DOCUMENTACIÓN

Clase: Inteligencia Artificial	Práctica: 2 [Árboles de decisión]
Alumno: Miguel Rodríguez Gallego	

Motivos de mi práctica

Quise intentar realizar un sistema de estados simple mediante el *decision tree* aportado por el docente.

Proceso de creación

Comencé por re implementar de manera simple el movimiento de la primera práctica de la infección de zombis.

Después creé el *decision tree* con el esquema de acciones claro del funcionamiento de la presa y del cazador como un subobjeto dentro del mismo.

Programé las funcionalidades de las estadísticas de stamina y rutinas de movimiento mediante el código previo, por lo que no me dio muchos problemas. Usé el target detector que cree en la práctica anterior que crea una esfera como collider que detecta a los objetivos que se encuentran en una determinada layer, dando el más cercano para perseguir.

Así entonces, le paso el target al árbol cuando lo detecta en el update, para así poder detectar que acción tomar.

Entonces, empecé poniendo que si la entidad tiene suficiente estamina, puede perseguir/escapar, o sea, entrar o no a las decisiones del árbol de decisiones, o repostar hasta tener estamina.

Si tiene estamina, realiza puede revisar si está cerca la otra entidad, y si es así, si está lo suficientemente cerca como para atacarla y perseguirla. Si no es así, tiene una rutina de vagar por el mapa tranquilo.

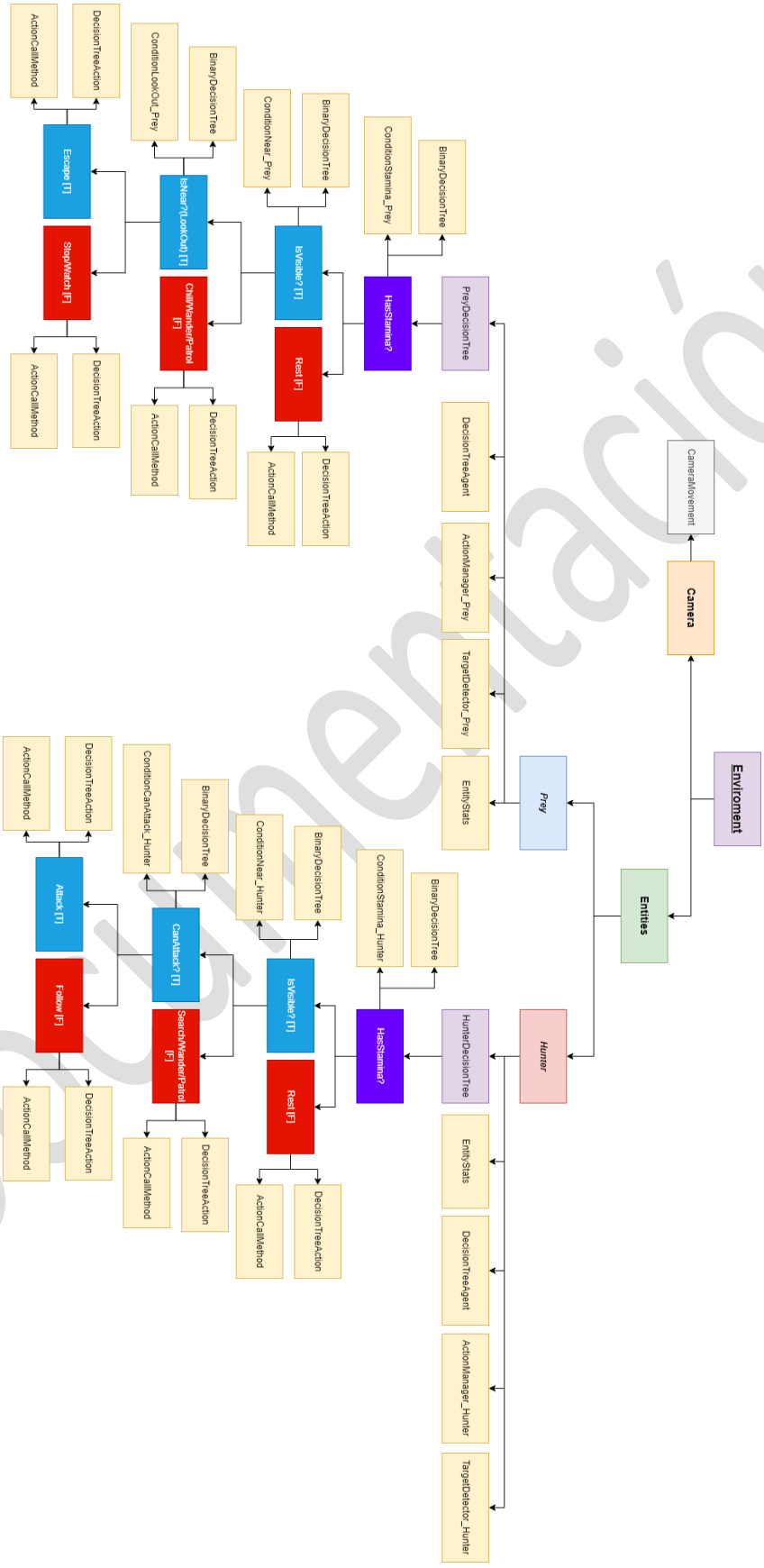
Me he referido como a uno solo porque ambos personajes funcionan igual, la única diferencia es que la presa, si tiene un cazador muy cerca se queda observando atento y quieto sin hacer ruido, mientras tanto, el cazador persigue a la presa hasta matarla si la detecta.

Por último, establecí unos canvas para visualizar el estado de las acciones dentro de la simulación.

Dificultades y como se resolvieron

Realmente la mayor dificultad fue entender como funcionaba el decision tree, por el resto fue bastante simple, fue cuestión de ir haciendo, no encontré mayor dificultad.

Diagrama de clases



- Camera
  - CameraMovement: Poder mover la cámara cuando presionas espacio y tmb pausarla con la misma tecla para que salga el cursor y puedas interactuar con el ambiente.
- Hunter
  - DecisionTreeAgent: *Root* del árbol, toma una decisión en cada frame de lo que debe hacer la entidad.
  - ActionManagerHunter: Donde se encuentran los métodos de las acciones principales del árbol, o sea, a donde llama el árbol para que estas se ejecuten.
  - TargetDetector_Hunter: Detectar los targets/prey más cercanos mediante un collider de una esfera. También calcula el target más cercano.
  - EntityStats: Manejo de las estadísticas de estamina del personaje.
  - BinaryDecisionTree: Nodo que toma una decisión con solo 2 posibles resoluciones.
  - ConditionStamina_Hunter: Condición para revisar si la entidad tiene estamina y se puede mover o no.
  - ConditionNear_Hunter: Revisa si el target está cerca, o sea, si lo detecta el portador.
  - ConditionCanAttack_Hunter: Revisa si está lo suficientemente cerca como para ir a atacar al target y matarlo.
  - ActionCallMethod: Llama a un método del objeto seleccionado en la escena.
  - DecisionTreeAction: Representa una acción a ser cogida.
- Prey
  - DecisionTreeAgent: *Root* del árbol, toma una decisión en cada frame de lo que debe hacer la entidad.
  - ActionManagerPrey: Donde se encuentran los métodos de las acciones principales del árbol, o sea, a donde llama el árbol para que estas se ejecuten.
  - TargetDetector_Prey: Detectar los targets/hunter más cercanos mediante un collider de una esfera. También calcula el target más cercano.
  - EntityStats: Manejo de las estadísticas de estamina del personaje.
  - BinaryDecisionTree: Nodo que toma una decisión con solo 2 posibles resoluciones.
  - ConditionStamina_Prey: Condición para revisar si la entidad tiene estamina y se puede mover o no.
  - ConditionNear_Prey: Revisa si el target está cerca, o sea, si lo detecta el portador.
  - ConditionLookOut_Prey: Revisa si el target está muy cerca, por lo que debe tomar la decisión de pararse a observarlo en silencio y quieto.
  - ActionCallMethod: Llama a un método del objeto seleccionado en la escena.
  - DecisionTreeAction: Representa una acción a ser cogida.

Autocrítica

El código es bastante limpio y está organizado, por lo q por ese lado pienso que está bien. Sin embargo, se que podría optimizarlo mucho más, ya que creo scripts parecidos para la detección de targets d cada uno, por no decir que son prácticamente iguales, y eso no me convence.

Sin embargo, todo funciona a la perfección, por lo que estoy contento de poder haberla acabado por completo.