PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

Proyecto “Down you go”

Benjamín Christian Vega Mardones

Profesor: Ignacio Araya

Asignatura: Estructura de Datos

Junio 2022

Índice

Resumen

El presente informe detalla la aplicación “Down you go”, la cual es un programa ejecutable de computadora programado en el lenguaje de programación C. Este documento presenta la información detallada de en qué consiste la aplicación, se indica cómo se utiliza el programa. Posteriormente se describe lo que se puede y no se puede hacer, cómo funciona internamente y el proceso de creación del programa. Finalmente se da una conclusión de todo el proyecto, desde la planificación hasta la entrega.

1 Introducción

La aplicación, de nombre “Down you go”, tiene fines de entretenimiento personal y está dirigida a personas que quieran dedicar un tiempo a ocio y distracción. El programa se contextualiza en la aventura de una persona que se adentra a una mazmorra con infinitas salas y un enemigo en cada una de ellas el cual debe derrotar para seguir avanzando. Los enemigos son generados aleatoriamente y poseen cualidades únicas. Los combates consisten en elegir una de 3 acciones; atacar, defender o usar un objeto. El combate termina cuando el personaje del jugador o el enemigo es derrotado. Si el jugador gana el combate, avanza, activando un evento aleatorio en el que puede suceder una variedad de cosas, desde obtener un objeto útil para el juego hasta algo puramente estético como obtener una descripción del ambiente, luego procede al siguiente enemigo y combate. El juego no tiene una forma de "ganar", cuando el jugador es derrotado la partida termina y se guarda su puntuación y nombre en un tablero de puntuación. La puntuación se calcula desde varias variables como el nivel del jugador, los enemigos derrotados o las armas obtenidas entre otras cosas. Esta puntuación luego aparece en el menú en la sección “HighScores”, la cual muestra las puntuaciones de partidas anteriores en orden cronológico.

2 Descripción de la Aplicación

La aplicación es un programa de computadora escrito en lenguaje C y programado en Visual Estudio Code. También conocidos como “Video Juegos”, tienen el fin de servir como un espacio de entretenimiento personal, dirigido a personas que quieran dedicar tiempo para distraerse y divertirse a la vez que se desafían a sí mismos u a otras personas a, por ejemplo, conseguir el puntaje mayor.

2.1 Historia y Contexto

El programa se contextualiza en la aventura de una persona que encuentra que la vida es aburrida, por lo cual se adentra a una mazmorra de la cual escuchó. Se dice que contiene una cantidad infinita de salas y peligros en cada una de ellas.

La persona que utiliza el programa (jugador) toma el papel de este personaje, quien toma lo primero que encuentra a su alcance como arma y de adentra en esta mazmorra. Cada sala tiene a un enemigo el cual el jugador debe derrotar para así seguir avanzando. Los enemigos de cada sala son generados aleatoriamente y poseen cualidades únicas. Los combates consisten en elegir una de 3 acciones; atacar, defender o usar un objeto. El combate termina cuando el personaje del jugador o el enemigo es derrotado. Si el jugador gana el combate, avanza, activando un evento aleatorio en el que puede suceder una variedad de cosas, desde obtener un objeto útil para el juego hasta algo puramente estético como obtener una descripción del ambiente, luego procede al siguiente enemigo y combate. El juego no tiene una forma de "ganar", cuando el jugador es derrotado la partida termina y se guarda su puntuación y nombre en un tablero de puntuación. La puntuación se calcula desde varias variables como el nivel del jugador, los enemigos derrotados o las armas obtenidas entre otras cosas. Esta puntuación luego aparece en el menú en la sección “HighScores”, la cual muestra las puntuaciones de partidas anteriores en orden cronológico

2.2 Lo que puede y no puede hacer

El juego se puede jugar únicamente individualmente, es decir solo una persona puede jugar a la vez. Dentro del juego se puede combatir contra una gran variedad de enemigos, cada uno único con sus propias fortalezas y debilidades, sin embargo, los combates son de uno contra uno, es decir no se puede combatir contra múltiples enemigos a la vez, se está limitado a uno por sala.

El juego guarda los puntajes de todas las partidas o “runs” terminadas en un archivo, el cual se puede mostrar desde el menú para comparar los puntajes de todas las runs pasadas.

El juego posee un sistema de guardado, lo que permite dejar una partida en pausa, sin perder el progreso incluso si se cierra el programa. Sin embargo, solo se guardará la partida cuando el jugador lo decida a través de una de las acciones disponibles en el Turno del Jugador en un combate, por lo cual si el programa es cerrado antes o después de ello el progreso nuevo no será guardado. Si se desea también se puede iniciar una nueva partida desde cero incluso si ya hay otra guardada, eliminándola en el proceso.

2.3 Menús

2.3.1 Menú Principal

El menú principal y el primero al que se encuentra el usuario cuando inicia el programa. Este menú indica al usuario ingresar un número correspondiente a una de las opciones disponibles, siendo estas “Nueva Partida”, “Cargar Partida”, “HighScores” y “Salir”.

2.3.1.1 Nueva Partida

Esta opción crea una nueva partida desde cero, no sin antes preguntar por la confirmación del usuario a eliminar la partida actual guardada si es que esta existe. Si el usuario no accede se vuelve al menú principal. En caso contrario el juego iniciará, creando una partida desde cero, iniciando una creación de personaje sencilla en la que se le pregunta al usuario por un nombre y raza para su personaje, el cual empieza de nivel 1 con estadísticas, armas y objetos aleatorios.

2.3.1.2 Cargar Partida

Si existe una partida guardada en el archivo de guardado esta será cargada y se empezará el juego desde el punto en el que se dejó, con el mismo personaje, armas y objetos, así como el mismo enemigo, sala, piso y puntuación.

2.3.1.3 HighScores

Muestra las puntuaciones de las partidas terminadas, con el nombre del personaje a su lado. También permite la opción de reiniciar todas las puntuaciones, eliminándolas todas a la vez.

2.3.1.4 Salir

Cierra el programa.

2.3.2 Menú de Combate

Este es el menú del jugador cuando está en combate contra un enemigo dentro de una partida. Muestra la vida actual del jugador, su nombre, nivel, y el nombre del enemigo, además de las opciones que el jugador puede realizar durante su turno. Este menú solo se muestra cuando es el turno del jugador durante el combate.

2.3.2.1 Atacar

Esta opción utiliza el turno del jugador para atacar al enemigo con un arma a elección de las disponibles en su inventario. Al atacar al enemigo se le restarán puntos de vida a este. La fórmula del cálculo del daño es la siguiente:

$$DañoRealizado = \big((AtkAtacante + AtkArmaAtacante) * Efectividad\big) - ReducciónDefensor \quad (2.3.2.1)$$

En la cual:

- "Efectividad" es igual a 1,5 si la efectividad del arma del atacante coincide con la raza del defensor y 1 en caso contrario.
- "ReducciónDefensor" es igual a la Defensa del defensor más la Defensa otorgada por el arma del defensor en el caso de que el arma atacante sea de daño físico, y Resistencia del defensor más la Resistencia otorgada por el arma del defensor en el caso de que el arma atacante sea de daño mágico.
- DañoRealizado no puede ser menor a 1.

2.3.2.2 Defender

Esta opción utiliza el turno del jugador para entrar en el estado defensivo, el cual reduce a la mitad el daño recibido por un ataque y tiene un 10% de probabilidades de reducirlo completamente.

2.3.2.3 Objeto

Esta opción utiliza el turno del jugador y le permite seleccionar un objeto de su inventario para utilizarlo. Si el jugador no posee ningún objeto el turno será utilizado igualmente y se le mostrará al jugador una indicación de que su inventario está vacío.

2.3.2.4 Información del Jugador

Esta opción no utiliza el turno del jugador, permitiéndole actuar nuevamente. Muestra la información del jugador, incluyendo su nombre, raza, nivel, experiencia, hp, estadísticas, armas y objetos.

2.3.2.5 Analizar al Enemigo

Esta opción utiliza el turno del jugador y muestra la información del enemigo actual, incluyendo su nombre, raza, nivel, experiencia, hp, estadísticas, armas y objetos.

2.3.2.6 Guardar y Salir

Esta opción guarda la partida actual y vuelve al menú principal.

3 Descripción de la solución

3.1 Estructuras de datos

3.1.1 RunManager

Contiene variables de enteros para la puntuación, el piso, la sala, el turno y el estado del juego. Se utiliza en la función “play” para almacenar los datos de la partida actual.

3.1.2 RacesFile

Contiene una variable de entero que representa la cantidad de razas disponibles y un puntero a un tipo FILE que se utilizará para el archivo “races.csv”.

3.1.3 ItemsListNode

Contiene una variable de entero que representa la cantidad de objetos disponibles y un puntero a un tipo List que se utilizará para guardar las variables tipo Item que se leerán desde el archivo “items.csv”.

3.1.4 WeaponsListNode

Contiene una variable de entero que representa la cantidad de armas disponibles y un puntero a un tipo List que se utilizará para guardar las variables tipo Weapon que se leerán desde el archivo “weapons.csv”.

3.1.5 Character

Contiene variables de enteros que almacenan la raza, el nivel, la experiencia actual, la experiencia para subir nivel, el estado del personaje, los puntos de vida actuales y el arma actual, además de un string que almacena el nombre del personaje, un puntero a un tipo Stats que se utiliza para almacenar las estadísticas del personaje, un puntero a 4 punteros tipo Weapon que se utilizan para almacenar las armas del personaje y un puntero a un TreeMap que almacena los objetos del personaje.

3.1.6 Weapon

Contiene dos strings, uno almacena el nombre y el otro la descripción del arma. También tiene variables de enteros que almacenan la rareza del arma, la durabilidad restante, la efectividad y el tipo de daño del arma, además un puntero a un tipo Stats que almacena las estadísticas del arma.

3.1.7 Item

Contiene dos strings, uno almacena el nombre y el otro la descripción del objeto. También tiene variables de enteros que representan el tipo de objeto, los usos restantes y un valor que varía dependiendo del tipo de objeto.

3.1.8 Stats

Contiene 5 variables de enteros los cuales almacenan los puntos de vida, el ataque, la velocidad, la defensa y la resistencia. Se utiliza para los personajes y las armas, representando sus estadísticas.

3.2 TDAs

3.2.1 Lista

Es utilizada dos veces, una para guardar las armas y otra para guardar los objetos leídos desde los archivos. La lista de armas almacena tipo Weapon y se utiliza cuando se obtiene una nueva arma, de la misma manera la lista de objetos almacena tipo Item y se utiliza cuando se obtiene un nuevo objeto, copiando el contenido en una nueva variable.

3.2.2 TreeMap

Es utilizado únicamente como inventario para almacenar los objetos de cada personaje. En este se utiliza el nombre del objeto como llave y la razón de elegir este TDA es para permitir sumar más usos de un objeto ya obtenido en caso de obtenerse de nuevo.

3.3 Funciones

La aplicación posee múltiples funciones que a su vez llaman a otras funciones sin un orden específico, por esta razón las funciones listadas a continuación están ordenadas alfabéticamente.

3.3.1 aiPhase

Recibe dos punteros a tipo Character, uno siendo el del enemigo actual y el otro el del jugador.

Verifica los puntos de vida actuales del enemigo y si son mayores a 50% de su vida máxima entonces atacará el 100% de las veces, si es menor al 50% tendrá un 40% de probabilidades de elegir atacar, un 30% de elegir defenderse y un 30% de elegir utilizar un objeto, si es menor al 20% tendrá un 70% de probabilidades de utilizar un objeto y un 30% de atacar. Si se decide atacar se llama a la función “attack”, al defender se cambia el estado del personaje a “defendiendo” o “2”, y para utilizar un objeto llama a la función “useItem” con argumento 1.

Si la función “attack” retorna 1, esta función también retorna 1, retorna 0 en cualquier otro caso.

3.3.2 attack

Recibe dos punteros a tipo Character, el primero siendo el “atacante” y el segundo siendo el “defensor”.

Reduce los puntos de vida de defensor restando a sus puntos de vida actual, la función “damageCalc”. Los puntos de vida del defensor no pueden ser menores a 0. Adicionalmente si el estado de defensor era 2, se cambia a 0 y se le otorga experiencia por defender. También se llama a la función “lowerWeaponDurability” para reducir la durabilidad restante al arma que utilizó el atacante.

Luego se verifica si los puntos de vida actuales del enemigo son iguales a 0, si es cierto entonces se le otorga experiencia al atacante y se retorna el valor 1. En caso contrario solo se retorna el valor 0.

3.3.3 chooseNextAvailableWeapon

Recibe un puntero Character e itera con un contador desde la última hasta la primera arma del personaje, verificando si esa arma es Nula, luego cambia el valor del arma actual del jugador a la del contador.

3.3.4 createChara

Recibe un puntero a tipo Stats llamado stats, dos enteros, uno llamado level y el otro llamado race y un string llamado name.

Reserva memoria para un tipo Character e inicializa todas sus variables en 0, el nombre name, el nivel level, la raza race y las estadísticas stats. Adicionalmente llama a la función createWeapon para crear un arma vacía de estadísticas 0 y asignarla al espacio de armas número 0 que se utilizará para cuando el personaje se quede sin armas. También inicializa el TreeMap con la función createTreeMap.

Retorna el puntero al nuevo personaje.

3.3.5 createItem

Recibe dos punteros a string, uno representa el nombre de un objeto y el otro la descripción, también recibe tres variables enteras llamadas type, uses y value.

Reserva memoria para un tipo Item e inicializa sus variables con los datos recibidos.

Retorna el puntero al objeto creado.

3.3.6 createPlayerChara

Se le pregunta al usuario por un nombre y una raza para el personaje, luego se llama a la función “genRanCharaStats” para obtener estadísticas y luego llama a la función createChara utilizando los datos obtenidos, finalmente cambia el valor de “isPlayer” a 1.

Retorna el puntero al personaje creado.

3.3.7 createWeapon

Recibe dos punteros a string, uno representa el nombre de un arma y el otro la descripción, también recibe nueve variables enteras llamadas rarity, durability, effectiveness, damageType, hp, atk, spd, def y res.

Reserva memoria para un tipo Weapon y para un tipo Stats, e inicializa las variables con los datos recibidos.

Retorna el puntero al arma creada.

3.3.8 damageCalc

Recibe dos punteros a tipo Character, el primero siendo el “atacante” y el segundo siendo el “defensor”.

Realiza el cálculo del daño que el atacante le infringiría al defensor con la fórmula de la figura (2.3.2.1). Luego si el estado del defensor es igual a 2, hay un 10% de probabilidades de que el daño se reduzca a 0, en el otro 90% se reduce el daño a la mitad.

Retorna el daño infringido.

3.3.9 deleteSave

Limpia el archivo “save.csv”, dejándolo en blanco.

3.3.10 genRanChara

Recibe un entero que representa el nivel.

Crea un nuevo personaje usando la función “createChara”, con una raza aleatoria y le otorga un arma aleatoria con la función “giveWeapon(getRanWeapon)” y un objeto aleatorio de tipo 1 o 2 con la función “giveItem”.

Retorna el puntero al personaje creado.

3.3.11 genRanCharaStats

Recibe un entero que representa el nivel.

Reserva memoria para un tipo Stats e inicializa sus variables, cada una siendo igual a un número al azar entre el 80% y el 120% del nivel.

Retorna el puntero a stats.

3.3.12 getRanItem

Recibe un entero que representa el tipo.

Verifica si hay al menos un objeto en la lista de objetos del tipo recibido, si es así entonces retorna un objeto aleatorio de la lista. En caso contrario retorna Nulo.

3.3.13 getRanWeapon

Recibe un entero que representa la rareza.

Verifica si hay al menos un arma en la lista de armas de la rareza recibida, si es así entonces retorna un arma aleatoria de la lista. En caso contrario retorna Nulo.

3.3.14 giveItem

Recibe un puntero a un tipo Character y uno a un tipo Item.

Verifica si el item ya está en el TreeMap del personaje, si es así entonces aumenta los usos restantes, en caso contrario inserta el item al inventario del personaje.

3.3.15 giveWeapon

Recibe un puntero a un tipo Character y uno a un tipo Weapon.

Verifica si el personaje ya tiene tres armas, si es así entonces le pregunta al jugador por el arma que desea dejar, si es la nueva solo se libera la memoria de esta. Si es una del inventario libera la memoria del arma escogida y asocia la nueva arma a ese espacio. Si aún tiene espacio para armas solo se asocia la nueva arma al espacio libre.

3.3.16 levelUp

Recibe un puntero a un tipo Character.

Incrementa el nivel del personaje recibido en 1 e incrementa las estadísticas del personaje en un valor aleatorio.

3.3.17 loadChara

Recibe un puntero a tipo FILE que contiene el archivo “sabe.csv” y un string que contiene una línea previamente leída desde el archivo.

Crea un nuevo personaje con la función “createChara” con los valores leídos en el archivo. Luego lee las siguientes tres líneas que representan las armas del personaje y crea armas para luego dárselas al personaje, si la línea es NULL no se crea esa arma.

Retorna el puntero al personaje creado.

3.3.18 loadItems

Recibe un puntero a tipo Character, uno a tipo FILE que contiene el archivo “sabe.csv” y un string.

Lee cada línea del archivo recibido y crea un objeto con los valores leídos para luego insertarlo en el inventario de objetos del personaje recibido, hasta que no haya más líneas.

3.3.19 loadRun

Recibe el puntero al tipo RunManager llamado runManager.

Abre el archivo “save.csv” y luego asigna los valores leídos desde el archivo a runManager, a continuación utiliza “loadChara” en dos veces, primero creando el personaje del jugador y luego el del enemigo actual, después utiliza “loadItems” con el personaje del jugador y el archivo “playerItems.csv” y con el personaje del enemigo y el archivo “enemyItems.csv”. Finalmente llama a la función “play” con runManager, el personaje del jugador y el personaje del enemigo.

3.3.20 main

Crea un puntero a una variable tipo RunManager llamada runManager, luego lama a la función “openAllFiles” para leer los archivos necesarios y luego muestra el menú principal, pidiéndole al usuario que ingrese un número para acceder a cada opción.

La opción 1 llama a la función “newRun” con runManager como argumento.

La opción 2 llama a la función “loadRun” con runManager como argumento.

La opción 3 llama a la función “showHighScores” y luego pregunta si se desea eliminar los puntajes guardados, eliminándolos si se acepta.

La opción 4 cierra el programa.

3.3.21 newRun

Recibe un puntero a un tipo RunManager llamado runManager.

Reserva memoria para runManager e inicializa sus variables, luego crea un puntero a una variable tipo Character de nombre “playerChara” con la función “createPlayerChara”, muestra el personaje creado con la función “printChara”, crea un arma de rareza 1 con “getRanWeapon” y se la otorga con la función “giveWeapon”, crea un objeto con la función “getRanItem” y se lo otorga con la función “giveItem”, crea un puntero a una variable tipo Character llamada “enemyChara” y finalmente llama a la función “play” con runManager, playerChara y enemyChara como argumentos.

3.3.22 openAllFiles

Llama a las funciones “readItems”, “readWeapons” y “openRaces” y muestra en pantalla un mensaje diciendo que los archivos se abrieron correctamente.

3.3.23 openRaces

Abre el archivo “races.csv” y calcula la cantidad de razas presentes dentro del archivo.

3.3.24 play

Recibe un puntero a runManager, playerCharacter y enemyChara.

En una iteración constante utiliza un switch verificando la variable de estado dentro de runManager.

En el estado 1 se incrementa el puntaje, se incrementa el valor de la sala y/o nivel actual, luego llama a la función “genRanChara” para crea un nuevo personaje enemigo de nivel igual a un margen del 20% del nivel del jugador con un mínimo de 1, finalmente cambia al estado 2.

En el estado 2 se comparan las velocidades de playerChara y enemyChara, luego se llama a las funciones “playerPhase” y “aiPhase” en orden, siendo la primera la del personaje con mayor velocidad. En cada llamado se verifica si el valor retornado por las funciones es igual a 1; Si “playerPhase” retorna 1 se cambia al estado 3 y se suma puntaje, si “aiPhase” retorna 1 el estado cambia a 4.

En el estado 3 se llama a la función “randomEvent”, se libera la memoria de enemyChara con “freeCharacter”, se muestra el estado del jugador con “printChara”, el valor del turno vuelve a 0 y el estado cambia a 1.

En el estado 4 se llama a la función “deleteSave”, se muestra en pantalla “Game Over”, se llama a la función “saveScore” con playerChara y runManager como argumentos, se libera la memoria de playerChara y enemyChara con “freeCharacter” y finalmente se retorna al menú.

3.3.25 playerPhase

Recibe un puntero playerCharacter, enemyChara y runManager.

Se le muestra al jugador las acciones disponibles y se le pide que ingrese un número para elegir una acción.

La primera acción es atacar. En esta se le pregunta al jugador qué arma quiere utilizar para atacar y luego se llama a la función attack. Si attack retorna 1 esta función también retorna 1.

La segunda acción es defender. En esta se cambia el estado del personaje del jugador a 2.

La tercera opción muestra el estado del personaje del jugador, llamando a la función “printChara”, “showWeapons” y “showItems” con playerChara como argumento.

La cuarta opción muestra el estado del personaje enemigo, llamando a la función “printChara”, “showWeapons” y “showItems” con enemyChara como argumento.

La quinta opción llama a la función “saveRun”.

3.3.26 printChara

Recibe un puntero a un tipo Character.

Muestra en pantalla el nombre, raza, nivel, experiencia, puntos de vida y estadísticas del personaje recibido.

3.3.27 printItem

Recibe un puntero a un tipo Item.
Muestra en pantalla la información del objeto recibido.

3.3.28 printWeapon

Recibe un puntero a un tipo Weapon.
Muestra en pantalla la información del arma recibida.

3.3.29 randIntLimits

Recibe dos enteros.
Retorna un número entero entre el primer y el segundo número incluyéndolos.

3.3.30 randomEvent

Recibe un puntero a un tipo Character llamado playerCharacter y un puntero a un tipo RunManager llamado runManager.
Activa un evento aleatorio, desde perder estadísticas hasta obtener objetos.

3.3.31 readItems

Abre el archivo “ítems.csv”, reserva la memoria para la variable global “itemsList”, crea sus 5 listas y crea un nuevo objeto por cada línea de “ítems.csv”, los cuales luego se insertan en la lista correspondiente a su tipo.

3.3.32 readWeapons

Abre el archivo “weapons.csv”, reserva la memoria para la variable global “weaponsList”, crea sus 5 listas y crea una nueva arma por cada línea de “weapons.csv”, las cuales luego se insertan en la lista correspondiente a su rareza.

3.3.33 saveCharacter

Recibe un puntero a un tipo Character y un puntero a un archivo.
Escribe en el archivo los datos del personaje recibido y de sus armas.

3.3.34 saveItems

Recibe un puntero a un tipo Character y un puntero a un archivo.
Escribe en el archivo los datos de cada objeto dentro del inventario del personaje recibido.

3.3.35 saveRun

Recibe un puntero playerCharacter, enemyChara y runManager.

Guarda la información de la partida actual utilizando las funciones “saveCharacter” y “saveItems” para cada personaje, luego usa “freeCharacter” con ambos personajes para liberar la memoria.

Retorna 0 si se guardó con éxito.

3.3.36 saveScore

Recibe un puntero a playerChara y a runManager.

Abre el archivo “highscores.csv”, calcula el puntaje de la partida y escribe en el archivo el nombre del personaje en la primera columna y el puntaje calculado en la segunda, luego cierra el archivo.

3.3.37 showHighScores

Abre el archivo “highscores.csv” y muestra el nombre y puntaje en cada línea.

3.3.38 showItems

Recibe un puntero a un tipo Character.

Muestra la información de todos los objetos que el personaje tiene en su inventario. Si no hay objetos muestra un mensaje diciendo que el inventario está vacío.

3.3.39 showWeapons

Recibe un puntero a un tipo Character.

Muestra la información de todas las armas que el personaje tiene en su inventario. Si no hay objetos muestra un mensaje diciendo que no tiene armas.

.

Conclusión

El programa desarrollado fue un gran desafío y tomó una larga investigación, planeación y tiempo para aplicar los conocimientos obtenidos en clase y escribir el código, adaptando los TDAs, solucionando problemas y errores que surgían durante el desarrollo y diseñando las funciones.

Se estima que se podrían haber agregado más características y optimizado tanto el tiempo de ejecución como el uso de memoria de haber dispuesto de más tiempo, pero al final se está satisfecho con el resultado y la elección de este.

Anexos

Fórmula