Javier Goldschmidt - VJT6A - Final - IA II

El GOAP se ubica en una taberna pirata.

OBJETIVO:
Que la IA logre escapar con la sangre de elfo sin morir en el intento.

Para eso hay un **estado del mundo** compuesto por las siguientes variables:

- HasEscaped (bool)
- Coin (int)
- Alive (bool)
- Drunkenness (float)
- Equipment (string)
- BrokenCabinet (bool)
- AngryOwner (bool)
- CabinetOpen (bool)
- PlayDarts (bool)

Hay una UI que te permite elegir presets para generar un GOAL World State que es el que se toma como referencia para tomar las decisiones. Ese GOAL World State es un objeto llamado “Preferences” que lo que hace es recopilar una lista de **Funcs** que devuelven true o false según el estado actual del juego satisface el **GOAL,** se utiliza para el **Satisfies** y la **Heurística** del **A***

No todas las combinaciones dan planes.

PRESETS:
Título: Steal elf blood breaking cabinet (rompiendo el gabinete: BrokenCabinet = true)
Acciones: Compra una botella -> rompe la puerta de vidrio del gabinete con la botella -> roba la sangre de elfo -> se escapa

Título: Steal elf blood stealing key (no romper el gabinete)
Acciones: Roba la llave de la habitación del dueño -> abre la puerta del gabinete con la llave -> roba la sangre de elfo -> se escapa
El dueño se entera de esto, AngryOwner = true

Título: Steal cabinet and die (no romper el gabinete, no enojar al dueño)
Acciones: Roba el gabinete -> los clientes de la taverna le matan (no llega al objetivo)

Título: Steal cabinet while people drunk (no romper el gabinete, no enojar al dueño, no morir)
Acciones: Comprar rondas de bebida para todos -> robar el gabinete -> escapar

Título: Steal elf blood winning cabinet key (no romper el gabinete, no enojar al dueño, no morir, no robar el gabinete sino la sangre de elfo)
Acciones: Retar al dueño a un duelo de dardos -> jugar a los dardos.

- Si gana a los dardos (bajas chances): gana la llave -> abre el gabinete -> se lleva la sangre de elfo
- Si pierde a los dardos: los clientes de la taverna le matan.

Título: Steal elf blood winning cabinet key drunk
Acciones: Comprar rondas de bebida para todos -> Retar al dueño a un duelo de dardos -> jugar a los dardos.
- Si gana a los dardos (altas chances): gana la llave -> abre el gabinete -> se lleva la sangre de elfo
- Si pierde a los dardos: los clientes de la taverna le matan.

ACCIONES EN CONCRETO:

Steal Key: Roba la llave del dueño de la taberna
- **Precondiciones:** Equipment == none
- **Efectos:** Equipment = key / AngryOwner = true

Get Breaking Object: Paga por una botella que puede ser utilizada para romper el gabinete
- **Precondiciones:** Coin >= 70 / Equipment == none
- **Efectos:** Coin = Coin - 70 / Equipment = BreakingObject

Pay For Grog: Paga por una ronda de bebidas, lo que sube el nivel de emborrachamiento de la taberna
- **Precondiciones:** Coin >= 10 / Equipment == none
- **Efectos:** Coin = Coin - 10 / Drunkenness += 0.2f

Challenge Owner: Invita al dueño de la taberna a un duelo de dardos
- **Precondiciones:** Equipment == none
- **Efectos:** Equipment = Darts

Play Darts: Invita al dueño de la taberna a un duelo de dardos
- **Precondiciones:** Equipment == Darts
- **Efectos:** Equipment = None / PlayDarts = True

Steal Key Without Angry Owner: La única forma de sacarle la llave al dueño sin que este se enoje es ganándole la partida de dardos
- **Precondiciones:** PlayDarts == True / Equipment = None
- **Efectos:** Equipment = Key

Open Cabinet: Abre el gabinete que contiene la sangre de elfo (el objetivo) usando la llave
- **Precondiciones:** Equipment == Key
- **Efectos:** Equipment = None / CabinetOpen = True

Break Cabinet: Abre el gabinete que contiene la sangre de elfo (el objetivo) rompiéndolo con un objeto contundente

- **Precondiciones:** Equipment == BreakingObject
- **Efectos:** Equipment = None / CabinetOpen = True / BrokenCabinet = True

Steal Elf Blood: Agarra y se lleva la sangre de elfo (objetivo)
- **Precondiciones:** CabinetOpen == True
- **Efectos:** CabinetOpen = False / Equipment = ElfBlood

Steal Cabinet: Agarra y se lleva el gabinete entero conteniendo la sangre de elfo (objetivo) y muere
- **Precondiciones:** Equipment == None
- **Efectos:** Alive = False / Equipment = Cabinet

Steal Cabinet Without Dying: Agarra y se lleva el gabinete entero conteniendo la sangre de elfo (objetivo) y mientras están todos borrachos y distraídos
- **Precondiciones:** Drunkenness >= 0.3f / Equipment == None
- **Efectos:** Equipment = Cabinet

Escape: Se escapa (objetivo)
- **Precondiciones:** Equipment == ElfBlood || Equipment == Cabinet
- **Efectos:** HasEscaped = True

ALGORITMO A*
Implementa Time-Slicing tanto para el GOAP como para el PathFinding de los personajes

TRABAS:
Los principales motivos que me trabaron durante la realización del trabajo fueron:

- Al realizar el .Clone() del WorldState (necesario para el algoritmo de A*) me había olvidado por completo que le había agregado mayor número de variables al WorldState por lo que efectivamente no se estaba copiando el objeto entero sino solo una parte. Esto hacía que el algoritmo de A* se “olvidase” de varios detalles del mundo anterior, por lo cual nunca llegaba a una planificación exitosa. Estuve debuggeando muchísimo tiempo esto hasta que finalmente logré dar con qué estaba causando el problema.

- Cuando cambié el sistema de A* del GOAP para que funcione por Time-Slicing con una corrutina en vez de que entregue la lista entera de una se me rompió todo el sistema de navegación de los personajes, teniendo que efectivamente refactorizar todo el código del mismo.

- Drunkenness es una variable float que indica el estado de emborrachamiento del mundo. La misma va de 0.0f a 1.0f, en un momento no me dí cuenta que en vez de 1.0f había escrito 10f, por lo cual el planificador no podía realizar algunos planes

debido a que supera la cantidad de dinero necesario para pagar por tantas rondas de grogg.

- En un momento le mandé a Emilio Dazza un mail consultando si el final cumplía todas las consignas. Me contestó que la heurística estaba mal hecha ya que la misma se componía del costo del camino + un bonus dependiendo de si se pudo escapar. Eso implicaba que en el 99% de las expansiones se guiaba por el costo y no por la heurística, haciéndolo funcionar básicamente como un dijkstra la mayoría del tiempo. La solución era hacer que las otras variables afecten también a la heurística, en función del objetivo seleccionado. El mayor problema con eso era que no tenía bien idea de cómo hacer eso, tenía que configurar todo un nuevo sistema que arme puntajes de heurística dependiendo de lo que el usuario hubiera seleccionado en la UI. Esto iba a ser una auténtica pérdida de tiempo.
  Lo estuve pensando durante bastante tiempo hasta que me dí cuenta que podía usar el GOAL State del mundo y comparar Current State con el Goal State para ver cuantas condiciones se habían cumplido, a mayor número de condiciones cumplidas, menor costo!
  Esto solito hizo que la performance del GOAP mejorase exponencialmente.