Goal Oriented Action Planning - Demo

How does the Entity work?
The Entity has 2 goals, namely ‘SatisfyHunger’ and ‘CollectFood’. This is so the Entity has a simple behavior where it walks around through the world, collecting food and feeding itself when it’s hungry.

The goal ‘CollectFood’ can be satisfied by the actions ‘CollectFoodFromBush’ and ‘CollectFoodFromCarcass’. ‘CollectFoodFromCarcass’ is preferred by the Entity because it has a lower cost (and provides more food), but the Entity does need a knife before this action becomes viable.

The goal ‘SatisfyHunger’ can be satisfied by one action, namely the action ‘EatFood’. Before ‘EatFood’ is viable however, the Entity needs to have food in its inventory.

```
public override void OnActionSetup(IGoap igoap, List<Condition> state) {
    entity = (Entity)igoap;

    // Only set the action to viable if the entity has any food
    isViable = entity.amountOfFood > 0;
}
```

How does the demo work?

Entity

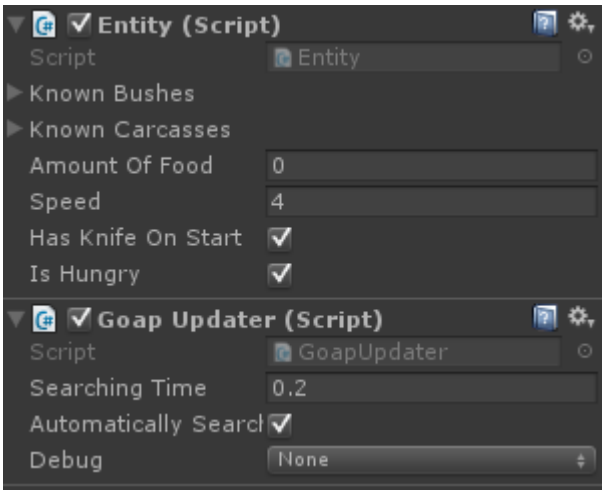
The Entity class has implemented the IGoap by including the 3 required lists, namely: state, availableGoals and availableActions. These lists are initialized and filled in the Awake function. After the availableGoals list is filled, for each Goal the OnGoalInitialized function will be called.

The state list is filled with one condition, namely the “hasKnife” condition. The value of this condition depends on the public boolean hasKnifeOnStart.

The Entity has two lists of Transform, named knownBushes and knownCarcasses. These are used by the Actions CollectFoodFromBush and CollectFoodFromCarcass. Every Transform indicates a spot which the Actions can use to harvest food.

In the Update function a simple timer runs, which sets the ‘isHungry’ value to true 4 seconds after the Entity has eaten.

On the Entity GameObject, both the component Entity and GoapUpdater are added.



Notes:

- Since Actions/Goals do not inherit van MonoBehaviour but from ScriptableObject, the correct way to add these to the availableActions/avilableGoals list is by ScriptableObject.CreateInstance<Action>()

Actions
What follows is a list of Actions the entity is given in the Awake function. Remember you can always update the list with available Actions on runtime.

Actions	*Preconditions*	*Effects*
MoveTo	-	[“inRangeOf”, transform]
CollectFoodFromBush	[“inRangeOf”, transform]	[“collectFood”, true]
CollectFoodFromCarcass	[“inRangeOf”, transform] [“hasKnife”, true]	[“collectFood”, true]
EatFood	-	[“satisfyHunger”, true]

Notes:

- Instead of using a precondition, the action ‘EatFood’ checks in the OnActionSetup if the entity.amountOfFood is bigger than zero.
- Both ‘CollectFoodFromBush’ and ‘CollectFoodFromCarcass’ update their precondition in the OnActionSetup. This way the precondition “inRangeOf” gets the correct value instead of the standard null value.

Goals
What follows is a list of Goals the entity is given in the Awake function. Remember you can always update the list with available Goals on runtime. Don’t forget to call the ‘OnGoalInitialize’ function for each available Goal.

Goals	*Succes*
CollectFood	[“collectFood”, true]
SatisfyHunger	[“satisfyHunger”, true]

Notes:

- The goal “SatisfyHunger” has the highest priority. This means the entity will opt for this goal as soon as it’s relevant.