UTN AVELLANEDA
Tecnicatura Universitaria en Programación

Trabajo Práctico Integrador
Laboratorio I

Tu objetivo es crear un juego de plataformas en Python utilizando la biblioteca Pygame. El juego deberá tener tres niveles con un personaje principal, enemigos, proyectiles y un jefe final.

Objetos del juego (Las imagenes son ilustrativas)

Jugador: Es el personaje del juego controlado por el usuario. Tiene que acumular tantos puntos como sea posible destruyendo los objetos del juego **Enemigo** mientras evita su contacto. También puede acumular puntos mediante **objetos especiales** del juego.
Puede moverse hacia la derecha, hacia la izquierda, saltar y disparar objetos del juego **Proyectiles** sobre los enemigos. Los puntos y las vidas se almacenan en propiedades personalizadas del jugador.

Como plus se valorará que el personaje pueda tener un ataque cuerpo a cuerpo con el/los enemigos.

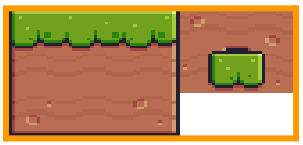
Enemigos: El personaje antagonista que cae en escena y elige aleatoriamente una dirección de movimiento. Rebota al chocar con los límites horizontales de la escena. Puede dañar al jugador y puede destruirse con objetos del juego **Proyectiles**.

Nota: Hay enemigos que pueden estar en escena.

Boss: El nivel final del juego deberá contar con un Boss. Con una lógica de ataque totalmente distinta a la de los anteriores.

Terreno: Objeto estático que configura las zonas transitables del jugador y del enemigo.

Plataformas: Objeto estático que configura las zonas donde el jugador y el enemigo se pueden parar.



En caso de poder programar plataformas con movimiento son bienvenidas al proyecto :D

Proyectil: generada por el jugador para destruir enemigos y por el enemigo para destruir al jugador. Se mueve en la dirección de quien la lanza hasta que alcanza un objetivo o los límites de la escena, en ambos casos es destruido.

Vidas: Inicia con tres vidas y pierde una de estas con cada proyectil recibido o con cada colisión con una trampa o enemigo. La recuperación de la salud del jugador solo es posible si encuentra **items especiales** a lo largo del nivel.

Puntuación: Se debe proveer visualización de puntuación, Dicha puntuación se incrementa por cada enemigo destruido, por cada elemento **Fruta** recolectado y al finalizar el nivel se multiplicará el tiempo restante por 100 y se sumará al score .

Cronómetro: Controlador de tiempo para los sesenta segundos disponibles para cada nivel. Cuando llega a cero, el juego termina.

Ítems: objeto auxiliar que forma parte de cada nivel a efectos de ser recolectado por el Jugador

Items especiales: objeto especial que permite otorgarle una nueva vida al Jugador. También se puede aplicar a la regeneración de proyectiles en caso de tener cierto límite.

Trampas: objeto auxiliar que forma parte de cada nivel a efectos de dificultar el libre movimiento del Jugador, al entrar en contacto con este le resta una vida.

Generador de enemigos: objeto auxiliar que suelta aleatoriamente objetos enemigos desde la parte superior de la pantalla.

Niveles: Objeto auxiliar encargado de agrupar a todos los elementos antes descritos a efectos de conformar un nivel. Dicha información deberá estar soportada en un archivo JSON. Esto permitirá guardar/recuperar una partida en ejecución.
los niveles tiene que tener distintos niveles (valga la redundancia) de dificultad, el primer y segundo nivel pueden ser similares en cuanto a su objetivo (ej: destruir a los enemigos y recolectar los items que se encuentren en el mapa) pero el tercer nivel tendría que haber una pelea con un jefe final, que este mismo tenga otros comportamientos destinos a los enemigos normales y otros “ataques” para que se diferencien de los enemigos clásicos de los niveles anteriores. Para terminar con este nivel deberá de destruir al jefe final y de ser el caso también recolectar los items (frutas) del nivel -> (este nivel podrá tener más tiempo que los anteriores)

Reglas asociadas a los movimientos del Jugador y del Enemigo.

El **Jugador** está restringido a cambios de izquierda a derecha y a saltos. En este caso, los saltos están limitados por una condición de contacto con el suelo o plataforma.

Los movimientos del **Enemigo** son autónomos y están determinados por dos reglas de comportamiento que dependen de los eventos de colisión. El primero ocurre con su primer contacto con el suelo, cuando seleccionará aleatoriamente una dirección de movimiento (izquierda o derecha). El segundo se activa si alcanza los límites horizontales de la escena (ya sea por colisionar con un elemento o por alcanzar los límites de la pantalla), momento en el que cambiará de dirección.

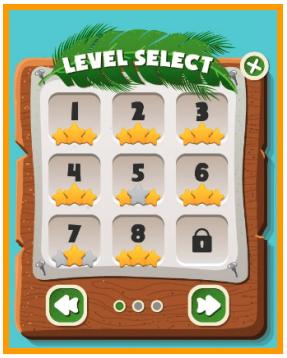
Audio

Efectos de sonido: el juego deberá contar con efectos de sonidos que acompañen las acciones del **Jugador** y del **Enemigo** (por ejemplo al saltar, colisionar con las balas, recolectar una fruta, etc.)

Música: también se deberá agregar música ambiental que se reproduzca durante el gameplay

Pantallas del juego

Selección de nivel: El juego deberá contar con una pantalla que permite visualizar por lo menos tres niveles distintos, cada nivel deberá proponer un nuevo mapa con distintos enemigos. Solo se podrá acceder a un nivel superior, si el anterior fue superado con por lo menos una estrella.



Configuraciones del Juego: En todo momento el juego deberá permitir encender y apagar, subir o bajar el volumen tanto los efectos de sonido como la música de fondo.

Idea: podrían agregar una opción dentro de ese mismo menú que nos lleve a un form y este nos muestre los controles con los que podremos interactuar en el juego

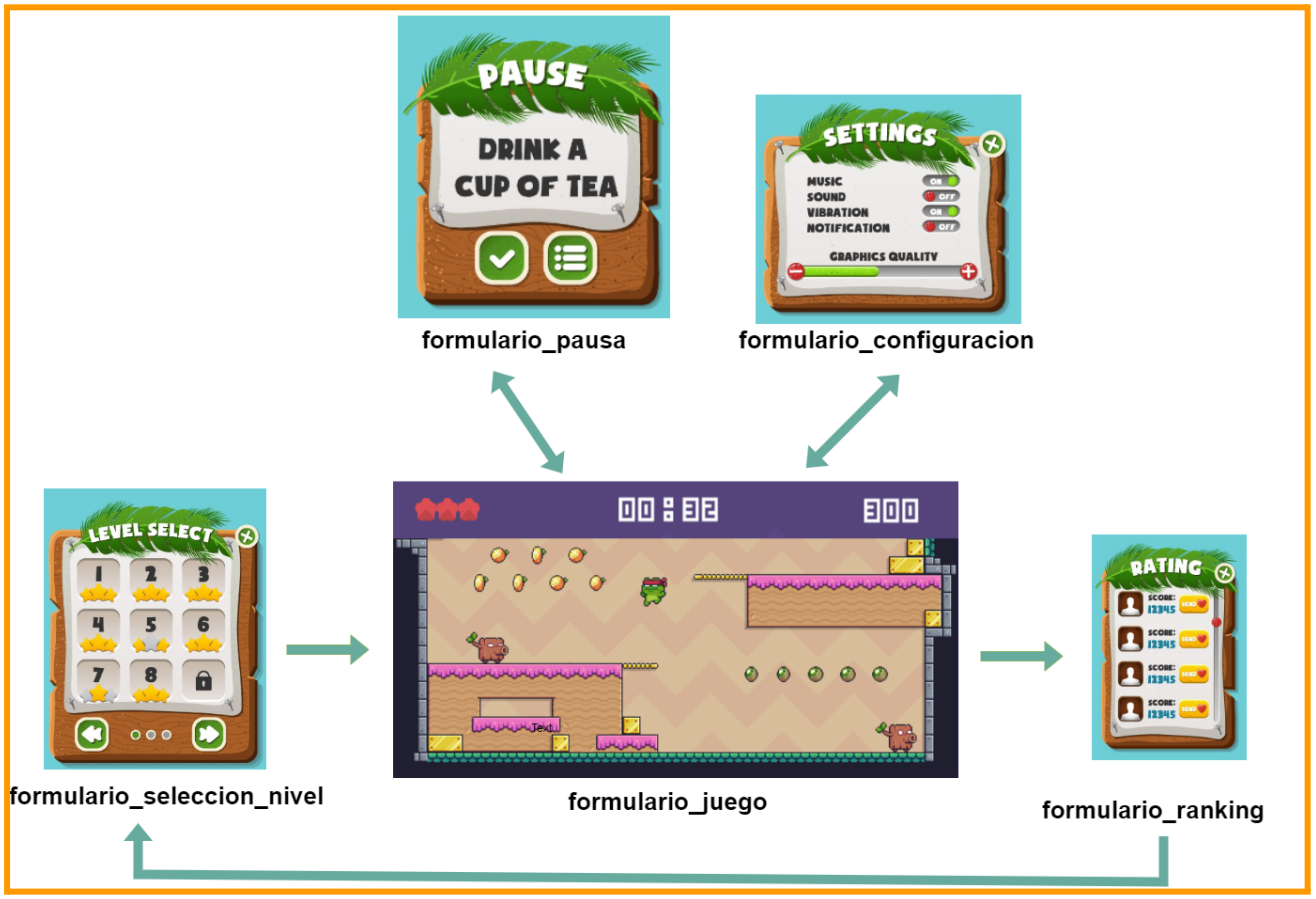
Pausar Juego: En todo momento el juego deberá permitir la posibilidad de ser pausado, para lo cual deberá contar con una pantalla que represente dicho estado.

Pantalla Principal: La pantalla deberá contener no solo el escenario, los enemigos, las recompensas, las trampas y al jugador, sino también la información de vida, tiempo y puntaje del juego.

Pantalla de ranking: Al finalizar cada juego si el puntaje realizado por el jugador así lo amerita se deberá registrar a este en el ranking. Dicha información deberá estar soportada en una base de datos Sqlite.

UTN AVELLANEDA
Tecnicatura Universitaria en Programación

El flujo de navegación entre los formularios debería ser de la siguiente manera:



UTN AVELLANEDA
Tecnicatura Universitaria en Programación

Posibles dinámicas de juego: A continuación se presentan posibles dinámicas que podrá tener el juego con los recursos disponibles. La dinámica es a elección/imaginación del estudiante. Los siguientes son solo ejemplos.

UTN
AVELLANEDA
Tecnicatura Universitaria
en Programación