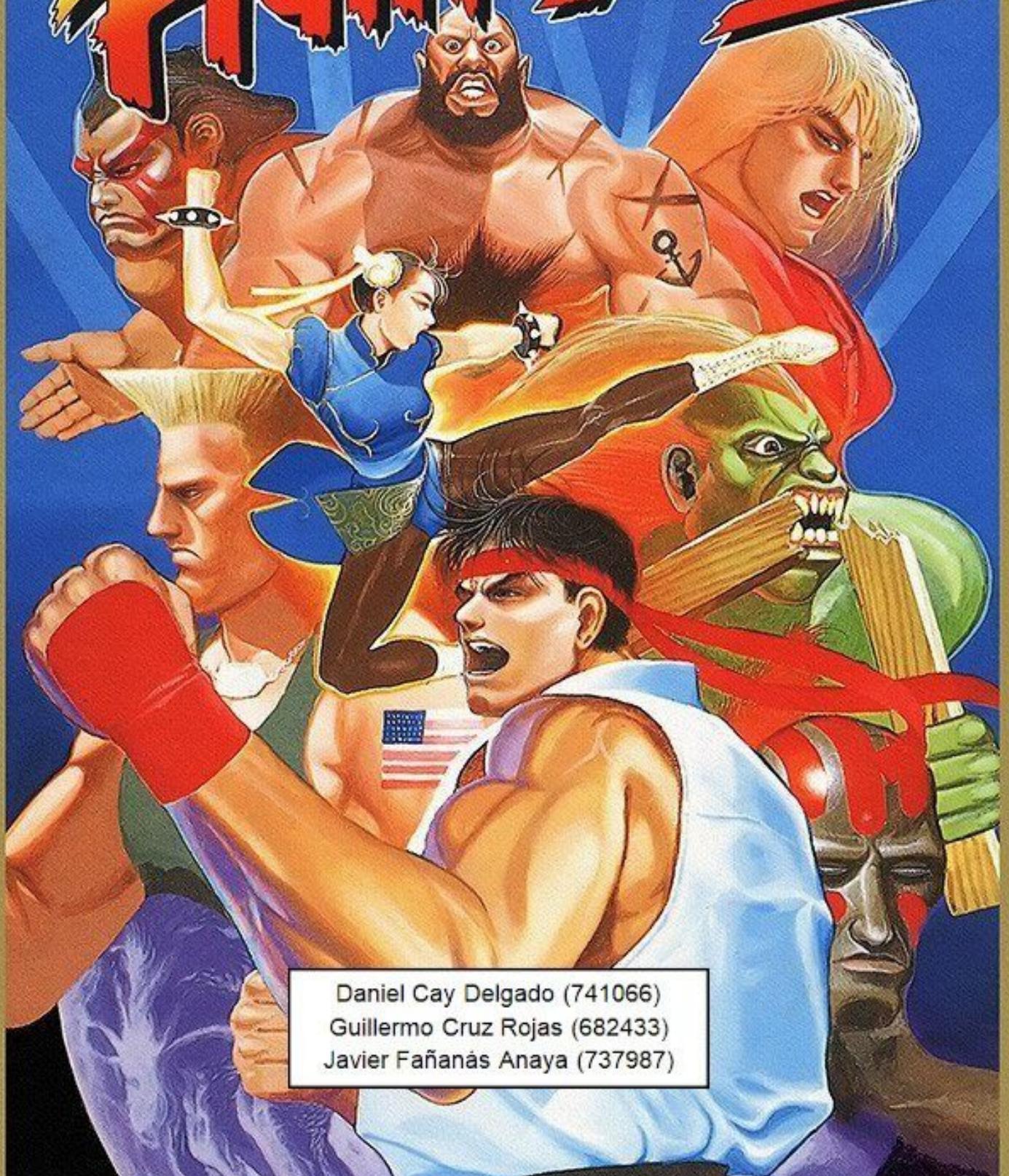


STREET FIGHTER III



Daniel Cay Delgado (741066)
Guillermo Cruz Rojas (682433)
Javier Fañanás Anaya (737987)

ÍNDICE

CONCEPTO DE JUEGO	2
REPARTO DE ROLES	3
CUESTIONES INICIALES DE IMPLEMENTACIÓN	3
ESTRUCTURA SOFTWARE	4
Diagrama de Clases:	4
Diagramas de Paquetes:	5
GAME LOOP	6
LÓGICA DEL JUEGO	6
GRÁFICOS	15
LUCHADORES:	15
ESCENARIOS:	16
INTRO:	18
AÑADIR O MODIFICAR ATAQUES Y ANIMACIONES	22
ANIMACIONES:	22
ATAQUES:	24
PERSONAJES Y ESCENARIOS	25
BLANKA:	25
CHUN LI:	26
RYU:	28
CÁMARA	29
PELEA	30
MOVIMIENTOS Y ANIMACIONES:	30
ATAQUES:	31
COMBOS:	34
HUD:	35
LÓGICA DE LA PELEA:	36
GUARDADO DEL JUEGO	37
MENÚ OPCIONES	38
MÚSICA Y SONIDOS	39
SISTEMA DE PUNTOS	40
INTELIGENCIA ARTIFICIAL	41
BACKDOOR	46
PROBLEMAS ENCONTRADOS Y SOLUCIONES	46
TAREAS RESTANTES	47

CONCEPTO DE JUEGO

Se trata de un juego de lucha 1 vs 1 en 2D. Si se juega contra otra persona en el modo “VS BATTLE”, el objetivo es ganar la pelea en cuestión, mientras que si se juega contra la CPU en el modo “GAME START”, el objetivo consiste en ganar las 2 peleas de las que se compone el modo historia logrando así terminar el juego, y pudiendo disfrutar de la cinemática disponible por cada personaje cuando el juego finaliza.

Para ganar una pelea, se deberán ganar 2 rondas antes de que lo haga el rival, puesto que es una pelea al mejor de 3, y la manera de vencer en una ronda consiste en reducir la vida de tu rival a 0, o en tener más vida que el enemigo cuando el tiempo llegue a 0.

Los puntos obtenidos en el combate juegan también un papel muy importante, ya que suponen una forma de constatar que se ha sido mejor que el rival en el modo “VS BATTLE”, y una forma de que quede constancia de lo bien que se ha jugado en el modo historia (“GAME START”). La forma de obtener dichos puntos consiste en golpear al rival, es decir, cada golpe, tiene su recompensa en forma de puntos. Además, si se gana una ronda, se recibirá un bonus determinado por lo rápido que se acabe con el enemigo y con la vida que no se haya perdido. Una vez se pierda en el modo historia o se termine, se podrá guardar la puntuación acumulada a lo largo de todo el modo historia introduciendo 3 letras que identificarán al jugador, de manera que todos sepan de lo que se ha sido capaz. Además, se podrá consultar el ranking de puntuaciones siempre que se quiera desde el menú en la opción “RANKING”.

¿Cómo se puede ser mejor que el rival?. La clave está en los ataques que se usen, ya que dependiendo de la situación unos serán más convenientes que otros. Tampoco se debe olvidar la defensa, protegerse y esquivar los golpes saltando y agachándose son técnicas que también pueden decidir de qué lado cae un combate. Además, el personaje seleccionado juega un papel fundamental, ya que según el estilo de juego del jugador, este se sentirá más cómodo con un personaje u otro. Comprobar cuál se adapta mejor a tus necesidades, si Blanka, Ryu o Chun Li solo depende de ti.

Todo esto va acompañado de un menú de opciones (“OPTION MODE”) que permiten hacer del juego la mejor experiencia posible adaptando el volumen, cambiando la dificultad del modo historia en entre fácil, normal, difícil y progresiva (en esta última la dificultad se va incrementando por cada pelea ganada), así como cambiar los controles al gusto del jugador y cambiar la resolución de la pantalla. Este menú puede ser accedido también durante la pelea, dando así aún más flexibilidad al jugador.

REPARTO DE ROLES

Inicialmente, debido a la inexperiencia de los 3 en el desarrollo de videojuegos, se decidió crear el esqueleto del juego conjuntamente, para una vez claras las bases, poder centrarse cada uno en el desarrollo de aspectos más específicos.

Javier: Desarrollo del combate y test del juego

Guille: Audio del juego, desarrollo de la intro y de diversos menús.

Dani: Desarrollo de menús y de la IA.

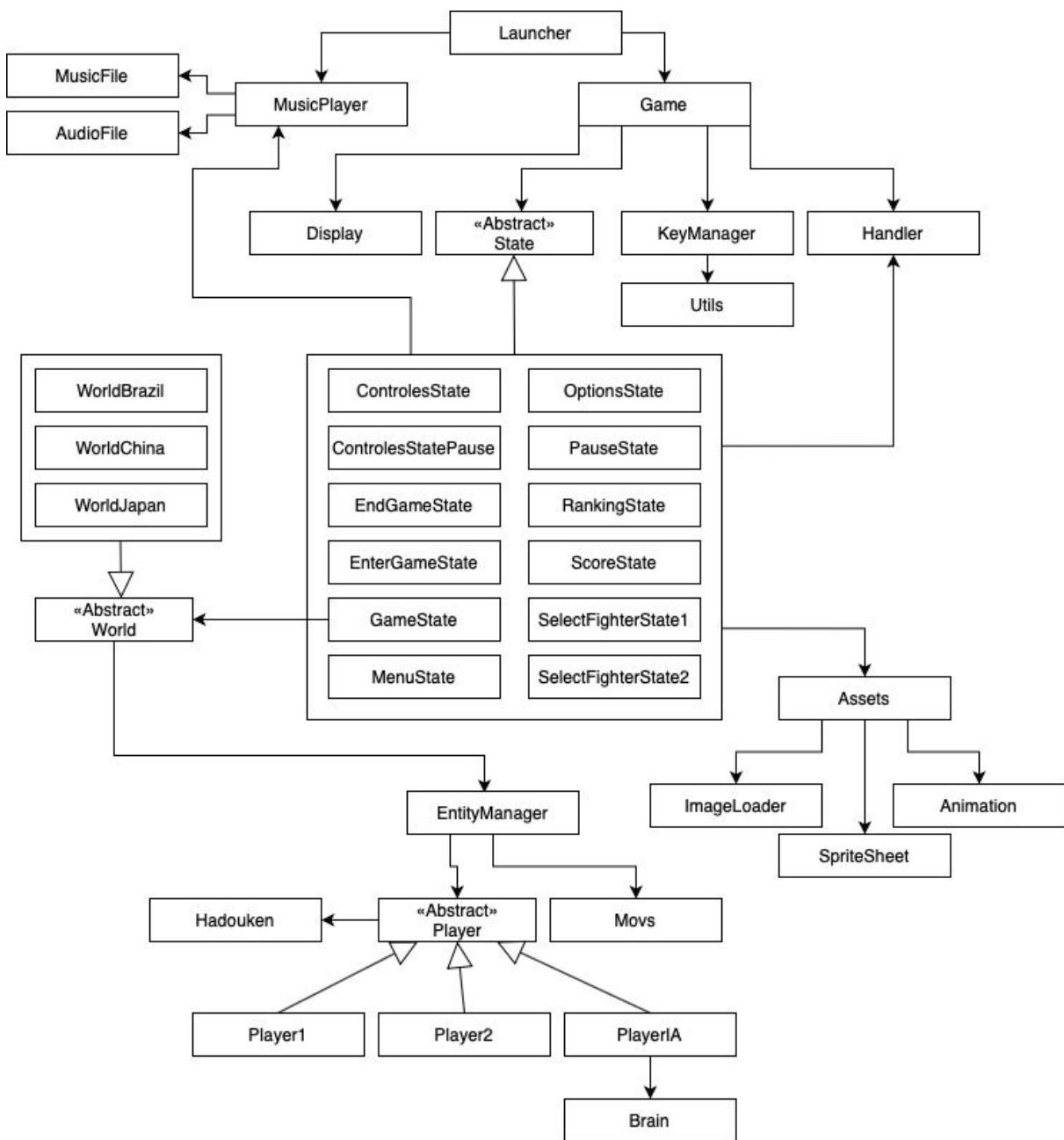
CUESTIONES INICIALES DE IMPLEMENTACIÓN

El juego se ha desarrollado para PC, y es compatible con Windows, MAC y Linux. La implementación se ha realizado en Java puesto que los 3 integrantes del equipo nos sentimos muy cómodos con este lenguaje, además de porque ha sido recomendado por parte del profesorado de la asignatura.

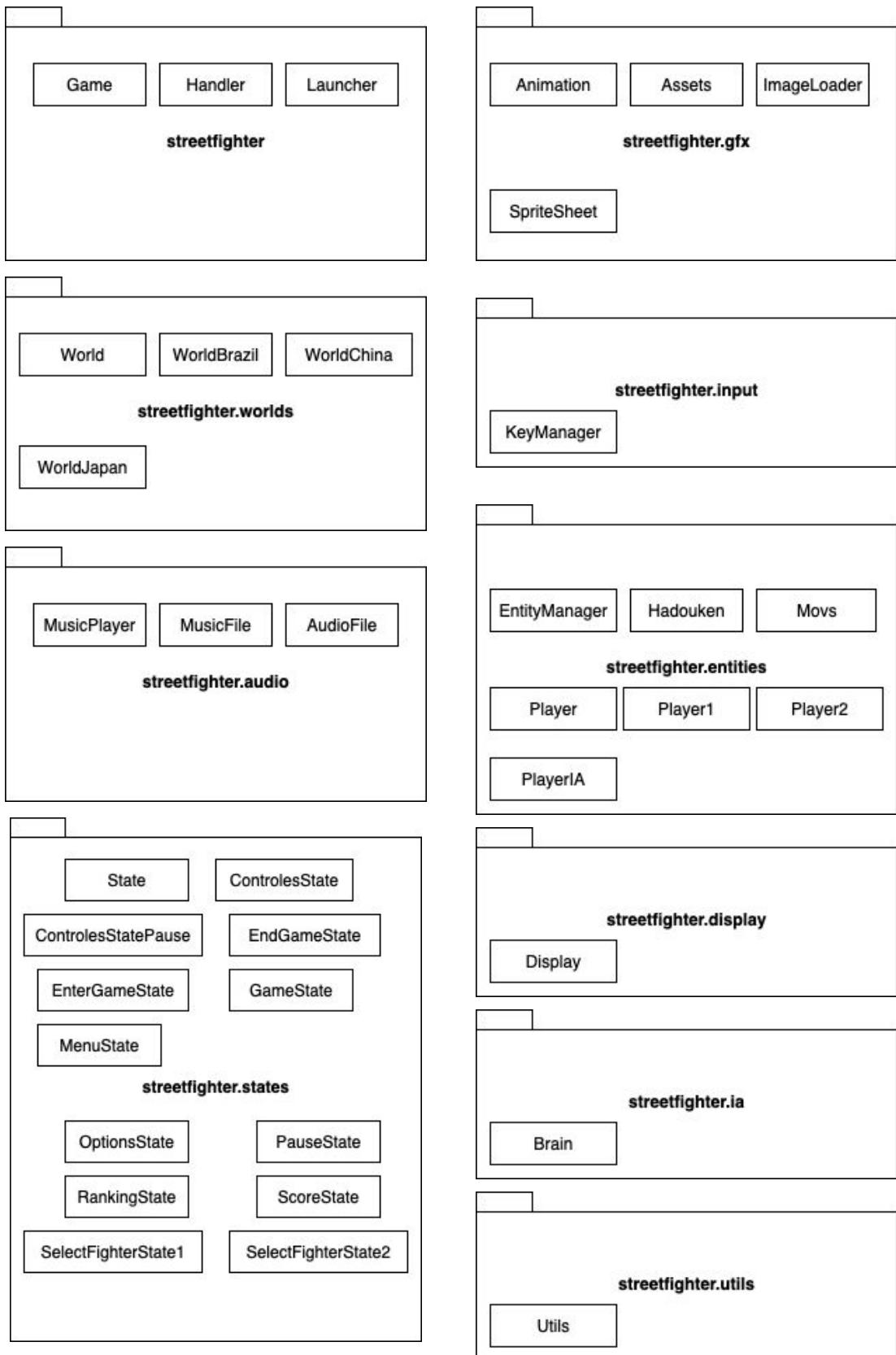
El juego se reproduce a 60 fotogramas por segundo. Por defecto, la resolución es 720p, aunque también se puede elegir la resolución pantalla completa o 480p.

ESTRUCTURA SOFTWARE

Diagrama de Clases:



Diagramas de Paquetes:



GAME LOOP

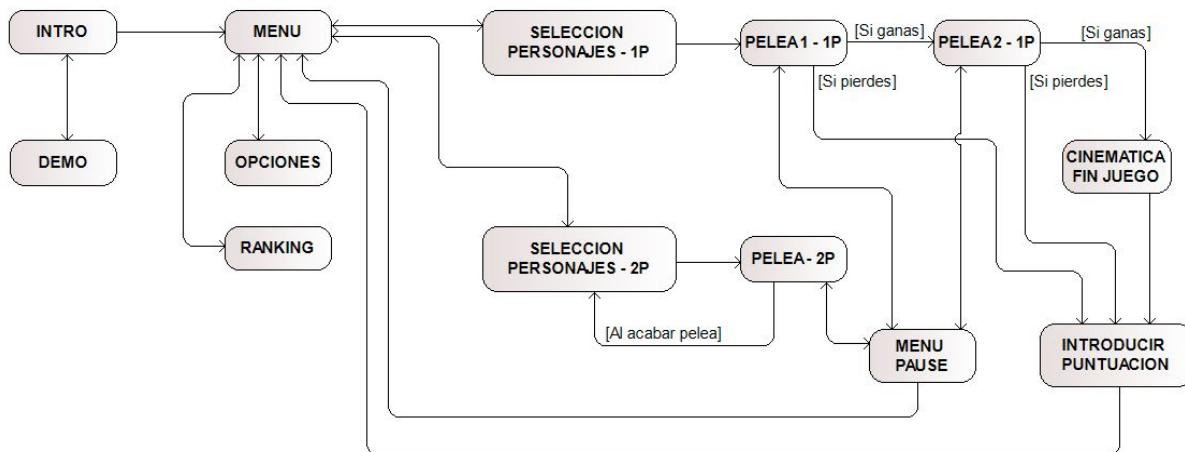
Está basado en el siguiente video:

<https://www.youtube.com/watch?v=vFRuEqEdO9Q&list=PLah6faXAgguMnTBs3JnEJY0shAc18XYQZ&index=4>

Es la estructura en torno a la que gira el videojuego, y se basa en la fase de tick y la fase de render. En la primera se actualizan las variables que haga falta, mientras que en la segunda se “dibuja” en la pantalla el resultado de actualizar dichas variables. Además, se controla que en cada segundo ocurran 60 veces ambas fases, es decir, el juego funciona con 60 fps.

LÓGICA DEL JUEGO

La implementación del juego se basa en una máquina de estados. En la clase Game, que es la que implementa el bucle del juego, se actualizan las variables y se renderiza la escena del estado correspondiente. A continuación se muestra la máquina de estados del juego:



El estado en el cual está el juego durante una pelea, está implementado en la clase GameState, que se encarga de actualizar las variables necesarias para la pelea y renderizar el escenario y los personajes que están peleando.

A continuación se muestran las distintas pantallas que forman el flujo principal del juego:

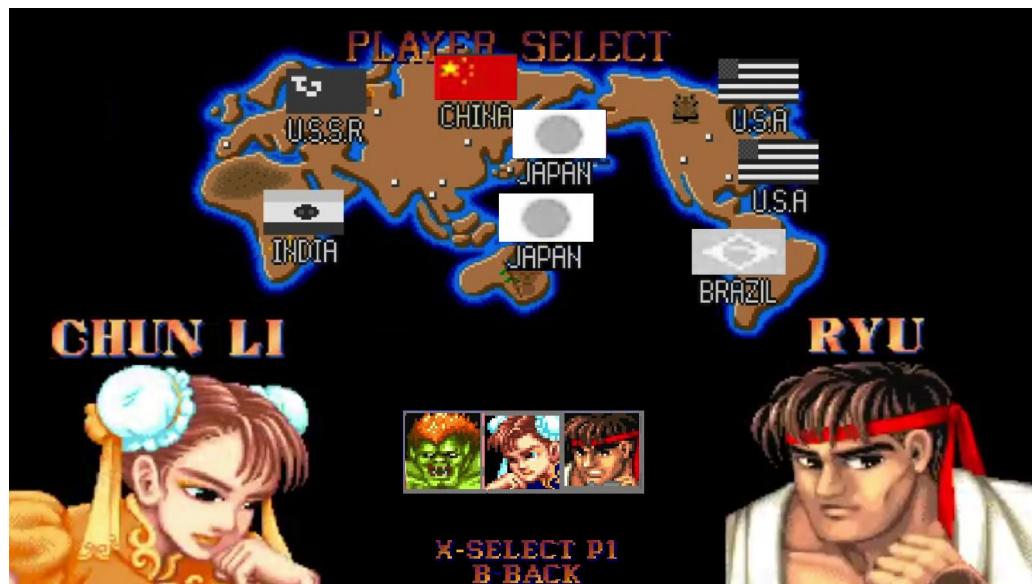
- **Intro:** Animación de dos luchadores peleando ante una multitud de gente, para posteriormente mostrar el logo del videojuego. Si tras esperar un rato no se pulsa el botón enter, comenzará el modo demo, o de lo contrario se pasará al menú del juego.



- **Menú principal:** Aparece el título del videojuego y se puede seleccionar el modo historia (*GAME START*), el modo pelea para dos jugadores (*V.S. BATTLE*), el menú opciones, mostrar el ranking de puntuaciones, o cerrar el juego.



- **Selección de personaje 1 jugador:** Si se elige el modo historia, se muestra la siguiente pantalla. Consiste en seleccionar el personaje deseado para comenzar la aventura.



- **Selección de personaje 2 jugadores:** Si se elige el modo V.S., se muestra la siguiente pantalla. Consiste en que cada jugador escoja su personaje deseado para comenzar a pelear el uno contra el otro.



- **Menú opciones:** Si en el menú del juego se escoge entrar al menú opciones, se muestra la siguiente pantalla. En ella se pueden personalizar las opciones del juego.



- **Menú controles:** Si en el menú opciones se escoge la opción CONTROLS, se muestra la siguiente pantalla, que permite personalizar los controles y además muestra información sobre las combinaciones de teclas que se pueden utilizar durante la pelea si se escoge la opción MORE INFO, así como funciones de otras teclas como la P para abrir el menú pause durante la pelea o ESC para salir del juego en cualquier momento.



CONTROLS

SPECIAL ATTACK	CROUCH > WALK LEFT/RIGHT > PUNCH
CROUCH PUNCH	CROUCH + PUNCH
CROUCH KICK	CROUCH + LOW KICK
AIR PUNCH	JUMP > PUNCH
BLOCK	WALK BACK
CROUCH BLOCK	CROUCH + WALK BACK

B BACK

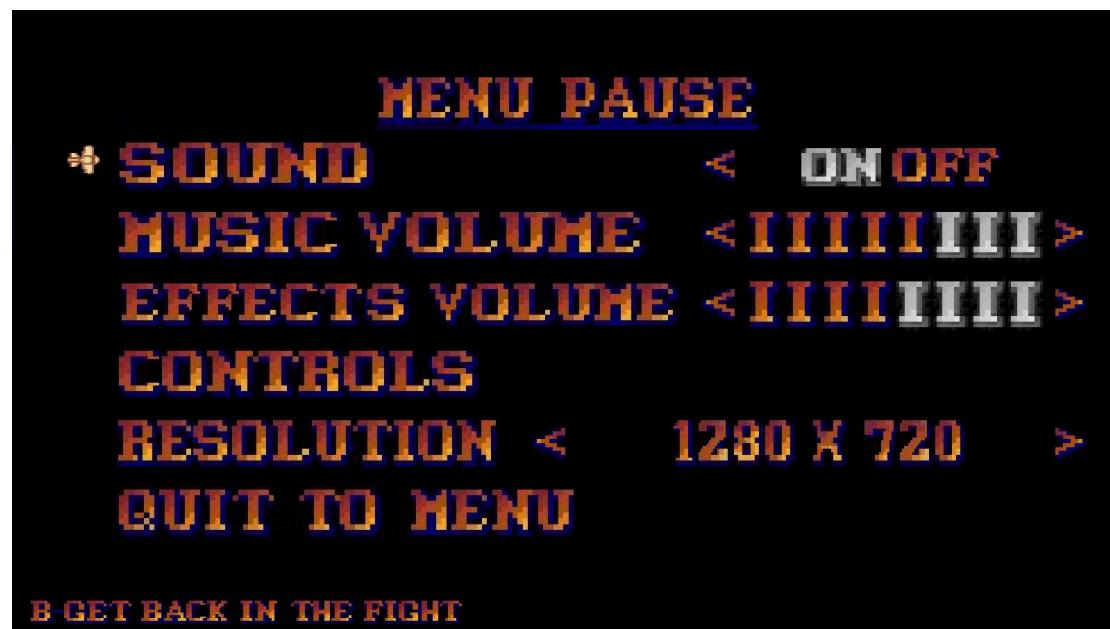
- **Pantalla VS:** Una vez que se tienen los personajes seleccionados para comenzar a luchar, tanto en el modo historia como en el modo V.S., aparece la siguiente pantalla con los dos personajes seleccionados.



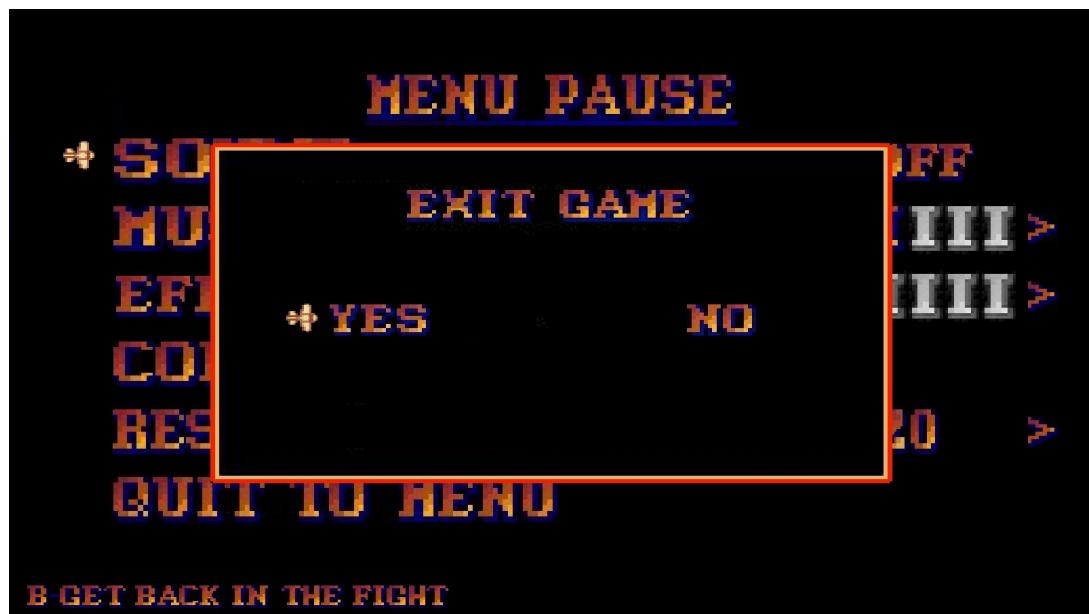
- **Pantalla de pelea:** Tras seleccionar los personajes y pasar la pantalla VS, comienza la pelea. El modo demo es igual a esta pantalla solo que aparece un mensaje de *PRESS ENTER* para salir de dicho modo.



- **Pantalla pause:** Durante cualquier momento en la pelea, si se pulsa la tecla P, se entra en el menú pause, donde se puede también cambiar algunas opciones del juego.



- **Tecla ESC:** En cualquier punto del juego, si se presiona la tecla *ESC*, aparece el siguiente mensaje para confirmar si se quiere salir del juego.



- **Pantalla resultado pelea:** Tras finalizar una pelea, se muestra una pantalla en la que la cara del personaje perdedor aparece dañada.



- **Pantalla intermedia en el modo historia:** Durante el modo historia, si se gana una pelea, se muestra la pantalla con el mundo y los países, mostrando el siguiente destino al que viajar para pelear.



- **Pantalla final tras ganar todas las peleas:** Si se consigue ganar todas las peleas del modo historia, se muestra una cinemática que depende del personaje que se haya escogido.





- **Pantalla introducir puntuación:** En el modo V.S., cuando finaliza una pelea, se vuelve al modo de selección de personajes, pero en el modo historia, tanto como si se pierde una pelea y no se puede seguir avanzando como si se ganan todas las peleas y se llega al final del juego, se le pide al jugador introducir su nombre para guardar la puntuación que ha conseguido.

ENTER YOUR INITIALS			
POS	SCORE	FIGHTER	NAME
5	50000	RYU	CCC
6	45000	CHUN LI	DDD
7	23300	CHUN LI	A--
8	20000	BLANKA	EEE
9	10000	BLANKA	FFF

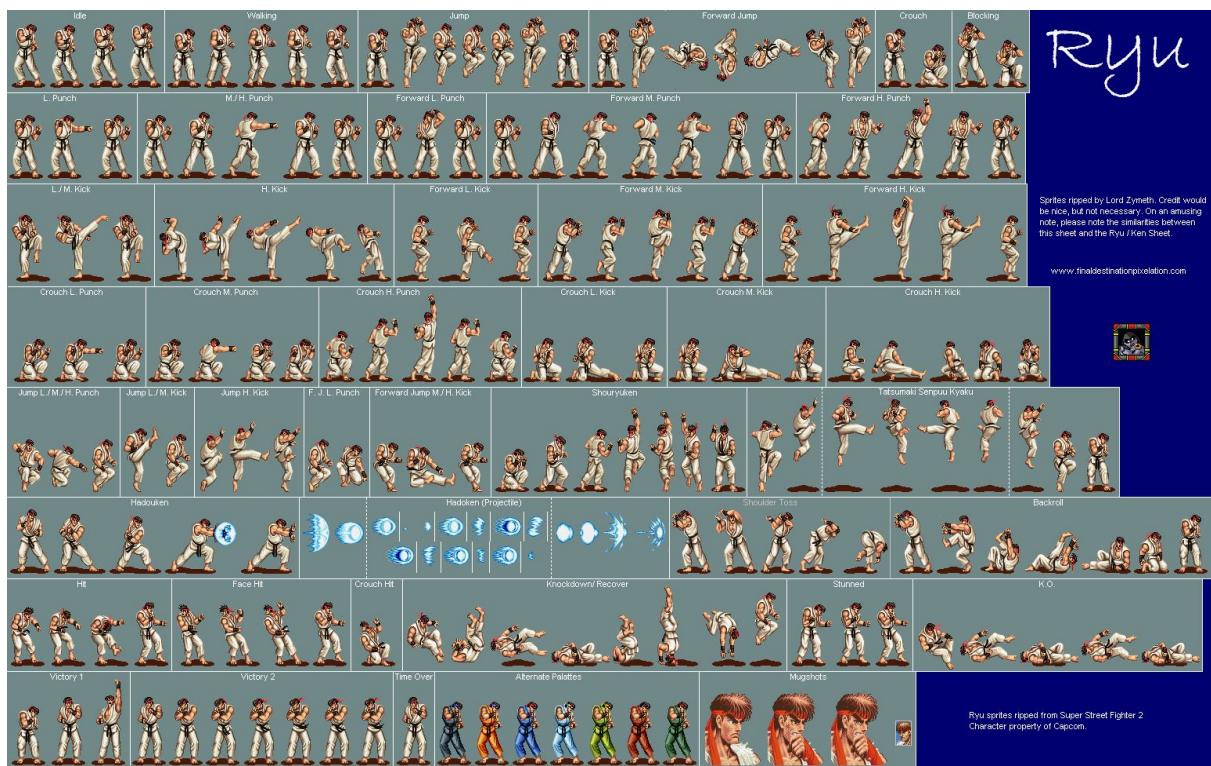
Tras introducir la puntuación se vuelve al menú principal del juego.

GRÁFICOS

En este apartado se va a explicar cómo se han diseñado los gráficos del juego, tanto el diseño de los personajes junto a sus animaciones y el diseño de los escenarios. Los sprites tanto de los personajes como de los escenarios, se obtuvieron del siguiente link: <https://www.sprites-resource.com/snes/supersf2/>

LUCHADORES:

Cada luchador está formado por un conjunto de animaciones. Cada animación almacena un conjunto de sprites y estadísticas específicas (Velocidad de la animación, tamaño de cada sprite...). Cómo se cargan estas animaciones se detalla en el siguiente apartado. En la siguiente captura podemos ver el conjunto de animaciones de RYU:



A continuación se puede observar los sprites que forman el puñetazo de RYU:



La animación se forma creando un vector de sprites, en los que además de la imagen se almacena el tamaño de cada frame de la animación (Por ejemplo, cuando está el puño estirado es una imagen más ancha que las otras dos) y la velocidad de la animación.

ESCENARIOS:

Cada personaje tiene su propio escenario. La lógica del juego es que durante una pelea, se carga el escenario del enemigo, en el modo historia, o el escenario del jugador 2, en el modo V.S. Los tres escenarios elegidos han sido Brasil para Blanka, China para Chun Li y Japón para Ryu, al igual que en el juego original.

A continuación se muestra el sprite sheet utilizado para construir el escenario Brasil, el de Blanka:



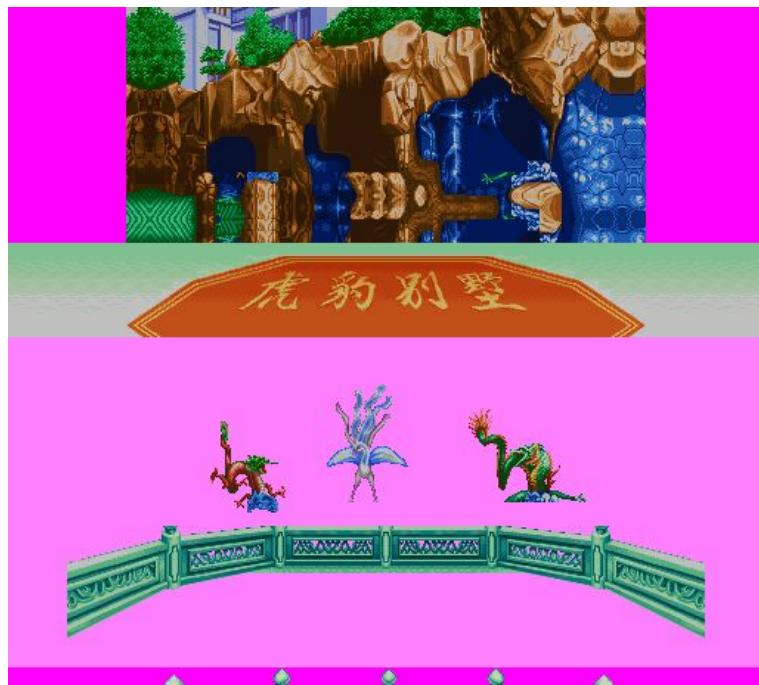
Gracias a que este sprite sheet incluía público, se ha podido incluir en el escenario con su correspondiente animación, en el que la gente aparece tomando fotos o animando.

Para proveer una sensación de profundidad, los elementos del escenario se han incluido a capas, es decir en la primera capa (la más cercana al espectador) se ha incluido el suelo, el

árbol en el que se encuentra la serpiente, y la casa en la que hay una gran pez colgando. En la segunda (la intermedia) se ha incluido la cabaña que se encuentra construida sobre el agua. En la última, se ha incluido el cielo y el agua. Así, cuando los personajes se mueven horizontalmente y es necesario que la cámara se mueva con ellos, se puede crear un efecto de paralaje, moviendo las distintas capas a distintas velocidades. Por ejemplo, la capa más cercana al espectador se mueve a mayor velocidad, la capa intermedia se mueve un poco más lenta, y la capa final aún más lenta, creando así el efecto de profundidad esperado.

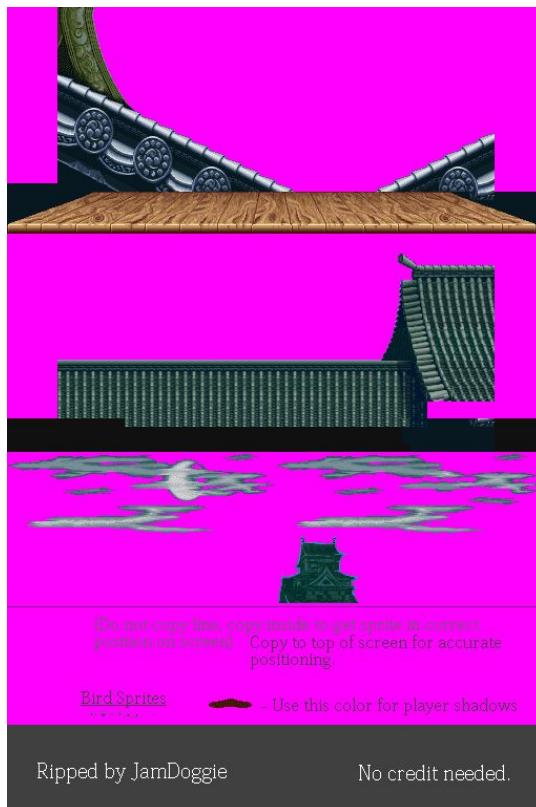
Lo mismo ocurre cuando los personajes saltan. La capa más cercana se mantiene intacta, pero a las otras dos se les da un pequeño *offset* para que se desplacen verticalmente con la cámara, y así dar una sensación de que en la capa más cercana se está elevando realmente la cámara.

A continuación se muestra el sprite sheet utilizado para el escenario China, el de Chun Li:



En este caso, la capa más cercana consiste en el suelo y la valla, y la segunda capa consiste en el fondo con las cascadas y los tres elementos marrón, azul claro y verde que aparecen en el medio del sprite sheet. De igual manera que en el escenario de Brasil las capas se mueven a distintas velocidades, tanto horizontalmente como verticalmente.

A continuación se muestra el sprite sheet utilizado para el escenario de Japón, el de Ryu:



En este caso se cuenta con cuatro capas. La primera y más cercana, el suelo junto a esos tejadillos diagonales que aparecen. La segunda, el tejadillo horizontal junto a el tejado de casa (en el medio del sprite sheet). La tercera, la pequeña mansión que aparece justo debajo de las nubes, y por último, la cuarta consiste en las nubes con la luna. Como cielo, ya que este sprite sheet no lo traía consigo, se creó con la herramienta GIMP un fondo que consiste en un degradado que comienza en un azul oscuro arriba del todo y se va suavizando el color progresivamente, dando así la sensación de oscuridad propia de una pelea por la noche.

INTRO:

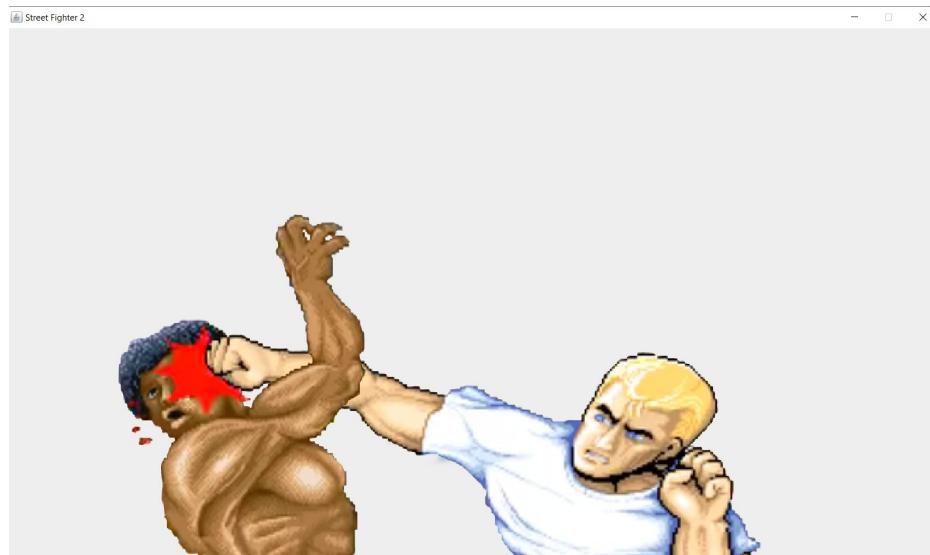
Para crear la escena que aparece nada más arrancar el juego, se ha buscado asemejarse lo máximo posible a la intro del juego original. Buscando por Internet no se encontró ningún sprite sheet por donde poder empezar, por lo que se optó por realizar capturas de pantalla de la intro original y junto a la herramienta GIMP recortar y quedarse con los elementos necesarios. Por ejemplo, para crear a los dos luchadores que aparecen, se recortó sus siluetas con esta herramienta y posteriormente se les dotó de animación para que pareciera que están en movimiento. A continuación se muestra las siluetas de ambos:



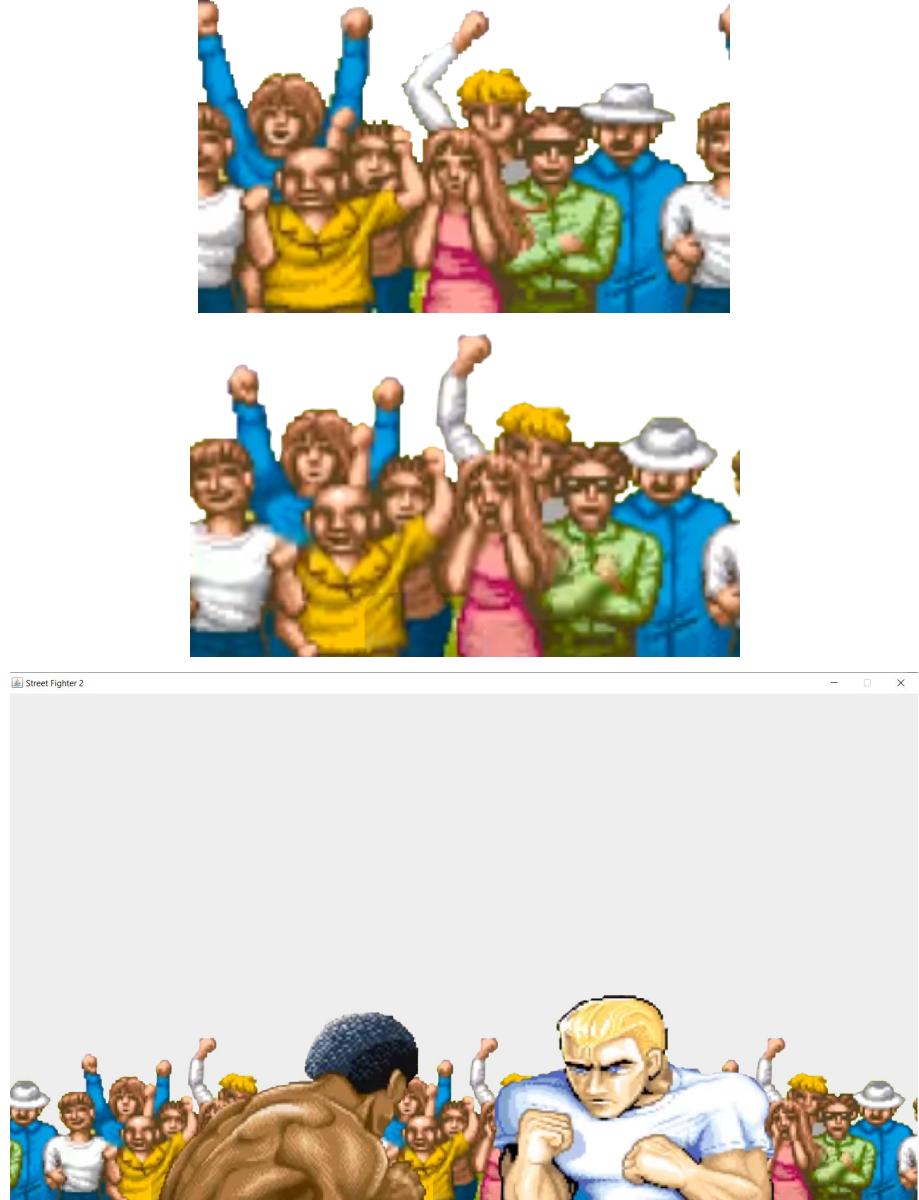
El mismo proceso se realizó para construir el momento en el que el luchador rubio le da un puñetazo al otro. Se realizaron capturas de pantalla de la secuencia original, y se recortaron los elementos necesarios.



Tras conseguir los elementos necesarios, el siguiente paso fue colocarlos sobre la escena y que quedase una secuencia homogénea, lo que llevó varias pruebas.



A continuación se buscó crear al público que aparece tras los luchadores y que además tiene un pequeño movimiento, por lo que se capturaron los momentos justos y posteriormente se reconstruyeron con GIMP las partes de público que pudiera faltar.



A continuación se añadieron los árboles que aparecen detrás del público, junto con el edificio y las nubes. El cielo también consiste en un degradado realizado con la herramienta GIMP.



Por último, como se puede observar se añadió el logo del videojuego en el edificio, y la intro finaliza con el fondo oscureciendo a negro progresivamente mientras el logo se mantiene y comienza a rotar creciendo de tamaño hasta que aparece el mensaje de *PRESS ENTER* para pasar al menú principal.

AÑADIR O MODIFICAR ATAQUES Y ANIMACIONES

El juego está programado de forma que introducir ataques y animaciones no suponga gran esfuerzo, así como poder modificar ciertos parámetros de los mismos sin ni siquiera volver a compilar el juego.

ANIMACIONES:

Previamente se ha descrito cómo se forman las animaciones, en este apartado se va a describir cómo se cargan en el juego. Para este proceso, previamente se cargan en el juego las imágenes que contienen todos los sprites de los personajes y sus animaciones. Indicamos las regiones de esta imagen para cada sprite de cada animación, así como algunas características desde el fichero ResConfig/config/fighters.assets.

Captura de las primeras líneas del fichero fighters.assets:

```
1 INIT_F
2 ID SPEED WIDTH HEIGHT [BLANKA]
3 0 170 300 350
4 INIT_ASSET [IDLE]
5 4
6 4 30 65 90 D D
7 76 30 65 90 D D
8 147 30 65 90 D D
9 218 30 65 90 D D
10 FIN_ASSET
11 D
12 INIT_ASSET [WALK]
13 8
14 292 30 65 90 D D
15 365 30 65 90 D D
16 433 30 65 90 D D
17 504 30 65 90 D D
18 574 30 65 90 D D
19 504 30 65 90 D D
20 433 30 65 90 D D
21 365 30 65 90 D D
22 FIN_ASSET
23 -100
24 INIT_ASSET [CROUCH]
25 2
26 942 30 61 90 D D
27 1016 30 61 90 D D
28 FIN_ASSET
29 D
```

El formato del fichero fighters.assets es el siguiente:

```
INIT_F      //Comenzamos a definir las animaciones de un luchador

ID SPEED WIDTH HEIGHT [FIGHTER] // Línea de información

INT INT INT INT    //Valores de la ID, velocidad, anchura y altura por defecto del luchador

// A continuación hay una lista de todas las animaciones del luchador, cada una de ellas con
// el siguiente formato:

INIT_ASSET [NAME] //Comenzamos a definir una animación

INT //Número de sprites que tiene la animación

//Para cada sprite:

INT INT INT INT [INT|D] [INT|D]
//Los 4 primeros enteros representan las coordenadas del sprite en el fichero que contiene
//todos los sprites del personaje. Los dos primeros son las coordenadas X e Y del punto
//superior de la izquierda del rectángulo que contiene el sprite, los dos siguientes la anchura
//y altura del rectángulo . Los últimos dos representan el tamaño en el que se quiere que se
//renderice el sprite (Anchura y altura). Siendo D el tamaño por defecto, y si se introduce un
//entero será D + Entero.

FIN_ASSET //Indicamos que se han introducido todos los assets para esta animación

INT //Velocidad a la que queremos que pasen de rápido los N assets de la animación
//introducida anteriormente. D = Velocidad por defecto del luchador. SI se introduce
//entero será D + Entero.

[Se describiría la siguiente animación, con el mismo formato, si la hay]

FIN_F //Cuando se han introducido todas las animaciones del luchador

[Se describiría al siguiente luchador, con el mismo formato, si lo hay]

FIN //No hay más luchadores
```

Se pueden ajustar las animaciones al comportamiento deseado sin ni siquiera volver a compilar el juego, simplemente modificando este fichero. Acciones como modificar el tamaño de una animación, hacer que se reproduzca de forma más rápida o más lenta, se realiza de forma bastante sencilla desde este fichero.

Si lo que se quiere, es introducir un nuevo ataque, definiremos la animación en el fichero que hemos descrito anteriormente y programaremos su comportamiento desde el fichero Player.java. Asignarle una tecla y un comportamiento específico se hace en pocas líneas, ya que los ataques se gestionan de forma casi automática definiendo su comportamiento en un fichero como vamos a describir en el siguiente apartado.

ATAQUES:

Para definir el comportamiento de los ataques se ha definido una clase para estos, que contiene para cada ataque sus estadísticas (Rango, daño, riesgo, a qué zona afecta, los frames de la animación en los que puede causar daños). Los ataques se cargan desde el fichero ataques.movs:

Captura de los tres primeros ataques definidos en el fichero ataques.movs:

Name	ID	Range	Damage	Risk	dH	dM	dL	Frame
punchM	3	175	25	40	0	1	0	2
kickL	6	225	20	30	0	1	1	2
kickH	10	225	20	30	1	1	0	3

Significado de cada parámetro:

- **Name:** Nombre del ataque
- **ID:** Id del ataque, coincide con la ID de la animación del mismo.
- **Range:** Valor que define el rango del ataque, en pixeles.
- **Damage:** Daño base que tiene el ataque, modificado por los stats del luchador.
- **Risk:** Riesgo que tiene realizar el ataque, utilizado en la IA
- **dH:** 1 si este ataque afecta a jugadores que están saltando/en el aire.
- **dM:** 1 si este ataque afecta a jugadores que están de pie.
- **dL:** 1 si este ataque afecta a jugadores que están agachados.
- **Frame:** frame de la animación del ataque a partir del cual este ataque no hace daño.
Por ejemplo, cuando el puñetazo se está recogiendo no tiene sentido que haga daño.

*La velocidad de los ataques depende de los stats del personaje.

A partir de estos datos se construye el objeto ataque correspondiente a cada línea, que gestiona automáticamente la lógica de la pelea como se describe en el apartado PELEA.

PERSONAJES Y ESCENARIOS

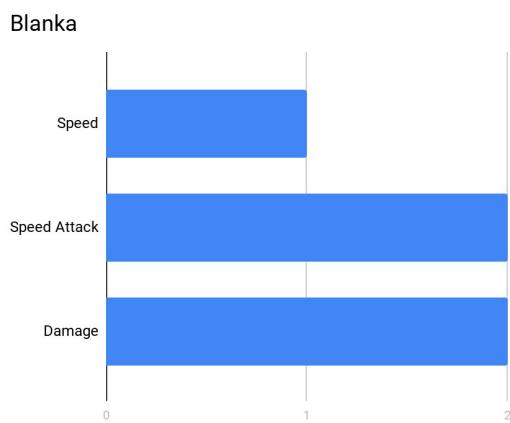
En el juego se han desarrollado 3 personajes con características y comportamientos diferentes. Además, para cada uno de los luchadores se ha creado su escenario de pelea. En este apartado se van a describir los distintos luchadores, sus características

diferenciales y sus escenarios. Todas las estadísticas y comportamientos desarrollados, se han implementado en función de las observaciones que se han realizado del juego original, y consultando también la documentación del juego original encontrada en el siguiente link: <https://gamefaqs.gamespot.com/snes/588700-street-fighter-ii/faqs/25190>

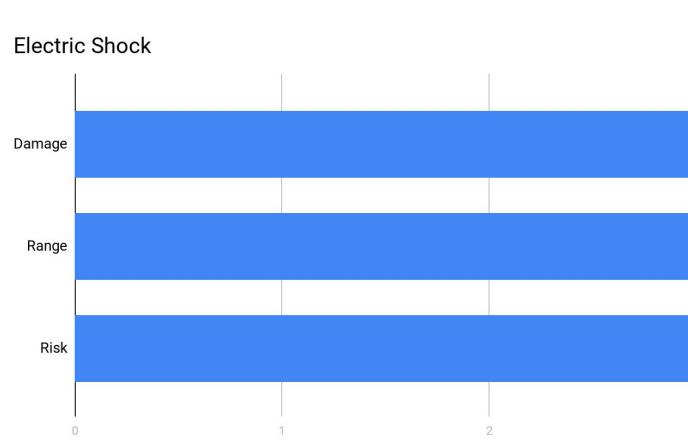
Cada personaje cuenta con un ataque especial que se describe en este apartado. En el siguiente apartado se describe la mecánica de la pelea, donde se detallaran todos los movimientos generales de los luchadores.

BLANKA:

Estadísticas generales del luchador:



Special Attack: Electric Shock

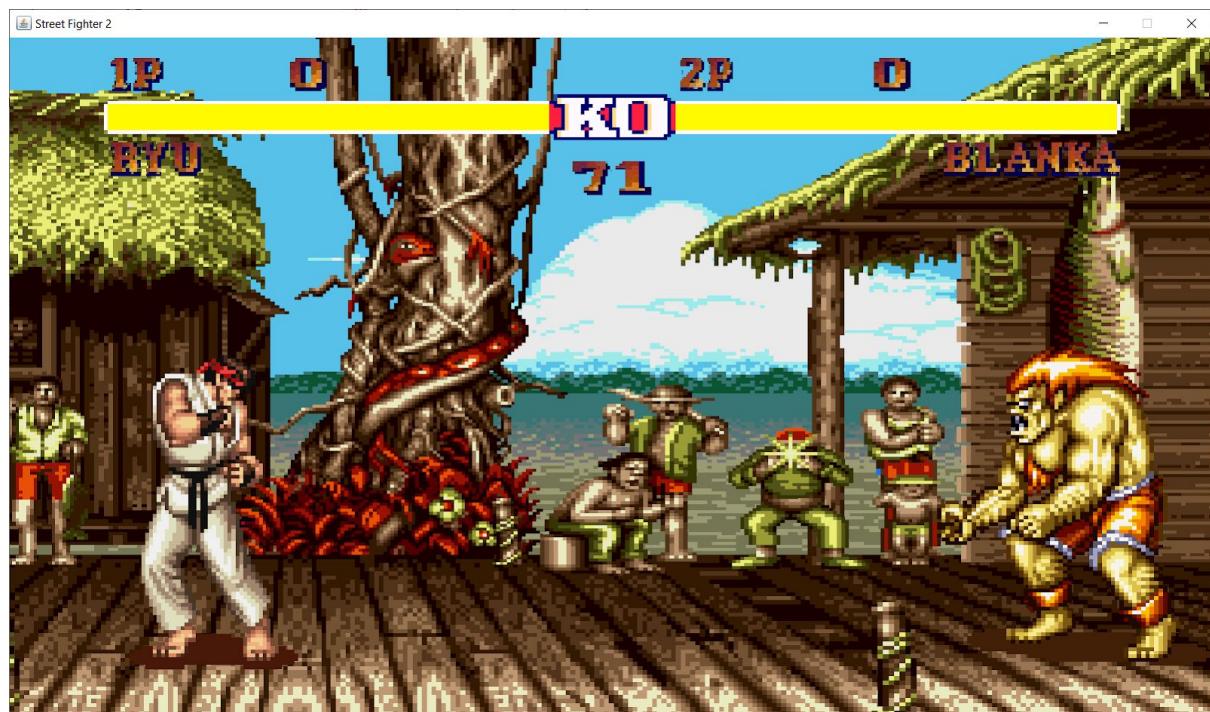


- Este ataque especial tiene mas rango que cualquier ataque físico, y causa daño de forma casi instantánea.
- Afecta si el contrincante está agachado o de pie.
- Mientras Blanka realiza este ataque, no puede realizar ningún movimiento, y puede recibir un Hadouken de Ryu o el Kikouken de Chun.

Comportamiento de la IA:

- La IA con Blanka trata de mantenerse cerca del contrincante, ya que es el luchador más lento. Además, contra más cerca está de su contrario es más probable que aproveche el Electric Shock.

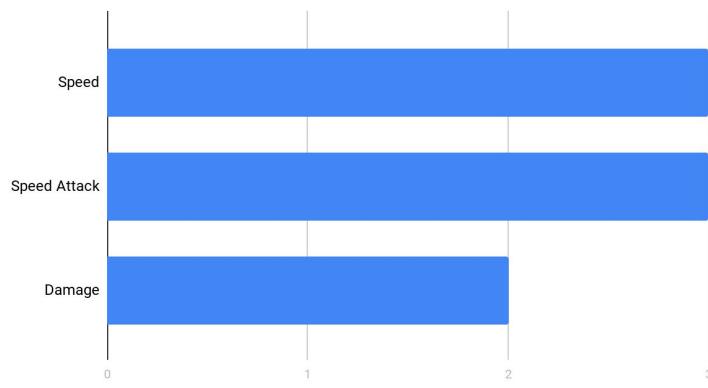
Frame de pelea en el escenario de Blanka:



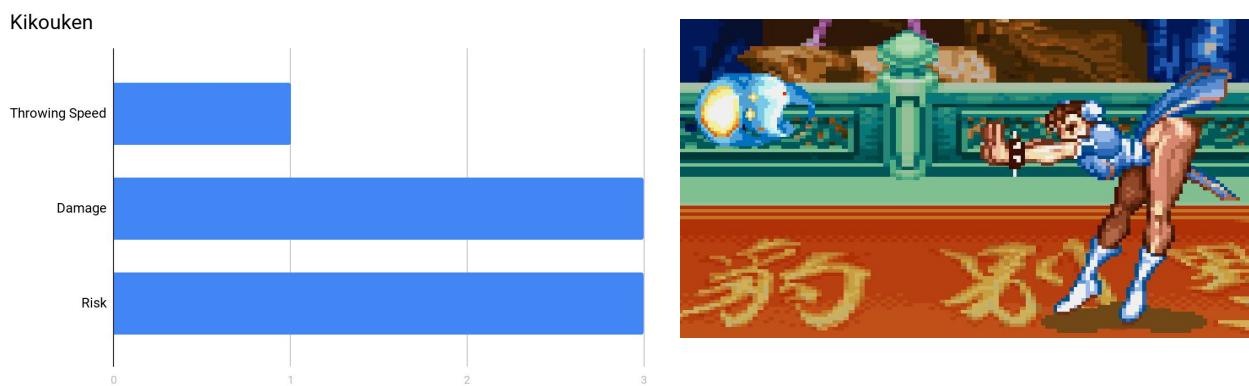
CHUN LI:

Estadísticas generales del luchador:

Chun Li



Special Attack: Kikouken



- Solo se puede esquivar saltando el proyectil.
- Proyectil con rango muy alto (Equivalente al ancho de la resolución del juego).
- Mientras Chun Li carga el ataque, se le puede golpear y así cancelar el Kikouken.

Comportamiento de la IA:

- Chun Li es la luchadora más ágil, la IA tratará de aprovecharse de esta ventaja tratando de esquivar más ataques que los otros luchadores. Especialmente, es más propensa a realizar saltos.
- Además, es propensa a alejarse para tratar de realizar el Kikouken.

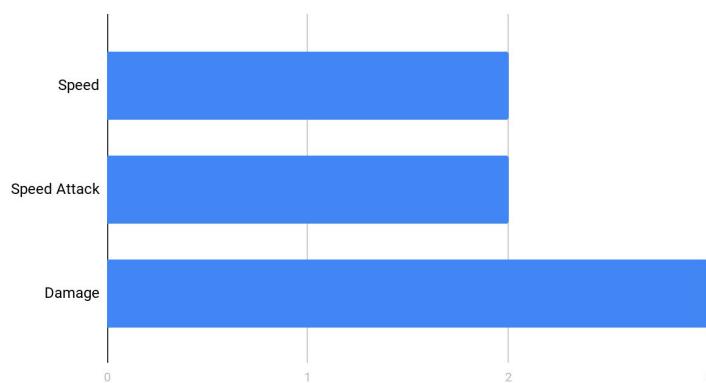
Frame de pelea en el escenario de Chun Li:



RYU:

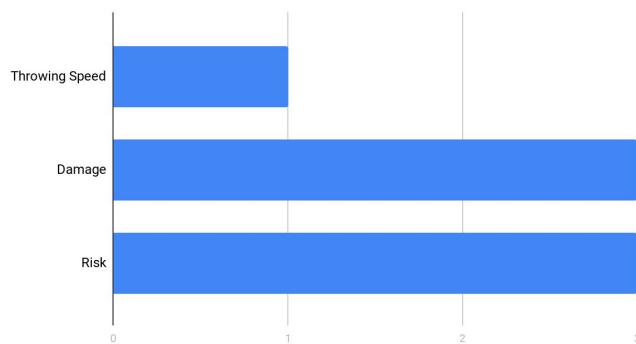
Estadísticas generales del luchador:

RYU



Special Attack: Hadouken

Hadouken



- Proyectil con rango muy alto (Equivalente al ancho de la resolución del juego).
- Solo se puede esquivar saltando el proyectil.
- Mientras Ryu carga el ataque, se le puede golpear y así cancelar el Hadouken.

Comportamiento de la IA:

- Tiene un comportamiento bastante equilibrado, aunque también tiende a alejarse para tratar de utilizar el Hadouken.

Frame de pelea en el escenario de RYU:



CÁMARA

La cámara consiste en una cámara con movimiento horizontal principalmente y un leve movimiento vertical cuando los personajes saltan que se basa en la posición y distancia de estos. Por medio de una variable que se recalcula constantemente en cada tick() se controla el offset que hay que sumar a los gráficos de los personajes y escenarios para que se cree una sensación de movimiento de la cámara.

Cuando el personaje que está más a la izquierda, se mueve hacia la izquierda continuamente de manera que va a salir de la pantalla, es decir, su posición en el eje de coordenadas X va a comenzar a ser negativo, el offset horizontal que se aplica pasa a ser directamente su posición. De esta manera la cámara se mueve con él. Es decir, si el personaje más a la izquierda se encuentra en la posición -100, el offset tomará ese valor, y todos los elementos que aparecen en la pantalla se renderizan sumando dicho offset horizontal, causando así la sensación de movimiento de la cámara. Lo mismo ocurre en el caso contrario, es decir cuando un personaje va a salirse de la pantalla por el lado derecho. Si la posición de este personaje supera la resolución horizontal de la pantalla, el offset pasa a ser la posición del personaje menos la resolución de la pantalla. Por ejemplo, si el personaje más a la derecha se encuentra en la posición horizontal 1380, y la resolución es de 1280, el offset horizontal es 100.

Hay restricciones, por ejemplo, para que los dos personajes aparezcan siempre en los límites de la cámara y no salgan de la visión de ésta. Para ello, se calcula la distancia entre los dos personajes. Si estos se encuentran a la distancia máxima permitida, es decir, la

resolución total horizontal, no se les permite moverse más hacia atrás, porque sino se saldrían del campo de visión.

Para el movimiento vertical de la cámara cuando los personajes saltan, se calcula un offset vertical. El valor de dicho offset es la diferencia entre la posición vertical del personaje cuando está en reposo y la posición vertical del personaje cuando salta.

Como se ha comentado, una vez que se obtiene el offset horizontal y el offset vertical, se debe aplicar al renderizar los personajes y los distintos elementos del escenario para causar una correcta sensación de desplazamiento de la cámara. La sensación de paralelo que se ha comentado anteriormente, es decir la diferencia de movimiento entre distintas capas del escenario, se consigue multiplicando estos valores de offset por un número racional entre 0 y 1. Así, el offset de las capas más cercanas al espectador se multiplican por números más altos, cercanos al 1, y conforme se va aumentando la profundidad, este valor decrece a valores más bajos, creando la sensación de movimiento entre capas.

PELEA

En este apartado se va a describir todo lo relacionado con el sistema de pelea. Se comentarán todos los movimientos posibles, el HUD de las escenas de pelea, las posibles opciones y la lógica que hay detrás de la pelea.

MOVIMIENTOS Y ANIMACIONES:

Lista de todos los movimientos/animaciones posibles durante la pelea:

- **Idle:** Estado de reposo, si no hemos pulsado ninguna tecla.
- **Walk:** Andar. El movimiento hacia adelante es a la velocidad máxima del luchador, mientras que si estamos retrocediendo, avanzamos a la mitad de la velocidad posible.
- **Crouch:** Agacharse, utilizado principalmente para esquivar ataques. Desde esta posición también se pueden realizar ataques.
- **Jump:** Salto vertical, para esquivar ataques del contrincante.
- **ForwardJump:** Si se pulsa la tecla de salto mientras el jugador está pulsando la tecla para moverse hacia adelante, realiza un salto hacia adelante con voltereta, lo cual permite esquivar ataques y en ocasiones rebasar por encima al rival para cambiar la orientación de la pelea.
- **BackJump:** Si se pulsa la tecla de salto mientras el jugador está pulsando la tecla para moverse hacia atrás, realiza un salto hacia atrás con voltereta, lo cual permite esquivar ataques y alejarse del rival. El forwardJump avanza el doble que el backJump.

- **Block:** Cuando el rival realiza un ataque de altura media (Ataque al pecho o hacia la cabeza), podemos bloquearlo pulsando la tecla de movimiento hacia atrás mientras el contrincante realiza el ataque.

- **CrouchBlock:** Si mientras el jugador está agachado, pulsa la tecla de movimiento hacia atrás, se realizará la animación de bloqueo agachado, bloqueando los ataques que se realicen sobre esta altura. No bloquea el puñetazo desde el salto o el ataque especial.

- **Hit:** Animación para cuando se recibe un ataque de altura media. Se percibe un leve retroceso y no se puede realizar ninguna acción mientras se realiza esta animación.

- **CrouchHit:** Animación para cuando se recibe un ataque de altura baja (Cuando estamos agachados). Se percibe un leve retroceso y no se puede realizar ninguna acción mientras se realiza esta animación.

- **HighHit:** Animación para cuando se recibe un ataque alto (Cuando estamos saltando). El luchador da una voltereta hacia atrás y cae de pie. No se puede realizar ninguna acción mientras se realiza esta animación.

- **KO:** Animación de KO para cuando el luchador no le queda vida y ha perdido la ronda o el combate.

Las animaciones idle, walk y crouch son las encargadas del movimiento, y pueden interrumpirse en cualquier momento. Las animaciones de salto no pueden interrumpirse por el propio jugador mientras se realizan, únicamente cambiarán si el luchador recibe un golpe mientras la está realizando.

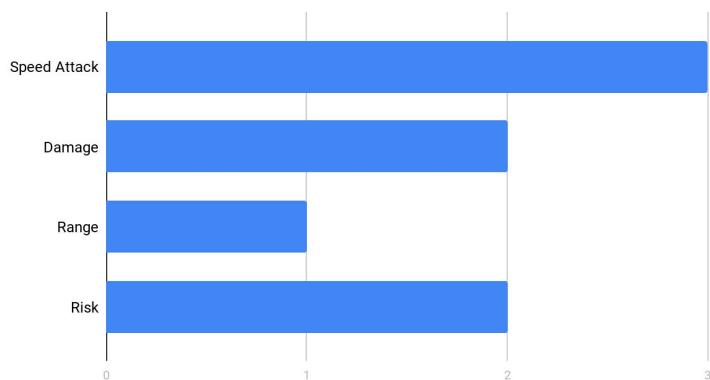
ATAQUES:

En este apartado se van a describir los ataques generales, que tienen todos los luchadores, a parte del ataque especial. Las estadísticas de estos movimientos se ven modificadas positivamente a partir de las stats de los luchadores.

NOTA: Si un ataque se esquiva saltando, al comenzar a saltar o al finalizar el salto, el ataque se puede recibir (Como si el jugador estuviera de pie).

Puñetazo medio:

Puñetazo medio

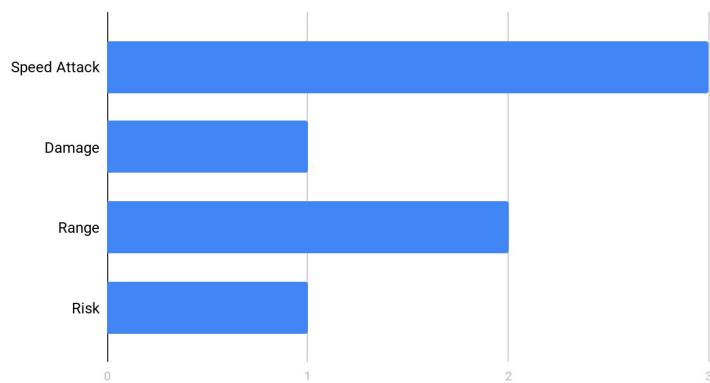


Esquivar el ataque:

- Agachado
- Saltando
- Bloqueando de pie

Patada baja:

Patada baja

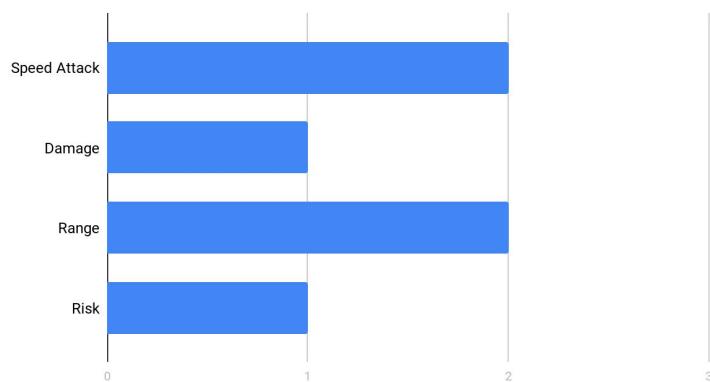


Esquivar el ataque:

- Saltando
- Bloqueando agachado

Patada alta:

Patada alta

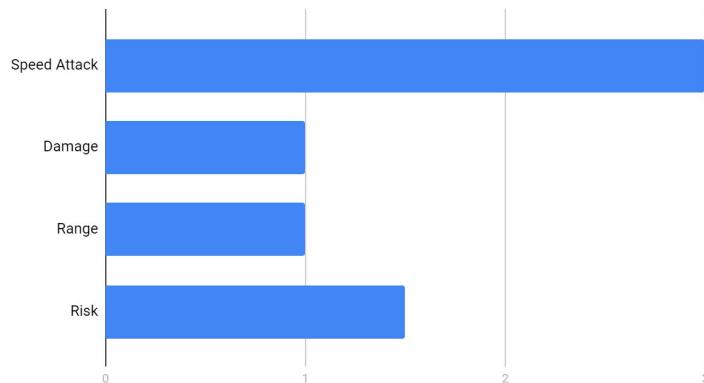


Esquivar el ataque:

- Agachado
- Bloqueando de pie

Puñetazo [Mientras el jugador está agachado]:

Puñetazo agachado

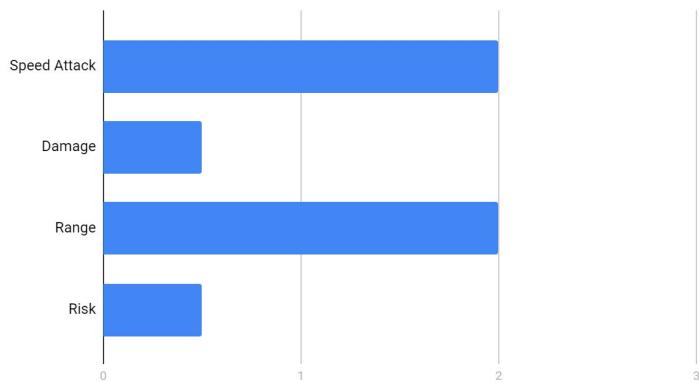


Esquivar el ataque:

- Saltando
- Bloqueo agachado

Patada baja [Mientras el jugador está agachado]:

Patada agachado

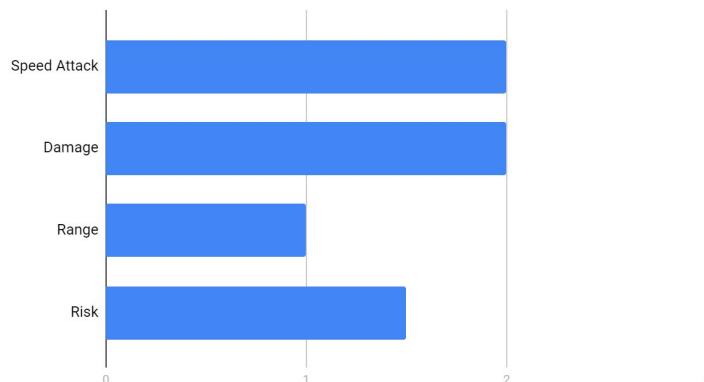


Esquivar el ataque:

- Saltando
- Bloqueo agachado

Puñetazo en el aire [Tras un salto del jugador]:

Puñetazo en el aire



Esquivar el ataque:

- Bloqueo de pie

Todos los ataques descritos también se pueden esquivar manteniendo una distancia con respecto al otro luchador mayor que el rango del ataque.

COMBOS:

Algunos de los movimientos y ataques descritos en los apartados anteriores se realizan mediante combos (Pulsando X teclas de forma consecutiva en un breve periodo de tiempo, o pulsando una tecla mientras se mantiene otra). Estos movimientos se realizan de la siguiente forma:

- **Ataque especial:** Agacharse, movimiento lateral (Hacia adelante o hacia atrás) y puñetazo. Este combo debe realizarse en menos de un segundo.
- **Puñetazo agachado:** Agacharse [Mantener] y puñetazo.
- **Patada agachado:** Agacharse [Mantener] y patada baja.
- **Puñetazo en el aire:** Saltar [Esperar a que se despegue el luchador del suelo] y puñetazo.
- **Salto hacia adelante:** Tecla de movimiento hacia adelante y salto.
- **Salto hacia atrás:** Tecla de movimiento hacia atrás y salto.
- **Bloqueo agachado:** A diferencia del bloqueo normal (Que como se ha indicado, se realiza pulsando la tecla de movimiento hacia atrás cuando el contrincante realiza un ataque a la altura del pecho o cabeza) el bloqueo agachado se realiza manteniendo la tecla de agacharse y pulsando la tecla de movimiento hacia atrás, se muestra la animación aunque el contrincante no esté atacando. El bloqueo agachado se puede dañar si el contrincante realiza un puñetazo tras salto o un ataque especial.

HUD:

Durante la pelea, se puede observar una serie de parámetros en la HUD, como se puede observar en el siguiente pantallazo:

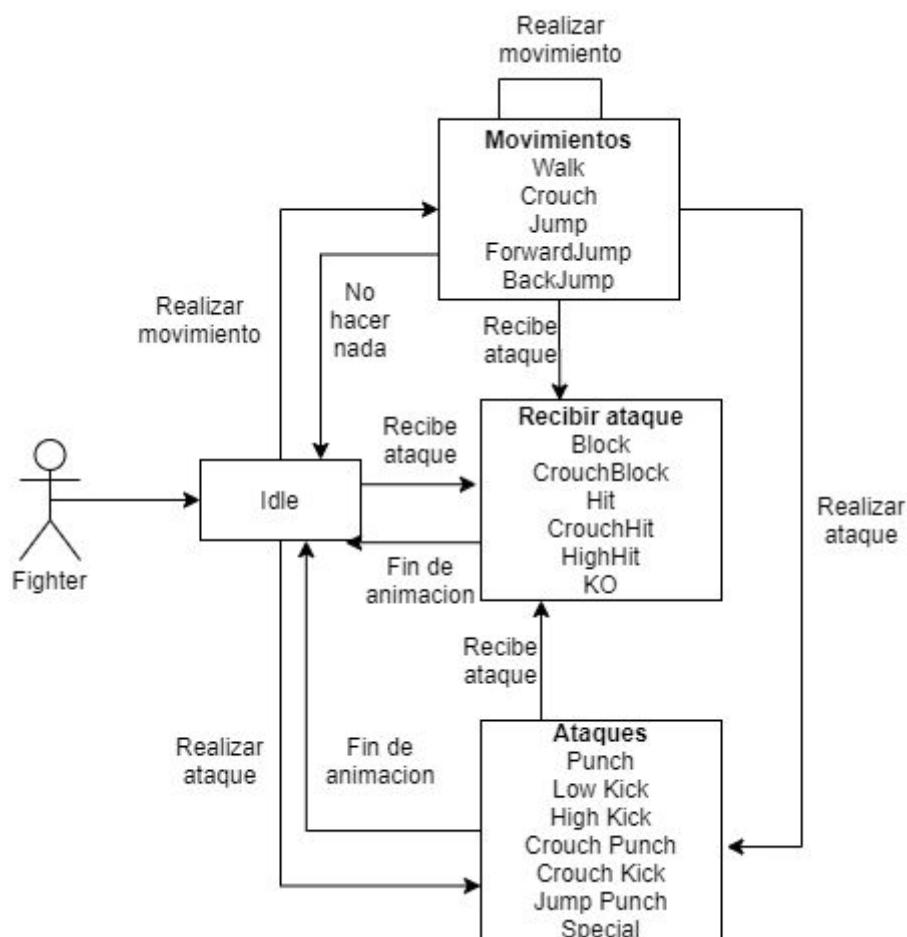


En la parte superior izquierda, se puede observar una barra amarilla que representa la vida del jugador 1 (Se puede observar que encima de la barra hay un 1P). En el centro de la parte superior de esta barra observamos la puntuación del jugador 1, 3000 puntos en este caso. En el lado contrario podemos observar la puntuación del jugador 2, que es 20100 puntos. Debajo de cada barra aparece el nombre del personaje que maneja cada jugador (Chun Li y RYU), y en el centro de la parte superior, debajo de KO, está el tiempo que le queda a la ronda, 86 segundos en este caso.

Además, este HUD también informa de las rondas que ha ganado cada luchador. Una ronda ganada es representada por el ícono que aparece a la derecha de la barra de RYU.

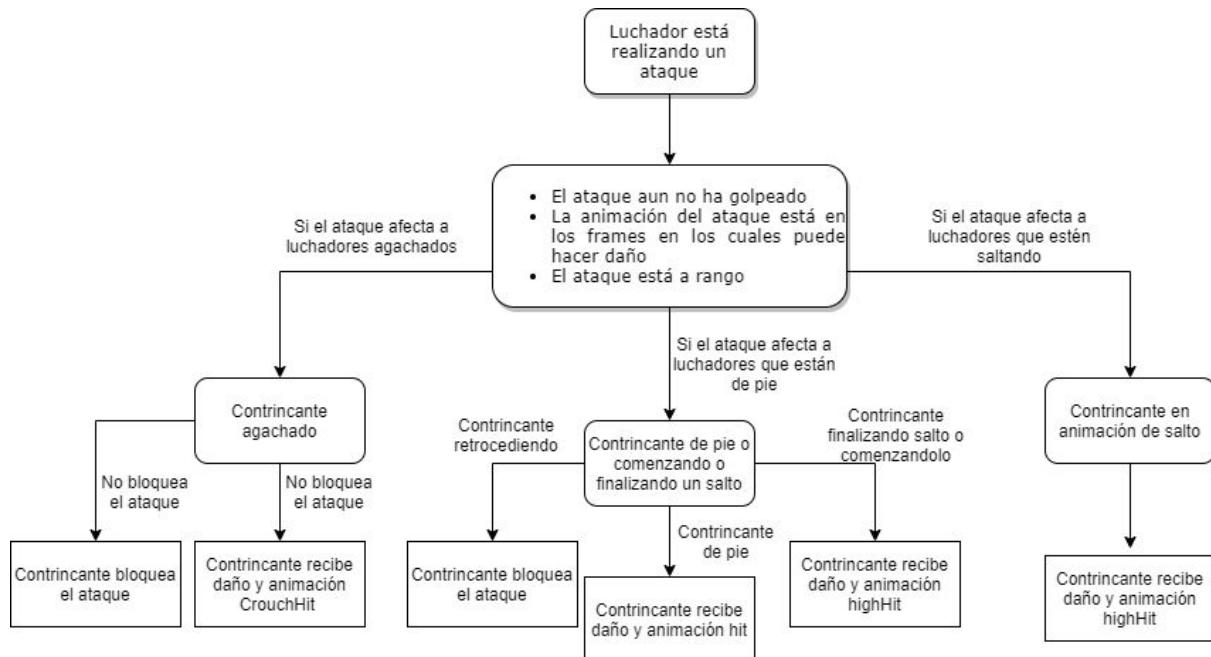
LÓGICA DE LA PELEA:

Durante la pelea, podemos definir el manejo de los personajes y la gestión de sus animaciones con el siguiente sistema de estados:



Cabe destacar que mientras estamos realizando un ataque, se puede recibir un ataque del contrincante, como podemos ver en el esquema de estados. Por ejemplo si lanzamos un puñetazo y no alcanza al rival, pero este nos lanza una patada de más rango y si que nos alcanza. De esta forma, la animación de nuestro ataque se anularía y recibimos el golpe.

Como se ha detallado en el apartado anterior, el juego dispone de una clase para almacenar los diferentes parámetros de cada ataque, por lo que nuestro sistema utiliza esta misma clase para gestionar si un ataque surge efecto o es esquivado, siguiendo el siguientes esquema (Desde la clase en la cual se gestionan todas las entidades del juego):



En cada tick se comprueba si el luchador 1 y el luchador 2 están realizando un ataque, y en caso afirmativo se sigue el esquema mientras se cumplan condiciones.

GUARDADO DEL JUEGO

Hay muchas configuraciones que es necesario guardar para que el juego pueda iniciarse correctamente e incluso pueda leerlas en ejecución si es necesario o modificarlas. Las configuraciones iniciales, se almacenan en el directorio interno dentro del .jar, llamado ResConfig/config. En este directorio, se encuentran los siguientes archivos de configuración:

- **animacionesID**: Almacena el nombre y el ID de las animaciones que un personaje puede realizar.
- **ataques.movs**: Almacena los distintos ataques que se pueden realizar en el juego, junto con el ID del ataque, el rango, el daño, el riesgo y si afecta al otro jugador cuando éste está saltando, agachado, o de pie.
- **dificultad**: Almacena la dificultad de la IA.
- **fighters.asset**: Almacena la velocidad de movimiento, la anchura, la altura y las secciones del sprite sheet de cada personaje que deben cortarse para obtener sus sprites.
- **keys**: Almacena las teclas utilizadas para que los personajes realicen acciones durante la pelea (controles).

- **resolucion**: Almacena la resolución actual del juego.
- **score**: Almacena el ranking de puntuaciones junto con al personaje y el nombre del jugador.
- **song**: Almacena el nombre de todos los archivos de música del juego.
- **sound**: Almacena el nombre de todos los archivos de sonidos o efectos del juego.
- **volumenJuego**: Almacena el sonido actual de la música y los efectos y si se encuentra muteado o no.

Cuando el juego se inicia, comprueba si en el mismo directorio en el que se encuentra el .jar, existe un directorio llamado .configStreetFighterII. Si existe, no hace nada, y lee la configuración de este directorio, y si no existe, lo crea, y copia los ficheros de configuración que se acaban de mencionar del directorio ResConfig/config en este nuevo directorio. A partir de este momento, el juego realiza todas las lecturas que necesita para cargar la configuración en el nuevo directorio creado. Esto es necesario hacerlo por que, por ejemplo, si el jugador quiere cambiar la configuración de las teclas para un cierto movimiento, es necesario sobreescribir el fichero que guarda esta información con la nueva tecla que ha introducido el jugador para que se almacene el cambio correctamente y sea persistente, y se vio que en Java no existe una forma (al menos hasta donde se pudo investigar) de sobreescribir los archivos internos de un .jar. Es por ello que se creó el directorio exterior .configStreetFighterII, pudiendo así el jugador personalizar la configuración del juego como desee y guardándose correctamente.

Los archivos configurables y que el jugador puede cambiar desde los menús de opciones o de pausa son:

- dificultad
- keys
- resolución
- score
- volumenJuego

MENÚ OPCIONES

El juego cuenta con un menú opciones al que se puede acceder a través del menú principal, y también cuenta con el menú pause durante una pelea que es muy similar al menú opciones, con la única diferencia de que durante la pelea no se puede cambiar la dificultad. La apariencia de ambos menús se ha podido observar en el apartado lógica del juego.

Todos los cambios que se realicen en ambos menús son persistentes, es decir, si se cambia cualquier cosa y se cierra el juego, al volver a abrirlo la configuración será la última que el usuario había editado. Los cambios que se pueden realizar son:

- **Sonido: on/off.** Permite silenciar el juego completamente si se desea.
- **Volumen de la música: Entre 0 y 8.** Permite ajustar la intensidad con la que suena la música del juego, como la canción de la intro, la canción entre menús, la canción de cada escenario, etc.
- **Volumen de los efectos: Entre 0 y 8.** Permite ajustar la intensidad con la que suenan los efectos del juego, el sonido del avión cuando te mueves por los menús, la voz de los personajes durante la pelea, etc.
- **Dificultad: Fácil / Media / Difícil / Gradual.** Permite ajustar la habilidad de la IA en las peleas del modo historia. Si se elige gradual, la primera batalla es en dificultad fácil y la segunda en dificultad media.
- **Controles.** Se accede a un nuevo menú que permite configurar cada uno de los ataques y movimientos de un personaje con la tecla del teclado que se desee. También muestra otra información como que durante una pelea, si pulsas P se abrirá el menú pause, si se pulsa la tecla ESC en cualquier momento se muestra un mensaje para salir del juego, y combinaciones de teclas para descubrir nuevos movimientos que pueden llevar al jugador a la victoria.
- **Resolución: Pantalla completa / Pantalla completa con ventana / 1280x720 / 640 x 480.** Permite cambiar la resolución a la que se desea ver el juego. La opción pantalla completa no muestra las opciones de minimizar o cerrar el juego (se puede cerrar con la tecla ESC), y pantalla completa con ventana sí.

MÚSICA Y SONIDOS

El juego cuenta con un segundo thread que se encarga de reproducir la música de fondo que suena constantemente y los sonidos de efectos.

El juego cuenta en total con 10 músicas diferentes que son las siguientes:

1. Intro.
2. Final de batalla.
3. Comienzo de batalla.
4. Pantalla de selección de personajes.
5. Escenario de Blanka.
6. Escenario de Chun Li.
7. Escenario de Ryu.
8. Cinemática fin juego Blanka.
9. Cinemática fin juego Chun Li.
10. Cinemática fin juego Ryu.

El juego cuenta con un total de 31 efectos de sonido diferentes entre los que se encuentran el sonido del avión cuando viaja de un país de otro, sonido de teclas entre menús, sonidos de inicio de ronda, sonidos de los personajes cuando lanzan un ataque o cuando son golpeados, etc.

Por medio de la clase Game, que maneja los dos threads principales, tanto el del juego como el de la música, el thread del juego puede avisar al thread que se encarga de la música de cuando debe comenzar una nueva canción o cuando debe sonar determinado efecto.

SISTEMA DE PUNTOS

En una combate, a parte de ganar o perder, también es importante el número de puntos conseguidos, ya que, al fin y al cabo, es en lo que se basa el ranking para clasificar a los jugadores.

POS	SCORE	FIGHTER	NAME
1	80000	RYU	MAN
2	60000	CHUN LI	AAA
3	55000	RYU	BBB

Menú de Ranking

En el **modo 1 jugador** ("GAME MODE"), cuando el jugador se pase el modo historia, o cuando pierda antes de terminarlo, tendrá la posibilidad de guardar la puntuación obtenida, y podrá consultarla más tarde en el menú de ranking.

En cambio, en el **modo 2 jugadores** ("VS MODE"), los puntos pasarán a un segundo plano, aunque los jugadores seguirán disponiendo de los puntos obtenidos a lo largo del combate.

ENTER YOUR INITIALS			
POS	SCORE	FIGHTER	NAME
4	50000	RYU	CCC
5	45000	CHUN LI	DDD
6	30000	CHUN LI	A--
7	20000	BLANKA	EEE
8	10000	BLANKA	FFF

Guardando puntuación del modo historia

Los puntos conseguidos en una pelea se basan en:

- Ataques que han dañado al objetivo (500 puntos cada uno)
- Bonus por ganar ronda:
 - TIEMPO: (Tiempo restante de ronda)*100*(multiplicador)
 - VIDA: (Vida restante ganador ronda)*10*(multiplicador)
 - Multiplicador: Depende de la dificultad en la que se esté jugando:
 - Fácil = 1
 - Normal = 1.2
 - Difícil = 1.4



Puntos del jugador 1 tras ganar una ronda

INTELIGENCIA ARTIFICIAL

Inicialmente, se estudiaron distintas opciones como utilizar una red neuronal o un autómata. La primera se descartó por nuestra inexperiencia en ese ámbito y la segunda porque era demasiado rígida (introducir nuevos golpes en cualquier momento iba a suponer reestructurar el autómata por completo).

Finalmente, se decidió desarrollar un **sistema que fusiona un autómata con un sistema basado en puntuaciones y aleatoriedad** para evitar un comportamiento repetitivo de la IA. Es un autómata porque al no usar redes neuronales era necesario expresar de alguna forma qué se quiere hacer de acuerdo a cada situación, pero también posee un sistema de puntuaciones que le da más flexibilidad, ya que para determinadas acciones como pegar, el añadir un nuevo golpe y que la IA lo tenga en cuenta solo supone parametrizar dicho ataque como el resto (y no reestructurar un autómata), como se verá a continuación:

En dicho sistema, **en cada tick del juego se debe decidir entre saltar verticalmente, agacharse, moverse horizontalmente, saltar con desplazamiento y pegar.**

A continuación se explica cuándo se produce cada una de las 5 opciones:

SALTO VERTICAL

Las 3 **situaciones** que se han planteado en las que un luchador debe saltar son:

- Cuando se le ha lanzado el ataque especial de Ryu o Chun, siendo la única forma de esquivarlo el salto. Cuanto mayor dificultad de la IA, más probabilidad de que los salte.
- Si el rival está agachado, habrá una probabilidad de que se realice el combo de salto+puñetazo.
- La luchadora Chun Li tendrá siempre una probabilidad de saltar en cualquier instante, puesto que observando videos del juego oficial se percibe esta actitud en dicho personaje.

AGACHARSE

Las 3 **situaciones** en las que la IA puede agacharse son:

- Cuando el enemigo da un puñetazo.
- Cuando el enemigo da una patada alta.
- Cuando el enemigo está agachado y pretendes golpearle con un combo agachar+patada baja ó agachar+puñetazo.

Cuanto mayor dificultad de la IA, más probabilidad de que se agache en estas situaciones.

DESPLAZAMIENTO HORIZONTAL

Para decidir si en cada instante la IA quiere acercarse o alejarse del rival se ha usado un sistema de votación en el que el valor de cada parámetro tenido en cuenta supondrá un voto para alejarse o para acercarse al rival. Los parámetros tenidos en cuenta son los siguientes:

- **Distancia al enemigo:** Si estás a más de 200 píxeles del rival, la IA se acerca. Se aleja en el caso contrario.
- **Vida de la IA:** Si tiene más del 30% de vida, se acercará mostrando una cierta seguridad y confianza porque aún se está en una situación favorable. Se aleja en el caso contrario.
- **Vida del rival:** Si tiene menos del 30% de la vida, se acercará mostrando agresividad por la situación desfavorable del rival, queriendo rematar la pelea lo antes posible.

Para solucionar los **casos de empate**, es decir, en los que tiene los mismos votos el desplazamiento hacia la izquierda que hacia la derecha, la IA posee 2 estados, uno agresivo y uno más reservado. Según en cual se esté, estas situaciones de empate se decantarán hacia un lado o hacia otro, es decir, determinarán si la IA arriesga un poco acercándose al rival, o se aleja siendo más conservadora.

Adicionalmente, el **bloqueo** de un ataque se produce cuando el rival golpea y el otro personaje se desplaza en la dirección contraria de donde está el enemigo que ha

golpeado. Por ello, el bloqueo se ha integrado de forma natural al desplazamiento, ya que en las situaciones que convenga alejarse del rival, si este lanza un golpe, si la IA se está intentando alejar se producirá un bloqueo.

SALTO CON DESPLAZAMIENTO

Junto con este desplazamiento horizontal, si se pulsa la tecla “UP” se va a producir un salto con desplazamiento, por lo que en determinadas situaciones, cuando el rival acorrala a la IA en uno de los extremos del mapa, realizará la voltereta (salto con desplazamiento) para saltar por encima del rival y aliviar así la presión que le estaba ejerciendo el contrario.

Cuanto mayor dificultad de la IA, más probabilidad de que se escape en esta situación.

PEGAR

Para decidir qué ataque quiere realizar la IA en cada momento, se ha desarrollado un sistema de votación en el que el valor de cada parámetro tenido en cuenta supondrá un voto o no para cada ataque. Los parámetros tenidos en cuenta son los siguientes:

- **Rango:** Cada ataque tiene asignado un rango, por lo que si un ataque está lo suficientemente cerca del rival como para golpearle, tendrá mayor probabilidad de obtener 2 puntos para la votación. Se le da 2 puntos porque estar a rango es crucial, de esta manera se evitan comportamientos en los que la IA golpea mucho al aire cuando el rival está muy lejos, cosa que transmite bastante irrealismo.
Además, los ataques especiales tienen algo menos de probabilidad de obtener puntos, ya que un uso excesivo de ellos puede ser contraproducente.
- **Riesgo:** Cada ataque tiene asociado un riesgo, que es también proporcional al daño que hace. Por ello, si un ataque tiene una 30% de riesgo, y la vida restante de la IA es mayor al 30%, la probabilidad de sumar 1 punto al ataque será mayor que si está por debajo de ese porcentaje de vida.

Tras sumar los puntos oportunos para cada ataque de acuerdo a estos parámetros, el que tenga más puntos será el que pretenderá realizar la IA.

Cuanto mayor dificultad de la IA, más probabilidad de que golpee.

El sistema de puntos desarrollado para pegar, permite que se puede añadir un nuevo ataque en cualquier momento y que este se integre en la IA sin problemas, ya que con su rango y riesgo, el sistema lo evaluará como uno más, sin necesidad de reestructurar todo el sistema como ocurriría con un simple autómata.

Para evitar un comportamiento repetitivo de la IA, todo lo comentado anteriormente está sujeto a una aleatoriedad, es decir, siempre habrá una probabilidad de que la IA no haga lo que en el fondo sabría que tiene que hacer. Esta probabilidad de comportarse

erróneamente será inversamente proporcional a la dificultad del juego. Por ejemplo, si el rival lanza un “hadouken” a la IA, aunque sepa que lo debería saltar, habrá más probabilidad de que lo salte en difícil que en fácil.

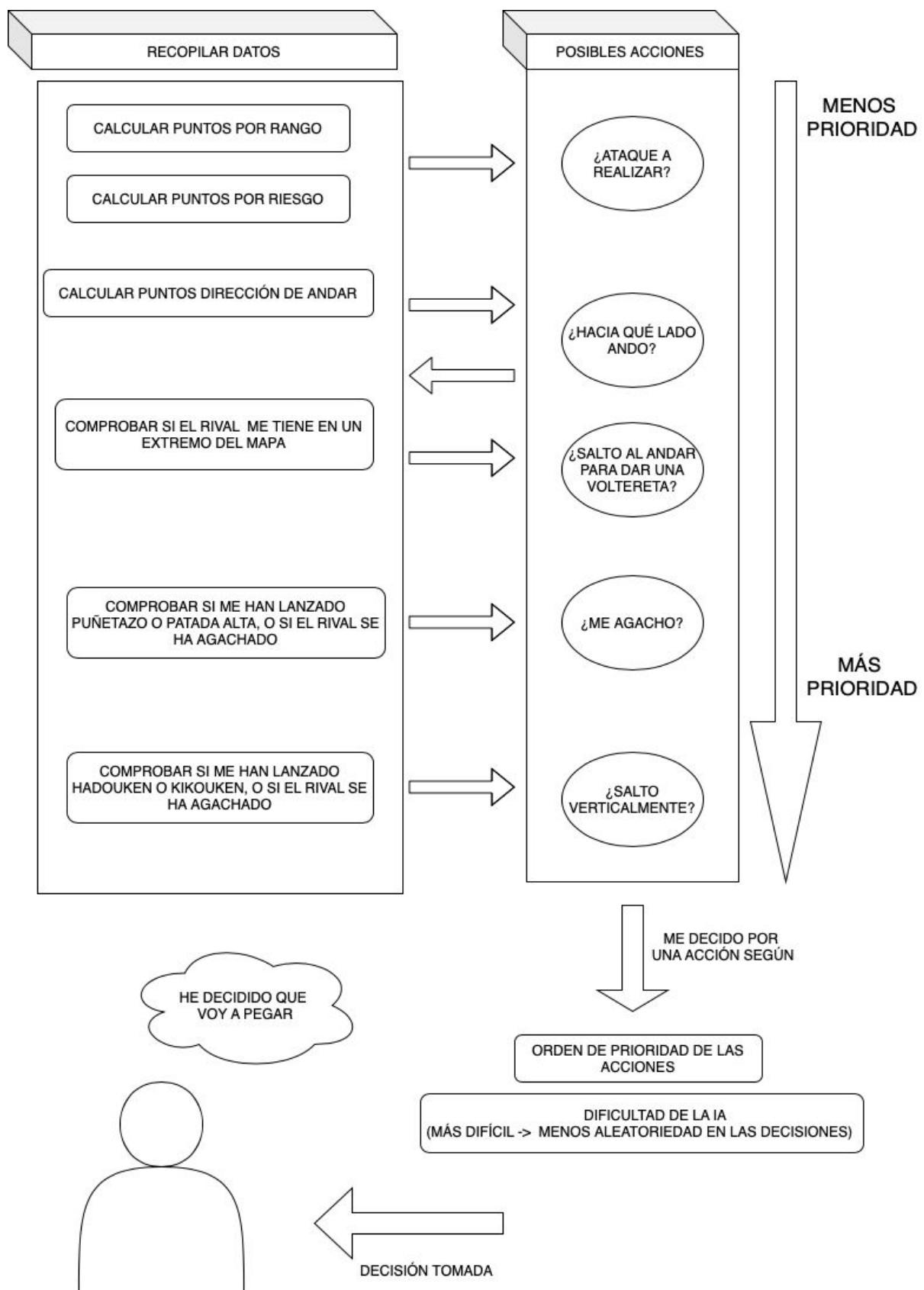
¿Cuál de las 5 posibles acciones va a realizar la IA en cada tick? La prioridad de acciones va en el mismo orden que se ha explicado, pero como se ha dicho, al existir esta aleatoriedad, se producen comportamiento impredecibles y sorprendentes que dan un mayor realismo.

Las diferencias entre cada dificultad de la IA radican principalmente en la presión que esta ejerce sobre el enemigo. Esta presión es directamente proporcional a:

- La probabilidad de que la acción del tick actual sea pegar.
- La probabilidad de que la acción del tick actual sea acercarse al rival.

Es decir, **más fácil, menos presión ejerce la IA sobre el rival.**

Diagrama que muestra cómo la IA toma la decisión de qué acción realizar en cada tick:



BACKDOOR

En cualquier momento del juego se pueden pulsar estas teclas para saltar a las cinemáticas finales del juego:

- **Tecla “5”**: cinemática Blanka
- **Tecla “6”**: cinemática Chun Li
- **Tecla “7”**: cinemática Ryu

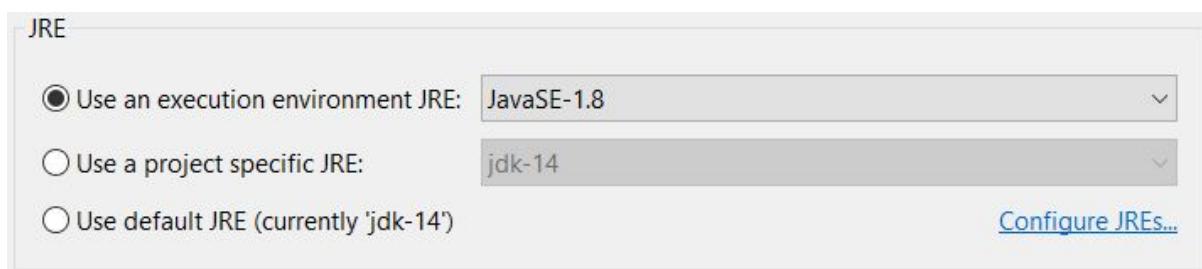
PROBLEMAS ENCONTRADOS Y SOLUCIONES

En este apartado se van a describir los principales problemas que se han encontrado durante la implementación del juego, así como sus soluciones.

- No se podía guardar el estado del juego en el JAR, ya que este es un archivo no modificable. La solución se ha explicado [aquí](#).
- La primera vez que se exportó el juego se hizo con una versión de java muy reciente, de la que no disponen la mayoría de ordenadores, y se obtenía el siguiente error al iniciar el juego:

```
java.lang.UnsupportedClassVersionError:
```

Esta es la versión de java que se ha utilizado finalmente:



- En la primera exportación del juego, en algunos ordenadores de vez en cuando el juego paraba la ejecución mostrando el siguiente error:

```
java.lang.OutOfMemoryError: Java heap space
```

Este error no lo pudimos encontrar antes de exportar el juego, ya que a los miembros del grupo nos iba correctamente siempre el juego. Se encontró el error

antes de subir la versión Beta, ya que compartimos el juego con amigos para tratar de encontrar errores.

Investigando por internet, se vio que se podía corregir este error lanzando el juego desde consola con la siguiente opción:

```
java -Xmx512M -jar "+path+"/StreetFighterII.jar
```

La opción -Xmx512M fuerza a que como máximo se le dedique a la aplicación 512 Mb de memoria a la máquina virtual de Java. Controlando este parámetro se soluciona el problema, ya que si no lo limitamos, en algunos ordenadores con poca memoria RAM, el juego daba ese error.

Para solucionar finalmente el error, se ha forzado a que el juego inicie con la opción -Xmx512M.

TAREAS RESTANTES

Las tareas que se harían con más tiempo para que fuera un juego más completo:

- Añadir todos los personajes.
- Añadir paleta de colores a los personajes.
- Añadir todos los escenarios.
- Añadir todos los movimientos de la pelea.
- Añadir pantalla de bonus.