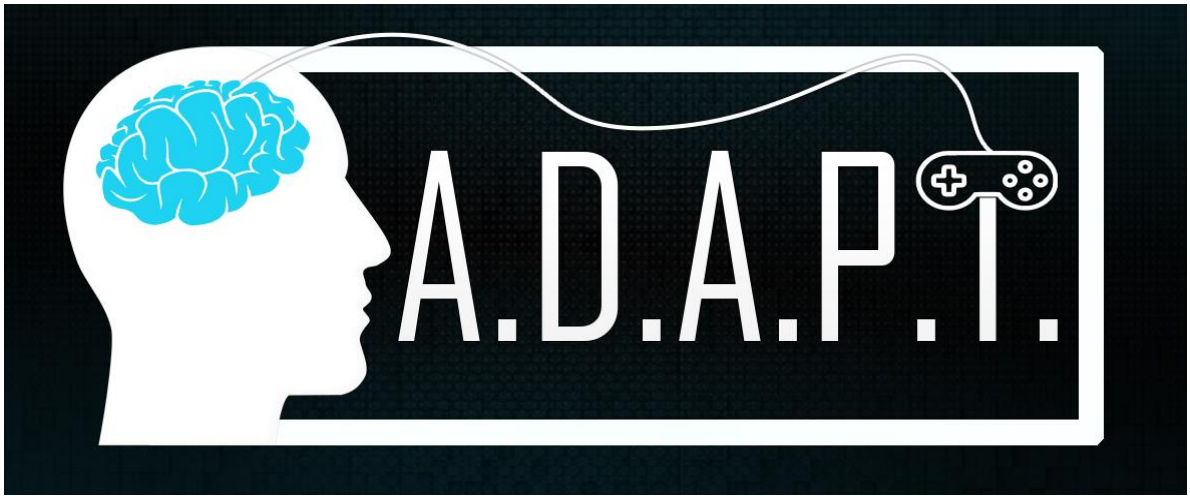
V. 1.0.



A.D.A.P.T. USER GUIDE

GOAL-ORIENTED ACTION PLANNING TOOL FOR ARTIFICIAL INTELLIGENCE IN UNITY

A.D.A.P.T. – UNITY AI TOOL
YAGOMIRA@GMAIL.COM

-INDEX:

1.- A.D.A.P.T. as GOAP tool :

Currently Artificial Intelligence (AI) is a technology that is constantly developing and improving. Used in many areas such as the video game sector, where through the analysis of the environment and the ability to adapt to it, behaviors similar to those of humans can be seen. This project is based on that idea, on developing a tool that allows in a simple way to be able to create an intelligent behavior and fill those deficiencies that the Unity video game engine does not offer in a simple way to the user.

To achieve this goal, one of the most innovative systems was integrated in terms of the creation of AI in video games, known as Goal-Oriented Action Planning (GOAP) which will allow those Non-Playable Characters Non-Playable Characters (NPC) to be provided with a set of actions, and depends on the situation one or other will be executed, with the target of achieve a predetermined goal.

2.- Some Concepts:

-Resources: the resources will be the actual Preconditions/Effects/Goals, A.D.A.P.T. use this for do more easy the concept of the actual states to achieve. Actually A.D.A.P.T. has **4 types** of resources:

1.-WorldResource: used for the desired locations, like some building, other Agent, … is the same as PositionResource but use GameObject.

2.-PositionResource: used for desired locations, use Transforms.

3.-InventoryResource: used for items can stack, like wood, minerals, …

4.-StatusResource: use for abstract situations for example, when a character is sick.

-States:

-Agent_States: used as local inventory for the actual agent, only this agent can access to the inventory.

-Global_States: global inventory, shared between all agents, for example: a gold storage of the town (*strategy games like Age of Empires*).

-¿How to perform a plan?: actually to perform a plan with the GOAP system you should achieve the preconditions when the actual state of the agent, for example:

-Agent has a state of inventory where: “Wood = 100”

-And a precondition to “Craft Wood Table” is “Wood = 100” then the Action can be perfomed because the preconditions are achieved.

-For other side, the Action has some effects of its execution, this effects are for example, after play the “Craft Wood Table” Action, then the actual state of the Agent: “Wood = 0” because the Agent use all of the wood in the Action.

-And a consequence of possible actions can achieve a goal, the goal can be anything you want, for example:

-“Stop work”, and this goal makes the agent cannot do more actions.

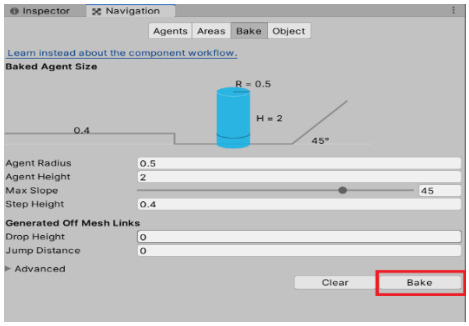
-And you can achieve this goal if you por example has a consequence of actions you can achieve because you fulfill with all preconditions:

“Make Wood Table” > “Go to Bed” > “Stop Work”

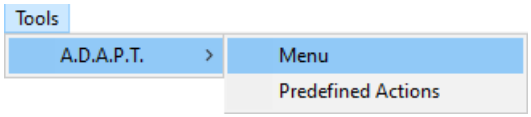
3.- Create a new agent from zero :

For transform your desired GameObject into an GOAP Agent you should follow the next steps:

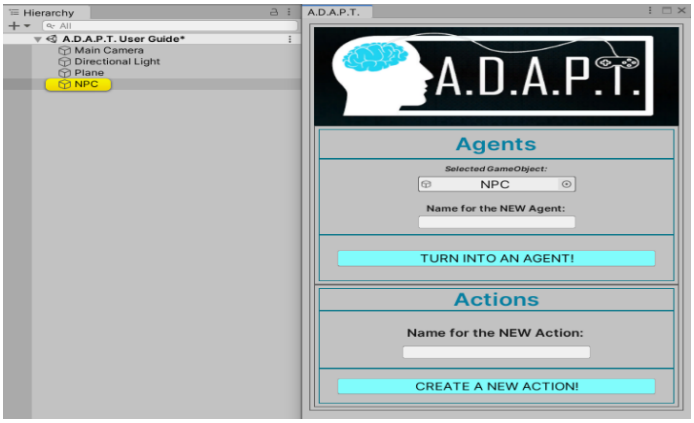
-FIRST OF ALL: IF YOU WANT TO CREATE A NEW AGENT, YOU SHOULD BAKE THE MESH TO NAVIGATE AS YOU DO WITH NAVMESH:



1.- Open A.D.A.P.T. Menu: `Tools > A.D.A.P.T. > Menu`



2.-With the Menu open, select your desired GameObject in the hierarchy:



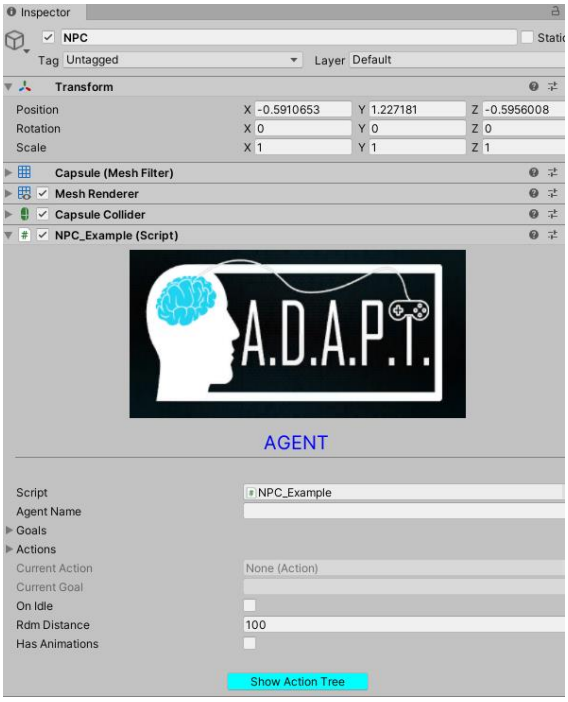
At the moment into the “Selected GameObject” box should appear your GameObject name.

3.-Now in the “Name for the NEW Agent” box, insert the name of the new agent:

*Take care with agents that exists, the tool will not duplicate the scripts, only check if exists.

4.-Click the button “TURN INTO AN AGENT!” and **<u>wait some seconds</u>** to the new script is being attached to your selected GameObject.

<u>-Your agent should look like these:</u>



5.- Now you have your new agent, with the correct code attached. The new script should look like these in the code editor:

```
using UnityEngine;
public class NPC_Example : Agent
{
    new void Start() //DON'T MODIFY ANY LINE OF THIS FUNCTION!!!
    {
        AddGoals();
        /************/
        base.Start(); //DON'T DELETE THIS LINE!!!
                      /************/
        ManageStates();
    }

    public void AddGoals() { }

    public void ManageStates() { }
}
```

*Format the document if the tabulation is incorrect.

-ABOUT AGENT-SCRIPT FUNCTIONS:

-Start() : **Shouldn't be modified**. Initiates the goals, states and all necessary variables from the base class.

-AddGoals() : here you cand add your goals to perform by the agent.

```
Example:

InventoryResource goal_2_resource =

(new InventoryResource("minedGold", 100.0f, 30, 500, false));

//InventoryResource(GoalName, ValueToAchieve, Priority, Limit,
isConsumable)

goals.Add(new Goal(goal_2_resource, false)); //Add the new goal.
```

* Check the actual predefined agent and actions to see the details about add new goals.

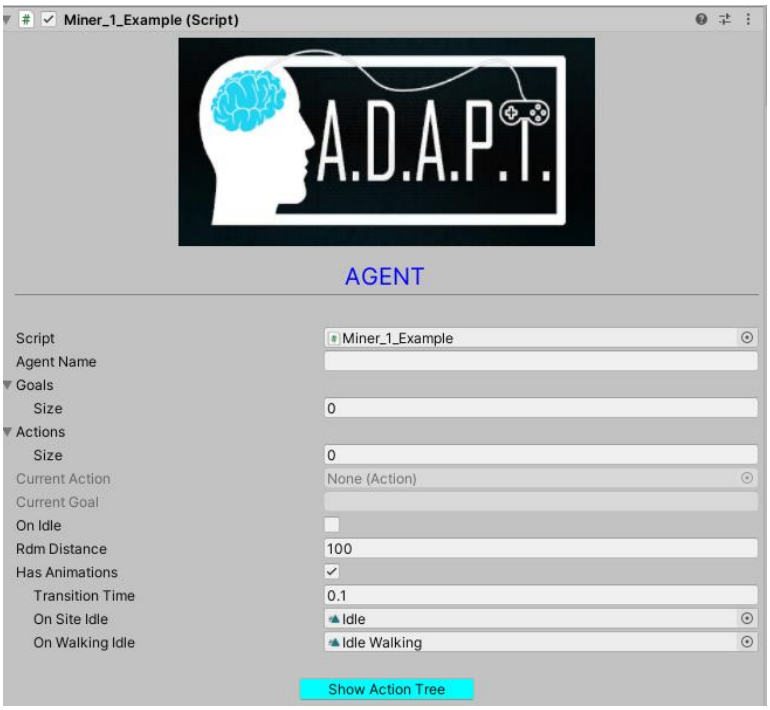
-ManageState() : here you can add your agent states. Global or local (like a shared or individual inventory).

```
Example:

global_states.AddInventoryItem("minedGold", 0f);

//Add as inventory the actual (Name, initialValue)

//Use: "global_states" for shared states and "agent_states" for
individual ones.
```

* Check the actual predefined agent and actions to see the details about add new goals.

4.- About Agents:

Here are some concepts you should know about the Agent:



-Goals[] : check the point ***"By code or by inspector"*** for see the information relative.

-Actions[] : YOU SHOULDN'T MODIFY IT!. This actions are the actions you will attach to the agent, and will appear here as summary.

-Current Action & Current Goal: current action performing and goal to achieve.

-OnIdle: in case of agent cannot find some plan to achieve, then the agent will enter in a "Idle" mode. If this variable is **true**, the agent will remain on the same site. If its **false**, the agent will move randomly across the map with a maximum of ***"Rmd Distance".***

-Rmd Distance: distance Agent can achieve randomly.

-ABOUT AGENT ANIMATIONS:

-HasAnimations: if you want to have a character with animations, set to **true**.

-Transition Time: time between play one animation and the next one.

-OnSiteIdle: Idle on site animation.

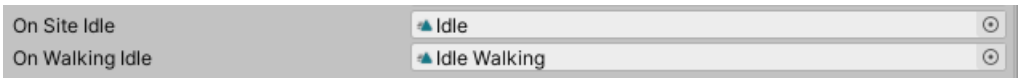
-OnWalkingIdle: walking idle state animation.

* If you use the same animation for the idle states (on the agents inspector) and the Action animation, change the name of one of them.

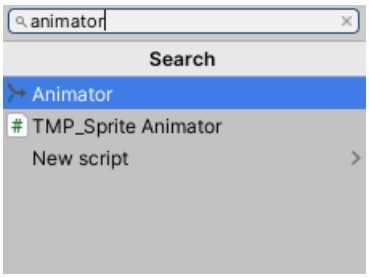
-ADD ANIMATIONS TO AN AGENT:

-If you want to play animations with the actions of the agent then follow the next steps:

1.- Add Idle animations to the Agent:



2.- Add an Animator to the Agent:

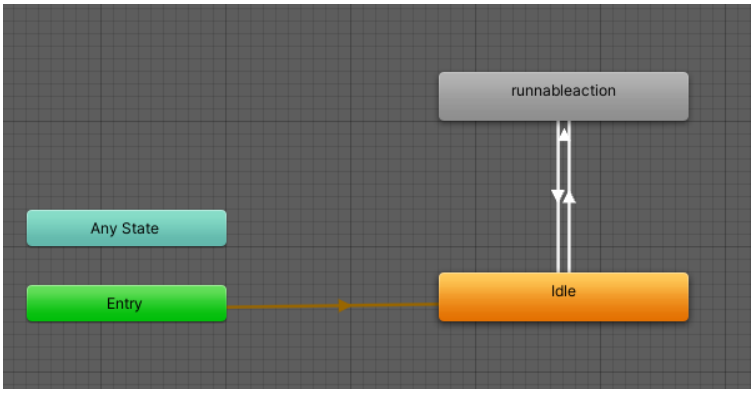


3.- Select the **A.D.A.P.T. Animator controller**:

Now you have your agent with animations!.

*In the action "Duration" field, you can add a value of duration if you want. By default, this field takes the duration of the animation, but if you insert a higher value, then the animation start a loop.

-ADVICE: DON'T MODIFY THE ANIMATOR. DON'T TOUCH THE ANIMATIONS INSIDE THE STATES. ONLY ADD IT VIA "Action Animation" OF THE DESIRED ACTION TO PLAY!

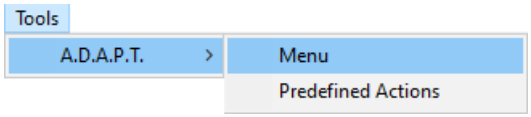


*The Animator use a simple state machine with two states and one loop. So you shouldn't worry about push the new states, A.D.A.P.T. will do for you!.

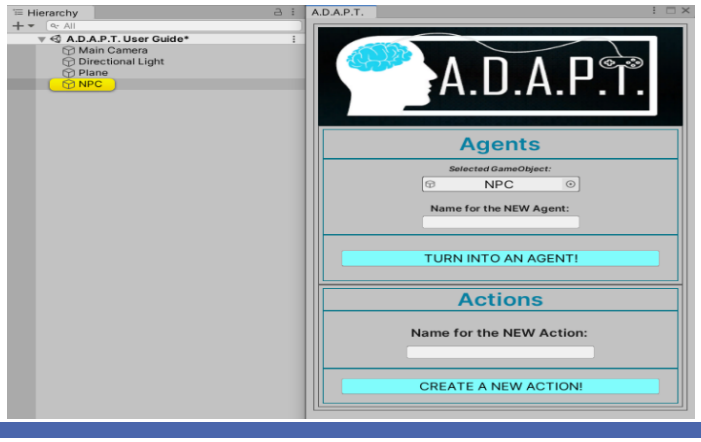
5.- Create new actions :

In case of need the creation of some new actions, you should follow the next steps:

1.- Open A.D.A.P.T. Menu: `Tools > A.D.A.P.T. > Menu`

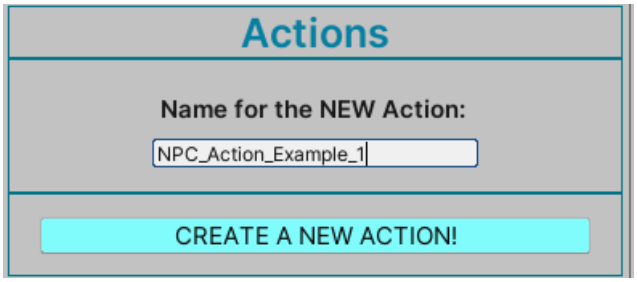


2.-With the Menu open, select your desired GameObject in the hierarchy:



3.-Check if the actual selected GameObject is an Agent. If is not, you should follow the previous steps of the previous chapter.

4.- Now in the “Name for the NEW Action” box, insert the name of the new action:



*Take care with actions that exists, the tool will not duplicate the scripts, only check if exists.

4.-Click the button “CREATE A NEW ACTION!” and **wait some seconds** to the new script is being attached to your selected Agent.

-Your agent should look like these with the new Action:

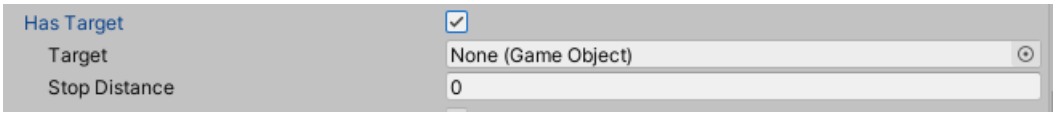
5.- Now you have your new action, with the correct code attached. The new script should look like these in the code editor:

```
using UnityEngine;
using UnityEngine.AI;
public class NPC_Action_Example_1 : Action
{
    string a_name = "ActionName";
    Agent agent;
    NavMeshAgent actual_agent;

    void Awake()
    {
        /******DON'T DELETE THIS LINES!!!******/
        actionName = a_name;
        agent = gameObject.GetComponent<Agent>();
        //WARNING MESSAGE!
        Debug.Log(" <color=blue> Action: </color> " + actionName + " <color=blue> has preconditions / effects added by code,</ color > <color=red> DON'T ADD MORE VIA INSPECTOR!.</color>");
        /************/

        /************/
        //HERE YOU CAN ADD YOUR PRECONDITIONS // EFFECTS
        /************/
        //In case of add preconditions/effects, uncomment the next lines:
        //preconditions_list.Add(GoTo_preconditions);
        //effects_list.Add(GoTo_effects);
    }

    public override void PerformAction()
    {
        //Uncomment next line if you need some navmesh:
        //actual_agent = gameObject.GetComponent<NavMeshAgent>();
        //Use 'finished = true;' when finish the action.
    }

}
```

*Format the document if the tabulation is incorrect.

*You shouldn't delete the lines between comments.

-ABOUT_ACTION:



-HasTarget: use it if the actual action need to move to desired location.

***IF ITS TRUE, THEN TARGET CANNOT BE NULL!.**

-Target: desired location to achieve.

You should add some Precondition of WorldPosition where the resource_value == Target.

*In case of don't do it, and exists some precondition with resource_value == null and is the tipo of World/Position resource, then it will take the actual Target as value.

-Stop Distance: desired distance from target to stop the Agent navmesh.

-ABOUT AGENT-SCRIPT FUNCTIONS:

-Awake() : only initializes the necessary variables and components.

-PerformAction() : HERE SHOULD BE YOUR CODE TO PERFORM!.

```
Example:

ResourceStruct precondition_1 =

new ResourceStruct("isNear", new WorldResource("isNear", target,
5, 50.0f

//new ResourceStruct(NameOfResource, new…

//…TypeOfResource(NameOfResource, GameObject, Priority, Limit))

preconditions_list.Add(GoTo_preconditions);


//Preconditions -> preconditions_list // Effects -> effects_list
```

* Check the actual predefined agent and actions to see the details about add new goals.

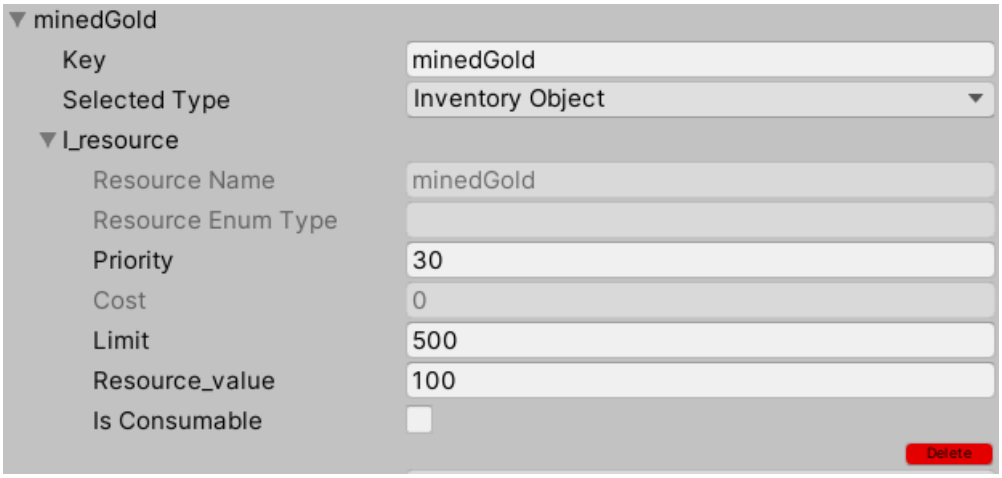
6.- By code or by inspector :

Actually you can use code or inspector to add the preconditions/effects and goals.

-Preconditions/Effects via inspector:

1.- Click the button **“Add Precondition”** or **“Add Effects”** respectively.

2.- Insert the desired values in the correspondent rows:



-Key: name for the desired resource (Effect/Precondition).

-Selected Type: type of the desired resource (*World, Position, Inventory or Status*).

In function of the Selected Type, one resource or another will be displayed: (**W_resource : World, P_resource : Position, I_resource : Inventory, S_resource : Status**).

-Inside resource:

-Resource Name: will copy the text of key.

-Resource Enum Type: type of resource loaded on execution.

-Priority: priority to achieve the actual precondition/effect, **assign more priority for the most important resources**.

-Cost: will be calculated. In case of **World/Position** resources: calculate the distance between the agent and the Target. In case of **Inventory** resources: calculate the cost in function of *resource_Value* amount of inventory, in case of a value of 1000, the cost will be 10 and so on. In case of **Status** resources: the cost will be 1 by default.

-Limit: used for the distance or inventory maximum amount to reach. In case of **World/Position** resources: **the target cannot be more far of the limit distance, or the action will not be part of the planner**. In case of **Inventory** resources: error message will appear if the resource_Value is bigger than the limit. In case of **Status** resources: the limit will be 0 by default, not use.

-Resource_Value: value to reach by states.

-isConsumable: you can use this variable for the consumable inventory resources: like potions, ammo, etc (**You should call it in the PerformAction code**).

* You can use the button "Delete" if you want to delete this specific resource.

-Preconditions/Effects via code:

-For add preconditions/effects via code, you should use the next structure:

```
ResourceStruct nameOfResource =new ResourceStruct(…);
```

-And inside the new ResourceStruct:

-Key: the name of precondition/effect.

-Resource:

-In case of World Resource: **new WorldResource:**

new WorldResource*("Same as Key" (string), Target (GameObject), Priority (int), Limit (float)*)

-In case of Position Resource: **new PositionResource:**

new PositionResource*("Same as Key" (string), Target (Transform), Priority (int), Limit (float)*)

-In case of Inventory Resource: **new InventoryResource:**

new InventoryResource*("Same as Key" (string), Value (float), Priority (int), Limit (float), isConsumable (bool)*)

-In case of Status Resource: **new StatusResource:**

new StatusResource*("Same as Key" (string), Value (bool), Priority (int)*)

-The result should view like this:

```
new ResourceStruct(new ResourceStruct("isNear", new
    WorldResource("isNear", target, 5, 50.0f));
```

-Finally add the new precondition/effect, to the actual list for view the new element in the inspector:

```
preconditions_list.Add(nameOfResource); //PRECONDITIONS*

effects_list.Add(nameOfResource); //EFFECTS*
```

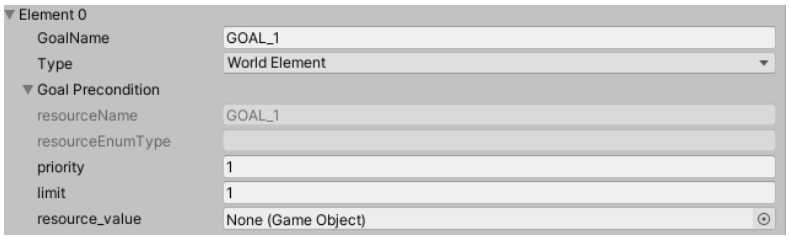
* Add to the list in the Awake() function.

**ADVICE: IN CASE OF ADD PRECONDITIONS/EFFECTS ONLY BY CODE AND NOT BY INSPECTOR, YOU SHOULD INSERT SOME ACTION THAT IMPLIES TO CHANGE THE VALUES OF STATES TO REACH THE PRECONDITION/EFFECT.

-Goals via inspector:

1.- Increase the size of Goals by +1.

2.- Insert the desired values in the correspondent rows as you do with preconditions/effects



-Extra rows:

-hasAction: in case of the goal should play some animation when is reached.

-goal_Action: animation to play.

-cost: calculated same as preconditions/effects.

-Goals via code:

-In the Agent script (**inside function AddGoals()**), you should use the next structure:

```
Resource nameOfResource = new Resource(…);
```

*Where resource can be: ***WorldResource, PositionResource, InventoryResource, StatusResource.***

-And inside the new Resource:

-Resource:

-In case of World Resource: **new WorldResource:**

new WorldResource*(“Same as Key” (string), Target (GameObject), Priority (int), Limit (float)*)

-In case of Position Resource: **new PositionResource:**

new PositionResource*(“Same as Key” (string), Target (Transform), Priority (int), Limit (float)*)

-In case of Inventory Resource: **new InventoryResource:**

new InventoryResource*(“Same as Key” (string), Value (float), Priority (int), Limit (float), isConsumable (bool)*)

-In case of Status Resource: **new StatusResource:**

new StatusResource*(“Same as Key” (string), Value (bool), Priority (int)*)

-The result should view like this:

```
InventoryResource goal = (new InventoryResource("minedGold",
                    100.0f, 30, 500, false));
```

For other side, **YOU CAN ONLY USE CODE TO ADD THE STATES**:

-In the Agent script (**inside function ManageStates()**), you should use the next structure:

-For shared states (GLOBAL): `global_states`

-For individual states (LOCAL): `agent_states`

-In this case, you have 4 type of items:

-worldElements: for World resources.

-positions: for Position resources.

-inventory: for Inventory resources.

-status: for Status resources.

-And you can use the next functions for **Add, Remove, Modify or Increase/Decrease** the values of the states:

AddWorldItem(string state),
AddPositionItem(string state),
AddInventoryItem(string state, float initialValue),
AddStatusItem(string state, bool initialValue): for initialize a state.

RemoveWorldItem(string state),
RemovePositionItem(string state),
RemoveInventoryItem(string state),
RemoveStatusItem(string state): for remove a state.

ModifyInventoryItem(string state, float newValue),
ModifyStatusItem(string state, bool newValue) : for modify the current value of a state.

IncreaseInventoryItem(string state, float newValue), DecreaseInventoryItem (string state, bool newValue) : subtract/add operations for inventory resources only.

* Except "Add" functions the other should be use in the PerformAction() function to modify the states as you desire.

-The result should view like this:

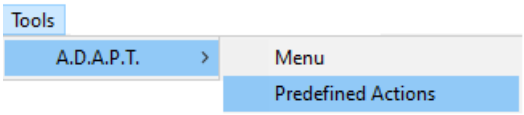
```
global_states.AddInventoryItem = ("minedGold", 0f);
```

* This is because the use of a unique instance to generate the global world state.

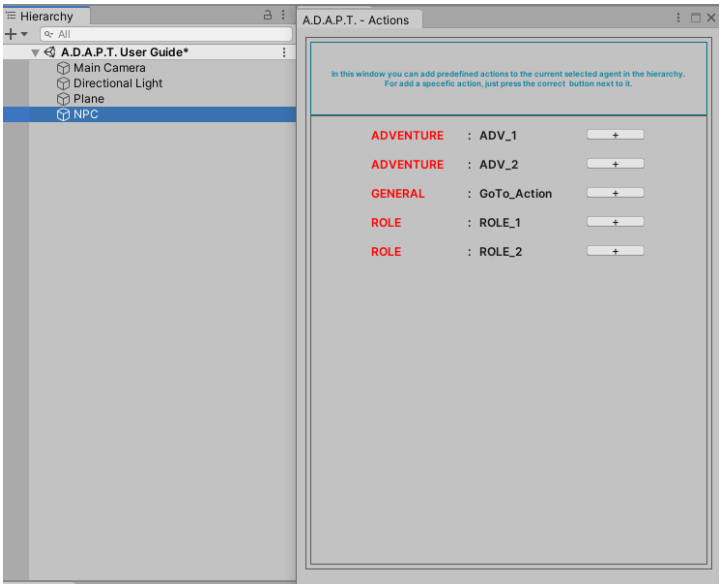
7.- Predefined Actions :

To add some of the predefined action to an actual agent (*the selected GameObject should be an Agent, if is not, then convert into one*), you should follow the next steps:

1.- Open A.D.A.P.T. Menu: `Tools > A.D.A.P.T. > Predefined Actions`

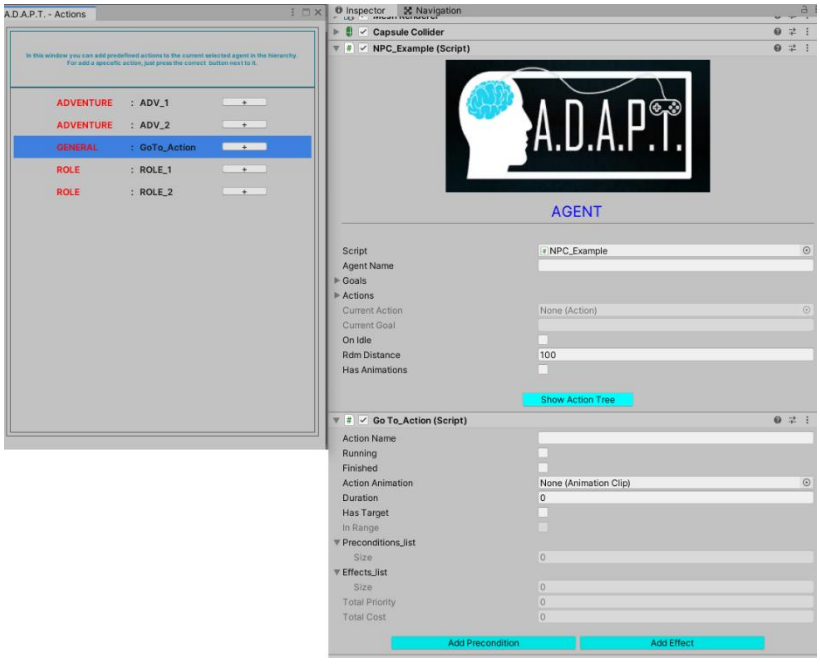


2.-With the Menu open, select your desired GameObject in the hierarchy:



3.-Click on the button **"+"** of the desired Predefined Action you want to add to the actual selected agent.

4.-Wait to the action is being added to the agent:

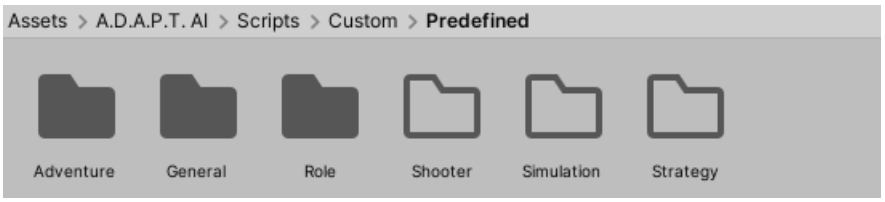


-Add your own Predefined Actions!:

You can add your own Predefined Actions to the Menu by moving your desired action to the path:

```
Assets/A.D.A.P.T. AI/Scripts/Custom/Predefined
```

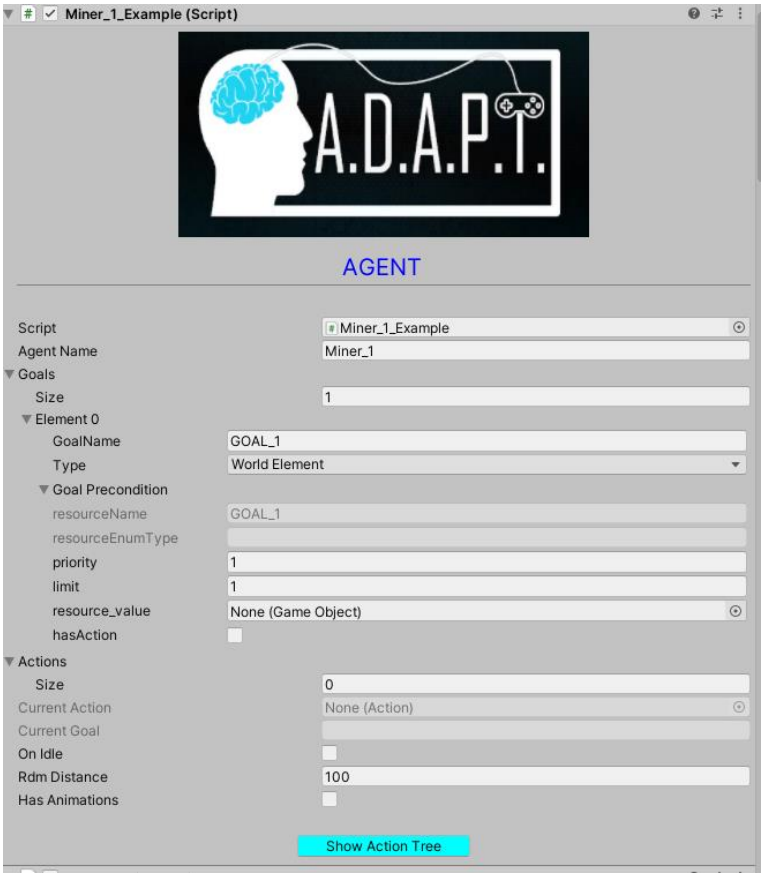
Then just throw your Predefined Action to one of the Game Genres:



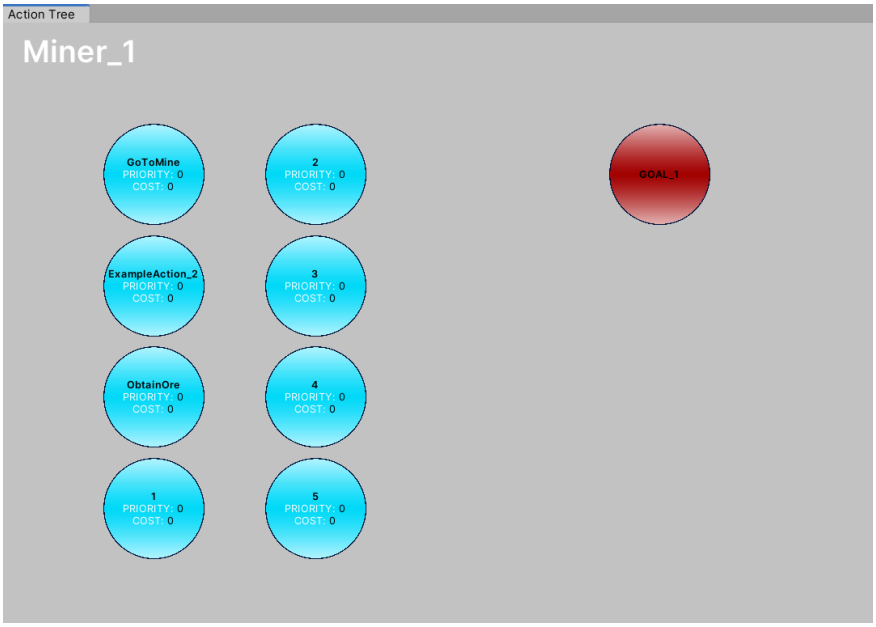
8.- Action Tree:

If you want to see the actual Action Tree of some specific created Agent (who you add some actions previously), follow the next steps:

1.- Select some created Agent to see it in the inspector:



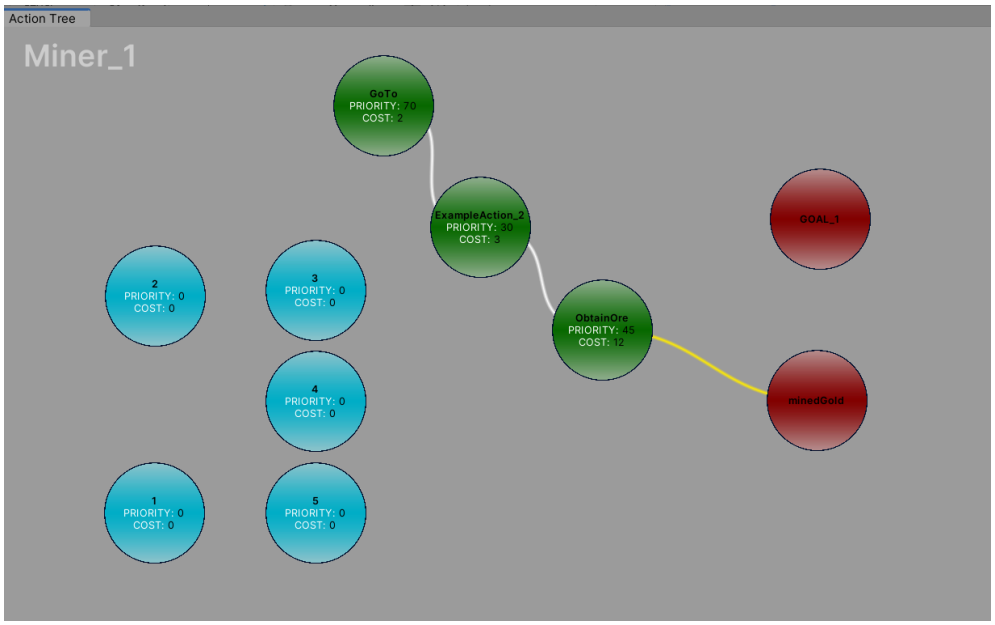
2.- Click the “Show Action Tree” button:



-In this case you will see the added actions and the goals added by inspector.

* If a goal is added by code it will be show in execution time.

3.- On execution time:



-You will see the actual Action plan to reach the possible goal:

-White lines: relationates actions.

-Yellow lines: relationates actions with the goal.

-Blue nodes: not achieve actions.

-Green nodes: actions who perfom the plan.

-Red nodes: goals.

* Nodes are draggable windows, so you can move it to any position if you want.

9.- In case of bug:

In case of any bug you can report it to my email: yagomira@gmail.com

Otherwise, here are some frequently bugs and solutions:

1.-In case of some error of “Index 0” caused by some relationated with Animations:

Close the Inspector window (A.D.A.P.T. and Unity also).

If the bug persists, restart the engine.

2.- Animation of the agent/action don’t play:

-Maybe you use the same animation for the idle states (on the agents inspector) and the Action animation, change the name of one of them.

-Add animator if you don’t have one (remind to select the specific **A.D.A.P.T. Controller**).

-Otherwise, check if you modify the “T-Pose” of the Agent Animator, **YOU CANNOT CHANGE THIS ANIMATION!.**

I hope this User Guide will be useful for you ☺. Thanks for use the tool!.

Sincerely,

Yago, developer of A.D.A.P.T. - AI Tool.