PROYECTO “BAD DICE”

Santiago Melo Aguirre

José Nicolás Ríos Pinto

Sara Gabriela Salazar Carvajal

REPOSITORIO: https://github.com/santiagomeloa/Proyecto-semestre-1.git

Resumen ejecutivo:

En el ámbito de la docencia frecuentemente se dificulta llamar la atención de los estudiantes, por lo cual se recurre a ideas didácticas que permitan comprender los temas de una forma fácil y divertida. Este desinterés se puede evidenciar en múltiples áreas del conocimiento, sin embargo, en el área de las matemáticas es más difícil realizar actividades del interés del estudiante. Para muchos estudiantes la idea de resolver múltiples ejercicios matemáticos suena poco agradable, y esto puede llevar a que terminen por no entender los temas, pues con la práctica se pulen las habilidades y el razonamiento matemático. Para solucionar este inconveniente, se creó utilizando el lenguaje de programación Python un videojuego del estilo RPG, con la finalidad de entretener a los estudiantes mientras aprenden como resolver ecuaciones básicas.

Funcionalidad de la solución computacional:

El programa cuenta con una pantalla de inicio, un escenario donde se localizan el personaje jugador y los enemigos, y un escenario donde el jugador es enviado luego de cumplir los requerimientos para enfrentarse al jefe final.
El juego en cuestión trata de derrotar una cierta cantidad de enemigos para llegar a detener un ser malvado. Luego de cada batalla, el jugador deberá ingresar el resultado de una ecuación que se mostrará en pantalla y, de ser correcta, será recompensado con un bonus a su ataque, de lo contrario será penalizado con una reducción del mismo. Luego de que todas las batallas han acabado, el escenario muestra una abertura, la cual al acercarse el jugador le transporta a la arena del jefe final. Si ha respondido las ecuaciones sin equivocarse, el jefe morirá de forma sencilla. Si no, el jefe acabará con el personaje jugador sin darle la posibilidad de luchar. En caso de haber realizado algunas de las ecuaciones se desarrolla la batalla normalmente.

Objetivos Específicos:

- Crear sprites para ser utilizados dentro del juego:
  Se cumplió completamente el objetivo, adicional a los sprites de personaje principal y enemigos, se diseñaron los escenarios y la música para el juego.
- Diseñar el entorno virtual del juego:
  Se cumplió completamente, se diseñó una pantalla de inicio, los botones y los escenarios para el juego.
- Desarrollar el funcionamiento de objetos y personajes:

Se cumplió completamente, los personajes tienen movimiento propio con una I.A bastante básica, que les permite moverse por la pantalla. Adicional a esto, cuentan con animaciones para el movimiento.

- Implementar un diccionario que permita el uso de diferentes ecuaciones en cada nivel:

  Originalmente se pensaba hacer un sistema de niveles, sin embargo, se decidió reducirlo a una ecuación después de cada batalla. Dejando de lado esto, se cumplió completamente el objetivo con la forma de mostrar las ecuaciones, ingresar los resultados y comparar con la solución.
- Implementar el sistema de batallas:

  Se cumplió completamente, el sistema de batallas se diseñó para presentar un obstáculo al jugador sin ser injusto, el daño hecho por los enemigos y la vida del personaje están balanceados, se tienen un total de 4 elementos curativos para usarse luego de cada batalla.

Descripción del código:

El programa está compuesto hasta el momento de los siguientes archivos:

· **Main.py**: En este archivo está el bucle principal del juego y la creación de Sprites junto con los distintos escenarios

·**Funtions.py**: El archivo de funciones sirve como su nombre lo indica para guardar todas las funciones que serán usadas durante la ejecución del juego

·**Sprites.py**: El archivo de Sprites tiene como función guardar todos los sprites u objetos que se requieren para luego ser llamados e instanciados desde el archivo main

·**Stages.py** El archivo de Stages tiene como función guardar todos los escenarios virtuales y llamado de funciones que se requieren para correr el juego.

El archivo que contiene el bloque de ejecución para poder correr el programa es el archivo llamado ‘main.py’, en él está toda la lógica de ejecución.

Las clases y funciones definidas en cada archivo son las siguientes:

·**Main.py**: En este archivo únicamente se define la función “main”, donde se ejecuta todo el bucle del juego, se muestra la pantalla de inicio y se llama las funciones del archivo “stages”.

·**Funtions.py**: Es el archivo que contiene las siguientes funciones:

- **Music:** se encarga de utilizar una función de pygame para reproducir música de fondo
- **Screen_size**: Lo que hace está función es determinar el tamaño de la pantalla en la que se está ejecutando el programa y así definir el tamaño de la pantalla.
- **Load_image**: Está función recibe como parámetro de entrada un archivo de imagen y lo procesa con ayuda de pygame para poder ser usado por esta misma. Además de esto, también es capaz de definir el tamaño de la imagen si se gusta, además de quitar el fondo de esta.

- **Move:** Esta función lo único que permite es el movimiento del personaje principal, lo cual es muy importante durante todo el juego.
- **Damage:** Esta función se usa para calcular el daño durante la batalla.
- **SpecialDamage:** Esta función se usa para el ataque especial del personaje jugador.
- **use_potion:** Lo que hace esta función es aumentar la vida del personaje jugador siempre y cuando posea "pociones".
- **Movement:** Esta función permite elegir el movimiento de los enemigos en la pantalla de forma random. Es una Inteligencia Artificial primitiva.

Sprites.py: En este archivo se encuentran todas las clases que se usarán a lo largo del programa. Todas las clases heredan de la superclase pygame.sprite.Sprite. Las clases aquí definidas son las siguientes.

- **Player**: Esta es la clase del personaje principal.
- **Bicho**: Esta es la clase de donde se derivan todos los enemigos básicos del juego.
- **Words:** Esta clase se usa de modelo para todos los textos que se mostrarán en pantalla.
- **Buttons:** Como su nombre lo indica, se usa para crear los botones implementados en el juego.
- **Stars:** Esta clase se usa en la pantalla de ingreso de las ecuaciones, para realizar el fondo con movimiento.
- **Boss:** Es la clase del jefe final del juego.
- **Points:** Esta clase se usa para los detalles de fondo de una pantalla del juego.

Stages.py: En este archivo se encuentran las funciones especiales para cada instancia del juego. Dentro de este archivo se encuentran las siguientes funciones:

- **Main_stage**: En esta función se crea el escenario inicial del juego. Inicializa el fondo, los 5 enemigos aleatorios y sus posiciones en la pantalla, así como la posición del personaje jugador. Dentro de esta se llama a la función Battle.
- **Equation**: En esta función se implementa la interfaz que permite el ingreso de las soluciones para las ecuaciones, así como la impresión en pantalla de la ecuación a resolver. Una vez se ingresa el valor, la función evalúa si es correcto o no y continúa el programa.
- **Battle**: Esta función se encarga de colocar al personaje principal y enemigo en sus posiciones de batalla, asi como crear los botones usados para luchar y la impresión en pantalla de la información relevante para la batalla.
- **Final_battle**: Se utiliza para la batalla contra el jefe final. Se realiza igual que con las batallas contra enemigos básicos, sin embargo, si el usuario acertó en todas las ecuaciones se le da una mejora de ataque. Si falló en uno o más se desarrolla la batalla normal y si no acertó ninguna la batalla se vuelve imposible de vencer.
- **End_battle**: Esta función se encarga de la creación del escenario final, usado para la activación de la batalla contra el jefe.
- **Dead**: Esta función se activa cuando el jugador pierde todos sus puntos de vida, muestra la pantalla de game over.
- **Credits**: Esta función muestra los créditos del juego.

El flujo del programa es el siguiente.

Lo primero que se hace es definir la constante “FPS”, que lo que hace es guardar las veces que la pantalla se va a actualizar por segundo, después se define la función principal llamada “main”, y en esta se encuentra toda la lógica del juego. Lo primero dentro de la función main es definir la música de fondo usando la función music, luego se hace la ventana del juego, esta variable se llama “screen”, después se define la variable “clock”, la cual ayuda a que la ventana se pueda actualizar, luego se pone un subtítulo en la ventana con el nombre del juego. Aquí es donde se empiezan a crear los Sprites, pero primero se crea un grupo de sprites que contendrá los objetos de la clase botón. Luego se crea el Sprite del botón ‘play’ y se importa el fondo de la pantalla de inicio. Una vez hecho esto se inicia el bucle del juego, que vendría siendo el ciclo principal formado por un while, donde primero se usa la variable clock mencionada anteriormente, y se lo pasa como parámetro la variable “FPS” y se dibuja el entorno virtual. Luego se crea un ciclo for con los eventos que estén sucediendo, esto para detectar si hay algún evento de tipo “QUIT” para terminar el juego. Adicionalmente, evalúa si el botón creado anteriormente es presionado y, cuando lo es, llamar la función main_stage desde el módulo “stages”. Main_stage genera el escenario principal del juego de manera similar a la primera pantalla. En primer lugar, inicializa la música, luego crea la ventana donde se desarrollará el juego y los grupos necesarios para el funcionamiento del escenario. Coloca la imagen de fondo y la imagen de batalla. Después de ello utilizando un for se crean los enemigos de la clase Bicho y los añade al grupo enemy_group. Luego de ello, se crea un Sprite para mostrar la vida del jugador usando la clase words. Se añade el jugador al grupo de todos los sprites y se oculta el mouse. Luego se hace un bucle infinito donde se realizarán las actualizaciones de pantalla y se realiza la función de batalla cuando el jugador colisiona con un enemigo. La función de batalla tiene las mismas bases de los demás escenarios, la creación de los botones que permiten pelear y la modificación de los valores de vida del jugador y el enemigo. Luego de que se desarrolla la función main_stage se actualiza el botón ‘play’ y se genera un segundo if para revisar si los enemigos fueron derrotados y, en caso de ser cierto, llama la función “transicion”, la cual a su vez lleva a la función “end_scenario” donde el jugador se enfrenta al jefe final. Luego de derrotarlo se llama a la función “credits” para mostrar los créditos finales del juego. Al final del ciclo while que contiene todos los procesos anteriores, está el método que permite actualizar la ventana. Luego, y ya, por último, está la condición if que evalúa si el nombre es igual a “main” le dé comienzo a la función main.