A ti te va a caer el Axl

Segunda Entrega

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

13 de marzo de 2009

Índice general

Capítulo 1

Dándole inteligencia a los reggaetoneros

1.1. 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 1.1 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.

1.1.1. 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 1.2)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 1.1: Nuevos obstaculos en el *Stage*

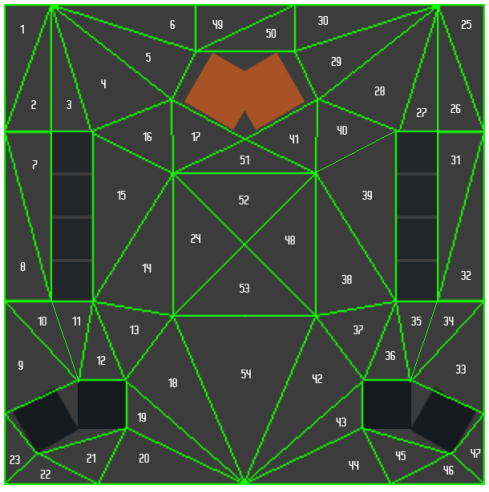

Figura 1.2: División poligonal

1.1.2. 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:

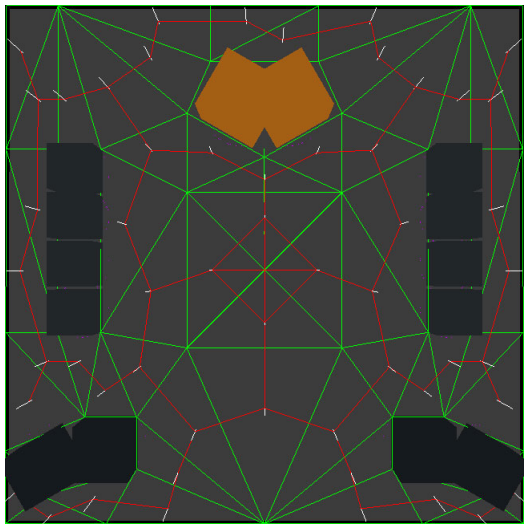

Figura 1.3: Nodos y caminos en el *Stage*

Las líneas rojas representan los caminos.

1.1.3. 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

[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

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).

1.2. 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.

1.2.1. 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.

1.2.2. Árbol de decisiones para el movimiento

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

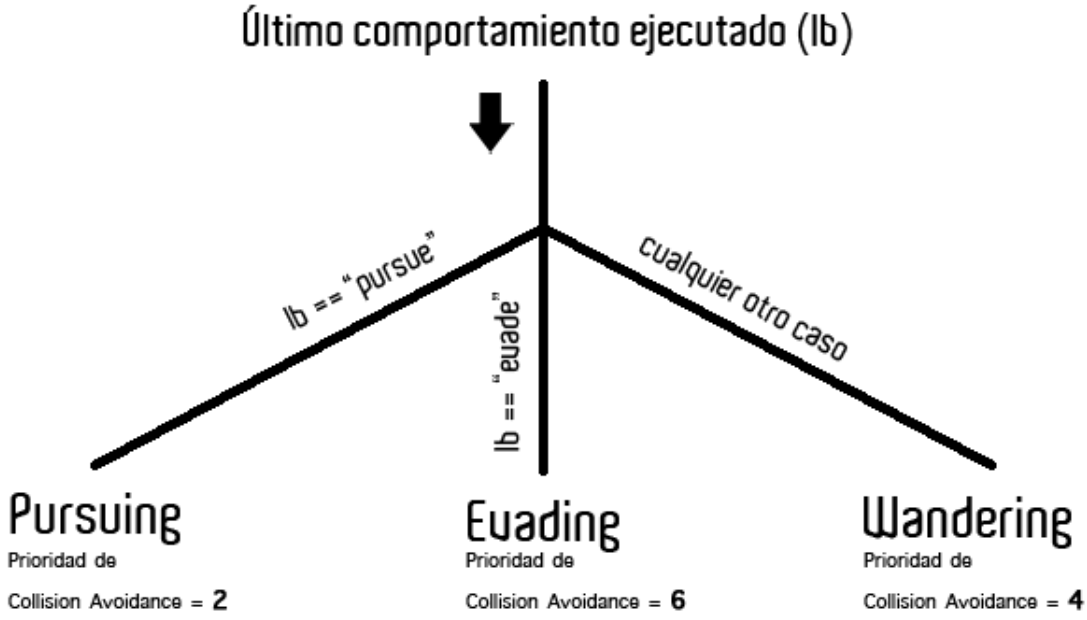

Figura 1.4: Árbol de decisiones para el movimiento

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 comportamientos 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 **??**.

1.2.3. Máquina de estados para la pelea

La máquina de estados corresponiente a las acciones de la pelea es la siguiente:

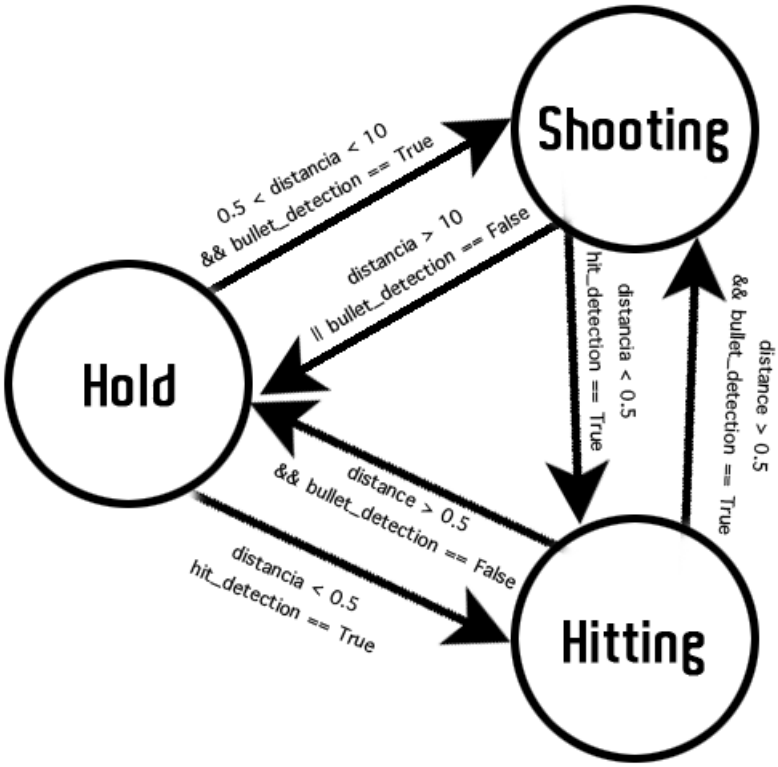

Figura 1.5: Máquina de estados

Capítulo 2

Detalles de Implementación

2.1. Nuevo modelo de fuerzas y movimiento sobre los reggaetoneros

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

$$p = p_0 + v * t \tag{2.1}$$

$$v = v_0 + a * t^2 \tag{2.2}$$

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:

$$a = \frac{\sum F}{m} \tag{2.3}$$

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:

$$\sum F = F_{bh} + F_{ht} + F_{bl} + F_{fr} \tag{2.4}$$

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 3

Problemas encontrados

3.1. 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.

Esta decisión mejora 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.

3.2. 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)