**Programación Orientada a Objetos**

**Práctica 2**

**Grupo 6**

**Brayan David Caballero Fernández**

**John Mauricio Mesa Echeverri**

**Eider Alejandro Peña Dagua**

**Santiago Rivera Mejía**

**CALABOZOS EN LA NACHO**

**Introducción**

Como primera medida se trata de transcribir el programa en JAVA de la práctica uno a Python tal cual estaba, lo cual conlleva a múltiples errores y a un código que no se ejecutaba completamente con éxito y que en cierto momento sentimos era intratable, por lo que se procede a programar nuevamente el juego casi desde cero, rescatando piezas del código anterior. En esta ocasión, a diferencia de la práctica uno, las mecánicas del juego no estuvieron centradas en una misma clase, y que así cada uno de los desarrolladores tuviera más independencia a la hora de programar, lo que nos trajo ventajas como también ciertos problemas, en especial a la hora de importar el código desarrollado por otros compañeros. Con mucho esfuerzo se logra rehacer la mayor parte del juego que se tenía previamente y se agregaron nuevas características.

La idea de la Práctica 2 se centraba en el desarrollo de entornos gráficos, haciendo uso de la librería TKinter. Se implementó, según las directivas de la práctica, una ventana de bienvenida que precede al juego como tal, llevándonos primero a un entorno de creación de personaje y luego a la ventana principal, donde contamos con un entorno gráfico para jugar al juego de texto, donde en reemplazo de menús de texto para seleccionar opciones contamos con sus respectivos botones, cajas de texto para la narrativa y frames para los respectivos combates que se han de desarrollar en diferentes momentos.

**Detalle y construcción:**

En el desarrollo de la interfaz crearpersonaje se implementaron cuatro widgets tipo Frames, uno para el titulo, y tres para los botones pertenecientes a las clases existentes en el juego, dispuestos en proporción al tamaño de la ventana usando place y sus atributos relx, rely, relwidth y relheigth, que controlan las dimensiones y posiciones relativas de dichos widgets, esto con el fin de que se mantenga cierto orden al maximizar o reducir el tamaño de la ventana. Los botones contienen imágenes representativas de cada clase, que, al oprimirlos abren una nueva ventana de tipo Toplevel() en la que se disponen los widgets tipo Entry para recibir los datos del personaje del usuario. Finalmente el botón continuar recoge los datos ingresados en los Entry y ejecuta la función crear() que utiliza dichos datos como parámetros del constructor en la clase Player.

Para la transición de la clase ‘clase’ de java a python se crearon <<constantes>> que representan las características propias de todas las clases en el juego, el constructor depende únicamente de un parámetro tipo int, luego, dependiendo de este parámetro, se ejecuta uno de los tres métodos que construyen la clase con sus respectivos valores por defecto, reemplazando la clase enum usada en java para tal propósito.

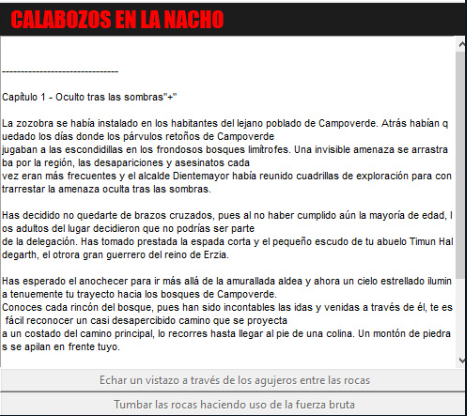
La clase NPC se conforma principalmente por dos metodos, el constructor que recibe como parámetros el nombre, la edad, la clase y el nivel, siendo estos tres últimos de tipo int. En este constructor se crea un objeto de tipo ‘Clase´ con el parámetro clase recibido del constructor, posteriormente se llama al segundo método de la clase que establece los atributos del personaje en base a su nivel y su clase.

En la clase Player se implementó un método que hace la función de sobrecarga del constructor, así, luego de instanciar un objeto de la clase Player, cuyo constructor instancia un objeto con los atributos por defecto, se llama al método Jugador, el cual recibe los parámetros nombre, edad y clase para luego calcular los valores de sus atributos.

**Procesos:**

* **Inventario:** el jugador puede modificar las armaduras, armas actuales, seleccionar objetos.
* **Narración:** Cumple la función que cumple un jugador normalmente en el juego de rol, al cual se le encarga guiar al resto de jugadores a través de distintos escenarios mediante la narración de situaciones. En este caso solo guiara a un jugador. Este también es el encargado de ejecutar las reglas de juego, como por ejemplo el desarrollo de un combate.

**Interfaz de narrador**



* **Combate:** enfrentamiento entre el jugador y un enemigo, esta funcionalidad nos permite realizar subfunciones como atacar, ver posición o rendirse.

El combate es gestionado dentro de la clase Narrador, cuando Narrador determina que habrá un combate, determina quienes participarán de dicha batalla (siempre será el personaje del jugador en contra de un rival controlado por el programa), el combate se da por turnos, se compara el atributo DES (destreza) de ambos luchadores , y aquel personaje cuya DES sea mayor actuará primero, durante el combate el jugador dispone de comandos especiales de batalla, como “atacar”, “pocion” y “escapar”. Cuando hay un ataque de cualquiera de los dos contendientes se determina si este ataque es exitoso o no lanzando un dado de 20 caras (función lanzarDados) y si el resultado es menor o igual al número obtenido mediante la siguiente fórmula, el ataque es exitoso y se procede a calcular el daño.

objetivo = 10 + nivel del atacante + clase de armadura del atacado - armadura del atacado

El daño se resta del atributo HP del atacado y es un valor correspondiente al atributo “dano” del arma usada por el atacante.

Cuando los “HP” (Hit points - puntos de vida) de alguno de los contendientes llega a 0, se da por terminado el combate y resulta victorioso quien conserve aún puntos de vida. Si el contendiente victorioso es el jugador, se le otorgan puntos de experiencia que le permitirán subir de nivel y mejorar sus atributos, también se le conceden monedas de oro con las cuales podrá adquirir en la tienda nuevas y mejores armas y armaduras. Estas recompensas se dan en relación al nivel del enemigo derrotado.

Se determinan los involucrados en el combate. Siempre sería el PJ vs una herencia de NPC.

Se decide quien tiene el primer turno sigue quien tenga un puntaje de destreza mayor entre los contendientes

En el turno del jugador, este podría: Atacar con arma principal, atacar con hechizos (habilidad especial), usar objeto del inventario.

En el turno del NPC este atacaría y usaría una habilidad especial cada x turnos

Cuando se ataca se determina si el golpe es exitoso o no de la siguiente manera: objetivo = 10 + nivel del atacante + clase de armadura del atacado - armadura del atacado, luego se lanza 1 dado de 20 caras,

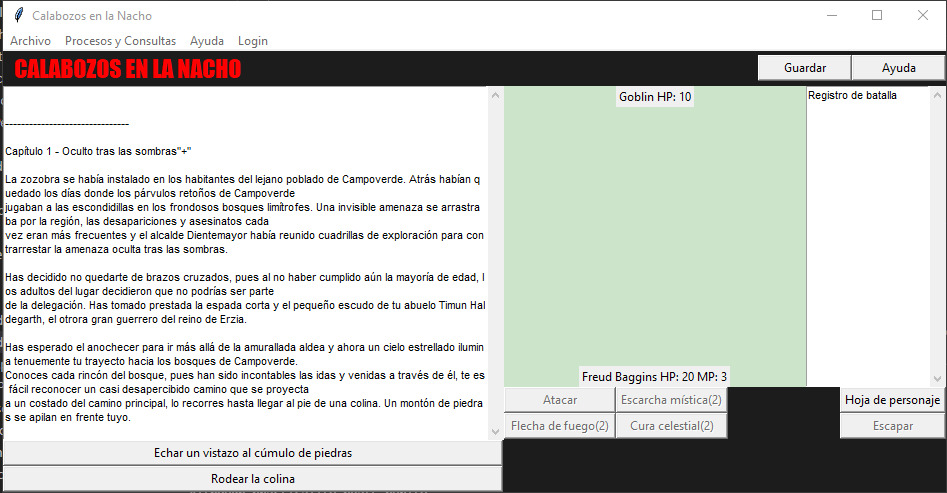
- si el resultado es menor o igual al número obtenido de la sumatoria anterior, el golpe es exitoso y se procede a determinar el daño realizado.

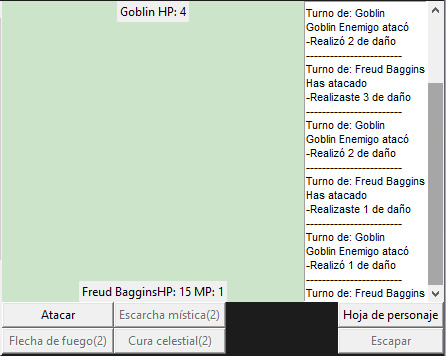
- El daño realizado es igual al daño del arma

- Luego de calcular el daño realizado se termina el turno y comienza el

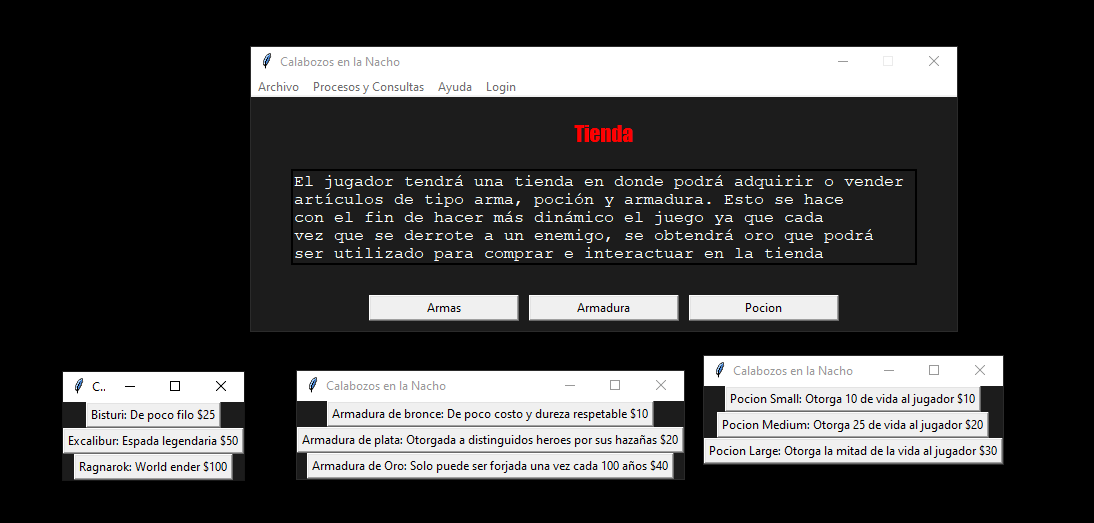
- El combate termina cuando uno de los combatientes obtiene HP menor o igual

**INTERFAZ DE COMBATE:**

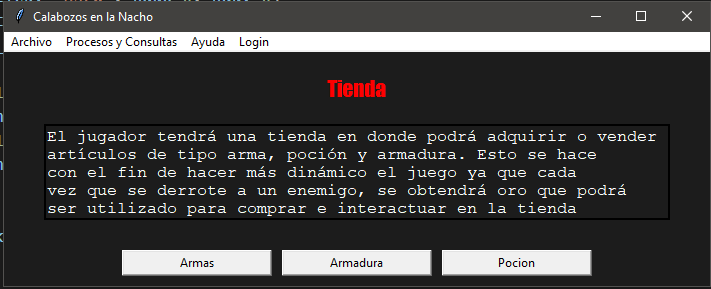
****



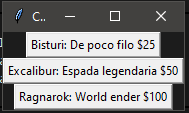
* **Tienda:** el jugador tendrá una tienda en donde podrá adquirir o vender artículos de tipo arma, poción y armadura. Esto se hace con el fin de hacer más dinámico el juego ya que cada vez que se derrote a un enemigo, se obtendrá oro que podrá ser utilizado para comprar e interactuar en la tienda



La transición a python para Armas, Pociones, Armaduras, Inventario y Tienda fue sencilla de hacer, las nuevas interacciones se realizaron al momento de usar la interfaz gráfica.

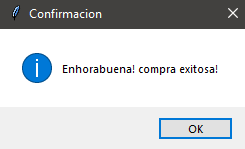


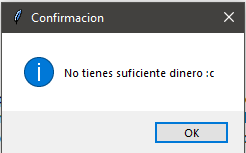
Para Tienda se pensó en desplegar las 3 opciones disponibles que son Armas Armaduras y Pociones, estas se muestran como botones, cada botón llama a una función diferente la cual despliega las armas/armadura/pociones disponibles en la tienda lanzan una ventana con las opciones disponibles para la compra.



Una vez se selecciona una de las opciones se llama al método correspondiente en tienda dependiendo de la clase del jugador, en este caso es “comprarWarrior” y luego se comprueba si es posible hacer la compra tras verificar el precio del arma con el dinero en el atributo wallet del jugador.

Después se le indica el resultado de la comprapor medio de una ventana emergente al llamar a la funcion “ayudaGood/ayudaBad” que se ejecuta dentro del mismo “comprarWarrior”.

l



* **Crear personaje:** se hace con el fin de que el usuario ingrese personalice más el juego.

**Manual de usuario**:

Antes de correr la aplicación debemos tener en cuenta algunos comandos numéricos como 1, 2… que se usaran para elegir opciones de narrativa. Y tendremos otros comandos de escritura que nos ayudaran en el juego. Los comando son \* solo se usan en combate.

|  |  |
| --- | --- |
| **Comando** | **Descripción** |
| help | Muestra la lista de comandos disponibles |
| save | Guarda el juego en la escena y con la configuración actual |
| hoja | Muestra la hoja de vida del personaje |
| atacar\* | Atacar al rival |
| pocion\* | El jugador puede usar una poción de recuperación de vida |
| escapar\* | El jugador intenta escapar del combate actual, tiene un 30% de probabilidaes de lograrlo |
| inv | Muestra los objetos del personaje en inventario |
| tienda | Muestra los objetos disponibles para comprar |

**Como jugar:**

Para jugar Calabozos en la Nacho has de leer primero la escena narrada en la pantalla principal, dicha narrativa te pondrá en contexto y a continuación podrás elegir qué camino seguir oprimiendo el botón que corresponda a la opción deseada, este te llevará a una nueva escena donde, mediante la misma mecánica, irás avanzando sucesivamente a través de la historia -!Elige distintos caminos para hallar los distintos desenlaces!- . Algunas veces tendrás que combatir, cuando esto ocurra se te habilitará la opción de batalla, donde dispones de ataques y habilidades para sobrevivir a dichas batallas. En la tienda podrás comprar equipamiento nuevo (armas, armaduras y pociones) que te permitirán ser más fuerte.

