Programación de Video Juegos

Universidad ORT, Montevideo Uruguay

Agustín Pazos - 185494

Martín Long - 184178

Fernando Acosta y Lara – 186601

Índice

Descripción

El juego realizado es un juego para más de 1 jugador, por turnos. En el cual tenés como objetivo eliminar a tu oponente antes que él te elimine.

Antes de empezar, se elige un mapa con el generador y se selecciona el lugar donde está cada jugador.

Cada jugador empieza con una cantidad definida de vida y pierde cuando esta llega a 0. Para esto cuenta con distintos tipos de armas. Cada jugador según su estrategia sabe cuál le conviene en cada momento.

Los jugadores en su turno pueden moverse por el mapa, está en cada uno asegurar el disparo o quedar en mejor posición para recibir el del rival,hay que recordar que una vez que disparas (o se termina tu tiempo) no puedes esconderte.

¿Cómo jugar?

El juego es simple, para jugar solo precisamos saber que nos movemos con las flechas, izquierda y derecha para dirección y para arriba para saltar.

Para efectuar disparos, apuntamos con el cursor del mouse y disparamos haciendo click.

Las armas se pueden cambiar mientras sea tu turno utilizando los botones numéricos 1 2 y 3. Por el momento se cuentan con solo 3 tipos de armas, pero probablemente en un futuro cuando crezca se podrá jugar con más.

Puedes atacar a tus enemigos tanto directamente, como al campo cerca de ellos. Según la modalidad de juego elegida, puedes aspirar a hundirlos en el “agua” (El fondo de la pantalla).

Requerimientos implementados en la versión final.

1) Elegir mapa en el generador de mapas.
2) Ubicar al jugador en el mapa.
3) Moverse por el mapa.
4) Saltar.
5) Disparar en dirección al mouse.
6) Cambiar de arma.
7) Disminuir vida cuando te disparan según arma.
8) Explotar mapa cuando es golpeado.
9) Disminuir vida por cercanía a la explosión.
10) Manejo de turnos.
11) Timer para el turno.
12) Muerte del jugador cuando llega a vida 0.
13) Mostrar jugador ganador.

Detalles a agregar sobre la solución.

Se utilizaron diferentes estrategias para facilitar la programación de las funcionalidades. El trabajo se realizó pensando en que el juego puede crecer.

A continuación, vamos a dar algunos ejemplos que nos pareció relevante documentar como fueron realizados.

Estado del jugador

El jugador a lo largo de la continuidad del tiene varios estados.

Caminando a la derecha, a la izquierda, saltando, quieto, esperando turno.

En el momento no son demasiados estados, pero sabemos que si crece el juego muy probablemente haya que crear nuevos estados. Por lo que decidimos manejarlo utilizando el patrón de diseño State Pattern.

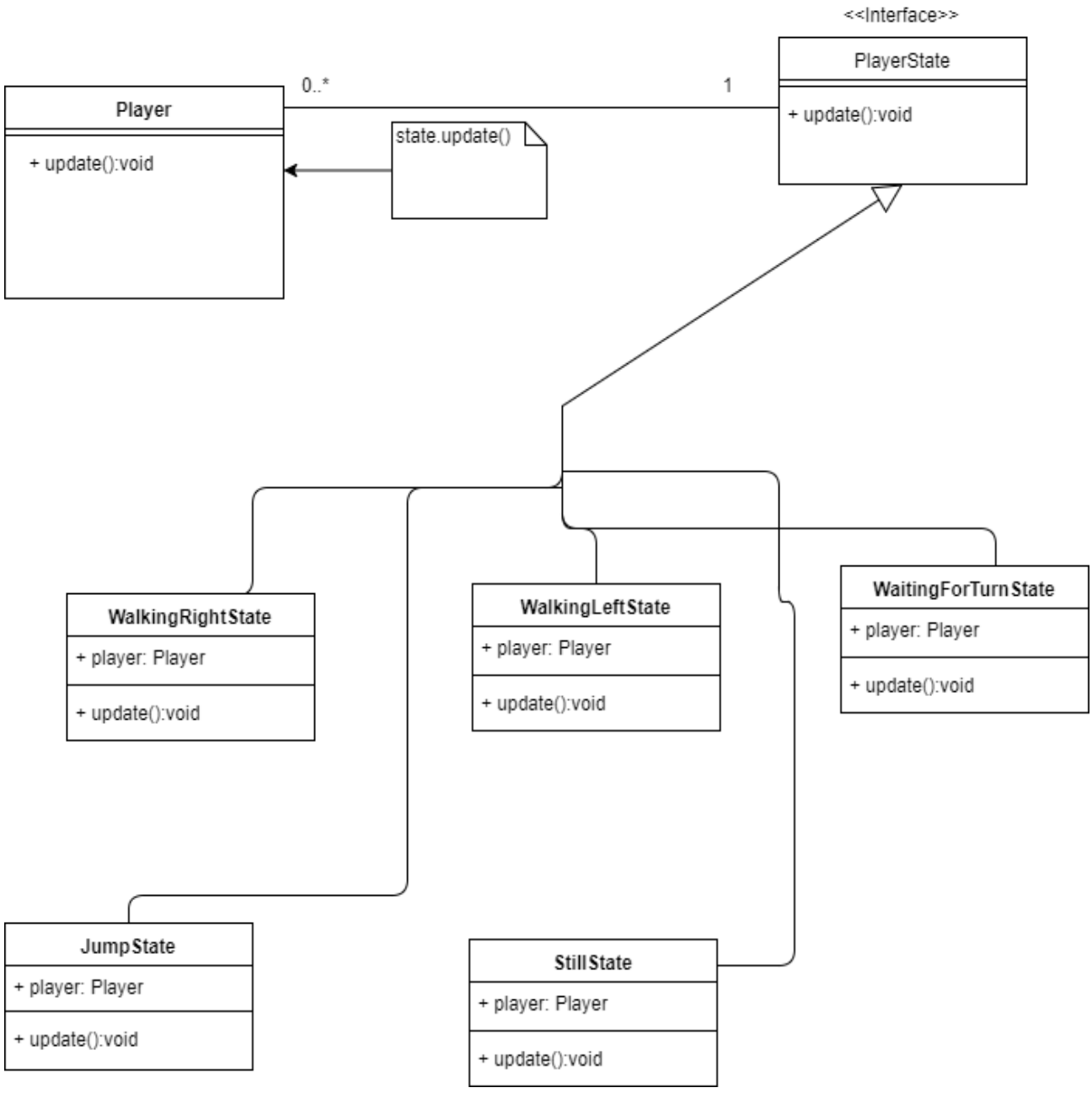
Para esto, contamos con una interfaz PlayerState con un método update, cada estado del jugador implementa esta interfaz. Esto nos permite, en caso de necesitar introducir un estado nuevo poder hacerlo sin necesidad de modificar, respetando lo esperado por el principio SOLID open-closed.

También obtenemos un código más claro y menos acoplado al separar el comportamiento del jugador, y que sea responsabilidad del estado del mismo.

También vale aclarar que el pasaje de estados se realiza desde los estados concretos, es decir estos conocen al jugador y son lo que “setean”.

Por último, cuando hablamos del estado del jugador, queremos aclarar que no es lo mismo que el FlixelState utilizado para manejar el estado del juego por ejemplo PlayState.

A continuación, podemos ver un diagrama de clases de cómo se modelo el patrón state en la solución,


<<Interface>>
PlayerState
+ update():void
0..*
1
Player
+ update():void
state.update()
WalkingRightState
+ player: Player
+ update():void
WalkingLeftState
+ player: Player
+ update():void
WaitingForTurnState
+ player: Player
+ update():void
JumpState
+ player: Player
+ update():void
StillState
+ player: Player
+ update():void

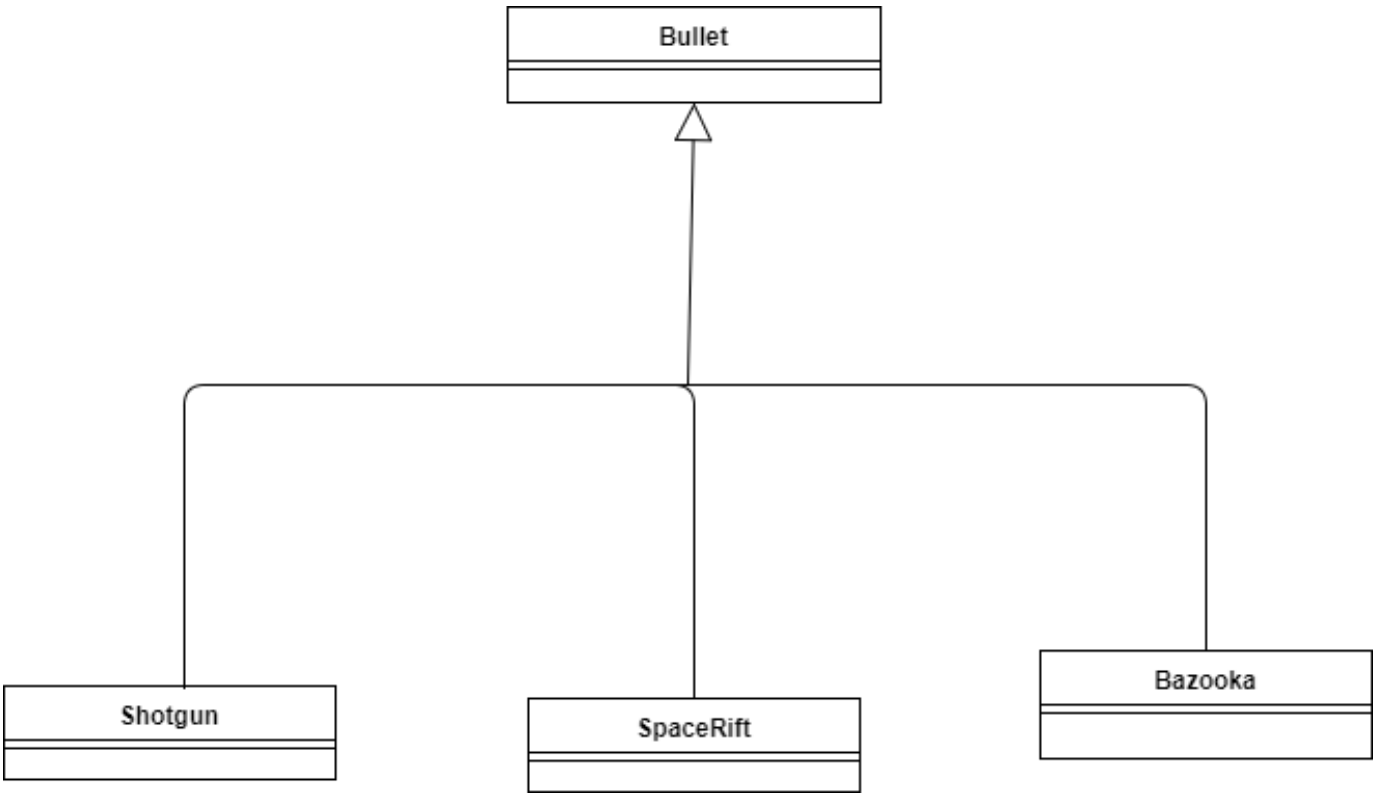
Diferentes tipos de balas

También nos pareció pertinente documentar como se manejaron los distintos tipos de balas, las características de cada una están en una clase separada, y mediante un enum llamado BulletType se maneja el tipo.

Esto también se podría haber desacoplado, favoreciendo los principios de diseño. Pero se realizó así por el motivo de no aumentar la complejidad.

De todas formas, si en un futuro se quiere aplicar algún patrón o estilo para el comportamiento no se deberían hacer demasiados cambios.

A continuación, podemos ver un diagrama de cómo funciona la jerarquía de las Bullets.



Manejo de turnos

Otra cosa que nos pareció bueno comentar, es que la responsabilidad de manejar los turnos está separada, en un Singleton (No tiene sentido tener más de una instancia de manejador de turnos pero si guardan estado, por lo que no nos convenía hacerlo estático). El turno, si bien tiene ciertas funciones desprolijas, está pensado para poder agregar más jugadores en un futuro, manteniendo a los mismos en una lista circular. Tener una cantidad N de jugadores, es algo que no se implementó para esta entrega, pero se tiene en cuenta que en caso de crecer el juego va a ser implementado.

Configuración del juego

La configuración del juego, se especifica antes de empezar la partida, en esta podemos definir si jugamos en modo “drown in wáter” que quiere decir que si toca el fondo el jugador muere, y también el límite de vida de cada jugador.

Para esto se tiene un Singleton a lo largo del juego, guardando estas configuraciones.

Detalles Generales

En nuestro código hay algunas medidas como por ejemplo la posición inicial de las balas que están realizadas con valores en base al ancho y alto del jugador, pero no son calculados dinámicamente. Por lo que un cambio en el tamaño del jugador, afectaría tener que cambiar estos valores y sabemos que eso puede generar problemas en algún momento.