TP3

EL DESAFIO FINAL

TERCERA PARTE

Introducción

Nuestra última tarea para terminar de mostrar nuestras capacidades para cubrir el desarrollo del remake de Silent Hill 2 va a estar asociada con uno de los momentos más tensos del juego, que se da cerca del clímax de la historia: **La boss fight con el par de Pyramid Heads.**

James siendo emboscado por los dos Pyramid Heads

En esta pelea, luego de que James logra escapar de la prisión y posteriormente del lobby del hotel, es enfrentado no con uno sino con dos de los enemigos más poderosos del juego. Cada uno materializa diferentes sentimientos que llevaron a

James a llegar a Silent Hill en el primer lugar. En varios puntos de la pelea se develan detalles muy importantes para entender el final, por lo que es vital que esté bien lograda en el remake.

En la versión original, el escenario de la pelea es pequeño y acorrala al jugador. Para esta nueva versión se quiere explorar un enfoque un tanto distinto. Se quiere que la pelea esté separada en dos escenarios distintos, los cuales se repitan hasta que el jugador logre completar varios niveles.

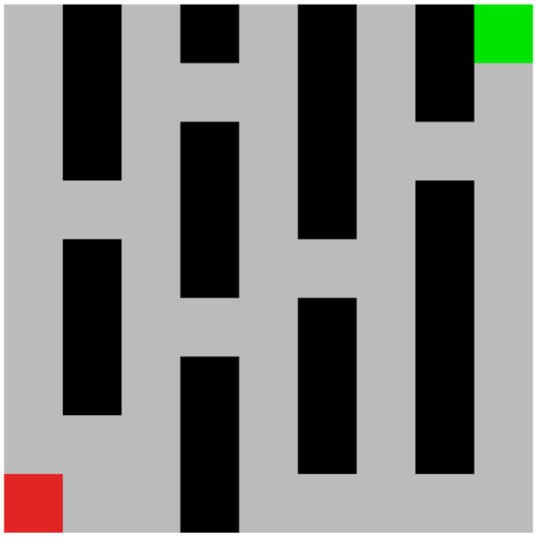
Funcionalidades a implementar

La última responsabilidad será **desarrollar un simulador interactivo** que servirá como base para el nivel anterior al jefe final del juego.

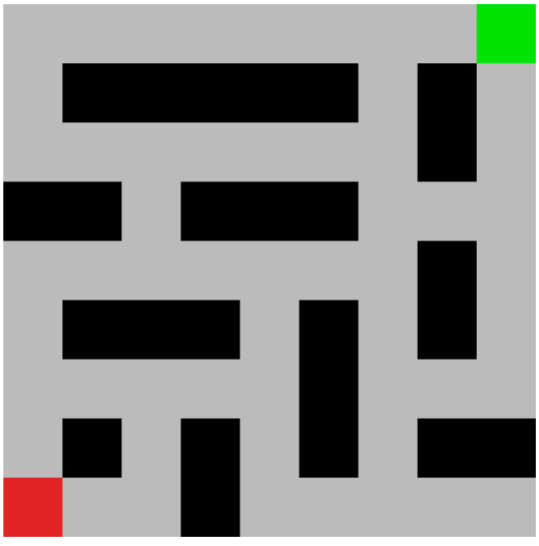
El objetivo de James será recorrer una serie de niveles laberínticos y así llegar a reencontrarse con ~~Mary~~... Sin embargo, esta tarea no será sencilla, ya que **Pyramid Head** estará acechándolo desde la oscuridad...

Concretamente, deberá recorrer el nivel, iniciando desde la “casilla” roja y teniendo como objetivo final la “casilla” verde. James solo puede moverse a otra “casilla” de forma **horizontal** o **vertical**. No se puede mover en diagonal. Moverse un casillero cuesta **10** puntos fijos.

El nivel jugable tendrá como base dos **layouts fijos**:



Layout 1



Layout 2

El layout se decidirá en base a la **altura del árbol de placas**. Si la altura es par, se utiliza el primero. Caso contrario, el segundo. Cada equipo deberá extender la firma de su árbol, proporcionando un método *altura()*, que obtenga el valor.

Al finalizar un nivel, James obtendrá **siempre** una placa, generada de forma aleatoria con un ID en el rango de 100 a 666. Adicionalmente, tendrá una chance del 20% de obtener una nueva arma, generada de forma aleatoria con una potencia de 10 a 100. Esta arma, de generarse, debe guardarse en el *inventario de armas*.

NOTA: ¿Se puede **siempre** generar una placa de forma aleatoria? ¿Qué problemas pueden ocurrir?

Como se mencionó anteriormente, **Pyramid Head** estará intentando dificultar el camino de James:

1. Los *dos* **Pyramid Head** tienen una chance fija del 50% de aparecer en el nivel actual. **Pyramid Head** solo puede y debe aparecer en un casillero válido, si la chance se cumple. Todos los casilleros son válidos, excepto los bloqueados, el inicial y final, y potencialmente el casillero que ocupe el otro **Pyramid Head**.
2. El casillero que ocupa **NO PUEDE** ser recorrido si James no tiene un arma equipada. Esto puede pasar porque no *tiene* un arma directamente, o porque el usuario decidió no equipar una (se desarrolla mas adelante). Si James tiene equipada un arma, puede pasar por el casillero. Esto destruye esa arma en el proceso y ahuyenta a ese **Pyramid Head** en ese nivel.
3. Moverse a los casilleros adyacentes al que ocupa **Pyramid Head** (esto es, verticales y horizontales, no diagonales) sin un arma equipada cuesta **cinco veces mas (x5)**. El costo *es el base* si se tiene un arma equipada.

James deberá recorrer **5** niveles generados aleatoriamente. Si logra llegar al final, se deberá felicitar al jugador, y se deberá mostrar el costo total que necesitó para llegar al final. Si en algún momento James se encuentra con una situación imposible (Pyramid Head bloquea de alguna forma la salida y no se cuenta con un arma) la partida se da por perdida, informándole al usuario.

Las condiciones iniciales del juego son libres.

Interacción con el usuario

El usuario podrá tomar decisiones en la medida en que no haya perdido:

1. Podrá moverse manualmente a una casilla adyacente válida. En caso de no ser válida, se debe informar.
2. Mostrar claramente en pantalla el **mejor recorrido posible (el que tiene costo mínimo)** para llegar de la casilla **actual** a la objetivo, junto con el costo, en base a si James tiene equipada un arma o no. Si el camino es imposible, también debe comunicarse. **OPCIONAL**: Mostrar el mejor recorrido *en general* (con o sin arma equipada, de poder equipar una).
3. Moverse a través de ese camino hasta el final, terminando el nivel actual.
4. Equipar y desequipar el arma **más fuerte** de James (y, opcionalmente, si el equipo lo implementó en la parte anterior, la **más débil**). De no tener un arma, mostrar un mensaje acorde.
5. Mostrar el puntaje actual (el costo total usado hasta el momento).

Interfaz visual

Aparte de las reglas mencionadas anteriormente, *no hay restricciones sobre las implementaciones que decidan hacer*. Para esta parte, no hay pruebas automatizadas, ni se exigirá una firma específica.

En esta línea, **hay libertad artística total** sobre cómo implementan la comunicación con el usuario, tanto para las entradas como para las salidas por pantalla. Los incentivamos a esmerarse en cómo representan el problema visualmente, haciendo uso de colores, emojis, sleeps, etc. Esto no es de carácter *obligatorio* (puede ser una interfaz sencilla, mientras sea funcional) pero puede influir en el puntaje final...

Algunas recomendaciones generales

1. El TP puede parecer muy largo y difícil al leerlo detenidamente por primera vez. Sin embargo, ya tienen (si vienen al día) todos los subsistemas y TDAs necesarios para implementar esta parte.
2. El TP, a nivel implementación, es más sencillo de lo que se pueden imaginar inicialmente. Les recomendamos **no irse por las ramas** con las potenciales clases que se podrían crear. En esta línea, lo mas idóneo es implementar una clase fachada que incluya todos los otros subsistemas y TDAs necesarios para obtener los resultados esperados.
3. Un acercamiento inicial es pensar que un grafo **es** el tablero/matriz/nivel/layout, donde luego James se estará moviendo. Como primera idea es válida, pero a la larga lleva a problemas de acoplamiento. *Un grafo es un grafo, y un tablero es un tablero*, que eventualmente podemos **representar** con un grafo. Les recomendamos que no *mezclen* ambas cosas. Esto no es nada nuevo porque ya vienen haciendo esto en los TPs anteriores (por ejemplo, el inventario de armas es una clase que *usa* un Heap pero **no es** el Heap).
4. Dicho esto, igual tiene mucho valor para ver por donde ir inicialmente pensar el problema como una matriz, pensando en índices, dibujando, etc...
5. Los layouts son fijos, por lo que no sería necesario escribir código modular y generalizable. Sin embargo, les recomendamos que piensen como se podría reducir ese código, para que no quede al final 500 líneas asignando vértices y aristas...
6. Piensen cuantos vértices y cuantas aristas son necesarias y cuantas tendrían como máximo. Adicionalmente, pueden usar mas vértices de los necesarios si de esta forma pueden identificar mas rápidamente un vértice, en base a algún criterio espacial/matricial.
7. Piensen qué algoritmo de camino mínimo sería mejor usar en este problema.
8. Aparte del método *altura()* del árbol, tienen todos los métodos necesarios en los TDAs anteriores para resolver esta parte. Sin embargo, pueden *consultar* sobre agregar algún método público que necesiten o simplifique algo. Como siempre, pueden extender los métodos privados.
9. Dejen la parte visual para el final. Primero asegúrense de que los resultados y el flujo de la simulación sea el esperado y esté libre de bugs.

Criterios de evaluación

TBA

Los criterios de corrección a nivel código no varían con respecto a las entregas anteriores.

Formato de entrega

El grupo deberá subir al campus una carpeta comprimida en formato .zip, incluyendo todos los archivos fuente del proyecto:

NOMBREGRUPO_TP3.zip

Adicionalmente, deberá subir todo el código final a la branch main de su repositorio.

La fecha límite de entrega y defensa es el **JUEVES 7/12**.

Puntaje de la tercera parte: **70** puntos.