Programación orientada a objetos, proyecto 2 **manual técnico**.

Fabian Ernesto Ledezma Ledezma

Juan Acosta López

María del Mar Villaquiran.

Pontificia Universidad Javeriana Cali

Facultad de ingeniería y ciencias

Programación orientada a objetos

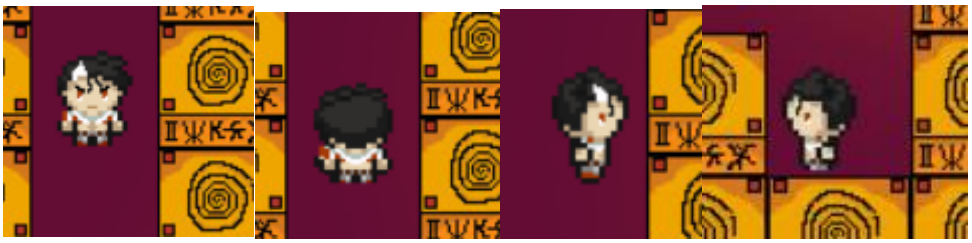
Mayo 2021

```
void Jugador::moverJugador( BITMAP * prota, BITMAP * buffer, int x, int y, int direccion){
    masked_blit(prota, buffer, 0, direccion, x, y, 32,40); //crea blit
}
```

Muestra el jugador en una posición y mirando hacia la dirección que se envía desde donde se llama a la función.

```
void Jugador::teclas(BITMAP * prota, BITMAP * buffer, int * x, int * y, int prohibidoDerecha, int pro
```

Detecta si se esta presionando alguna tecla y llama a la función moverJugador, indicándolo en que posición y en que dirección imágenes ->



```
void verificarInicioBatalla( list <int>, int, int, int *, int *, int * );
```

Verifica que el jugador toque un enemigo, por lo tanto empezaría la batalla y le diría al view que entre a la función ciclo batalla y se mostrará la siguiente pantalla en el videogame

->


```
void View::inicializarAllegro(){
    allegro_init();
```

Crea la ventana en el tamaño de 800 por 700 pixeles.

```
void Jugador::agregarItem(
```

Cuando el enemigo es derrotado la función es llamada y pone aleatoriamente un ítem en el inventario en el espacio que le corresponde

```
void View::ponerFondo(BITMAP * lobbyA, BITMAP
```

Pone el fondo dependiendo la fase en la que esté

```
void View::cicloPrincipal(){
```

Función que contiene el ciclo que controla todo el videoGame

```
//Verifica las colisiones depend
void View::verificarLimites(list
```

Función que le entra como parámetro una lista le limites prohibidos (donde el personaje colisiona) y unas variables por referencia que cambian a 1 si no es posible moverse hacia ese lado, por lo tanto parece que el personaje choca.

```
list<int> encontrarLimites( int m
```

Función que recibe una matriz y retorna una lista de limites donde el personaje no puede pisar, puesto que hay un bloque.

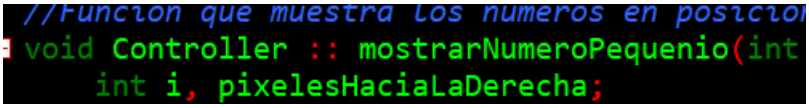
```
//Funcion que retorna la lista de pociones en pixeles de los enemigos de cada matriz que
list<int> Controller:: encontrarPosicionesEnemigos( int matrizNivel[10][14] ){
    int i, j;
```

Lo mismo que la anterior, pero retorna la lista de donde hay enemigos.

```
//Funcion que verifica la fase del juego, Fase = 1
void Controller::verificarFase(int * x, int * y, i
```

Mira si el personaje tiene que pasar a otro nivel, en caso de que si, cambia la variable fase, por lo tanto, cambiará el fondo.

Función que le entra un número, y una dirección, y pone en pantalla el numero en esas direcciones.



Lo mismo que el anterior pero con números pequeñitos.