Part 1: Agent Setup

What is a GOAP Agent?

The anatomy of an agent in GOAP contains 4 main elements:

- Goals: The state that the agents want to happen.
  - A state is simply a true or false situation
- Actions: The things the agent can do, each has a set of preconditions, effects and cost
  - A precondition is what determines if an action can be started or not — For example, `To Eat` you need a precondition of `HasFood.`
  - An effect is the result of action and that result can trigger a precondition to be true. — For example, The `Find Food` action sets the `HasFood` state to be true.
  - A cost is used by the planner to decide between which plan to take. A dynamic cost can help your agents achieve fuzziness logic.
- Planner: Plans a sequence of actions based on the current states to satisfy a goal.
- Agent States and World States, these states are persistent effects.

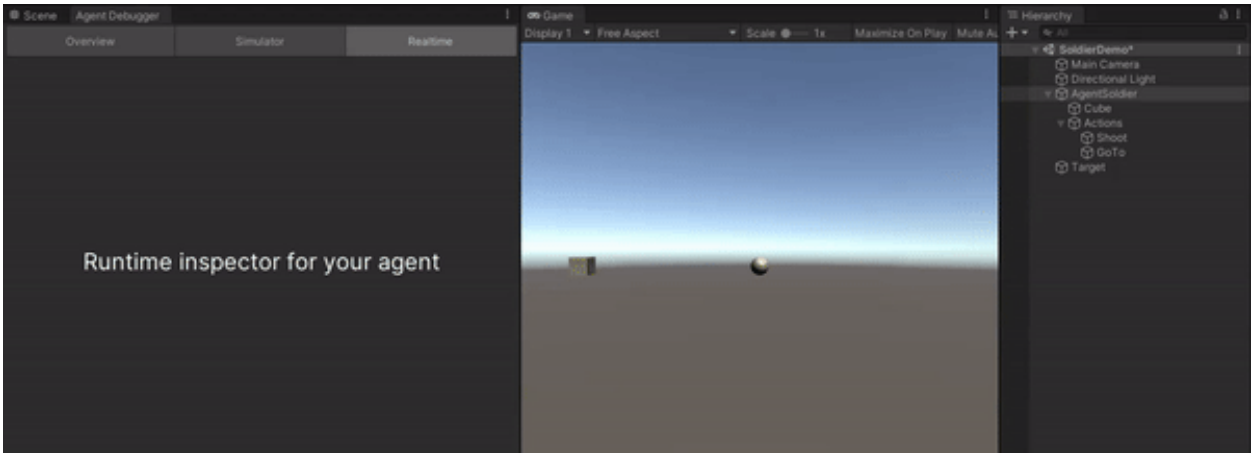
Designing the Agent

When creating an agent, you need to first have a design. So let's choose a design. For simplicity sake, in this tutorial, we will assume the Agent already has a target and will omit it from the preconditions.

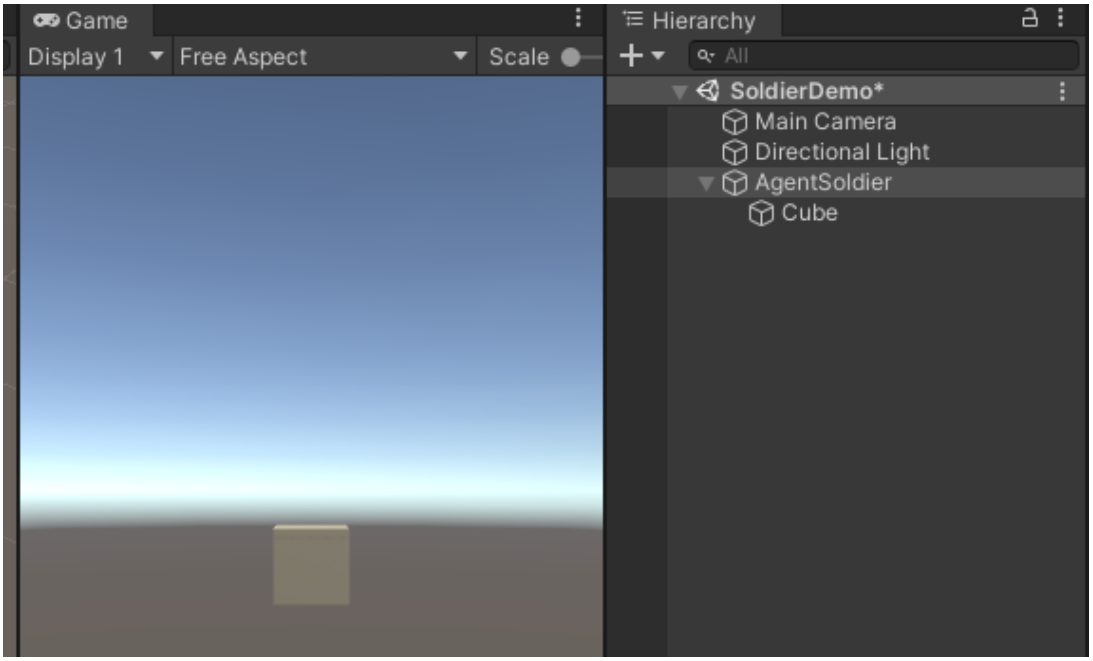
```
Agent: Soilder
Goals:
  - `HurtTarget`
Actions:
  - Shoot
    - Preconditions: `InShootingRange`
    - Effects: `HurtPlayer`
  - GoTo
    - Preconditions: `Not InShootingRange`
    - Effects: `InShootingRange`
```

Preview of What will be making

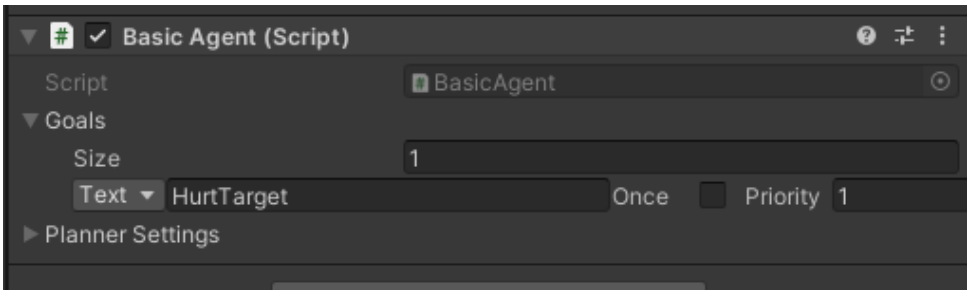
We will make a simple agent that follows a target and shoot it when it is within range.



In a new scene, add a new GameObject, name it Agent Soldier and add a cube as a child.



Add a **BasicAgent** component and then add a **HurtTarget** goal.



Under the **AgentSoldier** GameObject

1. Create an empty GameObject and call it **Actions**
2. Add an empty GameObject and call it **Shoot**
3. Create a **ShootAction C# class** and **attach it on the Shoot Game Object.**

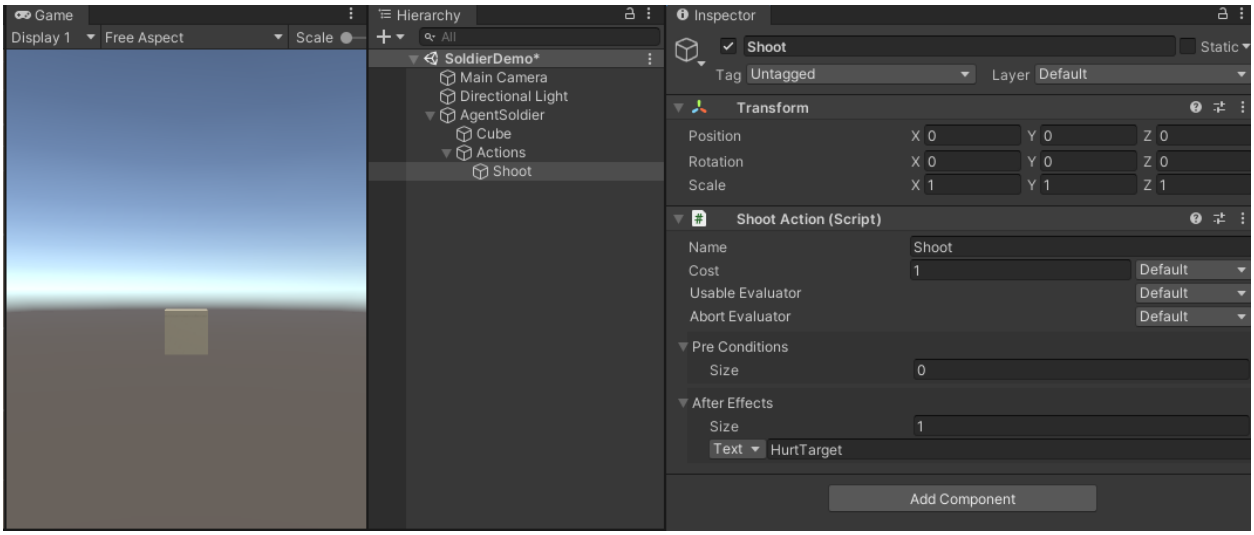
```
using SGoap;
using UnityEngine;

// Every SGoap Action inherits from either BasicAction or Action
public class ShootAction : BasicAction
{
    // This action has a cool down of 1 second every use.
    public override float CooldownTime => 1;

    // The planner by default skips actions that are cooling down unless we set this to true
    public override bool AlwaysIncludeInPlan => true;

    // Override Perform to execute the action.
    public override EActionStatus Perform()
    {
        Debug.Log("Shot");
        return EActionStatus.Success;
    }
}
```

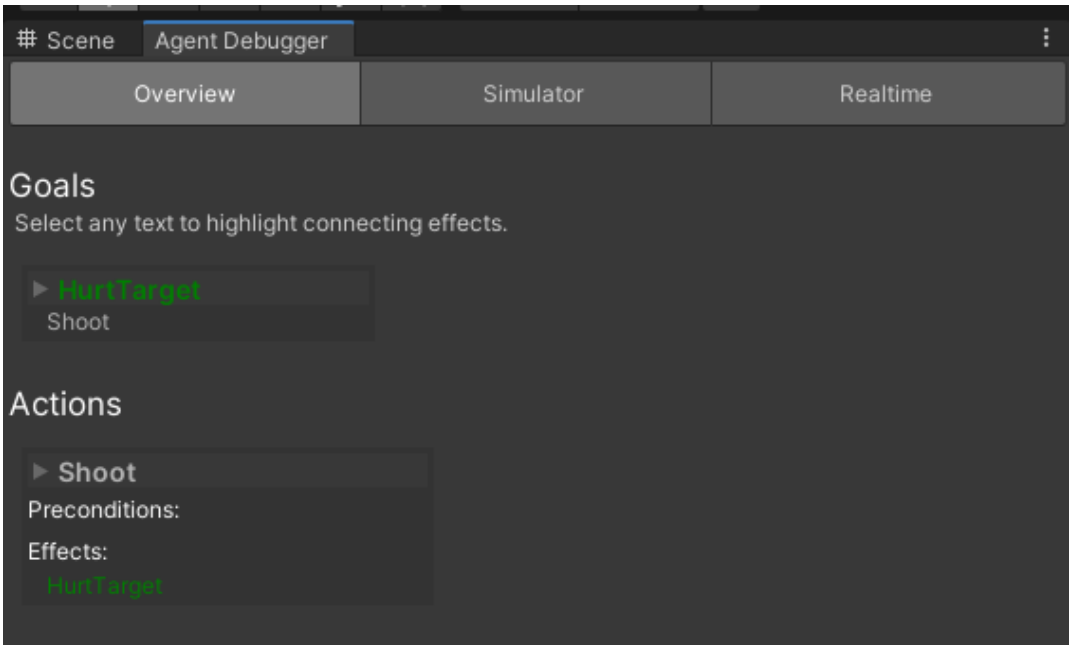
Add an effect with the text **HurtTarget.** This will let this action satisfy the goal **HurtTarget.**



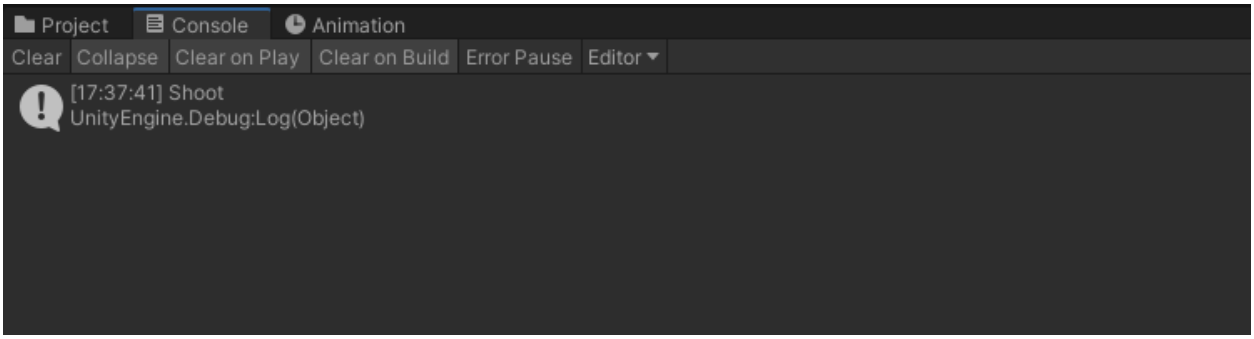
Open **Windows → Goap Agent Debugger,** make sure you've selected your **AgentSoldier** and then select the Overview Tab.

You should see the Goal, HurtTarget has an action that can satisfy it.

Clicking on the text will highlight similar text, such as **HurtTarget** is highlighted green.



If you press play now, you should see a shoot log every second.



In the next section we will add GoToAction and implement the ShootAction