A ti te va a caer el Axl

Daniel Barreto - #04-36723
daniel@ac.labf.usb.ve

6 de abril de 2009

Índice general

Introducción

"A ti te va a caer el Axl[1] *"*[2] es un juego desarrollado en 2D+1/2 que emula juegos de pelea entre un personaje principal y varios controlados artificialmente, sobre un "stage" o campo de tamaño limitado.

El objetivo principal del juego es vencer a todos los contrincantes que sean encontrados, para esto es necesario atacarlos y herirlos de forma que sean más suceptibles a ser lanzados fuera de la plataforma sobre la que pelean.

Para esta entrega se implementaron diferentes *Steering Behaviors* como persecución y evasión (*Seek, Flee, Pursue* y *Evade*), evasión de obstaculos (*Obstacle Avoidance*), comportamientos grupales o de manada (*Flocking*) que incluyen Separación, Cohesión y Alineación.

El juego esta construido en 3D usando OpenGL como herramienta para realizar el "rendering" de todos los objetos que este incluye (personajes, obstaculos, plataforma, etc).

[1] Axl Rose, vocalista principal de la banda de Heavy Metal Guns n' Roses
[2] Nombre provisional

Capítulo 1

Descripción del Juego

1.1. Metáfora

La metáfora del juego se trata de la difícil tarea que tiene **Slash** (personaje principal), el guitarrista principal de la banda de heavy metal *Guns n' Roses*, de acabar con los máximos exponentes del *Reggaeton*, a los cuales debe atacar con la música producida por su guitarra.

La trama se desarrolla en una tarima de conciertos de *Guns n' Roses*, en donde se encuentran obstaculos como amplificadores, una bateria y otros instrumentos. **Slash** tendrá que sacar a los reggaetoneros invasores lanzándolos fuera de la tarima, pero este objetivo no es tan fácil como parece, pués cada enemigo viene lleno de energía y su capacidad de quedarse dentro de la tarima está ligado a su nivel de energía actual, por esto, **Slash** tendrá que lastimarlos con su guitarra antes de intentar arrojarlos hacia afuera, de forma parecida al juego de Nintendo © *Super Smash Bros.*

1.2. ¿Cómo jugar?

Los controles y sus acciones se encuentran representadas en la siguientes tablas:

Cuadro 1.1: Controles de **Slash**

Tecla	**Acción**
a	Acelerar hacia la izquierda
d	Acelerar hacia la derecha
w	Acelerar hacia el fondo del escenario
s	Acelerar hacia el frente del escenario

Nota: Varios movimientos pueden ser ejecutados simultáneamente para accelerar hacia varias direcciones al mismo tiempo.

Cuadro 1.2: Acciones de **Slash**

Tecla	**Acción**
Espacio	Salto
h	Golpear
j	Disparar
k	Tocar Guitarra
y	ángulo de disparo
u	Disminuir ángulo de disparo
n	Aumentar fuerza de disparo
m	Disminuir fuerza de disparo

Capítulo 2

El juego y sus objetos

El juego consta de varios objetos que se describen a continuación

2.1. Personajes

Los personajes que participan en el juego pueden ser divididos en dos grupos: El personaje principal (**Slash**) y sus enemigos los reggaetoneros. Ambos cuentan con resticciones similares en cuanto a su máxima velocidad y aceleración, tanto lineal como angular, que puedan tener. Además cada personaje tiene una cierta cantidad de masa que lo hace más o menos suceptible a ser desplazado por golpes de otros jugadores.

Todos los personajes cuentan con un nivel de vida o energía, el cual es multiplicado por su masa cada vez que se detecta una colision de algún vector de fuerza que se le este siendo aplicado. El máximo nivel de energía para todos los personajes es de 100 puntos, y los golpes o ataques que reciban de otros personajes irán disminuyendo poco a poco dicho nivel.

La fórmula para calcular la masa de un personaje cuando reacciona frente a una colisión realizada por algún arma o bala es la siguiente:

$$M_c = M_p x E/100$$

Donde M_c es la masa referente a una colisión con un arma o proyectil lanzado por otro personaje, M_p es la verdadera masa del personaje y E es

su nivel de energía.

Para otro tipo de colisiones como los que vendrían dados por el choque con algún otro personaje, se usa la verdadera masa del personaje M_p.

2.1.1. Slash

El personaje principal del juego sólo posee un arma: su guitarra, con la cual puede golpear a los reggaetoneros como si fuera una espada, pero tambien puede disparar "balas" a sus enemigos más lejanos. No existen restricciones acerca del número de balas con las que puede contar **Slash**.

2.1.2. Los Reggaetoneros

Los reggaetoneros, al igual que slash, cuentan con un arma para disparar y golpear. A pesar de ser más pequeños que **Slash**, los golpes y balas que lancen causan más daños que las lanzadas por el personaje principal, sin embargo los reggaetoneros tienen que ser más cuidadosos, pues si se equivocan apuntando pueden herir a sus propios compañeros del reggaeton.

2.2. Proyectiles

Los proyectiles forman parte importante del juego pués son la manera más fácil y efectiva de hacer daño a los oponentes. Sólo se cuenta con un tipo de proyectil: la "bala" que describe un movimiento parabólico hacia el suelo dependiendo de la fuerza y el ángulo con la que fué lanzada. Además de las "balas" en esta sección describiremos otro tipo de herramienta con la que cuenta **Slash** para cumplir su objetivo: las "ondas de sonido" que, a pesar de no ser un proyectil como tal, funcionan como un ataque de largo alcance.

2.2.1. Balas

Las balas son simplemente esferas que tienen un radio, un peso y una cantidad de daño causante predefinido. El daño que causa una bala es constante, pero la fuerza con la que empuja al personaje que recibe la bala depende de la velocidad que lleve la bala en ese momento y el peso de la misma.

Las balas desaparecen del juego al tocar a cualquier jugador, a cualquier obstaculo o al tocar el suelo.

2.2.2. Ondas de sonido

Las ondas de sonido se modelan como circunferencias que poseen un radio incial y un coeficiente de expansión de dicho radio. Todas las ondas de sonido en el juego tienen inicialmente una intensidad fijada en 100 %, esa intensidad irá descendiendo a medida que la onda se expanda con su coeficiente de expansión.

2.3. Escenario o tarima

El escenario del juego es un espacio rectangular de tamaño fijo, en el cual se encuentran objetos que funcionarán como obstaculos a evitar para los personajes llevados por la inteligencia artificial.

Capítulo 3

Comportamientos y movimientos desarrollados

3.1. Los comportamientos no tan inteligentes de Slash

El comportamiento de **Slash** viene dado por la interacción del usuario con el juego, pero posee la posibilidad de realizar las acciónes "Saltar" y "Disparar".

Al saltar, se le da un valor predeterminado a la componente Y de su velocidad. En el momento del salto, la velocidad en el eje Y se ve afectada por la acción de la gravedad que actúa siendo una aceleración constante y negativa a lo largo de dicho eje.

La acción de disparar es dejada a manos del jugador. El usuario podrá elegir una fuerza y un ángulo para realizar los disparos mediante el uso del teclado. La trayectoria que siguen las balas variarán dependiendo de esas dos variables.

3.2. Los reggaetoneros

Las acciones que pueden desempeñar los reggaetoneros son las siguientes:

- **Vagar** (Wander): Los reggaetoneros pueden vagar por la tarima cada

vez que se encuentren sin ningún otro comportamiento con más importancia que tengan que realizar. Para simular este comportamiento se toma eventualmente una nueva dirección a seguir y se devuelve una fuerza o "steering" que lo acelera hacia dicha dirección.

- **Perseguir y Evadir** (Pursue and Evade): Con estos comportamientos, los reggaetoneros persiguen (o evaden) un objetivo, estimando la posición del objetivo 1 segundo mas adelante del tiempo actual, asumiendo que el objetivo permanece con la misma velocidad durante dicho tiempo.

- **Evasión de obstaculos** (Obstacle Avoidance): Los reggaetoneros tratarán de evadir, siempre que se pueda, los obstaculos que se les presenten en el camino. Entre los obstaculos a evitar, se fijarán también en los bordes del escenario, tomandolos como paredes imaginarias que tendrán que esquivar.

  Actualmente los reggaetoneros utilizan solo un rayo de visión dirigido hacia donde este dirigida su orientación, dicho rayo de visión se modela como un segmento de linea de tamaño 15.

- **Comportamientos en manada** (Flocking and Swarming): Los reggaetoneros se comportan como un enjambre o "swarm". Para poder simular este movimiento en conjunto son necesarios los siguientes comportamientos:

  - **Alineación** (Alignment): Se busca mantener una dirección de desplazamiento común para todos los integrantes del enjambre.
  - **Cohesión** (Cohesion): Los miembros del enjambre tratan de mantenerse juntos, tratando de acercarse a un centro de masa del enjambre en general.
  - **Separación** (Separation): Los miembros del enjambre tratan de mantener cierta separación entre ellos.

3.3. Comportamiento general, unión de varios comportamientos

Para lograr un comportamiento más natural en los reggaetoneros se fusionaron los comportamientos anteriormente detallados, utilizando el sistema

de “Blending” con arbitraje, en donde se agruparon los comportamientos y se establecieron prioridades entre dichos grupos.

De esta forma, cada reggaetonero se encarga de aplicar la fuerza dada por el grupo de comportamientos con mayor prioridad y que haya devuelto alguna fuerza superior a un umbral previamente establecido, que tenga entre sus comportamientos asociados.

Dentro de cada grupo de comportamientos se ejecutan los comportamientos pertenecientes a dicho grupo y se suman entre ellos, tomando en cuenta que distintos comportamientos en particular tienen un mayor peso en la suma resultante.

La implementación del “Blending” realizado será detallada posteriormente.

Capítulo 4

Introduciendo Sentidos al juego

Además de la posibilidad de tener un rango de visión, los reggaetoneros fueron dotados de la habilidad para oir ciertas ondas de sonido. A continuación se describe el funcionamiento de este sentido y las necesidades de usarlo.

4.1. El ataque secreto de Slash

Slash guarda un ataque secreto del cual sus enemigos no disponen: Tocar su privilegiada guitarra hace que los reggaetoneros queden mareados por dicho ataque y no puedan reaccionar ni pensar en estrategias o comportamientos a seguir.

El efecto producido por la guitarra de **Slash** se modela como una onda de sonido que parte desde la posición central donde se encuentra **Slash** y se expande a cierto radio disminuyendo su intensidad.

4.2. El umbral auditivo de los reggaetoneros

Los reggaetoneros por su parte cuentan con la capacidad de oir sólo hasta cierta intensidad de sonido. Si una onda de sonido los alcanza, ellos reaccionarán ante ella únicamente si la intensidad de dicha onda sobrepasa su umbral auditivo.

En el caso particular de las ondas emitidas por la poderosa guitarra de **Slash**, mientras los reggaetoneros tengan menor capacidad de oir sonidos estarán más a salvo de los efectos causados por la guitarra, lo que demuestra que para los reggaetoneros mientras más sordos estén, mejor.

Al recibir una onda de sonido proveniente de la guitarra de **Slash**, los reggaetoneros la procesan sólo si la intensidad de la onda es mayor a su respectivo umbral auditivo. De ser así, se marean una cantidad de tiempo en segundos que corresponde a la siguiente función lineal:

$$Mareo = IxC_m/100$$

Donde:

- I: Intensidad de la onda
- C_m: Coeficiente de tiempo de mareo

El umbral auditivo de los reggaetoneros también es en otros momentos del juego que serán explicados más adelante.

Capítulo 5

Dándole inteligencia a los reggaetoneros

5.1. Encontrar caminos (A*)

Un problema importante en el juego es dotar a los reggaetoneros de la suficiente inteligencia para poder navergar el mapa completo en búsqueda de caminos. Esta importancia se puede ver en la necesidad de poder encontrar a Slash cuando se esconda detrás de algún obstaculo del juego. En la Figura 5.1 podemos ver una visión superior de la disposición de los obstaculos en el *Stage*.

Para solucionar eficazmente este problema se tomaron y realizaron una serie de decisiones explicadas a continuación.

5.1.1. Malla de Polígonos: División del mapa

El primer paso escencial para proveer navegabilidad del *Stage* a los reggaetoneros fué dividir el mismo en **sectores triángulares** (Figura 5.2)para que la inteligencia artificial pudiese distinguir en que parte del escenario se encuentra.

Esta división se realizó a mano, sin seguir ningún algorítmo específico de división, buscando colocar los polígonos en la mejor disposición para que la inteligencia artificial pueda conseguir los caminos evitando al mismo tiempo los obstaculos.

Figura 5.1: Nuevos obstaculos en el *Stage*

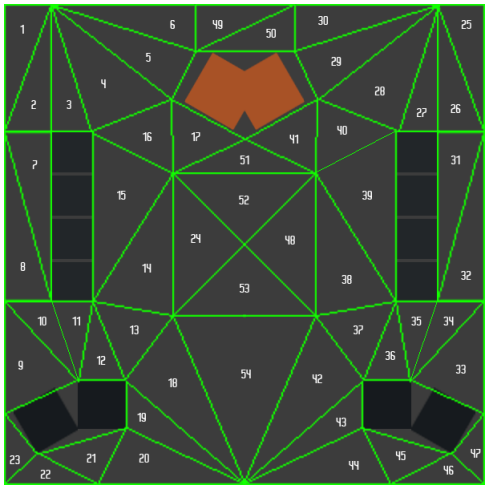

Figura 5.2: División poligonal

5.1.2. Enumeración de sectores, nodos y caminos

Para conseguir una verdadera noción por parte de los reggaetoneros sobre en que lugar del *stage* se encuentran, se enumeraron todos los sectores y se creó un grafo en donde los nodos son los **centroides**[1] de los triángulos (sectores) y los caminos entre dichos nodos fueron definidos arbitrariamente buscando que cada camino evite tener alguna intersección con el área ocupada por algún obstaculo.

El grafo obtenido se muestra a continuación:

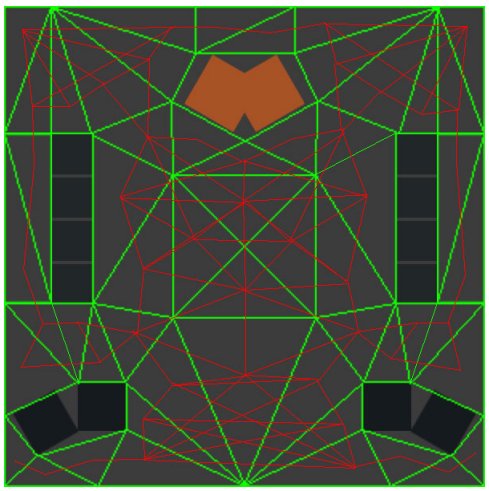

Figura 5.3: Nodos y caminos en el *Stage*

Las líneas rojas representan los caminos.

Inicialmente se definió un camino entre cada dos triángulos que compartieran un lado. Esta aproximación fué modificada y se crearon caminos entre cualquier par de nodos de forma que los caminos no se intersectaran con ningún obstaculo del juego.

Esta decisión fué tomada en base a la necesidad de recrear un movimiento más natural en el algorítmo de búsqueda de caminos mínimos hacia ciertos objetivos. Como los sectores triangulares podrían poseer áreas muy grandes en ciertos lugares del mapa, el camino a seguir sólo entre los lados de dichos

[1]El centroide de un polígono es el centro geométrico de el mismo, visto también como el punto promedio de todos los puntos del polígono

triángulos a veces parecía poco natural. La solución fue obviar los lados comunes de los sectores triangulares y trabajar unicamente con los nodos en el mapa.

5.1.3. Encontrando el camino inteligentemente

Una vez obtenido el grafo anterior, el problema presentado fué hacer que los reggaetoneros navegaran con cierta inteligencia los nodos y los caminos presentados y tomaran siempre el camino con menor distancia para llegar desde un punto dado a otro.

Para lograr esta tarea se utilizó el algorítmo **A*** como base de búsqueda de caminos del juego dado que, por ser un algorítmo de búsqueda que toma en cuenta una información extra (*heurística*), consigue de forma bastante rápida la solución óptima al problema.

La heurísitica utilizada en el algorítmo A* implementado fué tomar la *distancia euclideana* entre los nodos inicio y final entre los cuales se quiere conseguir el camino óptimo (con menor distancia recorrida).

5.2. Toma de decisiones

Cada reggaetoneros fué dotado de una implementacion de toma de decisiones que regula su comportamiento a través del juego dadas las distintas variables con las que se encuentre externas a él mismo.

5.2.1. Estados de un reggaetonero

Los reggaetoneros tienen dos conjuntos de acciones que realizar que se muestran a continuación:

Conjunto	**Acciones**
Movimiento	Wandering, Pursuing, Evading
Pelea	Hold, Hitting, Shooting

Ambos conjuntos son independientes entre sí, es decir, un reggaetonero puede estar disparando mientras persigue, escapa o vaga por el mapa, de la misma forma que puede golpear o simplemente mantenerse sin estado de

pelea.

Para implementar la toma de decisiones en cada uno de los conjuntos se tomaron aproximaciones distintas: para el movimiento se utiliza árbol de decisiones sencillo y para la pelea se implemento una máquina de estados con ciertas transiciones.

5.2.2. Árbol de decisiones para el movimiento

El árbol de decisiones implementado para el movimiento de los reggaetoneros es de la siguiente forma:

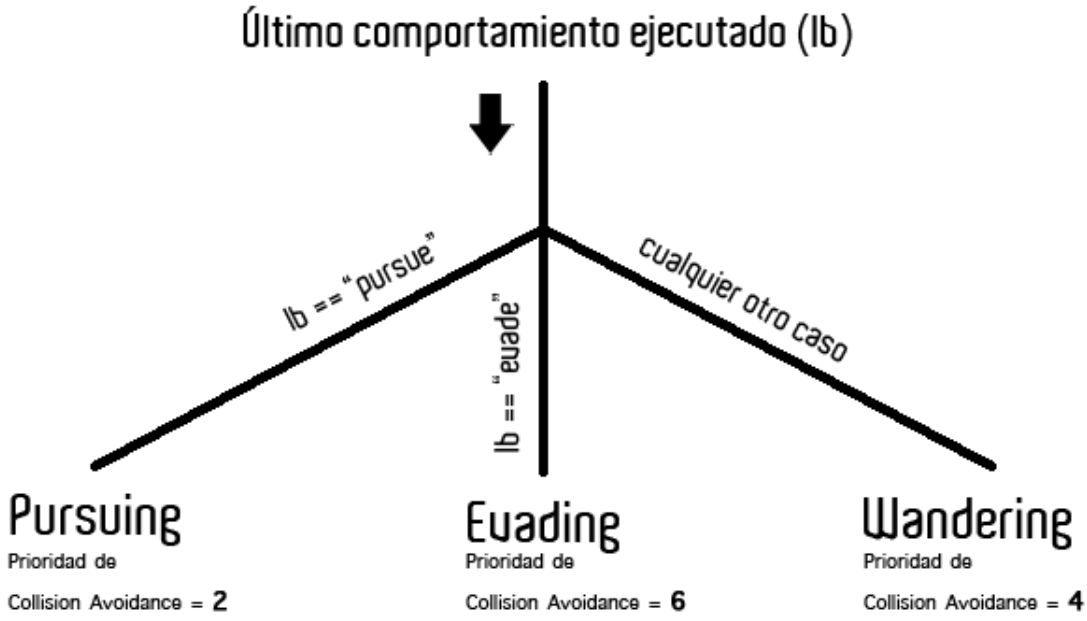

Figura 5.4: Árbol de decisiones para el movimiento

Básicamente el árbol de movimiento afecta la capacidad de esquivar obstaculos del reggaetonero. Cuando el reggaetonero esta persiguiendo a **Slash** está siguiendo los nodos de navegación presentados anteriormente, por lo tanto la probabilidad de que choque contra un obstaculo o pegue contra alguna de las paredes del stage es pequeña, por esto, se le baja la prioridad al comportamiento de evasión de obstaculos para que el *Blending* de compor-

tamientos lo tome menos en cuenta.

Por el contrario, cuando el reggaetonero esta huyendo, es mas probable que en su desesperación se tope con obstaculos y paredes que tenga que evitar, por esto, se le da la máxima prioridad al comportamiento de evasión de paredes y obstaculos.

Por último, cuando el reggaetonero está haciendo cualquier otra cosa que no sea perseguir o evadir, le da una alta prioridad a la evasión de paredes, pero no la máxima prioridad.

Para más información sobre las prioridades de los comportamientos en la inteligencia artificial del juego, ver la Tabla 6.3.1.

5.2.3. Máquina de estados para la pelea

La máquina de estados corresponiente a las acciones de la pelea se muestra en la Figura 5.5

5.3. La estrategia cobarde de los reggaetoneros

De vez en cuando los reggaetoneros se encuentran con una amenaza más peligrosa de lo que son capaces de manejar, y se ven en la necesidad de tomar estrategias evasivas para poder sobrevivir.

5.3.1. Super Slash

Nuestro personaje principal cuenta con la nueva característica de volverse super poderoso cuando logra realizar un "combo" frente a los reggaetoneros. Para lograrlo, **Slash** debe golpear a los reggaetoneros de forma seguida sin descansar hasta llenar cierta "barra de poder". Esta barra cuenta cuanto daño consecutivo ha podido realizar **Slash** en cierta cantidad restringida de tiempo. Si **Slash** logra llenar su barra de poder, adquirirá *Super Fuerza*, lo que hará que pueda disparar balas más grandes y que golpean con mucha más fuerza, tocar más fuerte su guitarra y hacerse mucho más pesado para que sea mas dificil sacarle del escenario.

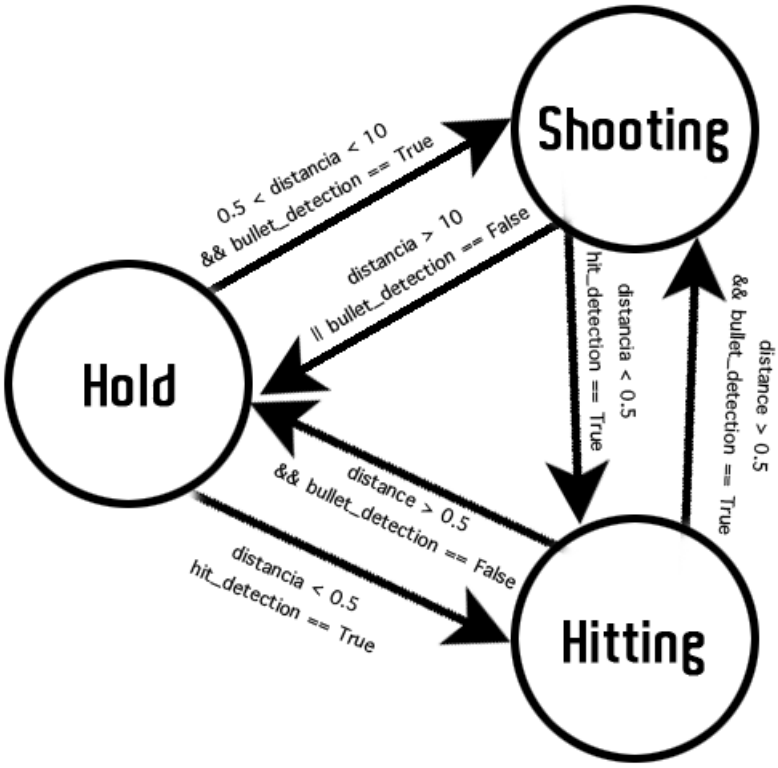

Figura 5.5: Máquina de estados para la pelea

Super Slash solo será capaz de mantener sus super poderes durante una cantidad limitada y corta de tiempo, así que durante este tiempo sus enemigos pasaran a tratar de realizar estrategias evasivas para cubrirse de los ataques de **Super Slash**.

5.3.2. La estrategia por sobrevivir

Al verse en la gran vulnerabilidad de no poder atacar a **Super Slash**, los reggaetoneros tratarán de conseguir ciertas localidades en el mapa cubiertas por varios obstaculos en la que saben que es mas difícil que puedan ser atacados por el enemigo. Estas localidades se encuentran mostradas en la Figura 5.6.

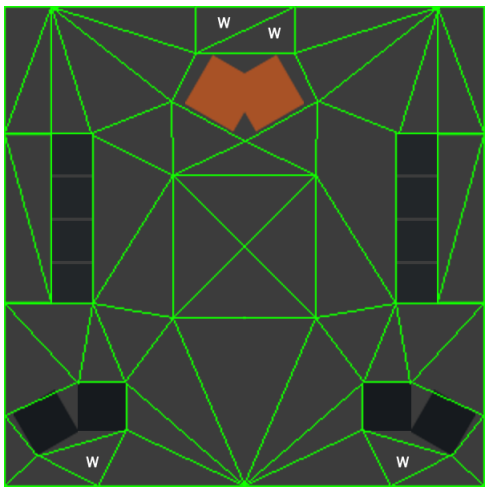

Figura 5.6: Puntos de protección en el *Stage*

Al momento de reconocer que **Slash** ha obtenido super poderes, los reggaetoneros buscarán el punto de protección mas cercano que encuentren y se dirigirán hacia él, pero antes de hacerlo le avisarán a los demás reggaetoneros que ese punto de protección ya esta tomado, para que ninguno de ellos vaya a cubrirse en el mismo punto y sean menos vulnerables.

5.3.3. El super defecto de Super Slash

Cuando **Slash** consigue sus super poderes, también consigue el importante problema de hacerse mas ruidoso cuando camina, y sus pasos se oyen

por los lugares en donde pasa.

Este defecto del super poder de **Slash** es tomado por los reggaetoneros para conseguir un poco de ventaja, pues una vez que se encuentran escondidos pasan a estar en un estado de vigilancia en el cual estan oyendo cualquier sonido que se les acerque.

Cuando un reggaetonero que ya esta cubierto en un punto de protección oye los pasos de **Super Slash**, pasa a atacarlo inmediatamente para por lo menos tratar de defenderse y bajarle aunque sea un poco la vida.

La detección de sonidos en vigilancia de los reggaetoneros es incluso un poco más refinada: Pueden detectar si el sonido viene de un lugar en donde no hay forma de que todavía **Super Slash** los pueda ver y así se quedan vigilantes sin descubrirse por error. Para lograr esto, los reggaetoneros analizan el punto de protección donde se encuentran parados y pueden ver los lugares cubiertos y no cubiertos que el punto de protección les ofrece. Si el sonido proviene de un lugar que no es cubierto desde el punto de protección, pasan a hacer una persecusión de la posición donde se originó el sonido mientras atacan. Si el sonido que oyen proviene de un lugar del cual se encuentran cubiertos, se quedan vigilantes esperando a que el sonido siga apareciendo desde allí.

5.4. El árbol de toma de decisiones completo

Tomando en consideración el estado de "mareo" obtenido al oir la guitarra de **Slash**, la toma de decisiones para los momentos de pelea, para el movimiento y para las estrategias evasivas, el árbol de decisiones que rige el comportamiento de un reggaetonero es el mostrado en la Figura 5.7

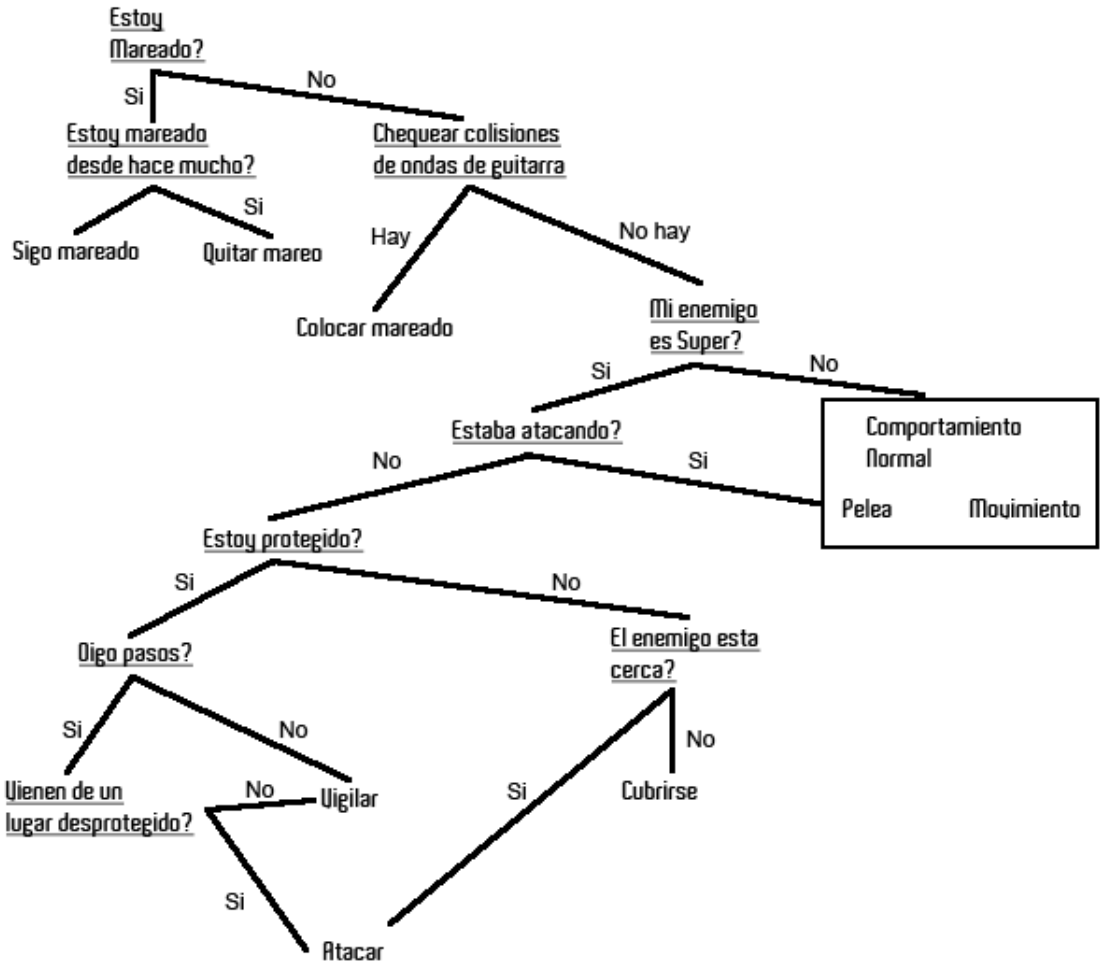

Figura 5.7: Árbol de toma de decisiones completo

Capítulo 6

Detalles de Implementación

6.1. Herramientas Utilizadas y Requerimientos

El juego está siendo desarrollado con *Python* como lenguaje de programación y *OpenGL* como librería gráfica, se está usando *Debian Linux* como ambiente de desarrollo y se corre en una computadora con 512Mb de Ram, tarjeta gráfica ATI Radeon Xpress 200M de 128Mb y procesador Intel Celeron de 1.46GHz.

Se están usando varias librerías auxiliares de *Python*, entre las cuales vale la pena mencionar:

1. **Pygame:** Conjunto de módulos de *Python* diseñados para escribir juegos.
2. **Psyco:** Módulo de *Python* para agilizar y acelerar la ejecución de código escrito en este lenguaje.

6.2. Modelos Geométricos Implementados

Para facilitar el trabajo en el campo geométrico Euclideano, se realizaron varios tipos abstractos de datos que modelan estructuras importantes como:

- **Vector3:** Tipo abstracto de datos que modela un vector en 3 dimensiones. Este TAD esta implementado para funcionar como un tipo de

datos nativo de *Python*, en donde ha sido implementada la suma y multiplicación por escalares y vectores. A parte de esto, cuenta con otras operaciones importantes como producto cruz, producto punto, longitud, orientacion, etc.

Este tipo abstracto se realizó basado en el trabajo de *Will McGugan* (`http://www.willmcgugan.com`), pero se incluyeron algunas modificaciones y correcciones necesarias.

- **Rect:** Tipo abstracto de datos que modela un Rectángulo en 2 dimensiones. Es utilizado para mantener el área ocupada por los personajes del juego, incluye operaciones como: unión, intersección, detección de colisiones entre rectángulos y entre puntos con rectángulos, clipping, entre otros.

  Este tipo abstracto de datos esta basado en el Modelo Rect que viene incluido en *Pygame*, pero fue reescrito desde cero para que pudiese trabajar con coordenadas flotantes pues el modelo de *Pygame* sólo trabaja con números enteros.

Para más información sobre estas estructuras, se puede revisar la documentación del código de cada uno de ellos, donde se explica como funcionan cada uno de los métodos.

Ambos se encuentran en la carpeta `physics` del proyecto.

6.3. Blending de comportamientos

Cada comportamiento fue escrito como una función, y para dar una interfaz general a cada comportamiento se escribió la clase `Behavior` que funciona como un contenedor de argumentos necesarios para ejecutar un comportamiento como su peso y el "handler" que hace referencia a la función que debe ser ejecutada para realizar el comportamiento.

Para realizar el "Arbitraje" entre comportamientos se diseño la clase `BehaviorGroup` que funciona como un conjunto de comportamientos que tienen una misma prioridad. Dicha clase define un método `execute` que se encarga de ejecutar uno a uno todos los comportamientos que contiene y de ir haciendo "Blending" entre ellos.

Por último, los personajes manejados por la inteligencia artificial definen un método `behave` que busca todos los `BehaviorGroup`'s asociados al mismo y los ejecutan, buscando el que tenga mayor prioridad y que devuelva una fuerza o "steering" superior a cierto umbral predefinido.

6.3.1. Prioridades de los comportamientos

Cuadro 6.1: Como se comportan los reggaetoneros

Grupo de comportamiento	**Prioridad**
Wander	1
Pursue	3
Evade	4
Collision Avoidance	5
Flocking	6

6.4. Manejo de fuerzas y movimiento sobre los reggaetoneros

Tras cada ciclo del juego, la velocidad y posición de un reggaetonero es calculada como:

$$p = p_0 + v * t \tag{6.1}$$

$$v = v_0 + a * t^2 \tag{6.2}$$

Donde p representa la nueva posición de un reggaetonero, p_0 la posición del ciclo anterior, v la nueva velocidad, v_0 la velocidad del ciclo anterior, t representa la cantidad de tiempo transcurrida entre cada ciclo y a la aceleración que lleva el reggaetonero en el presente ciclo.

Al inicio del juego, la posición de cada reggaetonero viene precargada en el programa, y su velocidad es 0. La aceleración es calculada cada ciclo siguiendo la fórmula:

$$a = \frac{\sum F}{m} \tag{6.3}$$

Donde $\sum F$ es la sumatoria de todas las fuerzas que actúan sobre el reggaetonero durante ese ciclo, y m es su respectiva masa.

La fuerza sobre los reggaetoneros se calcula como la resultante de la sumatoria entre 4 fuerzas principales que interactuan continuamente con cada uno de los personajes llevados por la inteligencia artificial. De esta forma podemos ver que:

$$\sum F = F_{bh} + F_{ht} + F_{bl} + F_{fr} \tag{6.4}$$

Donde:

- F_{bh}: Fuerza de comportamiento
- F_{ht}: Fuerza de golpes recibidos por otro personaje o por algún obstaculo.
- F_{bl}: Fuerza de balas recibidas
- F_{fr}: Fuerza de fricción

Capítulo 7

Problemas encontrados

7.1. Velocidades después de una colisión

Al chocar dos personajes el intercambio de velocidades no se realiza como debería, pués todavía no se toma en cuenta la masa del personaje y se asume que su masa "no existe" (o que es igual a 1).

A parte de esto, el choque entre los reggaetoneros hace que su orientación varíe abruptamente, mostrando un comportamiento errático por un período de tiempo corto.

7.2. Problemas con la cohesión de los "boids"

Cuando los reggaetoneros actúa en conjunto como una manada, la cohesion hacia el centro de masa calculado entre ellos genera una fuerza que a veces es mucho mayor a el resto de las fuerzas requeridas en el *Flocking*. Esto genera un comportamiento inestable en el resultado final.

La solución encontrada hasta ahora ha sido darle mucho más peso a la separación en relación con el peso dado a la cohesión.

7.3. Muchas balas = Juego lento

Cuando se disparan muchas balas, el juego empieza a ponerse lento mientras estas balas están en el aire. Una vez que tocan el suelo son eliminadas de memoria y el juego vuelve a recuperar su velocidad. Esto es debido a que las balas, como son esféricas, son pesadas para hacerles "Render", y probablemente también se debe al procesador de la computadora donde se está desarrollando el juego.

7.4. Optimización del algorítmo de búsqueda de caminos mínimos

Otro de los problemas encontrados es referente a la velocidad de procesamiento del juego: conseguir constantemente el camino mínimo para el movimiento de los personajes llevados por la inteligencia artificial requiere una cantidad de cálculos que se tienen que realizar continuamente.

Esto ocacionaba que, cuando los personajes persiguen o navegan el mapa siguiendo los grafos, el juego se pone lento. Para solucionar esto se realizó un estudio sobre los cálculos requeridos para llevar acabo el algorítmo de búsqueda de caminos mínimos (A*) y se vió que el cálculo de la heurística (la distancia euclidiana entre los puntos) era preprocesable antes de empezar el juego, debido a que los nodos no cambian de posición a lo largo del desarrollo del juego. De esta forma, al momento de cargar el nivel, los obstaculos y los personajes, también se crea el grafo, con sus sectores y nodos correspondientes, y se calcula la distancia entre cada uno de los nodos.

Luego, al agregar mucho más caminos entre los nodos (como se describió anteriormente en la Sub Sección 5.1.2, se pudo observar que no era suficiente precalcular la heurística euclideana entre todos los nodos pués, con muchos mas caminos, el costo de correr el algorítmo de A* aumenta. Para solucionar este problema se decidió tomar la misma aproximación de precalcular todo lo posible, y se decidio realizar una matriz de caminos óptimos que diera el camino óptimo entre 2 nodos con orden de acceso lineal. Construir esta matriz significó realizar el cálculo del algoritmo A* sobre todos los nodos al momento de cargar el juego, y dejar dichos calculos realizados en memoria principal.

Ambas decisiones mejoran ampliamente el comportamiento del juego al momento de tener reggaetoneros siguiendo caminos óptimos, debido a que tiene un impacto en el cálculo permanente de la heuristica entre cada punto y el nodo destino y además provee de forma directa (acceso lineal sobre una matriz) el costo de trasladarse entre un nodo y otro, y sobre el costo completo de calcular todo el camino.

7.5. Choques contra obstaculos

Para cada obstaculo en el juego, y para cada personaje, existe un radio de choque que se utiliza para detectar colisiones entre objetos. En el caso particular de un choque entre un personaje y un obstaculo, se trata de calcular cuanto es la fuerza normal que ejerce dicho obstaculo sobre el personaje que chocó contra él. El problema ha sido que cuando los personajes chocan de forma directa contra alguna de las paredes de los obstaculos, se aplica la fuerza normal que tiene dicha pared contra el personaje, pero si el personaje choca contra alguna esquina del obstaculo entonces se quedan ambos pegados y el personaje no logra salir, al menos de que otras fuerzas interactuen sobre él (por ejemplo, la fuerza de algunas balas).

7.6. Movimiento poco natural al buscar caminos mínimos

El origen y solución de este problema se encuentra detallado en la Sub Sección 5.1.2