔ Sábado 23 de Marzo:
Aprendizaje básico sobre PyGame con tutoriales de Youtube.
Duración: 4 horas

➔ Domingo 24 de Marzo
Estudio del código de Víctor
Duración: 0.5 horas

Elaboración del modelo del juego
Duración: 1 hora

Se empezó el código, logrando crear las pantallas del menú y del juego, la clase de los botones y se creó el primer botón (ir al juego)
Duración: 6 horas

➔ Lunes 25 de Marzo
Se cambió el botón a una imagen en vez de un rectángulo con el uso de sprites, y se creó un nuevo botón (salir). Se empezaron las clases Player y Platform en el archivo del juego, logrando que funcionara la gravedad y la colisión con la plataforma del suelo, al igual que los controles de saltar y moverse de izquierda a derecha.
Duración: 2.5 horas

➔ Domingo 31 de Marzo
Se intentó incorporar el subir escaleras pero no funcionaba, al final del día se pudo arreglar.
Duración: 3 horas (tiempo actual de trabajo en código)

➔ Martes 02 de Abril
Se incorporó el bajar escaleras. Se creó el botón de opciones y de la tabla de puntajes. Se creó la intro del juego pero sin animación.
Duración: 3 horas

➔ Miércoles 03 de Abril
Se cambió toda la lógica de usar botones para ir de una pantalla a otra.
Duración: 2 horas

➔ Jueves 04 de Abril

Se creó la clase Enemy, con todos sus movimientos y colisiones respectivas. Se implementó la generación automática de los mismos. Se agregó el reinicio del juego al colisionar el jugador con un enemigo. Se crearon los botones en la pantalla Settings para cambiar el nivel de dificultad del juego.
Duración: 5.5 horas

➔ Viernes 05 de Abril

Se creó el algoritmo para generar automáticamente el segundo y tercer nivel del juego.
Duración: 3 horas

➔ Sábado 06 de Abril

Se implementaron las vidas del jugador, y se crearon los botones en la pantalla Settings para configurar las mismas. Se creó el sistema de puntaje del juego. Se creó el método para resetear el juego al querer jugar de nuevo. Se cambió el jugador por sus respectivas sprites.
Duración: 6 horas

➔ Domingo 07 de Abril

Se agregaron los sprites de los enemigos y del boss. Se incorporó el cofre (lo que el jugador alcanza para ganar). Se creó la lista de puntajes con los archivos de texto.
Duración: 5 horas

Pruebas del juego:
0.5 horas

Redacción de la documentación externa:
2 horas

Documentación interna:
0.5 horas

Estadística de Tiempos

Función	**Horas**
Requerimientos/Diseño	1
Investigación de Funciones/Tutoriales/Aprendizaje	4.5
Programación	36
Documentación Interna	0.5
Pruebas	0.5
Elaboración del documento	2
Total	44.5

Conclusión Personal

La elaboración de este proyecto fue de gran provecho para mi crecimiento personal. Fue una pincelada de lo que va a ser el resto de la carrera, aprender sobre herramientas o lenguajes totalmente nuevos por nuestra propia cuenta, arreglar problemas y pasar horas en frente de una computadora. Pero más que todo, funciona extremadamente bien para darse cuenta de si uno de verdad quiere esto, y creo que por el momento, es afirmativo para mí.

El juego trajo mucho estrés, bastantes veces de pedirle ayuda a otras personas y varias horas en las que no se puede continuar debido a un error en el código. Y definitivamente me afectó emocionalmente.

Ahora manejo mejor el tiempo y aprendí a conllevar la responsabilidad de un proyecto importante y con un límite de tiempo-