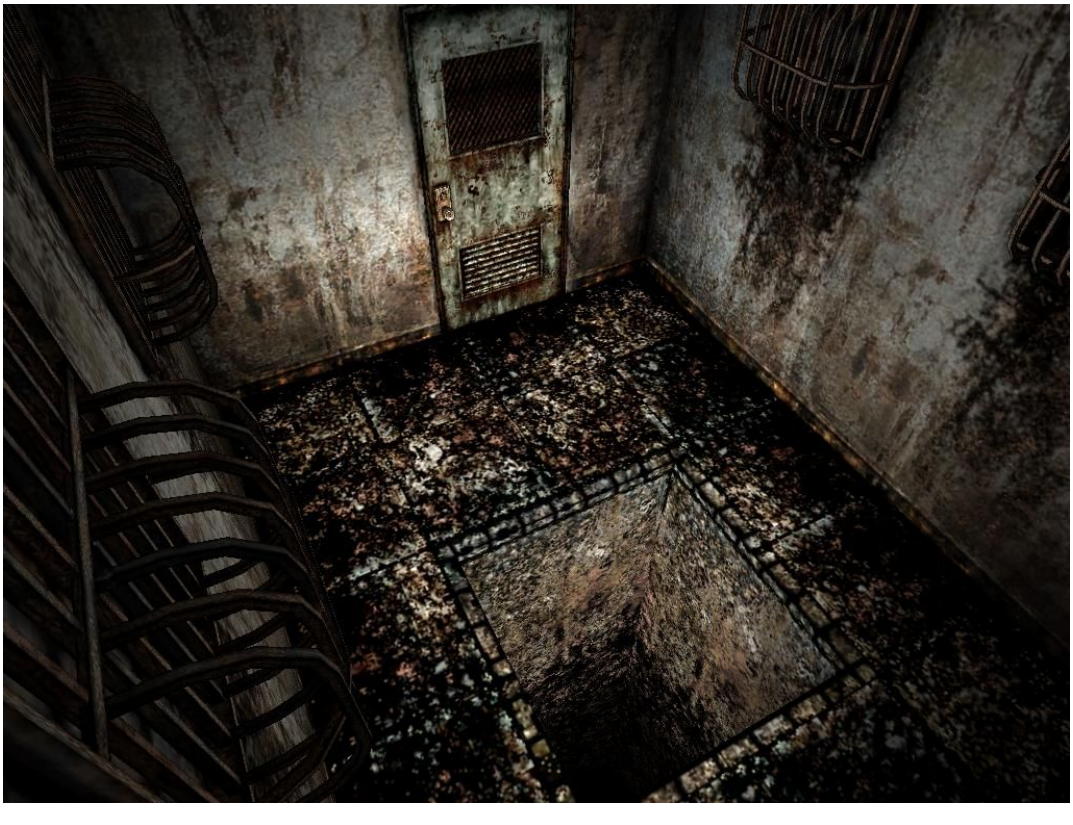
TP3

LOS DESAFÍOS DE JAMES

PRIMERA PARTE

Introducción

Llegado este punto ya nos merecemos un aumento. Logramos entregarle a nuestros jefes en Konami dos funcionalidades totalmente originales junto con una refactorización completa de una de estas. Es debido a nuestro gran trabajo que ahora nos pidieron que resolvamos otras tres nuevas mecánicas que van a ayudar a seguir construyendo la lógica del juego. La primera de estas mecánicas está asociada a uno de los niveles más climáticos del juego original, la **Prisión de Toluca**.



La prisión de Toluca se caracteriza por ser la primer área del juego donde todo empieza a tomar un giro oscuro, tanto a nivel narrativo como ambiental

En el juego original, James necesita juntar tres placas asociadas a diferentes personajes de la historia y colocarlas sobre una plataforma donde antiguamente se realizaban ejecuciones con el fin de obtener una llave hacia el siguiente nivel. Estas placas se encuentran dispersadas por toda la prisión y no se necesita recogerlas en un orden particular.

En esta nueva entrega, aprovechando que la tecnología actual lo permite, se buscó complejizar esta dinámica. Desde Konami nos comentan que ahora dentro de la lógica interna del juego se va a implementar un **Árbol de placas** para modificar como se presentan las habitaciones de la prisión al jugador en función de las que hayan sido recogidas. Esto con el fin de hacer que tanto jugadores nuevos como viejos tengan una experiencia distinta a la original, así como también distinta a la que pueda tener otro jugador que recoja las placas en otro orden. Este árbol se va a actualizar cada vez que James encuentre una placa, y se lo consulta antes de entrar a una habitación para modificar variables como cantidad de enemigos, objetos a encontrar, aspecto de la habitación, puntos de vida de los enemigos, etc.

James frente a la plataforma de ejecución sobre la que se colocan las 3 placas que necesita para continuar su aventura

Funcionalidades a implementar

Es entonces que nuestra tarea para esta nueva etapa de desarrollo es programar un **Árbol Binario de Búsqueda** donde organizar las placas. El árbol tendrá que ordenar objetos de tipo **Placa**, de los cuales se encargó otro equipo que tuvo la amabilidad de enviarnos el código asociado.

En resumen, nuestro **Árbol de placas** deberá contar con la siguiente interfaz:

- **Alta**
  - El árbol debe ser capaz de permitir el agregado de una placa recogida por el jugador y organizarla acorde a la función de comparación del árbol, por ID.
- **Consulta**
  - El árbol debe permitir que se le consulte si una placa se encuentra o no dentro de este, por ID.
- **Recorrido de descifrado**
  - El árbol debe ofrecer la posibilidad de recorrer su contenido bajo la lógica que permita descifrar el código enviado por Konami. **(*) Este punto se desarrolla más adelante.**
- **Eliminación (Opcional)**
  - Desde Konami nos explicaron que sería óptimo que el árbol también cuente con posibilidad de eliminar placas. **(*) Este punto también se desarrolla más adelante.**

Descifrado del código secreto

Si bien nuestros jefes en Konami están más que conformes con nuestro trabajo individual, para estas nuevas tareas nos asignaron un grupo de programadores de igual capacidad a la nuestra. Juntos van a tener que encarar este y los otros dos desafíos que quedan antes de terminar su parte en el desarrollo del Remake. Pero esta no es la única nueva noticia, nos revelaron también que detrás de esta tarea hay varios equipos más que están al acecho de convertirse en el equipo cuyo árbol sea el finalmente implementado en el juego.

Para incentivar un poco a los equipos, se prometió una **recompensa** para el equipo que primero descifre un mensaje empleando su árbol. A cada equipo se le va a otorgar una serie de archivos los cuales van a funcionar como prueba final del funcionamiento del TDA. Una vez el proceso de desarrollo se considere

terminado, se pondrá a prueba el código intentando descifrar el mensaje. El mensaje se descifra recorriendo el árbol de placas en un orden determinado entre los posibles (Inorder/Preorder/Postorder/En ancho) y, para hacerlo más difícil, **no se nos especificó el orden bajo el cual se descifra el mensaje**, por lo que deberemos probar múltiples recorridos hasta dar con el correcto.

Tests y primitivas

Para la **totalidad** de este Trabajo Practico (las tres partes), contaran con un repositorio donde encontraran:

- Todas las clases con la cual deberán implementar las diferentes funcionalidades.
- Las firmas de los TDAs a implementar, que no pueden ser alteradas.
- Los tests ofrecidos por la cátedra, que probaran sus TDAs.
- Para esta parte específicamente, una clase ~~**bGVjdG9y**~~ con un método ~~**Y2FyZ2Fy**~~ que cargará el archivo ~~**666.bin**~~ en su árbol.

Para ir al repositorio, hace clic aquí.

Se recomienda clonar o descargar el repositorio y comenzar a trabajar a partir de ahí. **NO** hacer pull requests sobre la plantilla.

Puntos recompensa

Sumados a los puntos del trabajo, se otorgarán puntos extra a los grupos que cumplan alguna de las siguientes condiciones:

- Desarrollar eliminación dentro del árbol: **5 Puntos**
- Descifrar el mensaje oculto primero: **10 Puntos**
- Descifrar el mensaje oculto segundo: **5 Puntos**
- Descifrar el mensaje oculto tercero: **3 Puntos**

IMPORTANTE: De llegar a descifrar el mensaje previo al anuncio del fin de la carrera (es decir, previo a que se anuncien los grupos ganadores) **enviar un mensaje privado a Julián para la fiscalización de los puntos.** Para obtener los puntos, **es necesario demostrar cómo se llegó a la solución.** De lo contrario, **el grupo queda automáticamente descalificado de la carrera.** El código que genera el mensaje debe estar versionado previamente. La solución debe respetar la firma brindada. Se recomienda fuertemente **NO** hacer reverse engineering para llegar a la solución.

Sobre el uso de Git y el versionado del proyecto

El uso de Git en este Trabajo Practico es **OBLIGATORIO**. **TODOS** los integrantes del grupo deben colaborar haciendo commits por igual. En base al desempeño individual, cada integrante tendrá un **peso porcentual** sobre el total de la nota del grupo.

En caso de hacer peer programming (esto es, desarrollar en grupo) deben usar la funcionalidad de coautor o especificar claramente en el mensaje del commit los integrantes involucrados.

Se recomienda ser ordenados de antemano, y desarrollar funcionalidades de a poco, evitando hacer commit de cientos de líneas de una sola vez y usando branches. Si necesitan inspiración, pueden revisitar la presentación de Git.

Todos los integrantes deben tener una cuenta asociada al mail FIUBA, ser colaboradores del proyecto, e invitar a su corrector para que pueda seguir el desarrollo. El repositorio debe tener visibilidad **PRIVADA**.

Entregas parciales

En caso de que el grupo logre cumplir con lo pedido en cada parte, puede coordinar con su corrector una entrega parcial, que será una instancia de feedback no formal (es decir, **NO** constituye a la nota y **NO** son obligatorias).

Las normas de la entrega final serán comentadas mas adelante (que no son muy diferentes a las anteriores...)

Puntaje de la primera parte: **40** puntos.

Música para ponerse en clima con la temática.