Pseudomemoria

Problema a resolver:

Los videojuegos nos permiten, de alguna manera, cierta libertad. Los jugadores pueden realizar ciertas acciones dentro de unos límites establecidos por los desarrolladores (se nos ocurre, por ejemplo, un juego de disparos en el que los desarrolladores nos prohíben disparar a los NPCs). Pero y si le diésemos a los jugadores una serie de mecánicas que pudiesen utilizar libremente y que tuviesen una acción aparentemente irreversible sobre ciertos elementos relevantes para el desarrollo de la historia (eliminar a un personaje principal, perder un objeto esencial para avanzar en el escenario). Parecería evidente que, llegado el momento, sería imposible llegar al final de la historia pensado, o que en este caso el argumento perdiese sentido por completo o la historia dejase de tener sentido.
Y aquí realmente reside el problema a resolver. A día de hoy, contamos con juegos en los que mediante lo que se conocen como “decisiones morales” podemos tomar diversos caminos en un juego en momentos determinados y delimitados en la historia y que afectan al desarrollo y al final de la misma, pero no dejan de ser caminos preestablecidos e historias generadas a partir de caminos claramente diferenciados y preestablecidos.
Nuestro problema va mucho más allá y busca cerrar historias, con la mayor coherencia posible, que viajarán desde una introducción hasta un desenlace sin importar lo que ocurra en el nudo. Los jugadores podrán realizar cualquier tipo de acción, como y cuando quieran y aún así tendremos que elaborar un sistema capaz de generar opciones que permitan al jugador llegar al desenlace antes o después.

Ese sería nuestro punto de partida, si bien es cierto que sería tremendamente interesante seguir profundizando en el estudio de este tipo de generación procedural y llevar la investigación, si fuese posible, todavía más allá. Aquí es donde podemos definir el problema final a resolver. Nuestra idea es elaborar un sistema capaz de resolver historias coherentes partiendo desde una presentación hasta un desenlace generado de manera totalmente procedural y aleatoria en función de las posibilidades que se nos han ido abriendo con las acciones del jugador durante la historia. Este desenlace, y ahí estaría el mayor de los retos, debería poner efectivamente punto final a la historia y el juego debería terminar de una manera coherente y consecuente con las acciones que el jugador a ido realizando sobre los elementos del juego y las decisiones que ha ido tomando de manera totalmente libre. Somos conscientes de que este problema sitúa nuestros objetivos en un nuevo y más escalón y somos conscientes de que para ello debemos de seguir un proceso de desarrollo que pasa por resolver primero el problema planteado con anterioridad. Una vez resuelto este sistema podríamos ponernos en la tesitura de tratar de resolver este nuevo y más avanzado problema, implementarlo y llevarlo a la práctica.

El videojuego como objetivo

Al definir nuestro problema hemos hablado de jugadores, mecánicas, etc. La solución que buscamos será aplicada a un videojuego y es que nuestro fin último es que nuestro desarrollo pueda ser directamente aplicada en un caso práctico y jugable, como parte de un videojuego, una experiencia interactiva por definición propia. Es en esa interactividad donde reside precisamente el interés de este proyecto y que hacen al videojuego un medio perfecto para llevarlo a cabo y demostrar su potencia y limitaciones. Las primeras aproximaciones serán totalmente conceptuales e iremos iterando tratando de implementar nuestras técnicas en casos cada vez más reales o cercanos a lo que hoy en día concebimos como videojuego. El fin último e idílico sería desarrolla en un futuro r una pequeña demo técnica jugable que nos permitiese demostrar que nuestros métodos y nuestra implementación es plausible, eficiente y eficaz, dando origen a un juego totalmente jugable y con un arco narrativo coherente.

Concepto inicial

La idea original de nuestro **Trabajo de Fin de Grado** es la implementación de un **planificador de narrativa** que sea capaz de, dada una historia, guiar al jugador a cumplirla, independientemente de las acciones y decisiones que este jugador tome. Nuestra historia presenta un estado **inicial** y estado un **objetivo** y cada uno de los cambios en el entorno constituye un nuevo estado de esta historia. Existen unos **operadores** que se pueden aplicar a estos estados para obtener nuevos. Cuando el estado en el que se encuentra el jugador sea igual al estado objetivo, se daría por concluida la historia. Para tratar de cumplir esto, el planificador haría una búsqueda por todos los posibles nuevos estados (aplicando todos los operadores) y usaría una heurística que le ayudará a saber qué cambios realizar en el entorno para poder guiar al jugador a un estado más cercano al estado objetivo.

Como extensión, también se discutió que, en caso de tener esta primera parte acabada, podríamos implementar un **generador de narrativa.** Este generador actuaría en aquellas situaciones en las que el jugador hubiese llegado a un “punto muerto” de la historia desde el cual sea prácticamente imposible dirigirle a cumplir el objetivo de esta. Si esto ocurriese, el generador de narrativa se encargaría de producir un nuevo final más apto a las circunstancias en las que se encuentre el jugador, ayudando así al planificador a dirigirle a cumplir este **nuevo objetivo.**

Uno de los retos de este trabajo es, seguramente, la implementación de la función **heurística** de la búsqueda, dado a que no solo debería encargarse de intentar mover la historia hacia el final, sino que preferiblemente debería priorizar aquellos estados que den lugar a una “mejor historia”. Esto presenta un problema ya que la calidad narrativa es completamente subjetiva y, por tanto, muy difícil de definir en ámbitos computacionales.

Prueba conceptual

Con este documento vamos a tratar de formalizar la estructura narrativa de un relato corto, intentando adaptarlo a unas estructuras lógicas que posteriormente utilizaremos en el proyecto final.

Vamos a enfocar el problema como una **búsqueda en el espacio de estados**. Este proceso, propio del campo de la Inteligencia Artificial, considera sucesivos estados de una instancia, con la meta de encontrar un estado final con las características deseadas. De manera formal, un **problema** consta de cuatro componentes:

- **Estado inicial**: estado en el que comienza la búsqueda. El **espacio de estados** es el conjunto de todos los estados alcanzables desde el estado inicial.
- **Función de expansión**: dado un estado x, esta función devuelve un conjunto de pares ordenados ⟨acción, sucesor⟩, donde acción es uno de lo **operadores** legales en el estado x, y cada sucesor es un estado que puede alcanzarse desde x aplicando dicha acción.
- **Función objetivo**: función que determina si un estado es un estado objetivo. Una **solución** de un problema es un camino desde el estado inicial hasta un estado objetivo.
- **Función heurística**: función que asigna un **coste** a cada camino y a cada acción. La calidad de una solución se mide con la función de coste; la solución óptima tiene el coste más pequeño de todas.

A partir del estado inicial y la función de expansión se genera un **grafo**, donde cada nodo representa un estado del espacio de estados, y cada arista el operador empleado para pasar de un estado a otro. El algoritmo de búsqueda utiliza la función heurística para determinar un camino de coste mínimo en el grafo hasta el estado final, que se identifica con la función objetivo.

Podemos entender el relato como una sucesión de hechos (estados) que conducen mediante distintas acciones (operadores) a un final (estado final). Efectivamente, podemos modelar nuestro relato como un espacio de estados, y por tanto podemos resolver nuestro problema mediante una búsqueda en el espacio de estados.

De esta manera podemos decir que las historias van a estar definidas por los estados por los que van a ir pasado. A su vez, cada estado será definido por las caract

Experimentación:

Para reforzar esta conceptualización e indagar en las posibles ramificaciones que podía tener un relato corto, decidimos realizar un experimento. Para ello nos planteamos un escenario de prueba similar al que tendríamos a nivel experimental para el proyecto ya implementado.

La idea es que los participantes jueguen con la historia y las posibilidades (mecánicas si quisiéramos extrapolarlo a un videojuego) que les damos para ver como

logran llegar de la presentación de la historia o estado inicial al desenlace o estado final preestablecidos, sin guiarlos de ninguna manera por el camino y sin poner mayores restricciones que los propios operadores y elemento de la historia.

Tomamos como base una aventura textual que aparece en uno de los últimos episodios de la temporada 4 de la serie *Mr. Robot (eXit)*, e intentamos adaptarla. Un integrante del grupo hace las veces de investigador. Este diseñó un relato aparentemente sencillo, con tres elementos básicos que modelan cada estado: personajes (A, B y C), lugares (Mazmorra, Túnel, Playa y Barco) y objetos (Cerilla, Nota y Ron). Así, un estado podría representarse como:

```
class Location {
    bool open;
}

class Item {
    string name;
    bool used;
}

class Character {
    string name;
    list<Item> inventory;
    Location currentLocation;
    bool standing;
    bool tired;
}

class State {
    list<Character> characters;
}
```

Se proponía un estado inicial en el que:

- A está en la mazmorra
- B está en la mazmorra
- B tiene una nota
- A tiene una cerilla
- La cerilla está sin usar
- La nota está sin usar
- El ron está sin usar
- La mazmorra está cerrada (un barril de ron bloquea el túnel)
- El túnel está abierto
- La playa está abierta
- El barco está abierto
- B está cansado
- A está activo

- B está de pie
- A está de pie

Utilizando los siguientes operadores legales, se pedía elaborar una sucesión de estados:

- Personaje X entra en lugar Y (MOVER)
- Personaje X da objeto Y a personaje Z (DAR)
- Personaje X utiliza objeto Y (USAR)
- Personaje X habla con personaje Y (HABLAR)
- Personaje X abre lugar Y (ABRIR)
- Personaje X cambia de postura (CAMBIAR)
- Generar personaje C (GENERAR)

El estado objetivo debía cumplir, al menos, dos de estas tres condiciones:

- La mazmorra está abierta
- B está en el barco
- A está en el barco

El resto de los integrantes del grupo harán las veces de probadores y elaborarán diferentes historias tomando como base los elementos clase y ese estado inicial definido por unos hechos concretos. Deberán, por tanto, ir creando una historia haciendo uso de los diferentes operadores con el objetivo de alcanzar el estado final u objetivos marcados por el investigador.

De esta manera podremos observar hasta qué punto o como puede escalar la generación de historias teniendo un número reducido de operadores, Operators, y elementos de tipo Class:

<u>Probador 1:</u>

A y B están en la mazmorra. A B se le cae la nota. A lee la nota, es una orden de asesinato para matarle. A sale corriendo, B le sigue despacio porque está cansado. A va al túnel. (mover) A mira dentro del barril que bloquea el túnel y coge una botella de ron. A tira el ron al suelo. (usar) B llega al túnel. (mover) A enciende la cerilla (usar) y PRENDE FUEGO a B. A va a la playa. (mover) A se monta en el barco y se va. (mover)

<u>Probador 2:</u>

Ambos personajes A y B se encuentran encerrados en la mazmorra. B habla con A sobre una nota que lleva encima. Dicha nota contiene instrucciones de cómo escapar de la mazmorra. Debido al cansancio de B, B le da la nota a A para que pueda leerla e interpretarla. A usa una cerilla para poder ver en la oscuridad y leer la nota. A conoce el modo de desbloquear el túnel gracias a la información de la nota. A abre la mazmorra y va al túnel. B también se sale al túnel. A y B usan el barril, desbloqueando el túnel. A y B han hecho demasiado ruido, y un guardia C aparece para detenerlos. A y B huyen, pero B está demasiado cansado y es atrapado por el guardia. B es trasladado a una nueva mazmorra. A logra subir al barco. A usa el barco para llegar a la playa.

Probador 3

Abel está activo y habla con Babilon que está cansado. Babilon quiere dormir, pero le dice a Abel que no puede dormir porque el sonido de las gotitas que caen del techo de la mazmorra no le dejan.

Abel se desplaza hacia el barril. Trata de levantarlo, no puede, pesa mucho. Necesitará que Babilon le ayude, pero necesita descansar. Hay una pequeña hendidura en el barril, no cabe una mano. El barril está lleno de Ron.

Abel habla con Babel, el cansancio le hace delirar, le entrega a Abel una nota y le dice que si no logra salir de allí que haga el favor de entregársela a su amada. Babilon le da Nota a Abel. Abel le dice a Babilon que si es al contrario busque a una mujer. Le da su nombre y el nombre por el que se refiere a ella de manera cariñosa. Es su hermana. Le da un mensaje de cariño para la mujer.

Abbel se vuelve a acercar al barril, usa la nota (hace un rollo con ella y la introduce por la hendidura del barril hasta que se empapa). Vuelve hacia Babilon y usa la nota empapada en alcohol, le ofrece un trago y la escurre liberando todo el líquido que esta había absorbido.

Así varias veces, Babilon se emborracha. Babilon se sienta, cae dormido.

Abel está cansado. Abel se sienta, Abel está sentado. Abel cae dormido.

Abel está activo. Abel está de pie.

Babilon está activo. Babilon está de pie.

Ambos se mueven hacia el barril. “Lo usan”, lo levantan y despejan la entrada.

Abel abre la mazmorra.

Abel y Babilon entran en el túnel.

El túnel está muy oscuro.

Abel tiene una cerilla, Abel usa cerilla y prende la nota, la luz les permite vislumbrar un agujero por el que colarse.

El agujero es muy estrecho, Abel habla con Babilon.

Babilon quiere escapar, Abel quiere escapar.

Abel se mueve a la mazmorra, Babilon se mueve a la mazmorra.

Cogen el barril, se mueven al túnel. Colocan el Barril en la apertura. Abel usa el papel ardiendo y lo tiran al barril, el barril explota, hace un socavón en el túnel, la luz entra por un hueco que ha dejado la explosión a media altura. Está a una altura inalcanzable para sortearlo un solo personaje.

Después del esfuerzo y por la borrachera Babilon está cansado.

Abel está activo.

Abel habla con Babilon. Acuerdan que Abel empujará a Babilon y luego este tirará de Abel para que los dos puedan llegar a la playa.

Babilon tiene la opción de pedirle a Abel sentarse y descansar para volver a estar activo y poder tirar después de Abel. Babilon oculta su cansancio y decide que es el momento de actuar.

Abel empuja a Babilon y Babilon llega al agujero en la pared. (Babilon está cansado). Babilon confiesa a Abel que está cansado y que no puede ayudarle a salir. Podría descansar, pero la altura es demasiado peligrosa y de la caída se haría daño condenando los dos a quedarse en el túnel. Abel maldice a Babilon. Babilon huye y se mueve a la playa.

En la playa Babilon ve un barco atracado. Camina despacio hacia el mismo. Antes de poder entrar al barco se encuentra con una mujer (Se ha generado personaje C en el acceso al barco previamente), la mujer es la capitana y le dice su nombre. Es el mismo nombre que le facilitó Abel en la mazmorra. La mujer le dice que es pirata, que viene a salvar a un hombre preso. Observa las vestiduras de Babilon y deduce que acaba de escapar de la mazmorra en la que estaban preso, que ha escuchado una explosión. Le pregunta por Abel.

Babilon miente a la mujer, le dice que él y Babilon eran buenos amigos, que él le salvó la vida a Abel y que Abel se la salvó a él, pero que en la explosión que les sacó del túnel perdió la

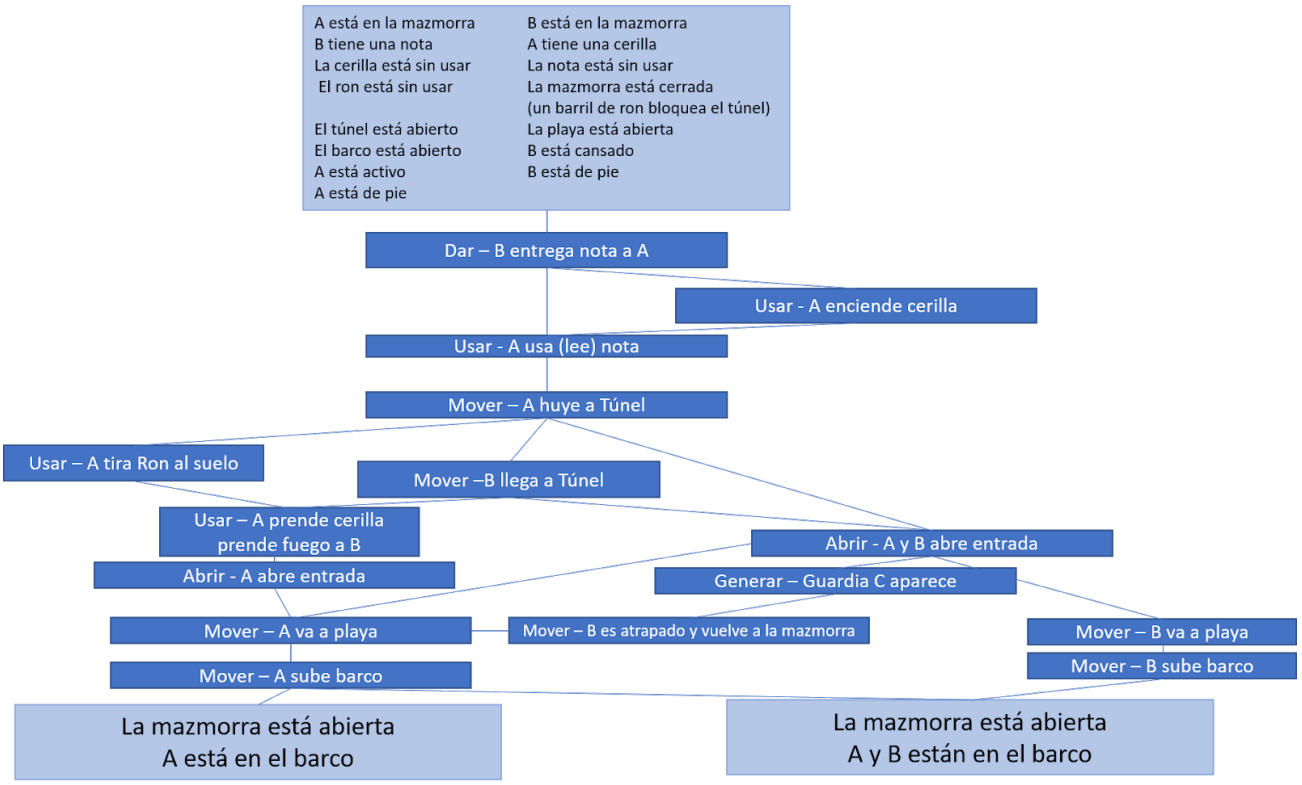
vida de manera heroica. Babilón hace memoria, en su cabeza revolotean distintos motes. Finalmente nombra a la capitana con el que cree que era el mote cariñoso con el que Abel se refería a su hermana, acierta. La capitana entiende que solo un buen amigo sabría como su adorado hermano Abel la llamaba cuando eran niños.

La capitana le concede acceso al barco, Babilon se mueve al barco.

Zarpan….

Conclusiones:

Podemos comprobar cómo a pesar de carecer de una variedad de clases complejas y multitud de operadores las posibilidades y los resultados obtenidos son totalmente distintos, no solo en argumento, si no en profundidad y extensión. Podemos pensar que si algo así ocurre en un ejemplo tan sencillo como este, los resultados obtenidos iterando una vez completada la implementación e incrementando el número de clases y de operadores pueden derivar en resultados extensos, complejos e incluso difíciles de manejar, tratar y procesar. Las historias generadas por cada miembro del grupo ponían de manifiesto una alta variabilidad combinatoria de los elementos de nuestro relato. Nos dimos cuenta de que debíamos reducir drásticamente el número de variables y operadores si queríamos trabajar con un grafo manejable a nivel computacional.



Teniendo en cuenta que utilizamos algoritmos de búsqueda y probablemente nuestro principal será A* es de gran utilidad visualizar el ejemplo anterior a modo de grafo y ver cómo se relacionan los diferentes estados aplicando operadores y de qué manera evolucionan las historias vistas de manera más gráfica.

Para el grafo hemos utilizado las historias del probador 1 y el 2.

Prueba inicial de implementación (bolsas de dinero)

En una primera instancia, a forma de prueba de implementación, construimos un entorno **sin jugador**, pero en el que ya exista un inicio y un objetivo, de forma que el planificador tenga que realizar cambios sobre ese entorno para llegar a este último. Por ejemplo, una de nuestras primeras pruebas consta de una cuadrícula 6x6 en la que existen NPCs y "bolsas de dinero". Ningún NPC tiene dinero en el inicio y el estado objetivo es que todos los NPCs tengan dinero, es decir, que hayan llegado a una posición donde existe una bolsa de dinero.

Con este concepto tan simplificado ya podemos empezar a trabajar en el planificador, el cual debe ir calculando qué cambios realizar (mover a un NPC, en este caso) en cada estado para acercarse al estado objetivo, observando todos los posibles estados a los que puede cambiar.

Trabajamos en la implementación de una **clase *State***, para definir un estado de esta historia. Esta clase tiene varios **atributos:** la posición de las bolsas de dinero y los

personajes (con su posición y si tienen dinero o no); y las **funciones** *Expand*, *Objective* y *Heuristic*.

- La función ***Expand*** se encargaría de, dado un estado, devolver todos los posibles nuevos estados a los que se puede llegar. Esto se hace realizando el producto cartesiano de todos los operadores que pueden aplicarse sobre el estado actual, lo cual resulta en todos los posibles nuevos estados. Con esta función, quedaría representada la historia en forma de Grafo, ya que los estados de una historia constituirían los vértices y los operadores las arístas.
- La función ***Objective*** dictamina si un estado es objetivo o no. En este caso, en esta función se comprobaría si los NPCs tienen dinero. En caso de que sí, entonces ese estado es el objetivo.
- La función ***Heuristic*** busca la bolsa de dinero más cercana, pues así se acerca más al objetivo.

Problemas de implementación

Hacemos uso de la librería Advanced Algorithms para C# para poder realizar algoritmos de búsqueda sin tener que invertir todo el tiempo que conlleva implementarlos. Sin embargo, al intentar hacer uso de esta, nos dimos cuenta de que para realizar una búsqueda en el espacio de estados la librería requería que construyesemos el grafo manualmente, es decir, construyendo cada vértice y cada arísta. La librería no parece contar con una función que, dados las cuatro elementos básicos de cualquier problema de búsqueda (Estado inicial y las funciones *Expand*, *Objective*, *Heuristic)*, construya el grafo por nosotros.

Este problema paraliza completamente nuestra prueba debido a que no podemos realizar una búsqueda y poner el planificador en marcha sin poder representar la historia como un grafo.