Proyecto Final Ciencia de la Computación I: Role Playing Game

Claudia A. Paredes Cornejo
Estudiante de Universidad Católica San Pablo
Arequipa. Perú
claudia.paredes.cornejo@ucsp.edu.pe

J. Santiago Aparcio Valdivia
Estudiante de Universidad Católica San Pablo
Arequipa. Perú
santiago.aparicio@ucsp.edu.pe

***Resumen*—Para proyecto final del curso Ciencia de la Computación I, decidimos hacer un Role Playing Game, de manera que el usuario pueda elegir el personaje que más le agrade y a partir de ello crear una historia, así el protagonista pueda interactuar con otros personajes, objetos y escenarios. Para la implementación de nuestro juego usamos las características básicas de Programación Orientada a Objetos, como son clases, a partir de ellas generar herencia, encapsulamiento, polimorfismo, etc. Además tomamos a la biblioteca Allegro [1] para insertar gráficos y sonidos y el IDE Code Blocks [2].**

***Abstract*—For the final project of the Computer Science I course, we decided to do a Role Playing Game, so that the user can choose the character that he likes the most and from that create a story, so the protagonist can interact with other characters, objects and scenarios. For the implementation of our game we use the basic characteristics of Object Oriented Programming, as they are classes, from them generate inheritance, encapsulation, polymorphism, etc. We also take the Allegro library to insert graphics and sounds and the IDE Code Blocks.**

***Index Terms* — RPG, Herencia, Encapsulamiento, MIDI, POO, NPC, Allegro.**

I. INTRODUCCIÓN

El motivo por el cual elegimos hacer un RPG es porque tenemos un gusto por juegos como Final Fantasy VII, Fall Out 4, The Legend of Zelda, etc., pues los encontramos muy dinámicos e interactivos. Nosotros somos los protagonistas del juego y vamos creando una historia progresivamente, cada vez que tomamos decisiones dependiendo de la situación, es así que exploramos "mundos" distintos hasta terminar el juego.
Nos agrada que un RPG nos dá la libertad de elegir lo que queremos hacer, es cierto que en muchos juegos hay un historia de por medio y de una forma u otra se "dirige" al usuario hasta que llega a su objetivo y termina el juego, sin embargo ese idea de independencia es lo que más nos gusta; así el usuario puede decidir qué hacer y construir un libreto nuevo cada vez.

II. TRABAJOS RELACIONADOS

Existen varios proyectos en internet sobre Role Playing Game, ya que es un tipo de juego que atrae bastante a los gamers, hacer uno propio no es muy complicado y además da amplias posibilidades sobre el estilo de juego. Encontré algunos proyectos sobre un RPG en Python [3], este proyecto en específico me agradó bastante por su simpleza y por la lógica de juego. También encontré un RPG en C++ [4], este a comparación del primero es más detallado, claro que solo es un archivo pero tiene más características. Revisé algunos otros proyectos para darme ideas, compararlos y así poder aportar y mejorar el proyecto.

III. PROPUESTA

Como se mencionó anteriormente, elegimos hacer un RPG como proyecto final porque este tipo de juego nos permite crear y modificar a los personajes; como también nos brinda la posibilidad de crear una amplia y extensa variedad de sistemas, modos y estilos de juego.
Nos agradó la flexibilidad que nos ofrece al momento de construir la historia, los personajes, los escenarios, niveles, retos, etc; dándole una gran posibilidad a nuestra creatividad.

Además al momento de ejecutar el juego, se puede conocer un

poco al usuario, pues se le pondrá a prueba en reiteradas situaciones, teniendo que realizar determinadas acciones y tomar decisiones para continuar en el juego.

Nuestra idea para realizar el juego fue crear un RPG en el que el usuario tenga la posibilidad de elegir el personaje que más le guste entre Mago, Guerrero, Pícaro y Elfo, todos estos personajes se mostrarán el principio del juego, cuando se inicia la partida.

El mundo que creamos tendrá un estilo medieval, estilo antiguo, retro con bitmaps de 24 bits, por tanto los escenarios, los personajes, diálogos se adaptarán a este estilo. Queremos que el protagonista empiece su historia en su casa, específicamente en su habitación, luego sale de ella y aparece en el bosque, aquí ya se dará la libertad al usuario que explore el mundo, los distintos escenarios. Los NPC's le irán dando pistas y opciones para el que el usuario tome sus decisiones. En caso un personaje de ataque (ej. un ogro) agote la variable vida (que se muestra al usuario en la barra de vida) el protagonista muere y se acaba el juego.

Además queremos implementar la posibilidad de que el protagonista pueda comprar objetos, para ello debemos tener un inventario y una tienda. Esta parte se explicará a mayor detalle posteriormente.

1. PATRÓN DE DISEÑO:

El patrón de diseño que decidimos usar es el MVC (Model View Controller) [5] porque se adapta a la forma que le dimos al juego y nos ayuda a separar los datos. Seguidamente explicamos cada componente y qué rol cumple en nuestro juego.

A. MODEL:

Para este componente, debemos tener claro que su función es almacenar la información del estado del juego en sí, y este modelo se va actualizando constantemente dependiendo de las acciones del usuario, como también puede que sólo se le esté consultando algo. Por tanto el controlador está muy en contacto con el modelo, tal como podemos observar en la Fig. 1.

B. VIEW:

En la vista se encuentra la interfaz del juego, es decir lo que el usuario ve básicamente; y esta misma se va actualizando a medida que el model cambia su estado. Para especificar, uno de los métodos de nuestro juego que se encargan de la vista son pinta_pantalla, pintar_pantalla, create_bitmap, etc.

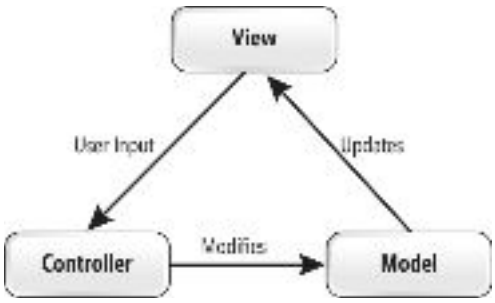


Fig. 1. Funcionamiento del patrón de Diseño Model View Controller (MVC).

C. CONTROLLER:

El controlador se encarga de detectar los datos ingresados por el usuario (input), en nuestro caso cuando el usuario toca las teclas de dirección para poder mover al personaje, cuando se presiona la tecla para salir, o cuando el usuario se acerca a un NPC y puede tocar la tecla de ataque o la tecla para hablar. Todos estos eventos los recibe el controller y hace que el modelo se modifique con estas nuevas acciones. Usualmente para identificar dichos eventos de usa el método KEY[tecla_a_presionar], se detecta cuando el usuario presiona determinados botones del teclado.

2. DIAGRAMA DE CLASES:

Para explicar nuestro código de una manera más dinámica he creado un diagrama de clases, proyectando las relaciones entre las clases, los métodos, los atributos, etc.

Se adjunta en el Diagrama UML en la página ANEXO 01, a continuación procedemos a explicarlo.

En la clase Controller tenemos cinco clases en las cuales se recibe información por el usuario y estas contienen numerosos métodos y atributos que trabajan con dichos datos. La clase Player trabaja específicamente con el protagonista, con sus distintas funciones (hablar, atacar, mover, posicionar) y las interacciones con su entorno, luego tenemos a la clase NPC, lo que significa Non-Player Character, es decir los demás aldeanos, que tienen numerosas variables y métodos, los más resaltantes son que cuando el protagonista se acerca y presiona la tecla Enter estos "hablan" desplegando un cuadro de diálogo, informando al protagonista sobre sus opciones; otro método es que cuando colisiona con una pared o cualquier superficie (incluso el mismo protagonista), inmediatamente revierte su dirección.

La clase MiMenu, se "pinta" lo que se mostrará al usuario cuando empieza el juego y cuando presiona la tecla escape, cuando se muestra al comenzar el juego, el protagonista tendrá

la opción de elegir el personaje y comenzar, el juego. Para salir del juego la clase recibe datos del teclado.
La clase Enemigo tiene herencia de la clase NPC ya en sí tienen la misma lógica pero el Enemigo tiene la capacidad de atacar, por ello tiene los métodos que detectan cuando el enemigo se desplaza, cuando es herido y cuando muere.
Por último tenemos a la clase MiJuego que es la clase que inicializa todo básicamente, como la escena, la biblioteca, el menú, etc, actualiza cada vez que recibe entradas del usuario.

En la clase View, coloqué las tres clases que se dedican netamente a cambiar la interfaz para el usuario, como son mensaje, en la que se muestran los diálogos, por ellos crea los recuadros, detecta el número de líneas que tendrá el string y método para cambiar el texto.
La clase TBoton que básicamente trabaja con los inventarios, los objetos disponibles que se le ofrecen al usuario y los botones se crean para que el usuario pueda elegir su producto, recibe datos de cuando el usuario selecciona con el mouse un determinado objeto..
Por último en la clase audio tenemos los métodos que reproducen distintos MIDI, dependiendo el escenario en el que se encuentre el personaje.

En la clase Modelo tenemos sólo dos clases, la clase Partida y la clase Objeto Venta. La primera se encarga de usar ficheros para guardar el estado general de la partida cada vez el usuario ejecuta el juego, por ello sólo tenemos dos métodos dentro de ella, como son cargar partida y guardar partida.
La otra clase Objeto Venta trabaja básicamente con la tienda, inventarios, eventos del mouse, usamos vector para poder almacenar los objetos del inventario, ordenarlos posteriormente.

3. CASOS DE USO:

A continuación se detalla cómo el usuario puede interactuar con el juego.
Al ejecutar el código, lo primero que se le muestra al usuario es el menú, en el cual tendrá tres opciones:

1. Nueva Partida
2. Cargar Partida
3. Salir

Para elegir una opción el usuario debe usar el teclado, las teclas direccionales y las tecla enter. Si se selecciona la primera opción se procede a la elección de personaje y luego comienza la historia; si se selecciona la segunda opción se cargarán los datos totales de la última partida, si nunca se ejecutó antes sería como iniciar una partida nueva. Si se selecciona la tercera opción se procede a cerrar la ventana de juego para acabar con la ejecución.

Luego de elegir el personaje a usar, se empieza el juega en el escenario de la habitación, ya el usuario puede manipular al protagonista a partir de las flechas direccionales para desplazarlo. Lo más lógico es que al no saber qué hacer en la habitación ya que en ese escenario no hay ningún detalle más que objetos con los que el usuario puede chocar o pasar "detrás" de ellos (para dar un aspecto más real al juego) el usuario guiará al personaje a salidas como son la puerta o las escaleras. Al salir de la habitación el escenario siguiente es el bosque, en el cual la música de fondo cambia, y también se reproduce un sonido cuando el protagonista sale de su casa, como si abriera la puerta.
Ya en el bosque el usuario explorará desplazándose en un escenario más amplio, en el camino verá varios NPC's tanto mujeres como hombres; si nos acercamos a ellos y presionamos la tecla enter, estos nos despliegan un cuadro de diálogo, contándonos sobre ellos u opciones de lo que podríamos hacer, si el usuario mueve al protagonista hasta la zona superior del mapa se encontrará con NPC's pero con barra de vida sobre ellos, esto ya da una pista al usuario que deberá enfrentarse a dicho NPC. La tecla de ataque (al igual que en la mayoría de juego de aventura de PC) es la tecla Espacio.
Considero que es hasta aquí lo que se podría predecir sobre las acciones del usuario, pues a partir de este punto el protagonista podría seguir distintos caminos, reslizar distintas acciones, como explorar mapas, hablar con NPC's, enfrentarse a los enemigos o comprar objetos. Eso ya dependerá del usuario y no le será difícil descubrir cuáles son los eventos de teclado para completar sus cometidos.
En caso el personaje sea herido reiteradas veces por un enemigo, su barra de vida va disminuyendo, si llego a 0 el personaje muere, se acaba el juego, GAMEOVER.

4. IMPLEMENTACIÓN:

El objetivo del proyecto era el poner en práctica los conceptos explicados en el curso Ciencia de la Computación I, que la mayoría de ellos eran las características básicas de POO (Programación Orientado a Objetos) y aprovechar algunas caracteristicas también de C++.
Una de las que podemos resaltar es el uso de clases en nuestro código, así como también el encapsulamiento en cada una de ella para poder separar la información en "cajas". Otra característica es la herencia que se dan entre clases como la clase Enemigo que hereda de NPC, ya que su lógica es la misma, lo único distinto es que el enemigo puede atacar, y por tanto también puede morir y desaparecer.

Un tema que también se trabajó en el curso es el manejo de archivos, que lo implementan en el header Partida.h para poder guardar los datos de la partida y cargarla nuevamente, para almacenar estos datos usamos packfile's que los llamamos ficheros, además aquí se encuentran numerosos método que se encargan pedir y almacenar estados constantemente.
Adicionalmente trabajamos con headers bastante, de forma que se les incluía en el main simplemente.
Otro aspecto resaltante es que utilizamos un grabber para poder insertar y cargar nuestros archivos.

Ya que usamos la biblioteca allegro, debemos tomar en cuenta que para los bitmaps, el color rojo RBG (255, 0, 0) lo traduce como una colisión, y el color fucsia RBG (255, 0, 255) lo traduce como superficie, es decir que los lugares que no estén de fucsia pueden ser "atravesados" por debajo, esta característica se aplicó para darle un aspecto más real al juego.
Para lograr estos resultados se debe trabajar el mapa en una plataforma editora de imágenes, en este caso se usó Photoshop [4] entonces por cada mapa tendré 2 bitmaps como se detalla en la Fig.8 y la Fig.9 .

Para implementar el sonido a las escenas se descargó distintos SAMPLES y MIDI's con estilo medieval o dependiendo de la acción que requiere como abrir puerta, mover una espada, un golpe, etc.

IV. PRUEBAS Y RESULTADOS

Todo lo descrito anteriormente se ha implementado en el proyecto, los personajes, sonidos, guardar partida, distintos escenarios.
Cabe resaltar que la idea inicial de nuestro proyecto fue un RPG pero con animación 3D, para ello pensábamos usar una plataforma llamada Unreal [7] ya que está en C++, pero definitivamente no estábamos listos para trabajar con objetos en tres dimensiones, ni disponíamos del tiempo para aprenderlo, así que descartamos a esta biblioteca.

Luego cambiamos a una animación 2D, y elegimos inicialmente la biblioteca SFML [8], sin embargo hubo algunos problemas para instalar la librería en la PC de mi compañero, por tanto vimos conveniente probar otra opción, fue así que exploramos para encontrar librerías que nos permitan ingresar contenido multimedia, hasta que encontramos Allegro, instalamos la librería en el IDE CodeBlocks y creamos una pequeña ventana de ejemplo. Fue ahí cuando determinamos oficialmente nuestra librería para el proyecto.

También se usó algunas funciones de la biblioteca allegro como inicia_allegro(). Para esto es necesario incluir <allegro.h>.
A continuación se adjuntan capturas de pantalla sobre el juego y características importantes que se implementaron en el código:

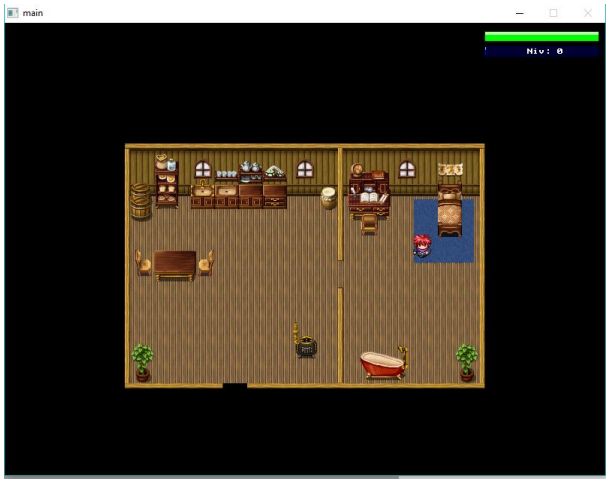


Fig. 2. Habitación de protagonista, captura y escenario inmediato al empezar nueva partida.

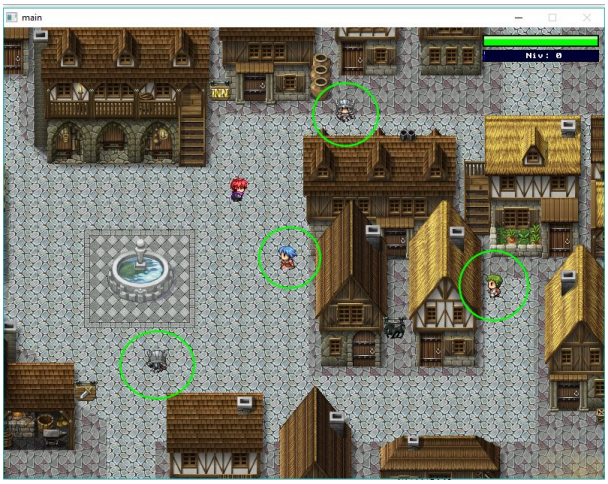


Fig. 3. Ubicaciones de los NPC's en el tercer escenario.

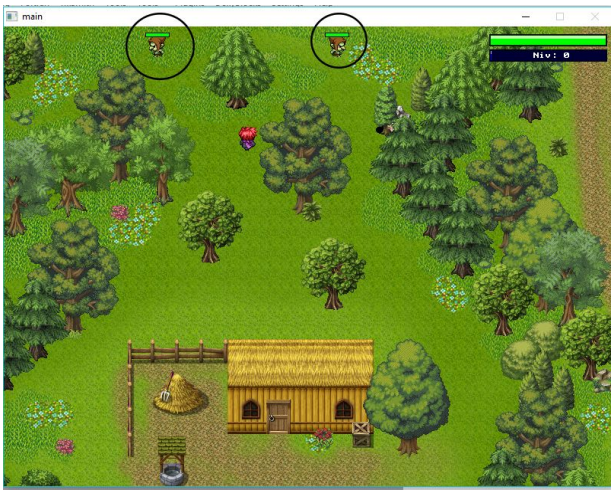


Fig. 4. Ubicaciones de los Enemigos en el segundo escenario.

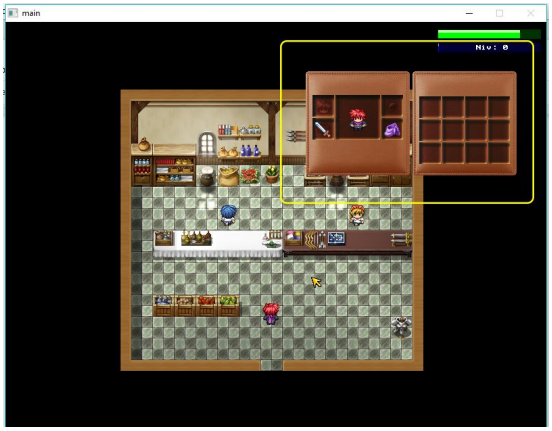


Fig. 5. Tienda e Inventario.

Fig. 5. Cuadros de Diálogo de NPC's.

Fig. 6. Bitmap de personaje.

Fig. 6. Método de ataque a enemigo, se evidencia en la barra de vida.

Fig. 7. Bitmap de escenario.

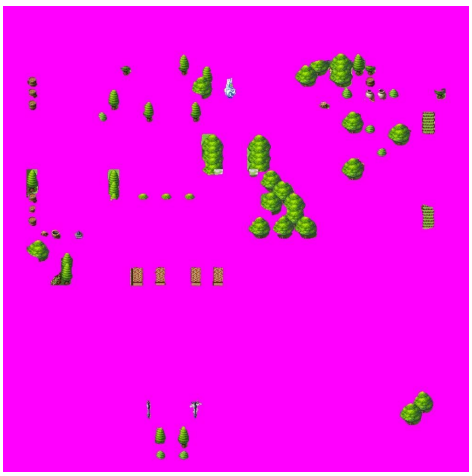


Fig. 8. Bitmap de superficies, los elementos verdes son aquellos que se pueden atravesar.

Fig. 9. Bitmap de colisión. El personaje chocará con todas las zonas rojas

V. CONCLUSIONES Y TRABAJOS FUTUROS

Nuestro proyecto trató de usar los temas estudiados en el curso, sin embargo en algunos casos no se logró implementar. Fue un proyecto que fue de avance progresivo, sin embargo tuvimos algunos percances que dificultaron el ritmo de avance.

El patrón de diseño si bien es cierto lo conocemos y entendemos bastante bien , no se logró implementar puesto que ya teníamos mucho código avanzado y debemos reestructurar con el MVC, lo intenté pero luego de cambiarlo, dejó de compilar o faltaban muchos elementos; por ello se prefirió dejarlo con aspecto de clases.

Nos agradó mucho programar un Role Playing Game porque nos dio bastante libertad de crear personajes, ambientes, estilos, etc.

Lo mejor que en el proceso ponemos en práctica la teoría del curso, haciéndolo más fácil de entender y aprendimos cada vez un poco más.

Para trabajos futuros tenemos la idea de crear una historia distinta para cada uno de los personajes a elegir, es decir una historia para el mago, otra distinta para guerrero, otra para el pícaro e igualmente para el elfo. Para ello necesitamos más tiempo para implementar más NPC'S, objetos y funciones. Tratamos de hacerlo para esta presentación de proyecto pero no tuvimos el tiempo suficiente; no obstante ya tenemos el libreto de la historia de cada uno de los personajes, se encuentra en el ANEXO 02.

VI. REFERENCIAS

[1] "Allegro - A game programming library –", Liballeg.org, 2017. [Online]. Available: http://liballeg.org/. [Accessed: 27- Nov- 2017].

[2] "Code::Blocks", Codeblocks.org, 2017. [Online]. Available: http://www.codeblocks.org/. [Accessed: 24- Nov- 2017].

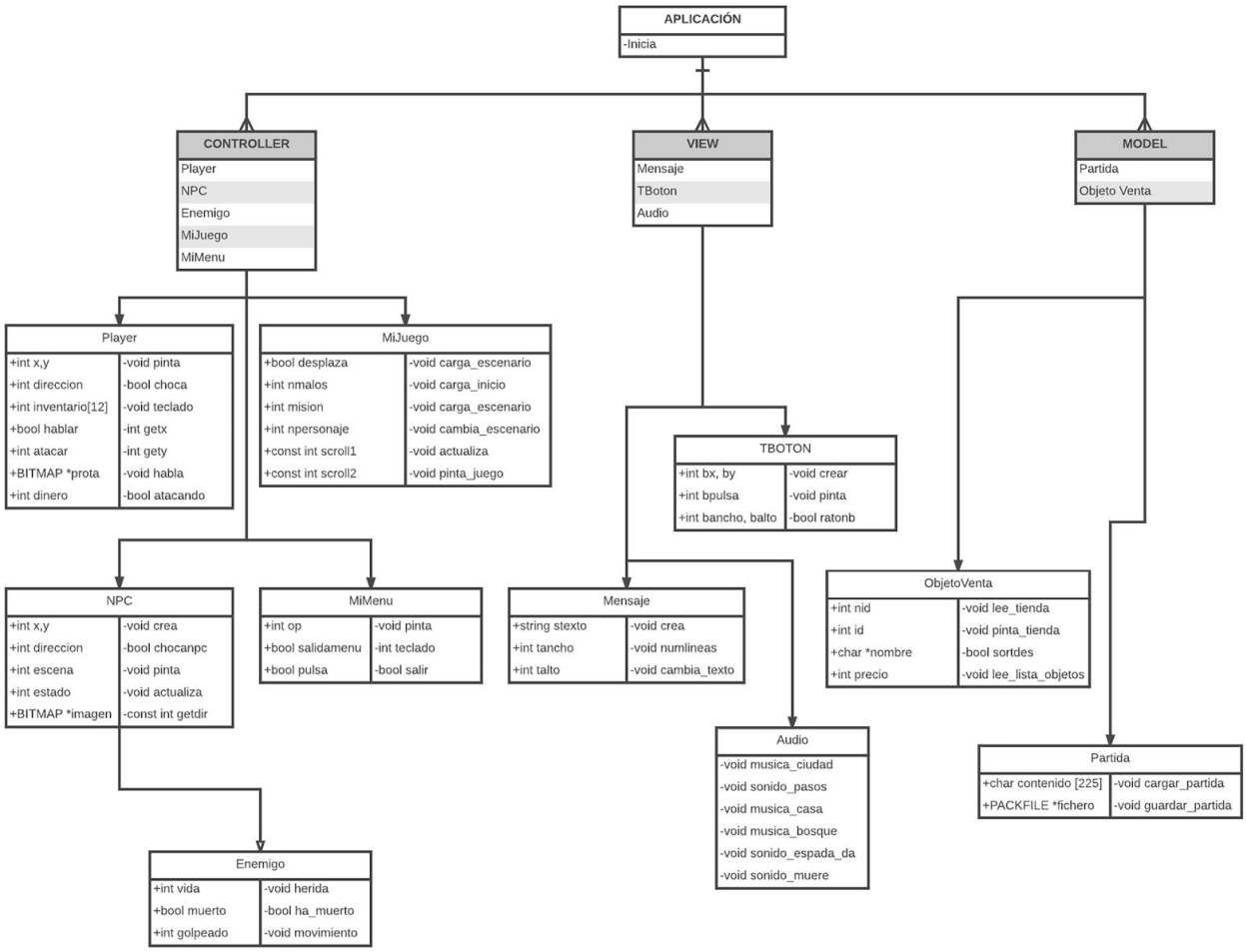
[3] F. Balducci, "Simplest RPG, in Python", Freedom Embedded, 2017. [Online]. Available: https://balau82.wordpress.com/2010/06/28/simplest-rpg-game-in-python/. [Accessed: 14- Nov- 2017].

[4] "C++ text-based RPG", Codereview.stackexchange.com, 2017. [Online]. Available: https://codereview.stackexchange.com/questions/80531/c-text-based-rpg. [Accessed: 06- Nov- 2017].

[5] Estructura de las Aplicaciones Orientadas a Objetos El patrón Modelo-Vista-Controlador (MVC). Madrid: Juan Pavón Mestras, 2017, pp. 5-18.

[6]]"Adobe Creative Cloud", Creative.adobe.com, 2017. [Online]. Available: https://creative.adobe.com/es/products/download/photoshop. [Accessed: 27- Nov- 2017].

v. ANEXO 01


APLICACIÓN
-Inicia
CONTROLLER
Player
NPC
Enemigo
MiJuego
MiMenu
VIEW
Mensaje
TBoton
Audio
MODEL
Partida
Objeto Venta
Player
+int x,y
+int direccion
+int inventario[12]
+bool hablar
+int atacar
+BITMAP *prota
+int dinero
-void pinta
-bool choca
-void teclado
-int getx
-int gety
-void habla
-bool atacando
MiJuego
+bool desplaza
+int nmalos
+int mision
+int npersonaje
+const int scroll1
+const int scroll2
-void carga_escenario
-void carga_inicio
-void carga_escenario
-void cambia_escenario
-void actualiza
-void pinta_juego
TBOTON
+int bx, by
+int bpulsa
+int bancho, balto
-void crear
-void pinta
-bool ratonb
NPC
+int x,y
+int direccion
+int escena
+int estado
+BITMAP *imagen
-void crea
-bool chocanpc
-void pinta
-void actualiza
-const int getdir
MiMenu
+int op
+bool salidamenu
+bool pulsa
-void pinta
-int teclado
-bool salir
Mensaje
+string stexto
+int tancho
+int talto
-void crea
-void numlineas
-void cambia_texto
ObjetoVenta
+int nid
+int id
+char *nombre
+int precio
-void lee_tienda
-void pinta_tienda
-bool sortdes
-void lee_lista_objetos
Audio
-void musica_ciudad
-void sonido_pasos
-void musica_casa
-void musica_bosque
-void sonido_espada_da
-void sonido_muere
Partida
+char contenido [225]
+PACKFILE *fichero
-void cargar_partida
-void guardar_partida
Enemigo
+int vida
+bool muerto
+int golpeado
-void herida
-bool ha_muerto
-void movimiento

v. ANEXO 02

- *MAGO:*

INICIA CUARTO DE CASTILLO, tiene que buscar medicamentos, baja el CASTILLO, sale al BOSQUE a buscar ingredientes, en el bosque se encuentra un OGRO el cual le da 2 opciones:
a) Atacarlo y probar suerte
b) retirarse y dejar morir al rey
CASO A: por suerte el ogro decide que eres alguien en quien confiar al no retroceder y te deja continuar al OASIS
CASO B: el rey se muere y te acusan de traición GAMEOVER.
CASO A CONTINUACIÓN: en el oasis recoges el primer ingrediente, luego para continuar tienes que ir al PUEBLO, ahí encuentras una MENDIGA que te pide limosna:
a) le das limosna
b) no le das limosna
CASO B: la mendiga resulta ser la alcaldesa del pueblo y te dice que no puedes entrar, no puedes comprar la siguiente medicina, regresas, te matan GAMEOVER.
CASO A: la mendiga se ríe y te dice que era una prueba, y te deja pasar, al entrar vas a la tienda y compras otro ingrediente, luego continuas y te encuentras con un COMERCIANTE te da un acertijo y te dice que, si lo resuelves, te dará el último ingrediente te dice ¿Que corre alrededor de una ciudad, pero nunca se mueve?
CASO A: respondes que no tienes ni idea y atacas al hombre, este resulta ser un hechicero poderoso y te mata antes de que tu logres atacar GAMEOVER.
CASO B: respondes una muralla, el comerciante te pide los otros ingredientes y en cambio te da la pócima para salvar al rey, vas al castillo y ganas.

- *GUERRERO:*

INICIA ENTRENANDO EN UN CAMPO con su maestro le dice que tiene que ir a buscar la espada maestra, le da una espada y tiene que escoger entre 2 escudos:
CASO A: escudo de madera, el joven va corriendo y lo matan un trol de un golpe apenas entra al bosque GAMEOVER
CASO B: escudo del maestro, este continua y el escudo lo protege del trol, y continua la historia, el chico va a un PUEBLO y encuentra un anciano que le da a escoger entre sus 2 manos y le dice que en una está el mapa que lo guiara a su zona:
MANO IZQUIERDA: el mapa no está el anciano se ríe y le dice al jugador que el camino es por la derecha, el jugador le hace caso y se muere al llegar a una emboscada GAMEOVER.
MANO DERECHA: el mapa está aquí el jugador va por la izquierda y ve un CASTILLO, al entrar al castillo DENTRO DEL CASTILLO ve una espada al centro, va corriendo y al pisar una trampa cae a una sala donde hay 2 espadas, tiene que escoger 1:
ESPADA DERECHA: era una trampa, caes un piso más abajo y hay púas GAMEOVER.
ESPADA IZQUIERDA: es la espada maestra, empieza a brillar y ganas el juego.

- *PICARO:*

  INICIAS EN TU CUARTO DE UNA CASA POBRE tu madre te dice que bajes al bajar te dice que si no consigues 3K monedas tendrán que dejar el pueblo por culpa de los ladrones, te dice que tienen 2 opciones, conseguir el dinero robándole a los ladrones o escapar:
  ESCAPAN: los ladrones los encuentran en la noche y al ver que intentan huir, los matan GAMEOVER.
  ROBAS A LOS LADRONES: sales al PUEBLO y ves la guarida de los ladrones y entras, al entrar un ladrón te dice que si vas por la derecha irás con el jefe y a la izquierda irás a unirte a los ladrones.
  JEFE: al ver que irás donde el jefe le preguntan si tiene una cita contigo, el jefe dice que no y por intentar huir, mueres en una balacera GAMEOVER.
  LADRONES: el ladrón te acompaña y los ladrones te dicen que tienes que pasar una prueba para ser un ladrón, tienes que matar a tu familia por la deuda, te dan un arma.
  CASO A: vas a tu casa e intentan huir, los ladrones los esperan afuera y los matan a todos GAMEOVER.
  CASO B: vas a tu casa y le dices a tu madre que se esconda, subes a tu cuarto y disparas por la ventana a los ladrones, coges su dinero y te vuelves el sheriff al matar a todos.

- *ELFO:*

  INICIA EN LA VILLA DE LOS ELFOS te dicen que eres el elegido para ser el príncipe, pero tienes que ir al pueblo vecino para presentarte como su príncipe tu padre te dice que vayas con guardia:
  VAS CON GUARDIA: el guardia era un traidor del otro pueblo y te mata para tomar el poder GAMEOVER.
  VAS SIN GUARDIA: le dices que solo te da más confianza, el guardia en un intento desesperado se lanza hacia ti, y falla su intento de asesinato y muere, entonces continuas solo por decisión de todos. Te diriges hacia el pueblo y tienes que pasar por el BOSQUE el cual tiene 2 caminos, uno lleno de pinos y el otro de árboles frutales:
  ÁRBOLES FRUTALES: vas por este camino y coges una fruta, esta fruta resulta venenosa para tu gente y te mueres antes de llegar al pueblo GAMEOVER.
  PINOS: Este camino era el correcto, a lo lejos ves el pueblo vecino y vas corriendo a alcanzarlo, al entrar te encuentras al rey amordazado y no sabes que hacer, hay 2 personas que dicen ser el príncipe de ese reino y no sabes cuál es:
  ATACAS AL DE LA IZQUIERDA: Era el verdadero príncipe, el farsante te ataca a ti y mueres y el reino cae en el mal GAMEOVER.
  ATACAS AL DE LA DERECHA: Era el farsante, se muere y el reino te venera como el príncipe legítimo del reino vecino y hacen una alianza potente, ganas.