



Manual

Introduction	5
Managers	6
Interactions	6
Interaction Manager	8
Conditions	9
Actions	9
Triggers	9
Dialogue	9
Dialogue Manager	12
Making Dialogues	13
Dialogue Lines	14
Responses	15
Dialogue DataBinding Components	16
Conditions	16
Actions	17

Triggers	17
Dialogue Interaction	17
Inventory	18
Items	20
Inventory Items	20
Placed Items	22
Inventory Manager	24
Equip Slots	25
Attachments	25
Inventory Item Loot	27
Conditions	28
Actions	28
Triggers	29
Crafting	29
Loot	31
Loot Table Item	32
Null Loot Item	33
Sub Table Loot Item	33
Virtual Currency Loot Item	34
Loot Interaction	35
Conditions	36
Actions	36
Behaviors	36
Interactive Objects	36
Chests	36
Conditions	39
Actions	39
Triggers	40
Destructibles	40
Actions	41
Doors	41
Conditions	43
Actions	44
Triggers	44
Moving Platforms	44
Actions	46
Spawn Camps	46

Actions	46
Spawners	46
Actions	48
Switches	48
Conditions	49
Actions	49
Triggers	49
Traps	49
Conditions	50
Actions	51
Stats	51
Base Stat	51
Derived Stat	51
Fuel Stat	52
Stat Modifiers	53
Progression	54
Progression Manager	56
Experience Tables	56
Stat Level Tables	59
Leveled Stats	60
Skill Trees	61
Conditions	63
Actions	63
Triggers	64
Behaviors	64
Loot Types	64
Stats	64
Combat	64
Combat Manager	65
Damage Application Stack	66
Other Combat CATs	72
Conditions	72
Actions	72
Triggers	72
Attack Slots	72
Conditions	73
Actions	73

Triggers	74
Abilities	74
Creating An Ability	74
Effect Groups	77
Special Conditions For Effect Groups	78
Special Actions For Effect Groups	78
Example Effect Group	79
Ability Casting Flow	81
Actions	83
DataBinding	83
Triggers	83
NPCs	83
Attack Behavior	83
Flee Behavior	84
Follow Behavior	85
Over Head Indicator Behavior	86
Patrol Behavior	87
Patrol Point	88
Wander Behavior	90
Merchant Interaction	91
Actions	92
Factions	92
Conditions	93
Actions	93
Triggers	93
Threat List	93
Conditions	96
Actions	96
Triggers	97
Examples	97
Abilities	97
Attack Slots	100
Complete Game	101
Crafting	103
Dialogue	105
Factions	110
Interactive Objects	112

Doors	112
Destructibles	115
Chest	116
Spawners	117
Trap	118
Moving Platform	119
Inventory	120
Loot	124
NPCs	127
Follower	127
Flee	128
Merchant	129
Wanderers	130
Patrol	133
Progression	136
Threat List	136

Introduction

With CAT RPG Builder you get access to all the nuts and bolts needed to make almost any RPG-type game you can dream up. We're talking fully developed features such as Quests, Dialogue, Items and Inventory. Each system is ready to ship, but extensible to fit your project. And of course, all of them use the base CAT system of Conditions, Actions and Triggers. CAT RPG builder has so many useful features and game systems built in that you will find the development of most game genres can be simplified. We did the work for you!

Some of the features CAT RPG Builder can put at your fingertips include:

- NPC Behaviors – create Merchants, Patrolling guards, or big bad monsters
- Abilities – build spells and other player/NPC abilities with complex behaviors
- Combat – the Combat system is ready to manage damage, health, attacking and all the other bits needed for slayage
- Crafting – hide ingredients for players to collect, create and distribute recipes for new items and let players outfit themselves with Items they create
- Dialogue - a robust system for branching dialogue between players and NPCs. It is chock full of hooks for triggering Quests, rewarding items, or just about anything else you can imagine.
- Factions – Govern the push and pull of alliances and animosity in your world

- Loot – you can't have an RPG without Loot! Create Loot Items, Tables and Bundles to give your game a deep and rewarding Loot experience
- Interactive Objects – Switches, Doors, Chests, Destructible Objects, Movers and more are ready for you to pop into your game world and make it come alive
- Spawners and Spawncamps – put together encounters for your players to trigger and then spawn for the ultimate challenges
- Items - sharp swords, glorious helms, currencies, crafting ingredients, socketed axes and more can be created with the robust Items system
- Progression – what would an RPG be without experience and leveling? You can make it as simple or as complex as you like with the build in Progression system
- Skill Trees – Our Skill Tree system allows you to create any kind of Skill Tree variation you can dream up
- Stats and Fuels – the bread and butter of RPG mechanics. Keep your min/max'ers up late at night as they try to unwind your dastardly maths!
- And more!

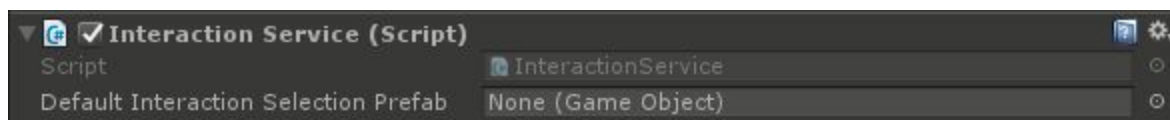
Note: Quests are part of the original CAT Game Builder.

Managers

Managers are a new type of component for CAT RPG. They attach to characters typically (PCs or NPCs) and manage some system for that character. For example, the Ability Manager keeps track of what abilities the character has and whether they can cast them.

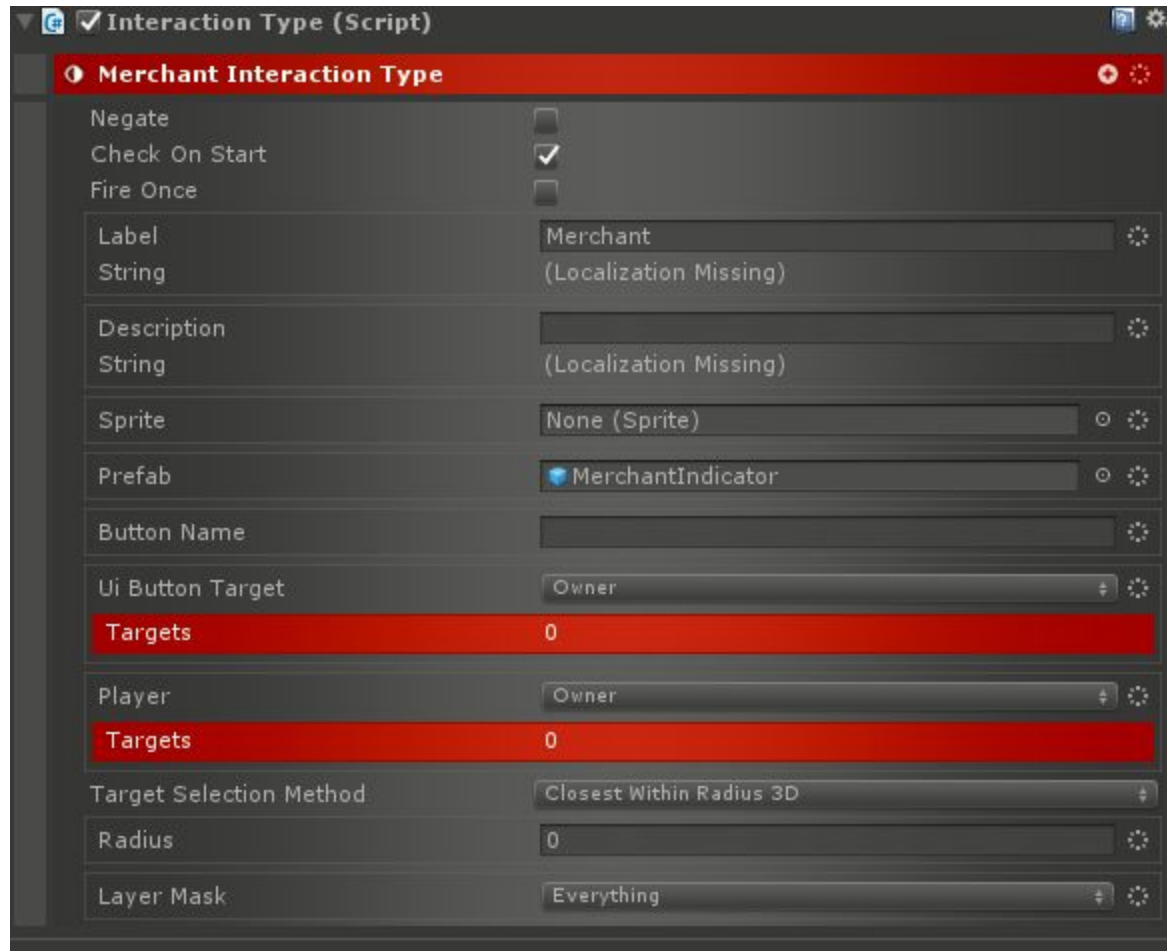
Interactions

The Interaction System manages Characters and inanimate objects that the player can interact with. It allows multiple Interactions on each thing and uses priority and availability to determine a default Interaction. Example Interactions would be speaking with an NPC, picking up an Item, or opening a door. In order to use this system, add an Interaction Service to the Conductor in the scene.



There's only one option, and it's optional. If specified, the Default Interaction Selection Prefab is what will be displayed when the player is asked to select from a list of Interactions in the case that multiple are available.

For each type of Interaction, an Interaction Type must be childed to the service.



Interaction Types define generic properties for a type of interaction. These include the player facing name of the Interaction, a description, a Sprite to use (in the Interaction Selection Window), and a prefab to use as an indicator that this interaction is available.

There's also the Button Name option which can specify an input button which will cause the Interaction to be triggered based on the Target Selection Method.

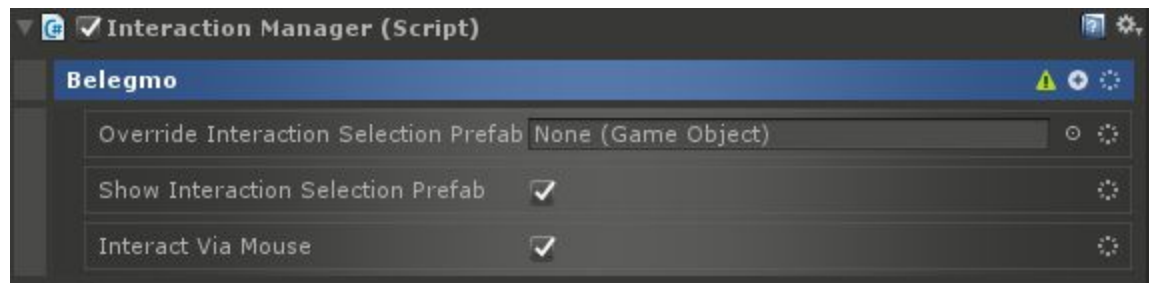
If UI Button Target points to a UI button (for example on the HUD), it can be used to trigger the interaction.

The Player option should be set to the Player Game Object when Button Name or UI Button Target are used. This allows the system to determine the closest interactable target.

Target Selection Method can be set to 2D or 3D and then a radius is defined along with a layer mask. Any Game Object within that radius which is Interactable will be queried when the Button Name or UI Button Target are activated.

Interaction Manager

To apply interactions to a Game Object / Character, the Interaction Manager component must be added.



Override Interaction Selection Prefab makes it possible to change the Interaction Selection Prefab for this Game Object / Character.

Show Interaction Selection Prefab determines whether the prefab will be shown if multiple Interaction types are available when the Game Object / Character is interacted with.

Interact Via Mouse will allow the Interaction to be activated by clicking on the Game Object / Character. A collider is required for this to work.

Underneath the Interaction Manager, Interactions can be added.



There are different Interactions for things like Dialogue, Combat Targeting, Looting, etc. Conditions can be childed to them which will be used to determine whether the Interaction is

available at any given moment. The Evaluation Requirement parameter sets how many conditions must pass before the Interaction is available.

The other common options on all Interactions are the Interaction Type which needs to point to a valid Interaction Type under the Interaction Service, and the Priority setting. The Priority helps identify which interaction is the most pertinent if multiple are active at any given point.

Conditions

- Has Interaction Type Condition - Checks whether the target has a specific interaction type. Doesn't check for availability.
- Is Interactable Condition - True if the target has any available interactions optionally of the specified type.

Actions

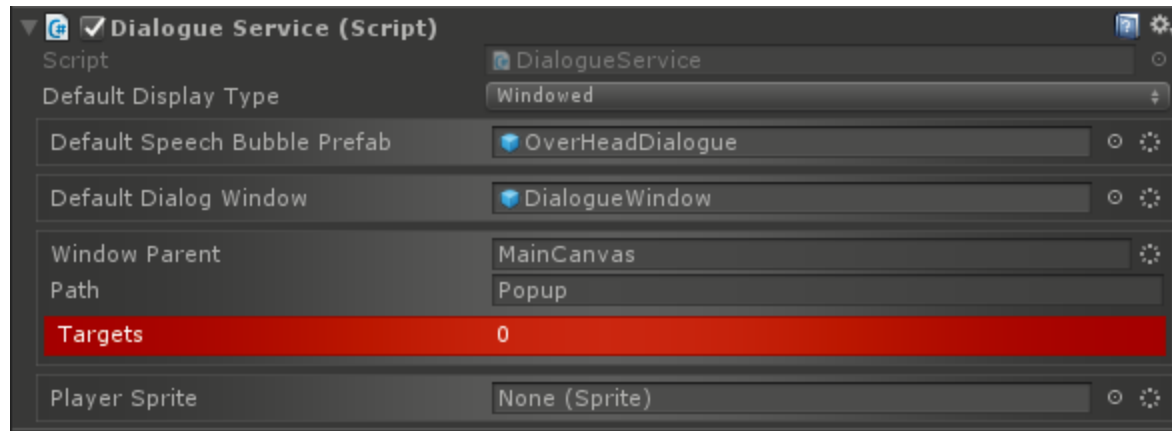
- Add Interaction Action - Adds a new interaction to a target with an Interaction Manager.
- Enable Interaction Action - Causes interactions of a specific type to be enabled or disabled on a target. The interactions will still adhere to any conditions.
- Hide Interaction Selection Window Action - Hides the interaction selection window of a target if it is open.
- Interact Action - Trigger an interaction of an optional type with the target.
- Remove Interaction - Removes an interaction of a given type from the target.

Triggers

- Has Interaction Type Trigger - Fires when the target has a specific interaction type. Doesn't check for availability.
- Is Interactable Trigger - Fires when the target has any available interactions optionally of the specified type.

Dialogue

Dialogue in CAT RPG starts with adding the Dialogue Service to the Conductor.



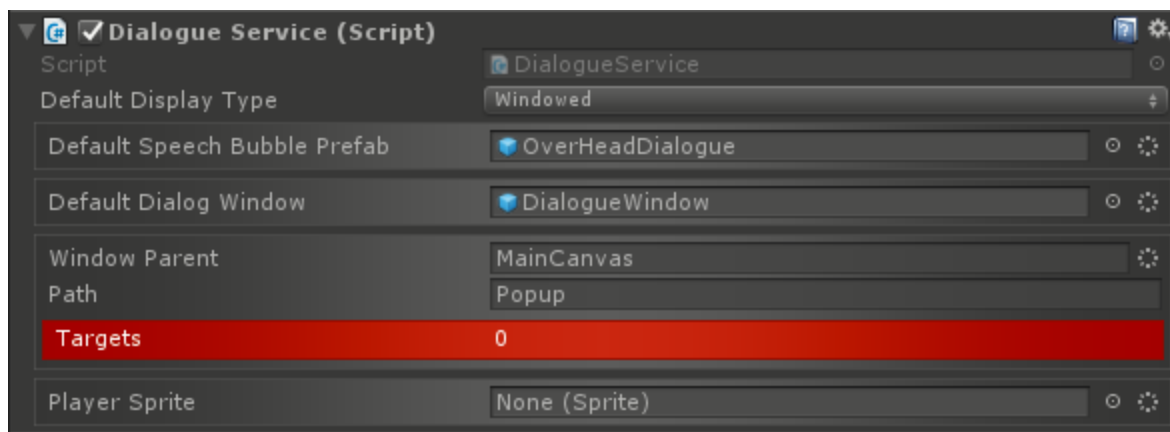
There are some important default settings. One thing to note is that the Dialogue System supports two display types: overhead and windowed. Overhead dialogue is as it says-- the character's lines will pop up over their heads like this:



Windowed dialogue is more common and looks like this:



Returning to the Dialogue Service options:



Default Display Type specifies whether to use Windowed or Overhead by default. This can be overridden in a number of ways, but this will be the default for the game.

Default Speech Bubble Prefab specifies the default Prefab to be used for Overhead Dialogues. As with the Display Type, this can be overridden at other levels.

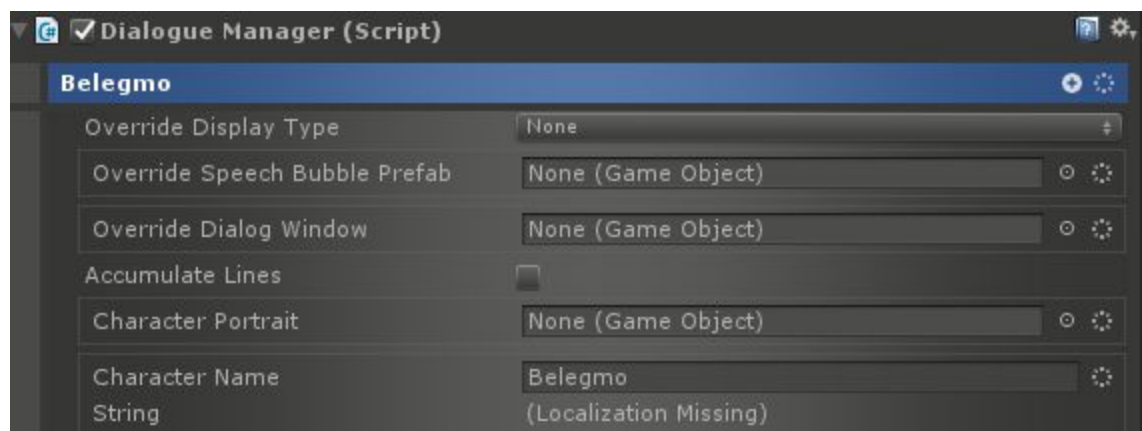
Default Dialogue Window is the Prefab for the window to show for Windowed Dialogue. There are a number of custom DataBinding components which should be used with this window.

Window Parent is to specify the parent under which to spawn the Dialogue Window when in Windowed mode.

Player Sprite can point to a Sprite to use to denote the player in the UI.

Dialogue Manager

The Dialogue Manager attaches to any Character that will have Dialogue.



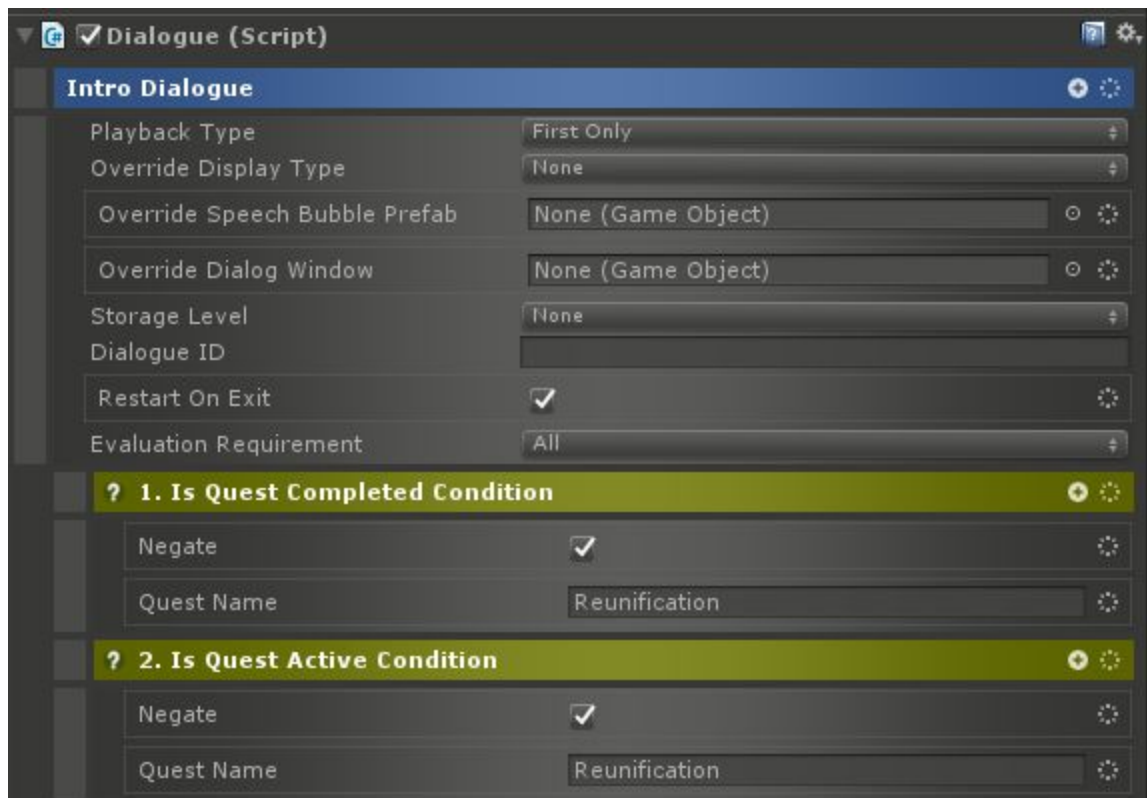
It contains some override options for the settings in the Dialogue Service. These overrides will apply to any Dialogue shown by this Character, but can be further overridden at other levels. Display Type, Speech Bubble Prefab, and Dialogue Window can all be overridden.

The Accumulate Lines option can be turned on in order for old lines to stay on screen as the player progresses through the Dialogue.

Character Portrait can point to a Prefab which can be used to display an image or model of the character when they are speaking within the UI. Similarly, the Character Name can be used to display who is speaking in the Dialogue Window UI. Both of these options can be overridden at other levels.

Making Dialogues

The next level is to actually construct Dialogues. There are three parts to them: Dialogues, Dialogue Lines, and Responses. A Dialogue represents a single conversation with that character and can contain one or more Dialogue Lines. Each Dialogue Line is a line in the conversation which can have zero or more Responses. Responses are what the player can use to respond to the Character's Dialogue.



Dialogues for a given Character must be childed to that character or a Game Object with a Dialogue Manager Component. There are several options at the Dialogue level:

Playback Type specifies how to display the Dialogue Lines in this Dialogue. If set to Serial, each line will be displayed in order. This is useful for simple dialogue, or multiple lines that should be played in sequence. The next option is First Only. This will display the first available Dialogue Line in this Dialogue. Then, if nothing else moves the Dialogue along, it will exit.

The next three options are to override Display Type, Speech Bubble Prefab, or Dialogue Window for this Dialogue.

The Dialogue system is persistent if a Storage Service is present. This means it will record which Characters the player spoke with and what lines were seen. This is useful for Conditions or Triggers to change the game state depending on who the player has spoken with. In order to persist this, the Storage Level and a unique Dialogue ID must be specified.

Restart On Exit will cause the Dialogue to restart at the beginning after exiting. If turned off, if the player only views the first 3 lines of the Dialogue and then exits, the next time they bring up the Character's Dialogue, it will start at the 4th line.

The Evaluation Requirement is related to the ability to add Conditions under Dialogue. Conditions under the Dialogue will be evaluated to determine if the Dialogue is available at any given moment. The Dialogue Manager will display the first (top down) Dialogue that is available unless it has been overridden. In the example above, both conditions must be true because Evaluation Requirement is set to All and the two conditions check whether the Reunification Quest has not been completed and whether the Reunification Quest is not currently active for the player.

Actions and Stop Actions can be childed to Dialogues. The Actions will run when the Dialogue is displayed, and the Stop Actions will be run when it is done.

Dialogue Lines

Every Dialogue must have one or more Dialogue Lines childed to it. These display a single line in the Dialogue and allow player interaction through Responses.

The image shows a configuration panel for a dialogue line titled "3. Welcome". The panel contains several settings:

- Line String:** "Welcome, traveler, to River Vale. Please enj" (Localization Missing)
- Override Display Type:** "None"
- Delay:** "0"
- Next Line Name:** (Empty)
- Override Speech Bubble Prefab:** "None (Game Object)"
- Override Dialog Window:** "None (Game Object)"
- Voiceover:** "None (Audio Clip)"
- Override Character Portrait:** "None (Game Object)"
- Override Character Name String:** (Empty) (Localization Missing)

The first field in the Line is the actual text of the Line itself. This is what will be displayed to the player.

Below that are options to override Display Type, Speech Bubble Prefab, and Dialogue Window. These are as discussed previously and will override settings in the Dialogue, Dialogue Manager, or Dialogue Service.

Delay can be set to something greater than 0 in order to have the Dialogue Line display for a given number of seconds before advancing to the next. This can be useful to move Dialogue along with there are no Responses.

Next Line Name can be specified to point to the name of the next Dialogue Line that will be displayed after this one. This is somewhat of a default as Responses can override this.

Voiceover can point to an Audio Clip which will be played whenever this Dialogue Line is displayed.

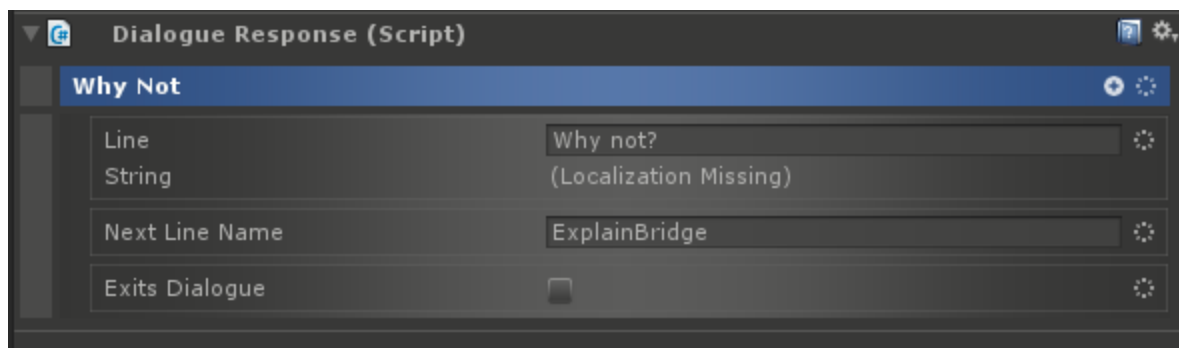
Finally, Character Portrait and Name can both be overridden at this level.

Like Dialogues, Dialogue Lines can also have Conditions underneath them which will be checked in order to determine if the Line can be displayed.

Actions and Stop Actions can be childed to Dialogue Lines. The Actions will run when the Line is displayed, and the Stop Actions will be run when it is done.

Responses

Dialogue Responses allow for player choice in the Dialogue System. Each Dialogue Line can have zero or more Responses childed to it.



Similar to the Dialogue Line, the Response starts out with a field to specify what text will be displayed to the player.

Responses can also move the Dialogue to a specific Line if the player chooses that Response via the Next Line Name. They can also exit the Dialogue if selected with the Exits Dialogue setting.

Both regular Actions and Else Actions can be childed to a Dialogue Response. Actions will be run if the Response is chosen by the player and Else Actions will be run if a different Response is chosen.

If the Dialogue this Response is under has Storage Settings defined, then the player's choice of Response will also be stored.

Dialogue DataBinding Components

The Dialogue System also has several specialized DataBinding Components to make it easier to construct Dialogue Windows and Overhead Speech Bubbles.

- UI Dialogue Character Name DataBind - Used with a Text component, this will bind to the name of the Character that is speaking the line.
- UI Dialogue Line DataBind - Used with a Text component, this will bind to the line being spoken.
- UI Dialogue Player Name DataBind - Used with a Text component, this will bind to the Player's name.
- UI Dialogue Response Line DataBind - Used with a Text component, this will bind to a Dialogue Response's text.
- UI Dialogue Responses Layout Group DataBind - Used with a Layout Group component, this will display the Responses for a given Line.

Conditions

- Dialogue Response Condition - Finds the dialogue with the specified DialogueID and determines if the response with the specified ResponseID has been selected by the player.
- Has Dialogue Been Active Condition - Finds the dialogue with the specified DialogueID and determines it has ever been activated.
- Has Dialogue Condition - Determines if the Target has a dialogue with the specified DialogueID on it.

- Has Seen Dialogue Condition - Determines if the dialogue specified by the DialogueID has been active.
- Has Seen Dialogue Line Condition - Determines if the dialogue line specified by the DialogueID and line name has been active.
- Has Spoken With Condition - Determines if the player has spoken with the target.

Actions

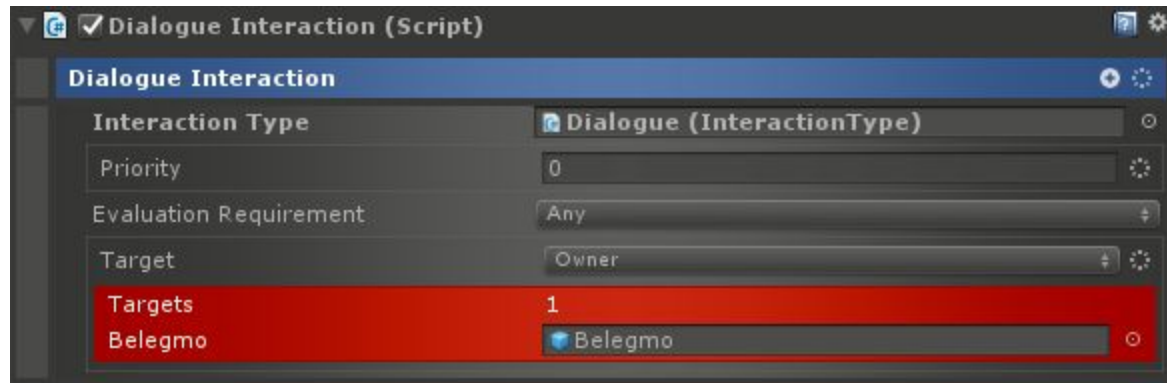
- Clear Accumulated Dialogue Lines Action - If a Dialogue Manager is set to Accumulate Lines, this will clear the Lines.
- Dialogue Response Behavior - Waits for a button press on the target and then marks it as the selected Dialogue Response assuming it has a Response databound to it.
- Hide Dialogue Action - Can be pointed at either a Dialogue Manager or a Dialogue and will hide either.
- Next Dialogue Line Action - Moves the currently active dialogue to the next line.
- Reset Dialogue Action - Completely resets the state of all Dialogue.
- Revert Dialogue Action - Reverts a previously overridden dialogue back to the default.
- Set Dialogue Action - Override the default dialogue of a target with a new one specified by name.
- Show Dialogue Action - Show the dialogue for a target.
- Skip To Dialogue Line Action - Skip to a new line in the currently active dialogue of the target.
- Skip To Dialogue Line By Index Action - Skip to a new line by index in the currently active dialogue of the target.

Triggers

- Dialogue Line Trigger - Triggers when the specified dialogue is currently active.
- Dialogue Response Trigger - Triggers when the specified response on the specified dialogue has been selected.
- Dialogue Trigger - Triggers when the specified dialogue is currently active.
- Has Dialogue Trigger - Triggers if the target has a dialogue with the specified DialogueID.
- Speak With Trigger - Triggers whenever the target has any dialogue attached to it active.

Dialogue Interaction

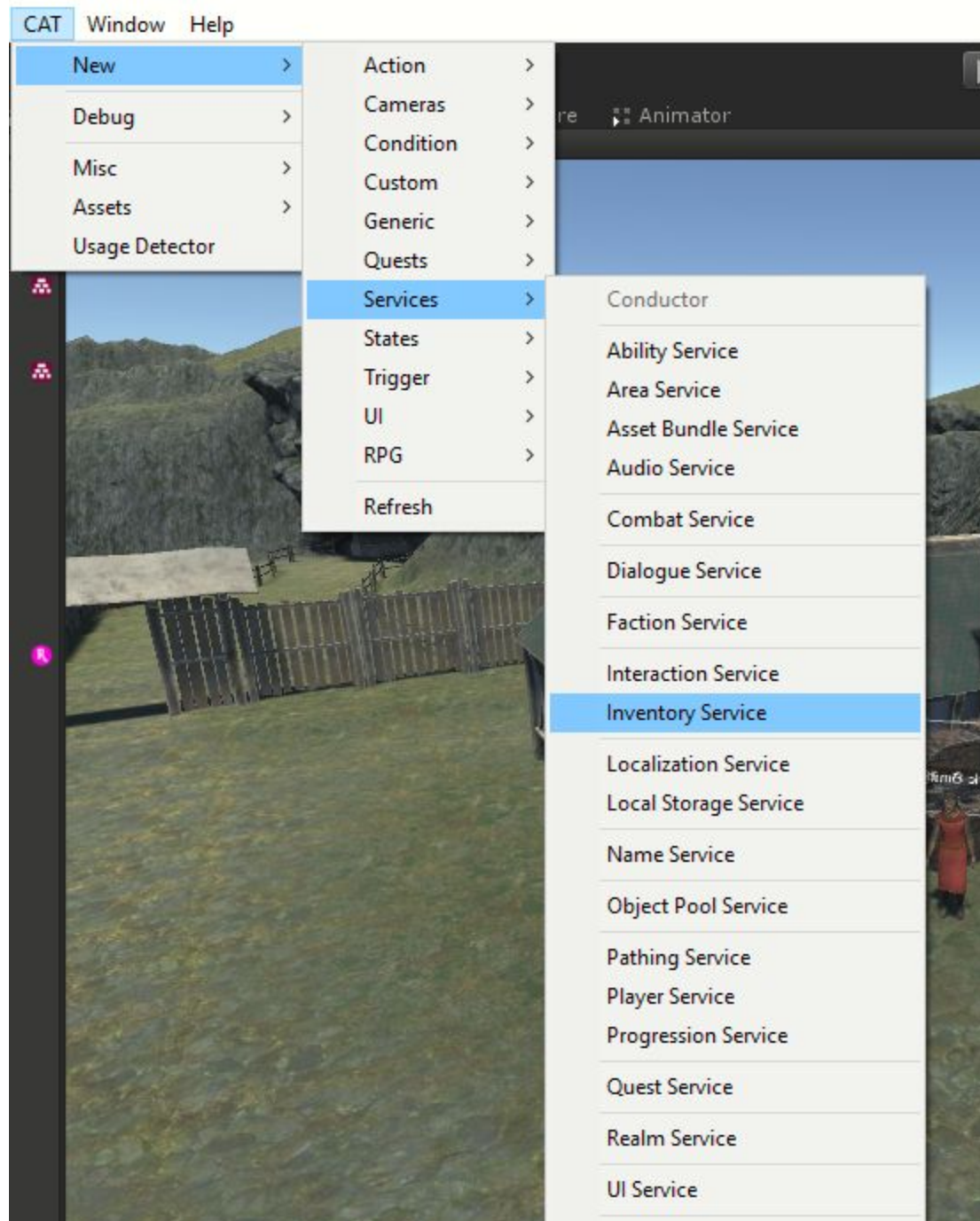
The Dialogue System also includes a custom Interaction called the Dialogue Interaction. It can be used to activate dialogue on a Character using the Interaction System.



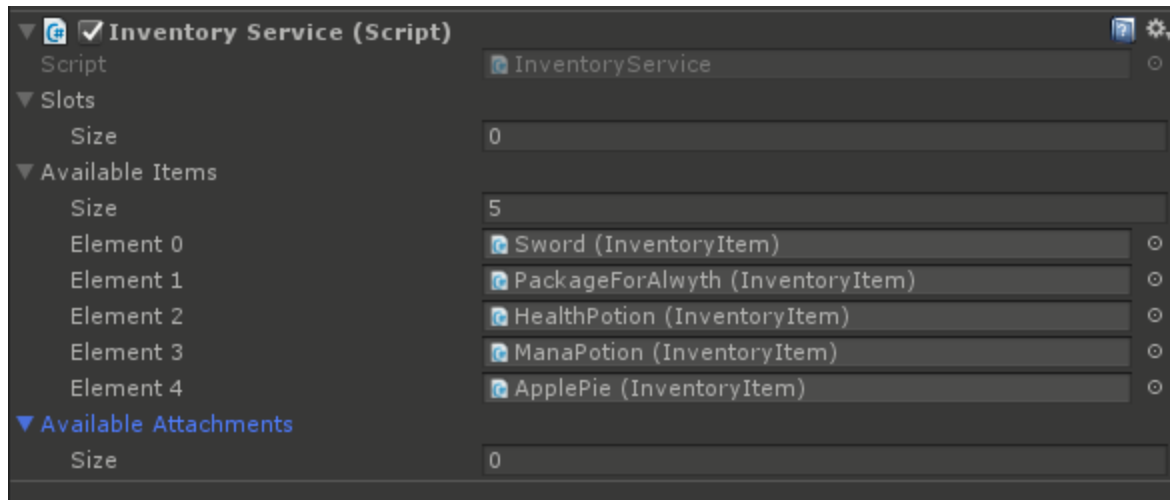
The parameters are the same as any other Interaction. See the Interaction System for more information on use.

Inventory

The Inventory System supports features common to inventory in most RPGs. Items can be created and placed in the level or given out as loot. Characters including the player have a discrete inventory to hold those items in. The Inventory System also supports equip slots and visually showing equipped items like swords or guns. To get started with the Inventory System, include the Inventory Service in a scene:



Configuring the Inventory Service is optional if Asset Bundles are being used.



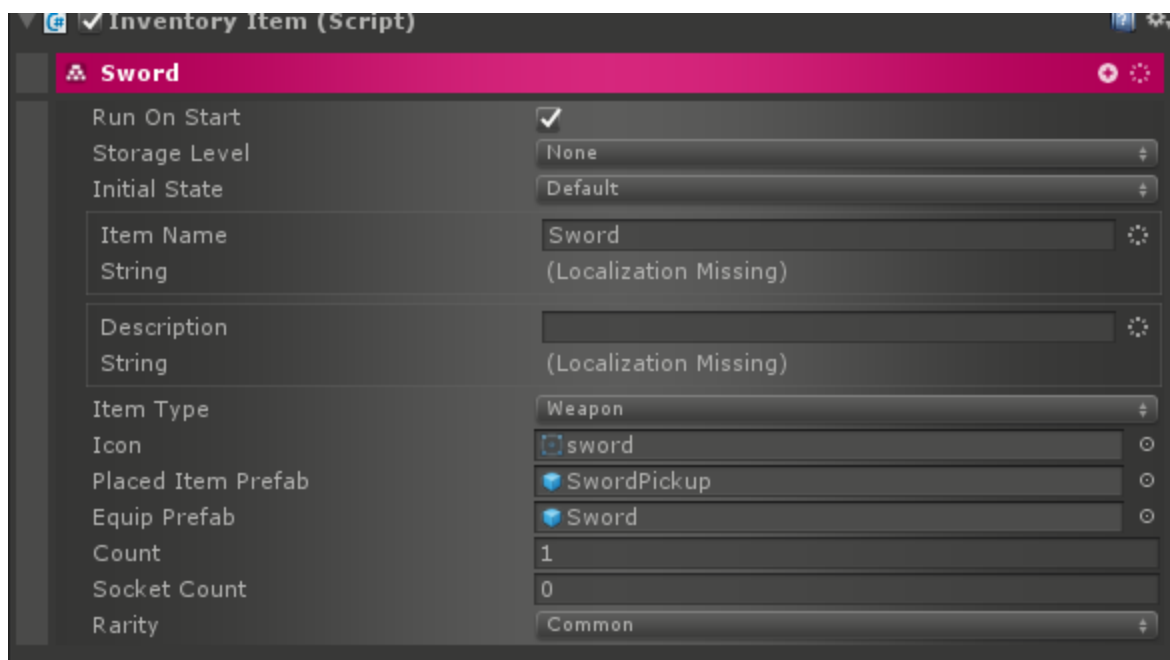
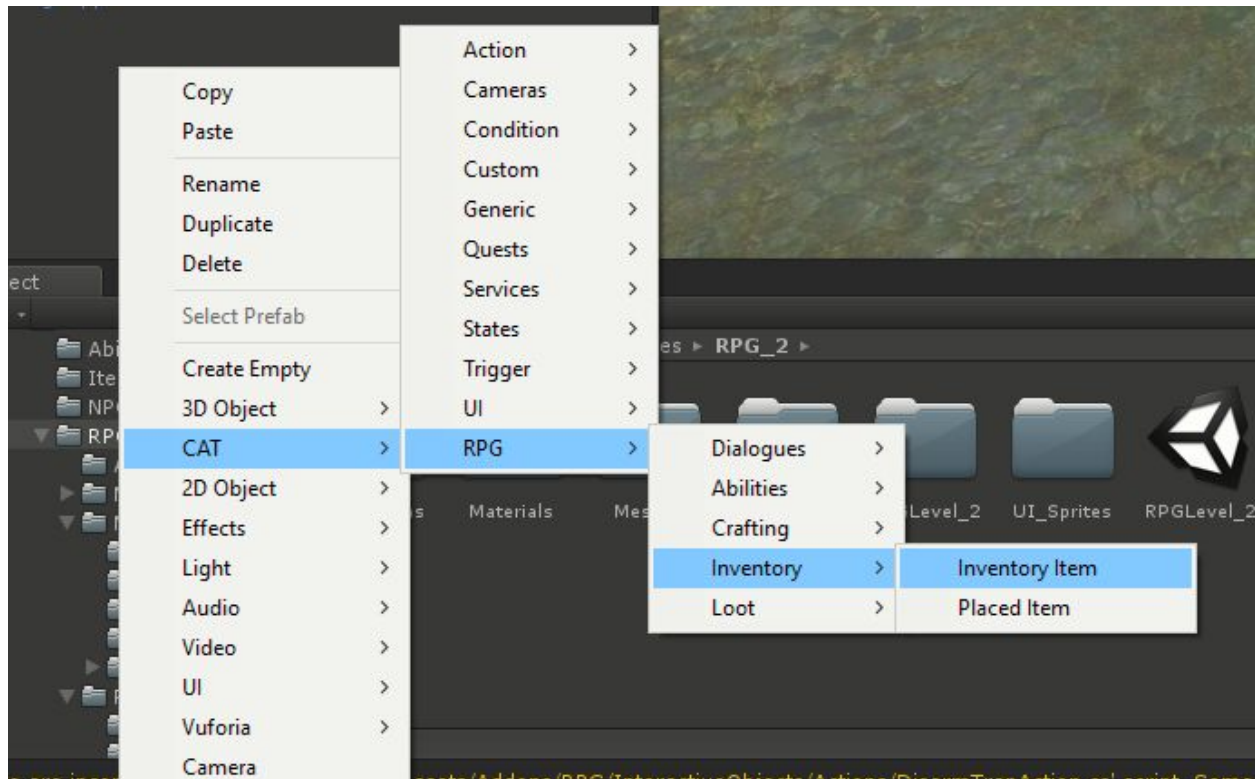
If not, then all Items in the game must be added to the Available Items list. Same for all Slots and all Attachments. Typically, it will be easier and less error prone to use Asset Bundles when using the Inventory System.

Items

There are two parts to creating an Item. The first is the Inventory Item, which is the definition of the Item and what actually sits in Inventory. The second is the Placed Item, which is optional and represents an Inventory Item when placed on the ground or in the level somewhere.

Inventory Items

To create an Inventory Item, use the CAT menu:



Inventory Items are also State Machines and all the functionality of State Machines can be used on them. However, they have 3 special states: On Equip, On Use, and Default. Normally, Inventory Items are in the Default State, and often, this state will be empty. When the player picks up an Inventory Item, it is moved into the OnEquip state. Some Inventory Items may have

an ability to do something associated with them. These are accessed through using the Item. Any Actions placed in the OnUse State will be run when the Item is used.

Items have a localizable name and description which can be set. They also have an Item Type. Inventories have Slots which can accept only certain Item Types. To show what an Item looks like in Inventory, define the Icon. Then, to have it droppable, set the Placed Item Prefab to the Placed Item for this Item.

There's also the Equip Prefab, which should point to a Prefab / model that will be attached to whatever socket is defined for an Equip Slot that the item is put into. For example, this is how when a sword is placed in the right hand Equip Slot that it appears in the character's right hand.

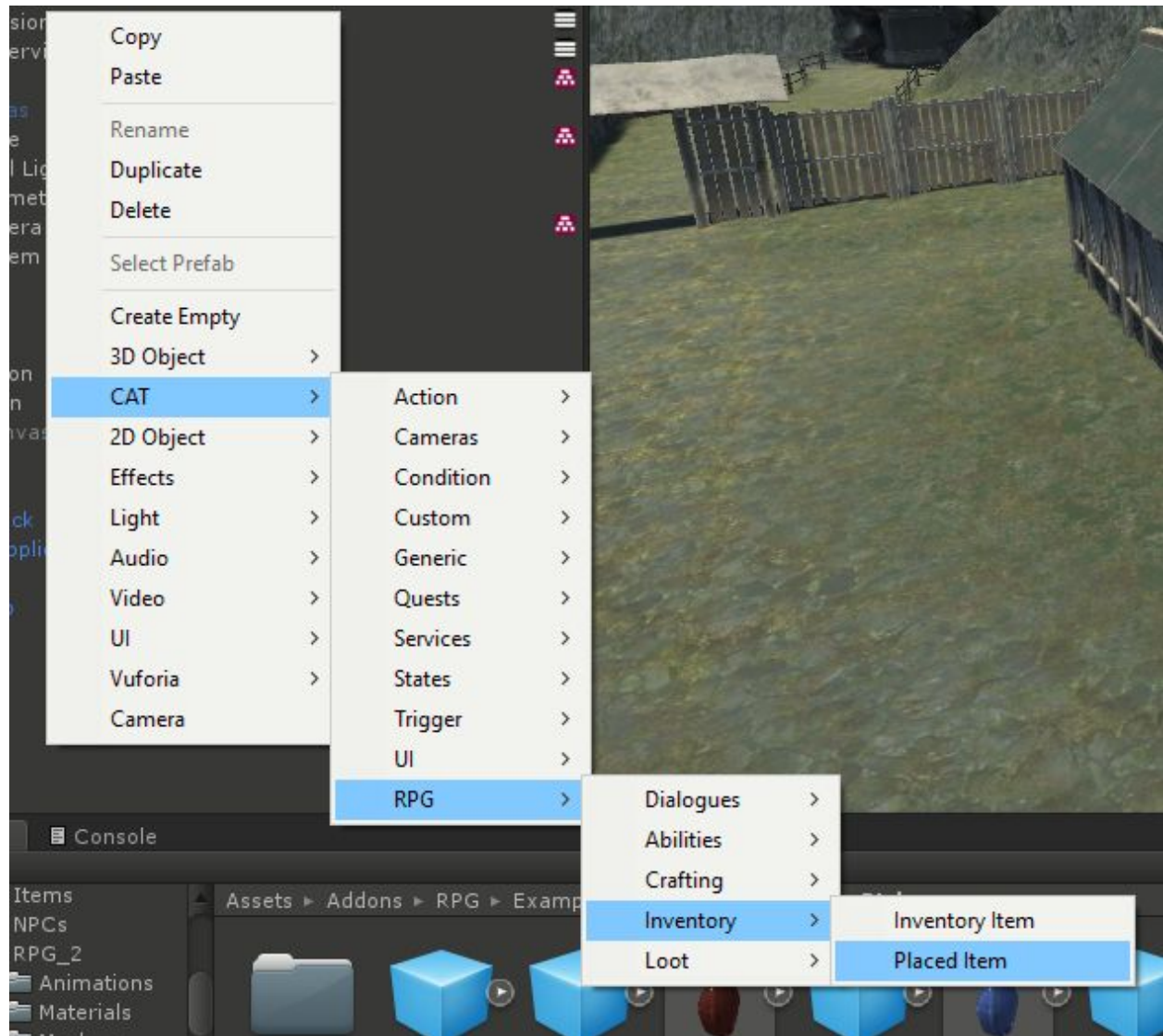
Count is for Stackable Items (Item Type is Stackable). This sets how many Items this Inventory Item represents.

Socket Count defines how many Sockets are available on this Item for upgrades.

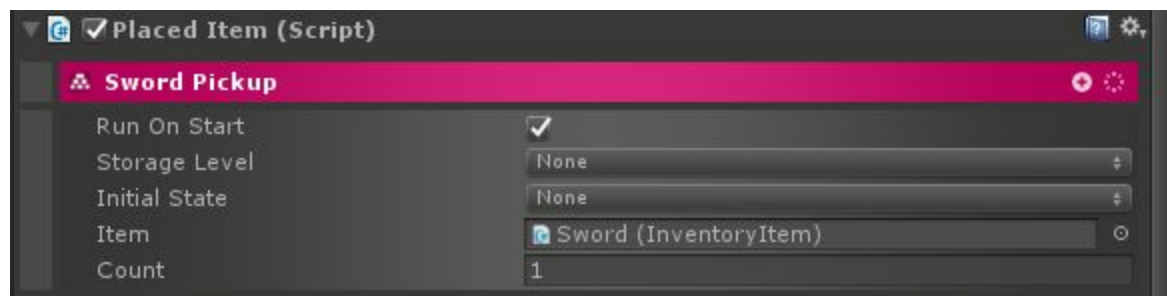
Finally, Rarity can be used to differentiate between Common and Rare items.

Placed Items

After creating an Inventory Item, the typical next step is to make a Placed Item for it. If the game does not have items that can be picked up off the ground, then this step is optional. To create a Placed Item, use the CAT menu:

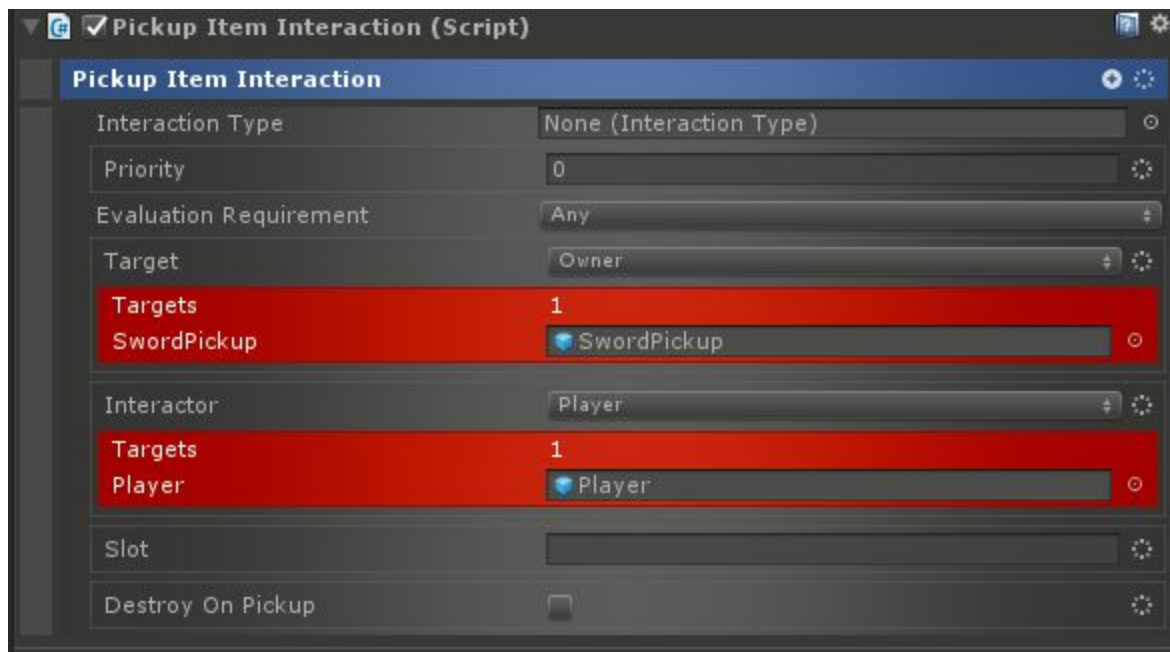


The Placed Item will look like this:



Placed Items are also State Machines. The extra parameters are a link to the Inventory Item, and a Count which will be applied to the Inventory Item when added to the Character's

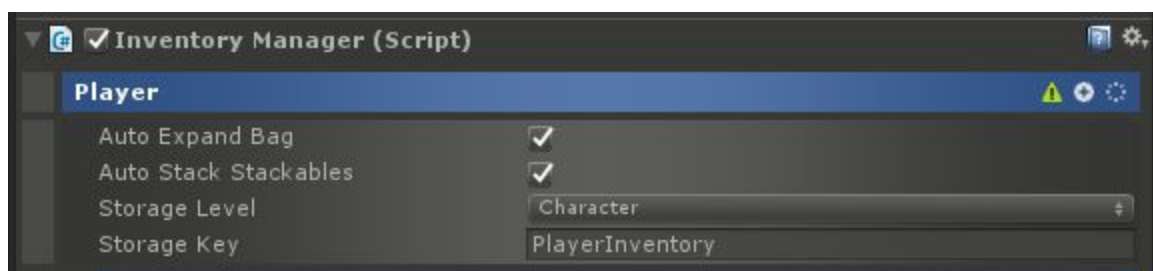
Inventory. The easiest way to make a Placed Item able to be picked up is to add an Interaction Manager and then a Pickup Item Interaction:



Target should point to the Pickup Item (typically Owner). Interactor can either point at Player or Targeted. If specified, Slot should be the name of an Equip Slot to put the item into. Otherwise, an available bag slot will be used. If Destroy On Pickup is set, then the Pickup Item will be destroyed as soon as it is picked up.

Inventory Manager

Each Character that has Inventory should have an Inventory Manager Component added.



This contains some standard Persistence System settings, but it also has two others. Auto Expand Bag is used to make the Character's bag add new slots whenever something is added to their Inventory and there is no existing bag slot to store it in. Use this for unlimited Inventory

size. When Auto Stack Stackables is checked, new Stackable Items that are added to the Inventory are automatically stacked if there are existing Stackable Items of the same type already in the inventory.

Equip Slots

The next step is to add Equip Slots to the Character. These are childed to the Inventory Manager:

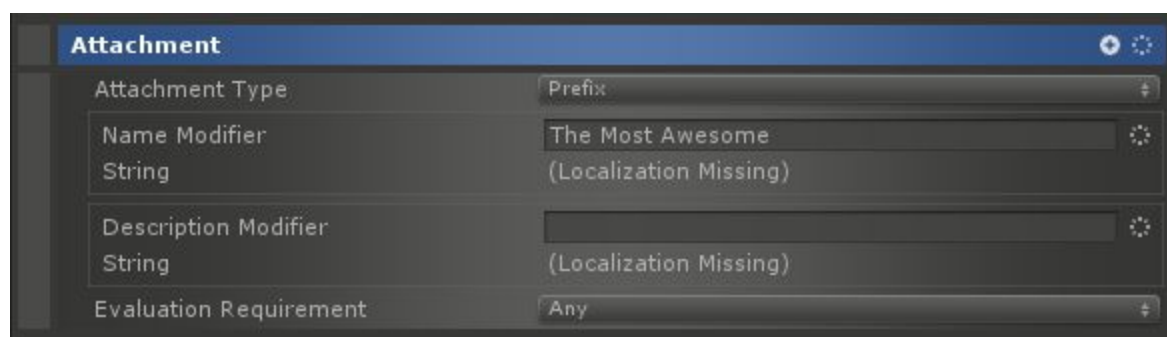
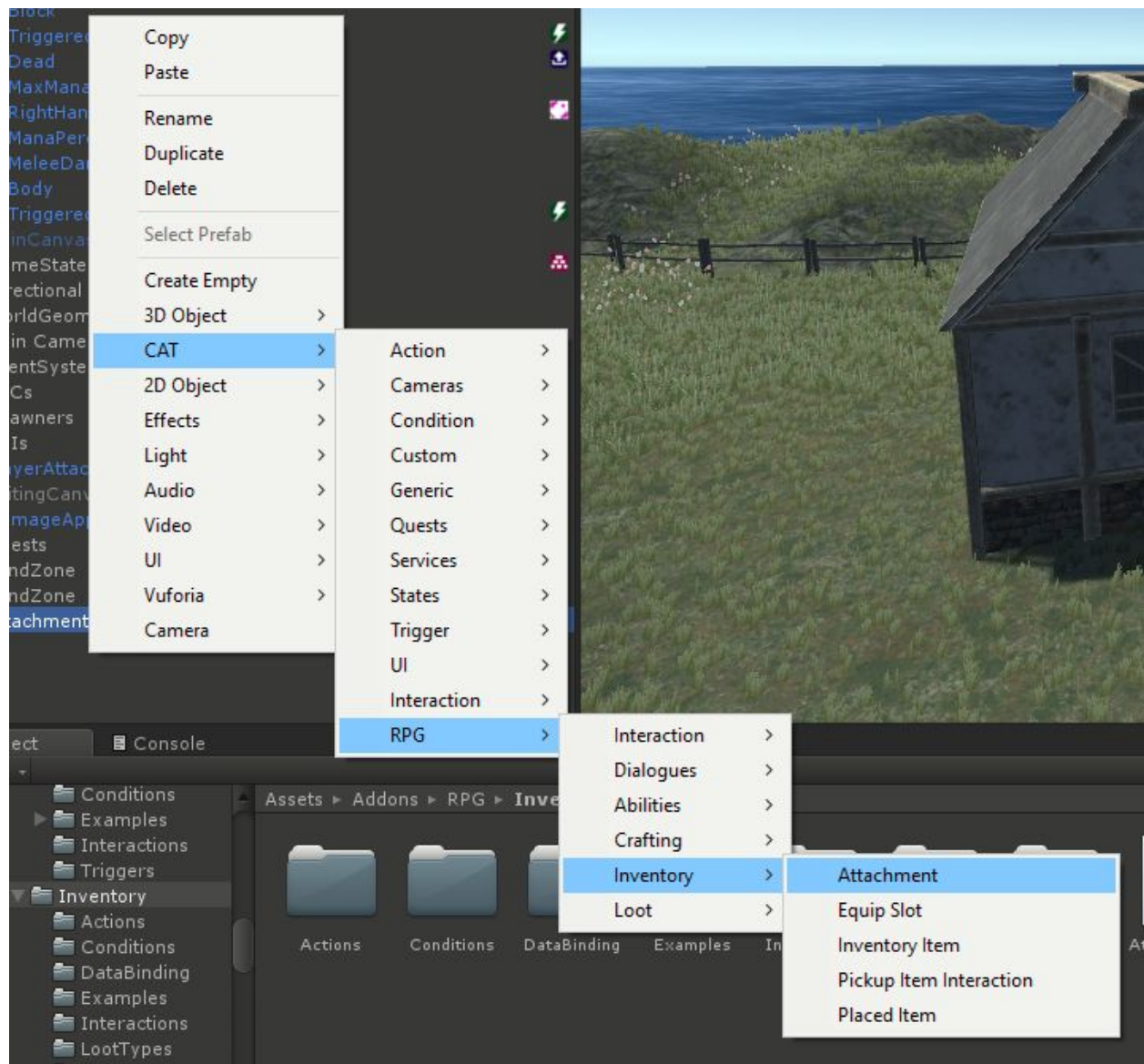


The Slot Type specifies whether this is right hand, left hand, both hands, an adornment (clothing or armor), or a bag. Allowed Item Types can be used to restrict what can go in the slot. The Attach Point specifies where to add any Equip Prefabs.

Attachments

Inventory Items can have Attachments. These can be Prefixes, Suffixes, Sockets, or Attributes. Prefixes and Suffixes when attached, will modify the name of the Inventory Item by either prepending or appending respectively. Typically, Prefixes and Suffixes are built in to the Item when it is created. Sockets are Attachments that are typically added by players. Inventory Items have a limited number of Sockets available to fill. Attributes may be added by players or may be generated as part of loot. There is no limit to the number of Attributes that can be added to an Item. When added by players, it would typically be in the case of upgrading items.

To create an Attachment, add it via the menu:



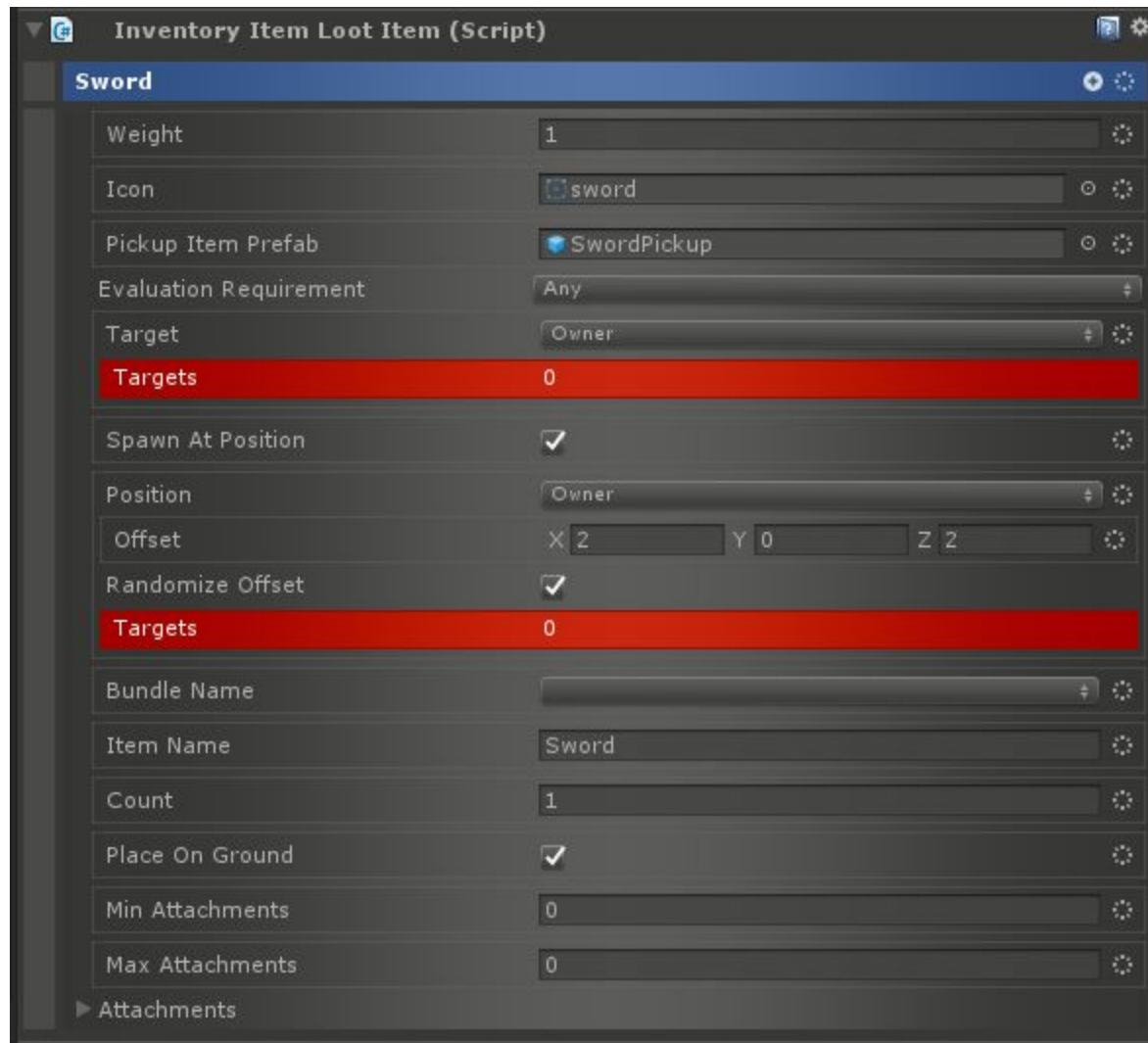
Attachment Type was previously discussed and sets whether this is a Prefix, Suffix, Socket, or Attribute Attachment. Name Modifier will be used in the case of a Prefix or Suffix. Description Modifier is added to the description in all cases.

Conditions can be childed to the Attachment and when this is done, based on the Evaluation Requirement, a number of them must be true in order to attach the Attachment to an Item.

Actions may also be childed to the Attachment. They will be run when the Attachment is attached to an Item. Typically, Continuous Actions should be used here. They will be run with the Owner set to the Inventory Item they are attached to.

Inventory Item Loot

Using the Loot System, Inventory Items can be distributed as part of Loot rolls using the Inventory Item Loot Item.



The Weight defines how likely this Item will be rolled in the Loot Roll. A higher weight means more likely. Icon should be set to the Item's icon and Pickup Item Prefab should be set to the Placed Item (if Spawn At Position is set). Target should typically be set to Owner, but it refers to the Character to give the Item to.

Spawn At Position will spawn the Pickup Item Prefab at the specified position. Bundle Name and Item name should be used to point to the Inventory Item to give out. If not using Asset Bundles, Bundle Name can be blank, but the Item should be present in the Inventory Service's list of Items.

Count defines the number of items to give out. Place On Ground will raycast to put the item on the ground. Min and Max Attachments sets the number of attachments to randomly generate from the list of Attachments. These will be attached to the Item when looting.

Conditions

- Can Add Attachment Condition - True if the attachment can be added to the item.
- Can Equip Item Condition - True if the item can be equipped by the target.
- Can Pick Up Item Condition - True if the item can be picked up by the target.
- Can Use Item Condition - True if the item can be used on the target.
- Have Item Condition - Check if the target has an item.
- Have Item Quantity Condition - Check if the target has a quantity of an item.

Actions

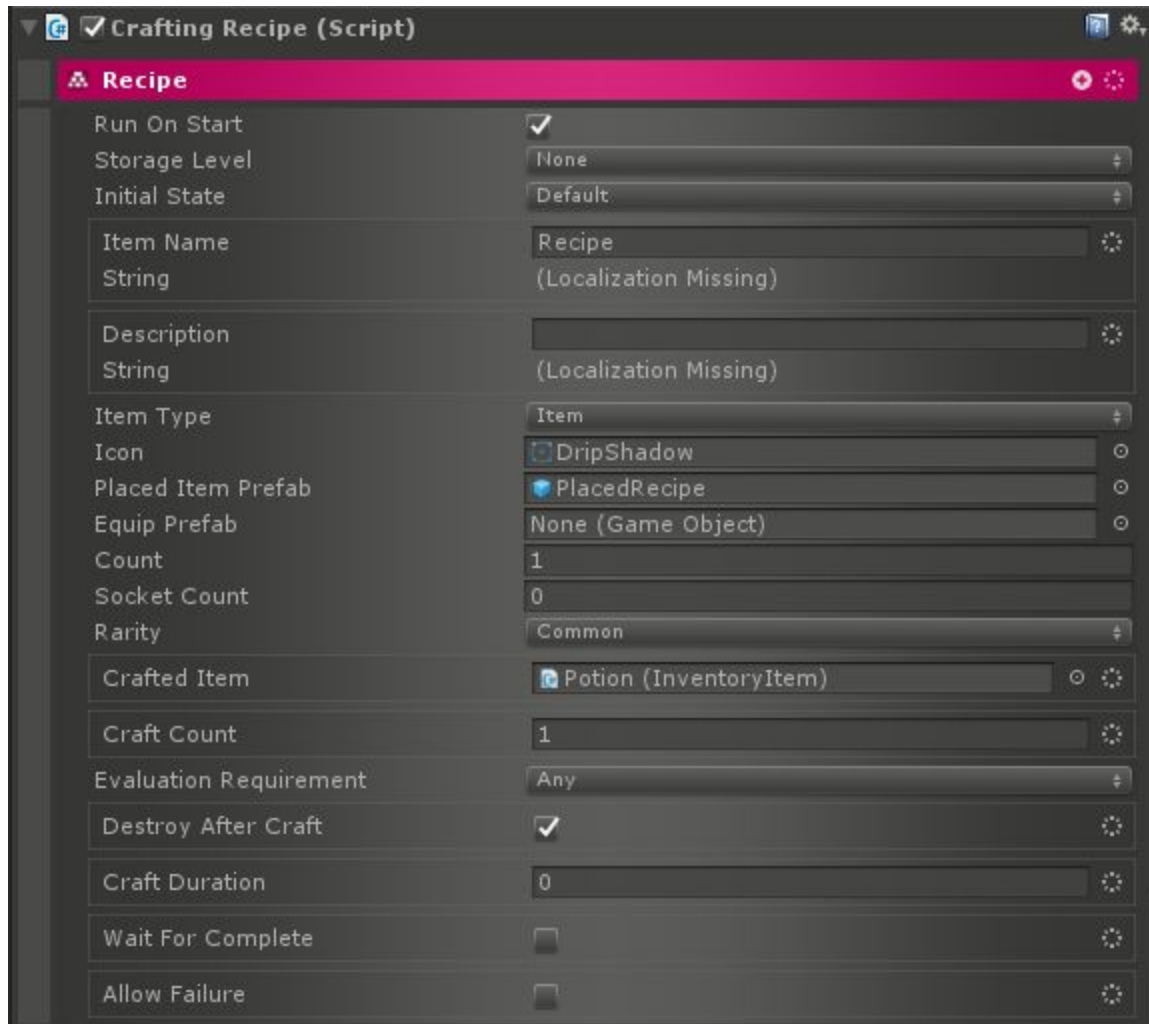
- Add Attachment Action - Add an attachment to an item.
- Add Item To Storage Action - Moves an item from target to storage.
- Clear Inventory Action - Clears the inventory of the target.
- Drop Item Action - Drops an item from target's inventory.
- Equip Item Action - Places an item in an equip slot.
- Give Item Action - Give target a new instance of an item and equip it.
- Move Item Action - Move an item from one equip slot to another.
- Pick Up Item Action - Cause target to pick up an item and put it in a slot.
- Remove Attachment Action - Removes an attachment from an item.
- Remove Item Action - Removes an item from target's inventory.
- Remove Item From Storage Action - Removes an item from target's inventory.
- Remove Item Quantity Action - Removes a quantity of an item from target's inventory.
- Use Item Action - Use an item on a target.

Triggers

- Have Item Quantity Trigger - Fires when the target Character has a specific count of an item.
- Have Item Trigger - Fires when the target Character has an item.
- Picked Up Trigger - Fires when an item is picked up.
- Use Item Trigger - Fires when an item is used.

Crafting

The main component of CAT RPG's Crafting System is the Recipe. Recipes are actually just a special type of inventory item. Required ingredients are childed to them as Inventory Items with their counts set appropriately.



Most of the options are the same as an Inventory Item. The new options are:

- **Crafted Item:** The item that results from crafting.
- **Craft Count:** The number of items produced at a time.
- **Destroy After Craft:** If true, then the recipe will be destroyed after it is used.
- **Craft Duration:** If greater than 0, the craft will take this long to complete.
- **Wait For Complete:** If true, then the craft will wait for a `CompleteCraftAction` to target the Recipe before completing.
- **Allow Failure:** If true, then if a `FailCraftAction` targets the Recipe while crafting, then the craft will be considered a failure and no items will be crafted.

Recipes also have a new state called `OnCrafting`. This state is active while the recipe is in the process of crafting.

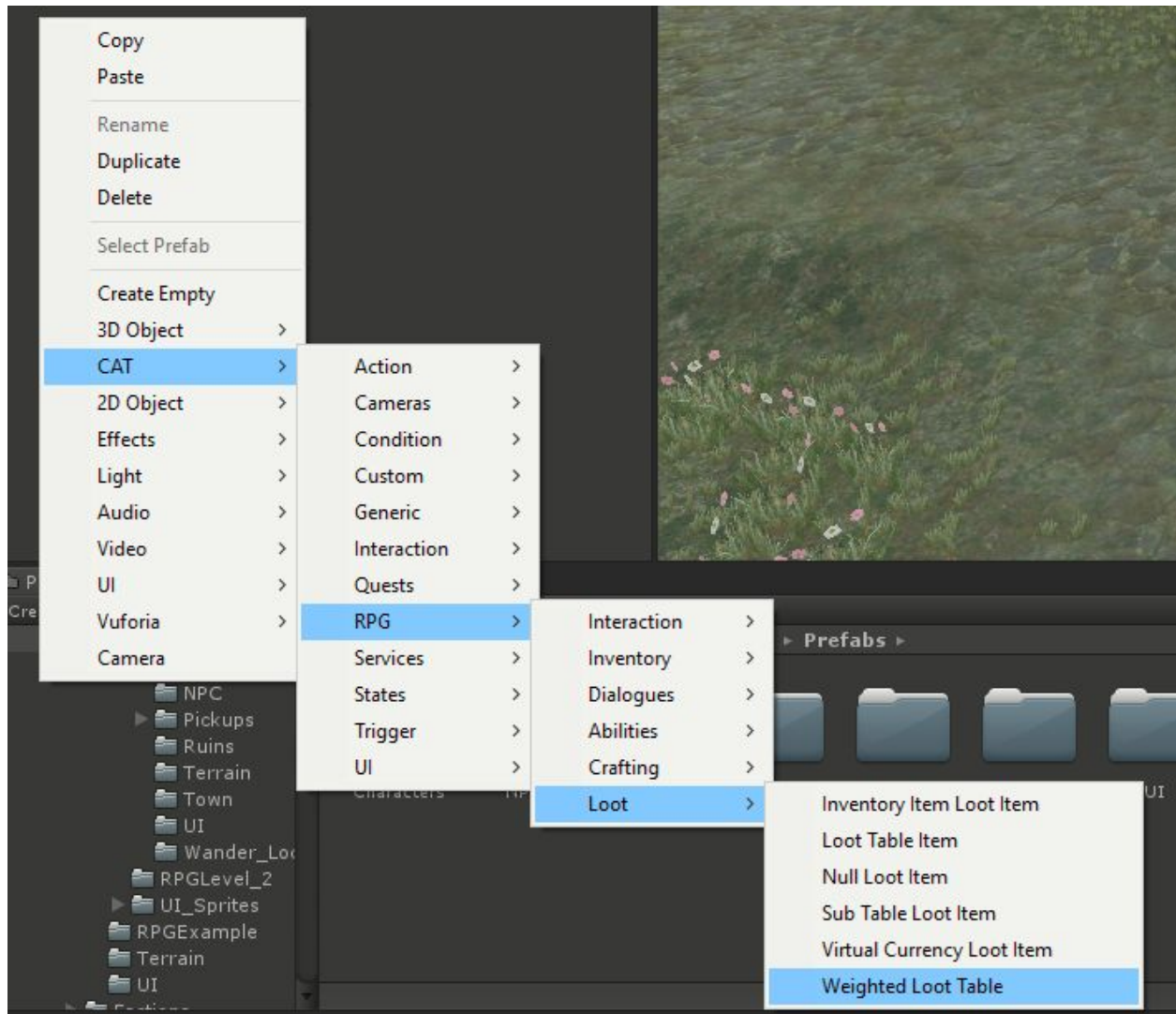
Conditions can be attached to a Recipe as well. These are evaluated in the `Can Craft Condition` or `Trigger`. Both take a Crafter and a Recipe to check. They check both whether the Conditions pass and whether the Crafter has the ingredients required.

There are two additional Actions associated with the Crafting System:

- Craft Item Action: This will check the conditions and availability of ingredients and then initiate Crafting.
- Get Recipe From Ingredients Action: This will find a Recipe which matches can be crafted with the given ingredients.

Loot

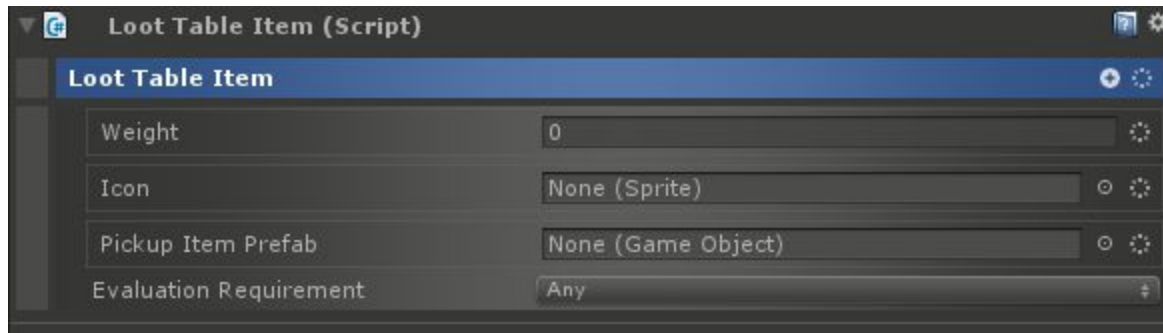
The Loot System in CAT RPG mainly consists of a Weighted Loot Table and Loot Table Items along with a Loot Interaction to distribute the Loot. The best way to get started with the Loot System is to create a Weighted Loot Table:



There are no parameters on the Weighted Loot Table, but each Loot Table Item added to it has a Weight. When rolling for Loot, all the Weights in the table are added and a random number is generated up to that total. Then, the Loot Table Item to give out is decided based on the roll. So, the percent chance a Loot Table Item will be given out is equal to its Weight divided by the Weight of all Loot Table Items in the Weighted Loot Table. There are several types of Loot Table Items.

Loot Table Item

All other Loot Table Items are based of of this, so they retain the parameters and functionality but extend on it:



The Weight as discussed previously determines how likely this item will be selected. The Icon can be used in cases where the Looted Items are displayed to the player in the UI. The Pickup Item Prefab can be something like a Placed Item or other Prefab which will be spawned when the loot is given out in certain circumstances.

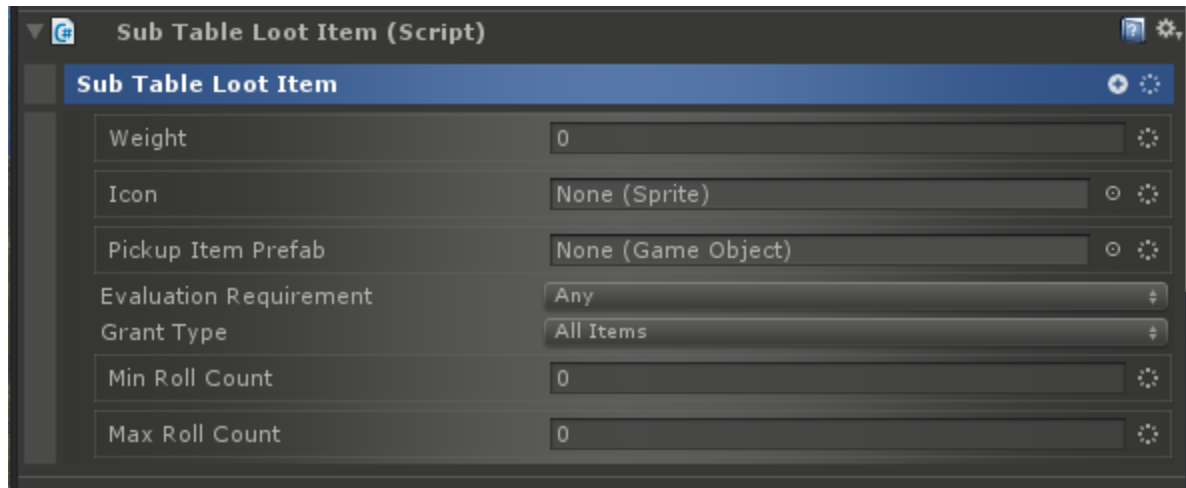
Actions can be childed to the Loot Table Item. They will be executed if the Loot Item is given out. Conditions can also be childed to the Loot Table Item. In accordance to the Evaluation Requirement, they will determine if this Loot Table Item is available in any given Loot Roll.

Null Loot Item

This is an empty Loot Table Item which will always return nothing. This is useful in cases where there is a chance that nothing will drop. It has the same parameters as the base Loot Table Item.

Sub Table Loot Item

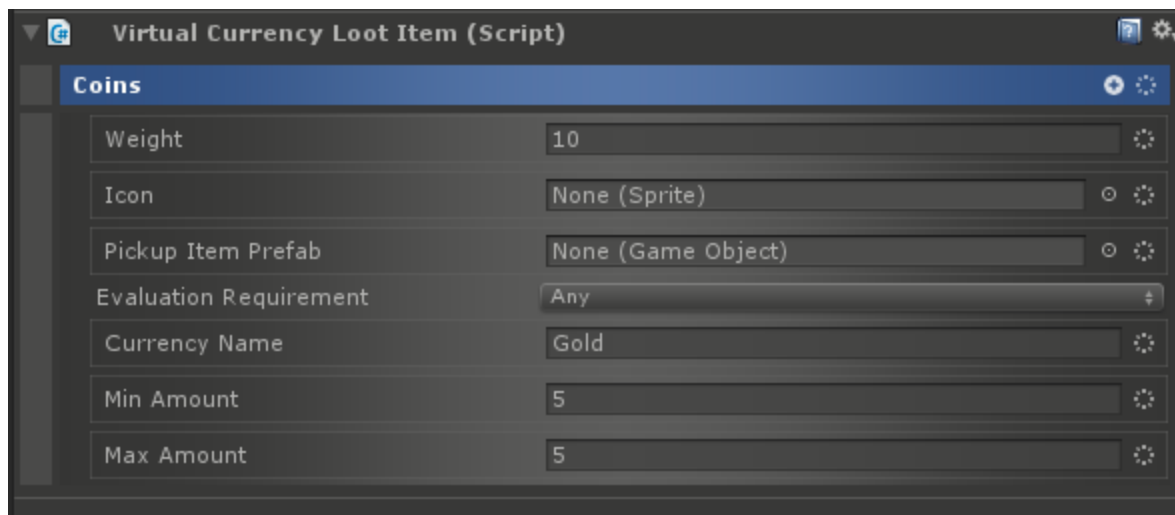
In cases where one entry on the Loot Table may need to give out multiple items, the Sub Table Loot Item can be used. One or more Weighted Loot Tables can be childed to this Item.



The additional options are Grant Type, which can be set to grant all items in each childed Weighted Loot Table, or a number of rolls off of each. In the latter, Min and Max Roll Count determine the number.

Virtual Currency Loot Item

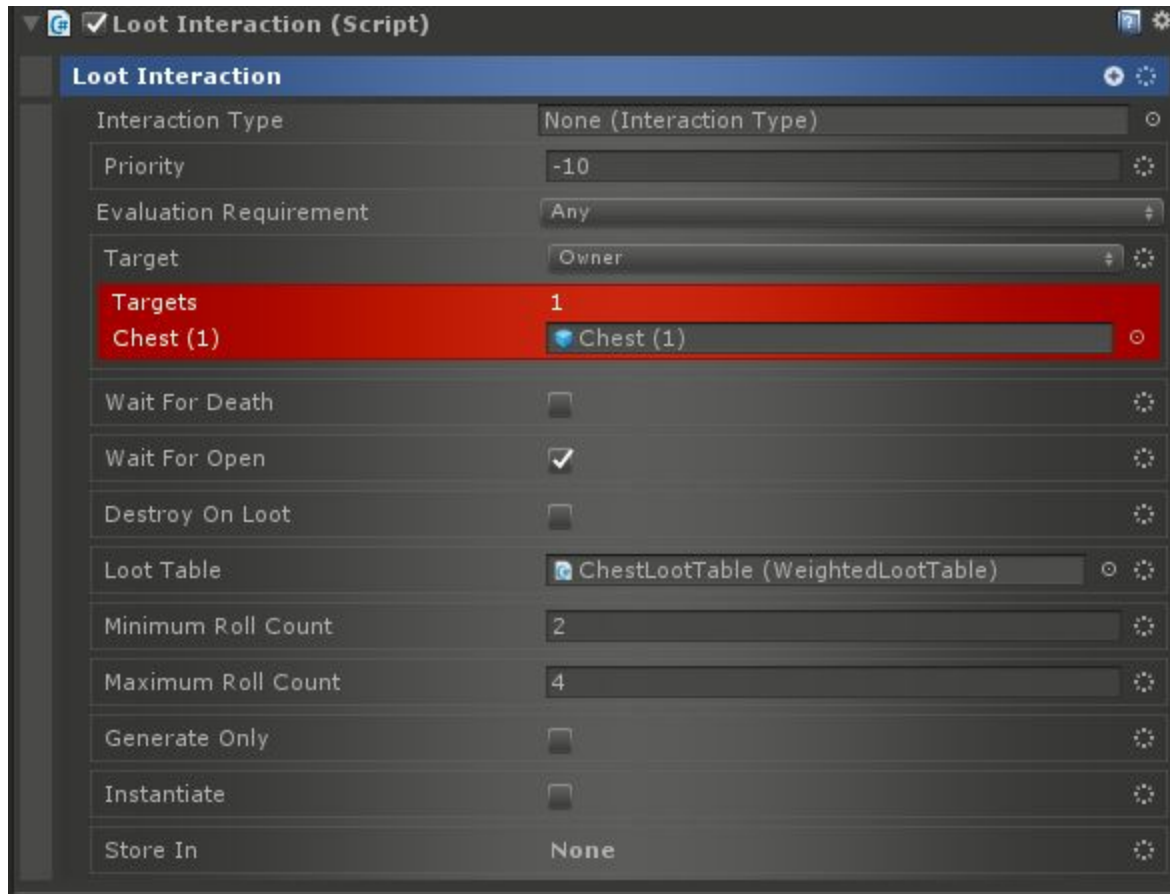
This can be used to directly grant currency to the player as a loot item.



The additional options are the name of the Currency and the amount to give which will be a random number between Min Amount and Max Amount.

Loot Interaction

This interaction should be added to any Character, Chest, Destructible, or other Interactive Object that will have loot.



Wait for Death will cause the Interaction to trigger when the Character it is attached to dies. Wait for Open will similarly cause the Interaction to trigger when the Door or Chest it is attached to is opened. Destroy On Loot causes the target to be destroyed when the Interaction is activated.

Loot Table should specify the Weighted Loot Table to use, and Minimum Roll Count and Maximum Roll Count specify how many rolls to do on that table. If Generate Only is checked, the Loot Items will be rolled and added to Store In but not otherwise distributed. If Instantiate is checked, the Loot Items' Pickup Item Prefabs will be created in the world.

Conditions

- Has Loot Condition - Check if a given target has any loot.

Actions

- Generate Loot Action - Roll some loot on a table and store it in a value.
- Grant Loot Action - Roll loot and grant it to the target.

Behaviors

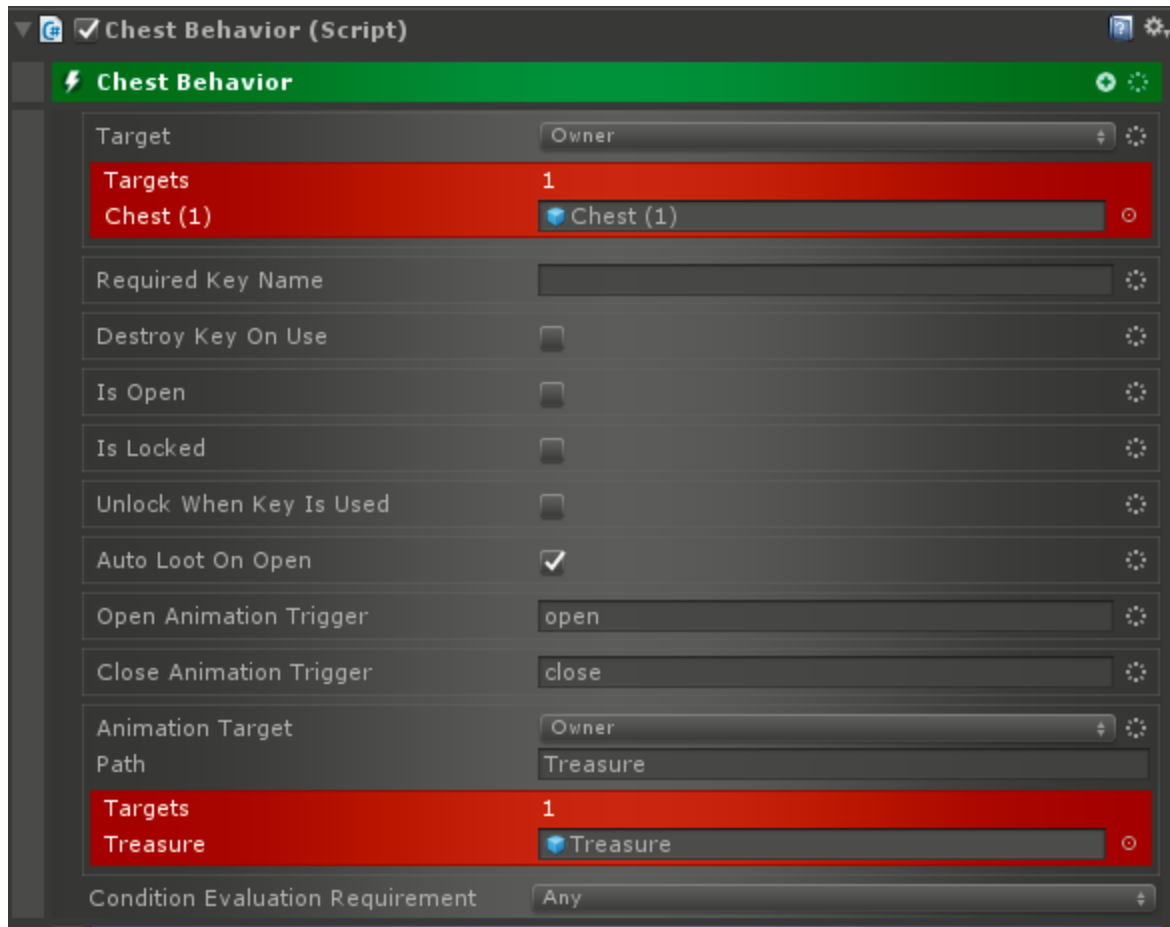
Behaviors are a new type of Action in CAT RPG. They are continuous actions (i.e. they run until something stops them), and they are typically even higher level than other Actions in CAT. They can be added to States in a State Machine or anywhere else Actions are allowed. They make their target perform some complex behavior like flee, pursue, show combat text, etc.

Interactive Objects

Interactive Objects aren't a system like the other folders. They're more a collection of behaviors to simplify the process of creating and connecting doors, switches, NPC spawners, and other useful things.

Chests

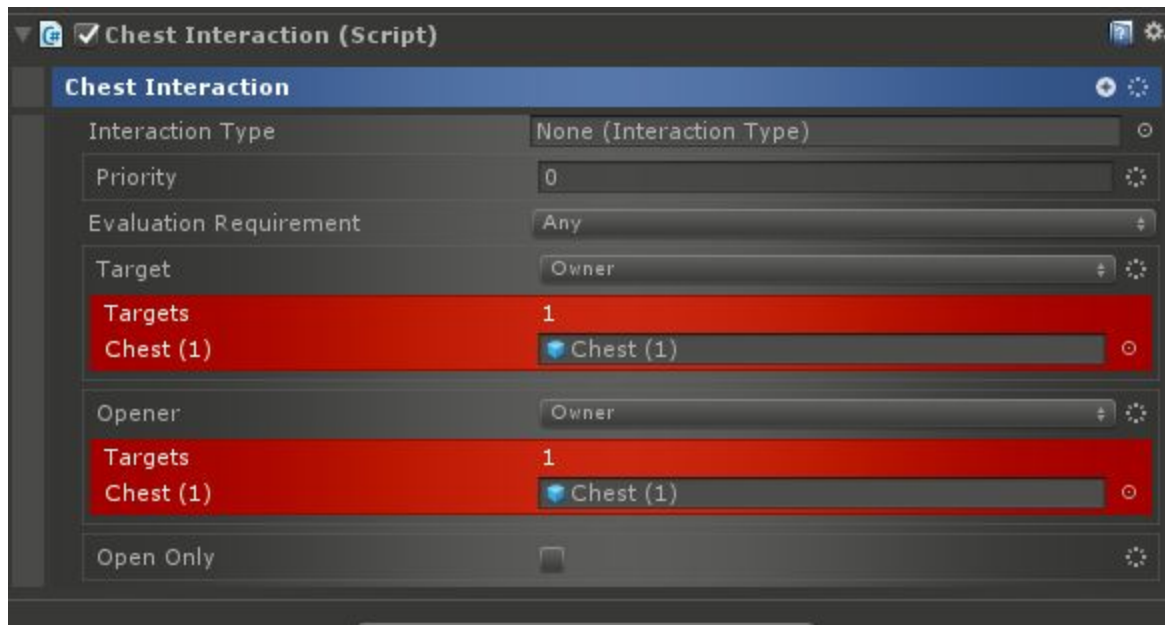
Chests are defined by using the Chest Behavior. This Behavior can be used to make an interactive object give out loot when opened. Chests can be locked and require keys. They can be trapped, and the Behavior can drive the animation of the chest when opening or closing.



There are quite a few options for the Chest Behavior. Target specifies the Game Object to turn into a Chest. Required Key Name sets the name of an Inventory Item that can be used to unlock the Chest. Is Open defines whether the Chest is open or closed. Is Locked can be used to lock the Chest and optionally require a key. Unlock When Key Is used will set Is Locked to false when a key is used on the Chest. Auto Loot On Open will cause the opener of the Chest to get any loot attached to it when the Chest is opened. Open Animation Trigger is the name of the animation trigger parameter to set when opening. Close Animation Trigger is the name of the animation trigger parameter to set when closing. Animation Target can be used to point to the Game Object which will be animated. Finally, Condition Evaluation Requirement specifies how many conditions must pass in order for the chest to be able to be opened.

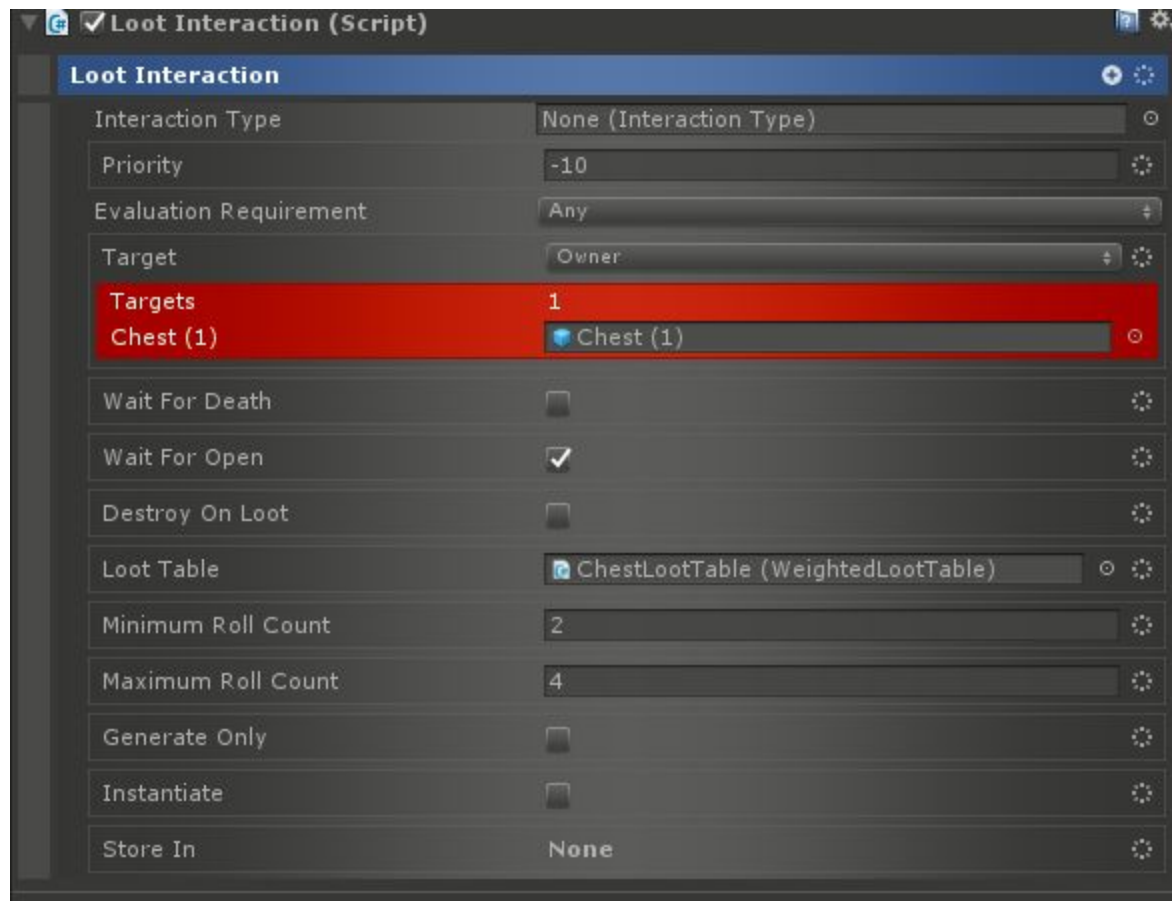
Conditions can be childed to the Chest Behavior in order to only allow opening in certain circumstances. These conditions are executed with the opener of the chest as the target. Actions and Stop Actions childed to the Chest Behavior will be executed when opening or closing respectively. Additionally, one or more Trap Behaviors can be childed to the Chest Behavior. In this case, they will be triggered whenever the Chest is opened (and will roll trigger chance / check their own conditions as usual).

In order to interact with a Chest, the simplest method is to apply a Chest Interaction.



Typically, the Opener should be set to either Player (in single player games) or Targeted (in multi player games). Target of course should point to the Chest, and if Open Only is set, the interaction will only cause the Chest to open instead of the default which is to toggle between open and closed.

The other aspect of setting up a Chest is adding Loot. This is done by adding a Loot Interaction to the Chest:



When doing this, specify “Wait For Open” to make sure that the loot is unavailable until the Chest is opened. The rest of the parameters should be as normal for Loot Interactions.

Conditions

- Can Open Chest Condition - Checks if a Chest can be opened.
- Is Chest Locked Condition - Check if a Chest is locked.
- Is Chest Open Condition - Check if a Chest is open.
- Is Chest Trapped Condition - Check if a Chest is trapped.

Actions

- Lock Chest Action - Set the locked state of a Chest.
- Open Chest Action - Open, close, or toggle the state of a Chest.

Triggers

- Trigger that fires when target Chest is open.

Destructibles

The Destructible Behavior allows a Game Object to be destroyed and broken up into pieces. This can be triggered with another action or if attached to a Combatant, it can be triggered by Death. The latter can be useful for crates or other things that shatter when attacked. To make an object Destructible, add the Destructible Behavior:



The Target should point to the object to be destroyed. Destruct On Death causes the Target to be destroyed when it dies via the Combat System. Destroyed Pieces is a list of pieces to spawn when the Target is destroyed. Spawn Type can be set to All Pieces which means every piece in the Destroyed Pieces list will be spawned or Random Pieces where a number of random selections from the list will be spawned. If the latter, then Min and Max Random Piece Count are applied to determine how many pieces will be spawned.

Outward Force specifies how much force to give to the spawned pieces, and Explosion Radius, if specified, will create an explosive force instead of a normal force. This should be used with Explosion Position to determine where the explosion originates.

A Loot Table can be attached along with a minimum and maximum Roll Count. If specified, then the Loot Table will be rolled a number of times when the object is destroyed and the loot will be distributed.

Triggers can be childed to the Destructible Behavior. In this case, the Evaluation Requirement will be used to determine how many triggers must fire in order for destruction to be triggered.

Actions childed to the Destructible Behavior will be executed when the object is destroyed.

Actions

- Destroy Destructible Action - Causes a Destructible to be immediately destroyed.

Doors

The Door Behavior can be used to turn a State Machine into a Door. Doors can be opened and closed, they can be trapped, locked, and transport the Character passing through to a new Area or Realm. To create a door, add a Door Behavior to a State Machine.



The Target parameter should specify the object to turn into a Door. Required Key Name can be used to require a Key in order to open or unlock the Door. Set Destroy Key On Use to have the key be destroyed when used.

Is Open determines if the Door is opened or closed. Is Locked determines if it is locked or not. If Unlock When Key Is Used is set, then the Door will be unlocked any time the key object is used.

Destination Scene can be used to load a new scene when traversing the door and the Mode sets how the scene is loaded. Load Progress can also be used to track progress while loading. Destination Area ID and Realm ID can be used to transport the Character to a new Area or Realm. Destination Door specifies what door the Character should appear at on the other side. Set this to None if the door is just something that can be walked through.

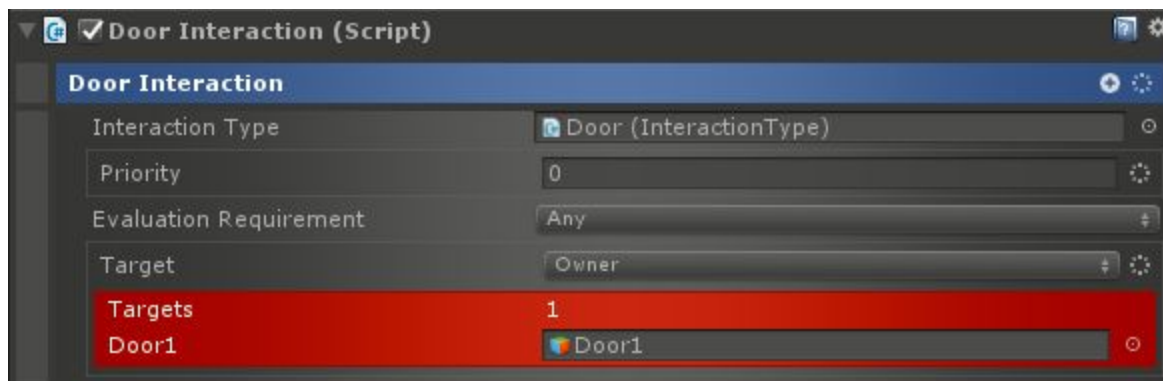
For animation, the Animation Target can be specified as the object to animate, and the Open and Close Animation Triggers will be triggered on its Animation Controller respectively. If Closed Collider Target is set, then it will be enabled when the door is closed and disabled when open.

Conditions gating the opening of the door can be childed to the Door Behavior. If this is done, then Condition Evaluation Requirement can be used to determine how many must be true before the door is able to be opened.

Actions and Stop actions can also be childed to the Door Behavior. These will be executed on open and close respectively.

Finally, Trap Behaviors can be childed to the Door Behavior to add traps to the door. These will be triggered whenever the door is opened.

In order to use the door, the Door Interaction can be added to it:



Conditions

- Can Open Door Condition - Checks if a Door can be opened.
- Is Door Locked Condition - Check if a Door is locked.
- Is Door Open Condition - Check if a Door is open.
- Is Door Trapped Condition - Check if a Door is trapped.

Actions

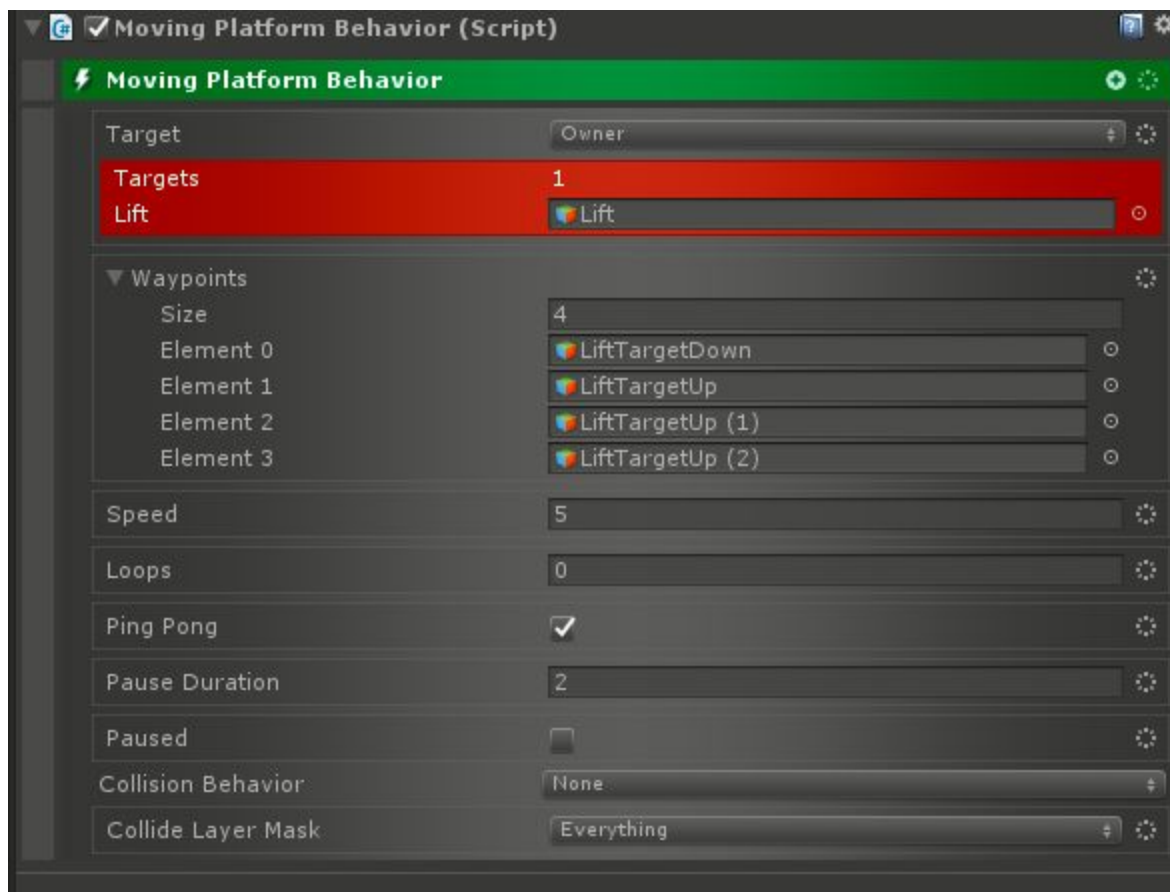
- Lock Door Action - Set the locked state of a Door.
- Open Door Action - Open, close, or toggle the state of a Door.

Triggers

- Door Trigger - Trigger that fires when target Door is open.

Moving Platforms

Moving Platforms are a way to transport Characters from one point to another using a platform. They can follow a path, be paused, and have different actions when stepped on. To create a Moving Platform, add a Moving Platform Behavior to a State Machine:



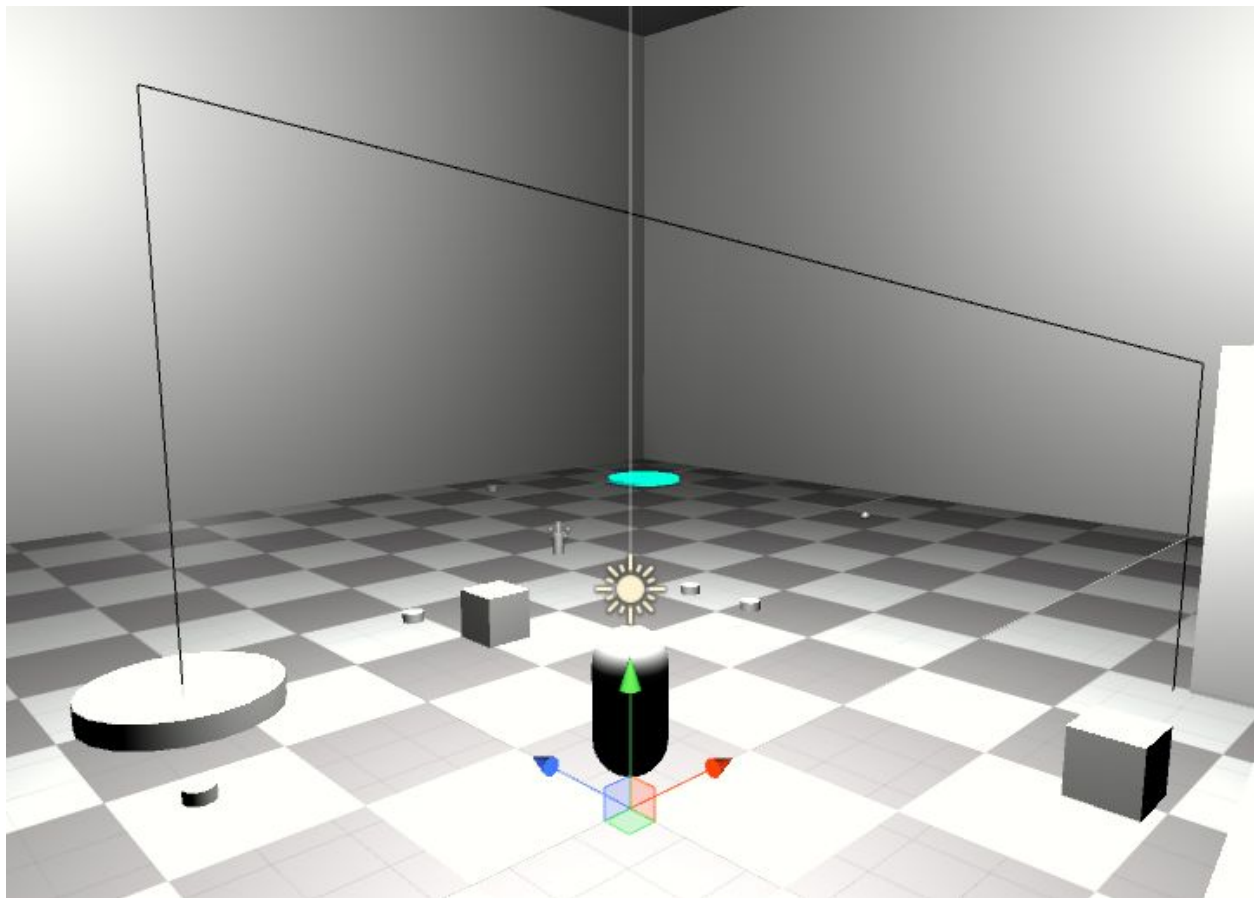
The Target here should be pointed to the Moving Platform object. Waypoints contains a list of Game Objects which signify where the waypoints are for the platform to move to. The order here is the order it will travel to them.

Speed defines how fast the Platform moves. Loops if nonzero will cause the Platform to loop through the waypoints that many times before stopping. If Ping Pong is checked, then the Platform will move through the waypoints forward and then go through them in reverse order.

Pause duration sets how long the Platform waits at each waypoint. Paused can be used to temporarily stop the Platform.

Collision Behavior defines what the Platform does with it collides with something that matches the Collide Layer Mask. The default is None which means there is no change in action. If set to Stop, the Platform will stop when something collides with it. Usually this is the Player colliding. If set to Fall, the Platform will enable gravity on an attached Rigidbody, causing it to fall.

When waypoints are defined, the Moving Platform will draw gizmos in the editor to show its path:



Actions

- Pause Moving Platform Action - Pauses or unpauses a Moving Platform.

Spawn Camps

A Spawn Camp is a collection of Spawners that work as a group. This can be useful for enabling or disabling a group of Spawners at a time or based on a trigger. The Spawners should be childed to the Spawn Camp. To create a Spawn Camp, just add the Spawn Camp Behavior to a State Machine:



Target is the object to make into a Spawn Camp, and Paused is whether the Spawners are paused currently. Additionally, Triggers can be added under the Spawn Camp which based on the Evaluation Requirement can pause or unpause all the Spawners. Actions and Stop Actions can also be childed to the Spawn Camp. They will run when unpaused and paused respectively. Finally, there's the Destroy All On Stop option which when set will destroy all spawned things as soon as the Spawn Camp Behavior is stopped.

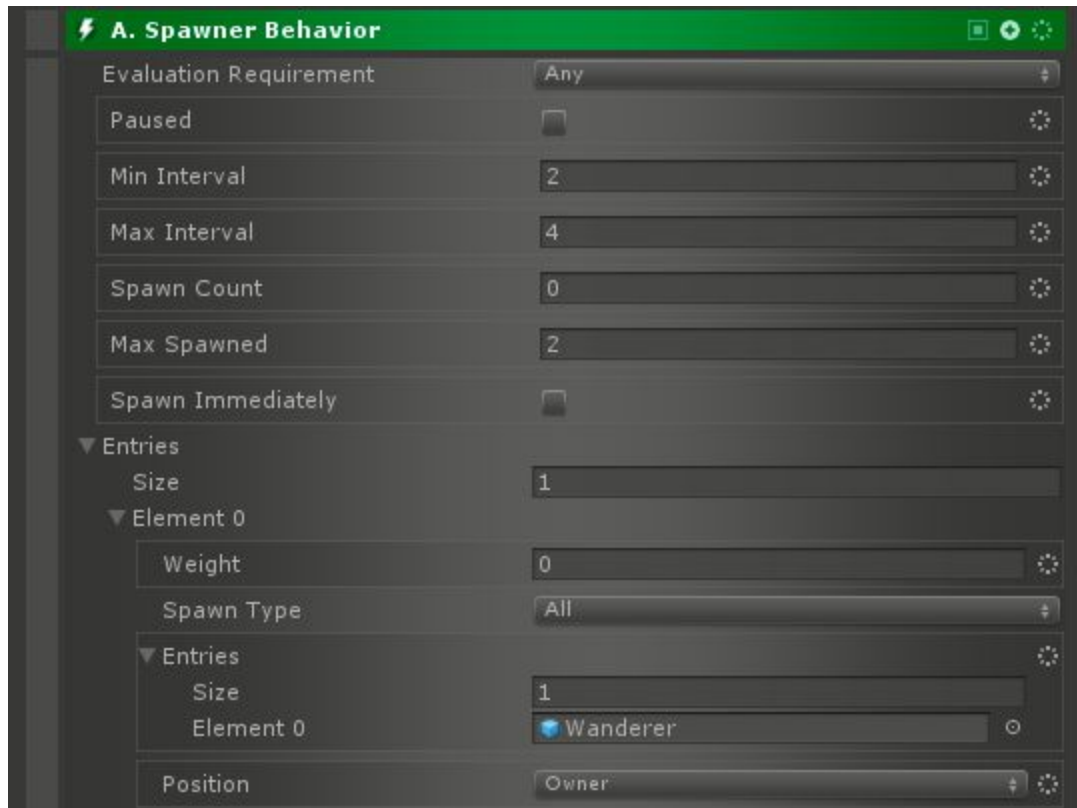
Actions

- Destroy Spawn Camp Spawned Things Action - Destroy all things spawned by the target Spawn Camp.
- Pause Spawn Camp Action - Pause or unpause spawning for a Spawn Camp.

Spawners

A Spawner can spawn in one or more prefabs (usually enemy Characters). There can be a delay between spawns, a maximum number of spawns at one time and a maximum number spawned for the lifetime of the Spawner.

To create a Spawner, add a Spawner Behavior to a State Machine or a Spawn Camp:



Paused specifies whether this Spawner is actively spawning or not. Min and Max Interval defines the time in between spawns. Spawn Count can be used to set the maximum number of spawns for the duration of this Spawner. Max Spawned sets the maximum number of spawns active at a time.

If Spawn Immediately is checked, then the Spawner will spawn something as soon as it is active. Entries contains the list of prefabs to spawn. It is weighted and each element also contains multiple prefabs. The Spawn Type on each determines if all or a single random entry are spawned.

Triggers can be childed to the Spawner and using the Evaluation Requirement setting, the Spawner will be paused or unpaused depending on how many Triggers are active.

Actions

- Destroy Spawner Spawned Things Action - Destroy all things spawned by the target spawner.
- Pause Spawner Action - Pause or unpause spawning for a spawner.

Switches

Switches are pretty typical Interactive Objects. They have an on and off state and the player can interact with them to toggle that. They can control other Interactive Objects such as Doors, Chests, or Moving Platforms.

To make a new Switch, add a Switch Behavior to a State Machine:

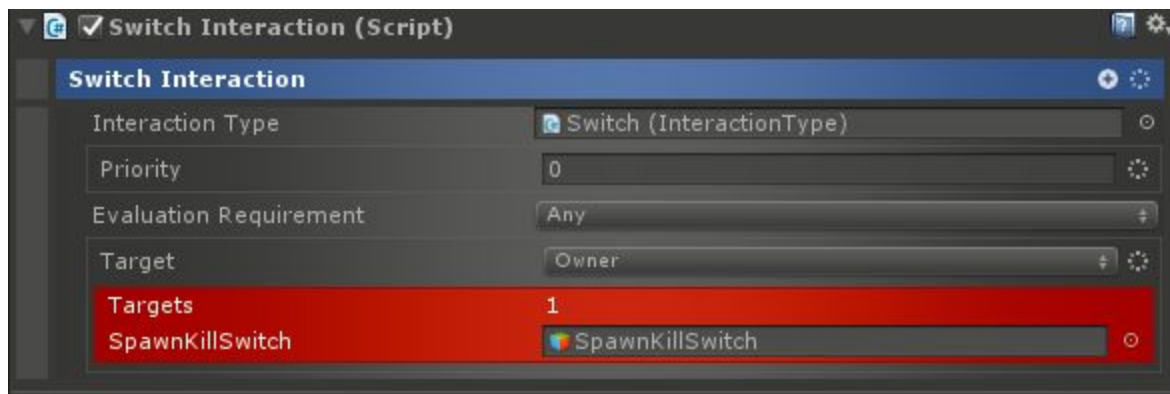


The Target should point at the actual Switch. State is whether the Switch is on or off. Is Momentary can be used to set whether the switch is a pushbutton or a toggle. Notification Target can be used to notify other Interactive Objects of when the Switch changes state.

Conditions can be childed to the Switch. Using the Evaluation Requirement to determine how many must be true, they will gate whether the state of the switch can be changed.

Actions and Stop Actions can also be childed to the Switch. They will run when the Switch is turned on or off respectively.

To make a Switch interactable, add the Switch Interaction to it:



There aren't any special parameters here, and as long as there's a collider and Interact via Mouse is enabled on the Switch's Interaction Manager, or there is some button for the Switch Interaction Type, the Switch should be able to be interacted with by the player.

Conditions

- Switch Condition - Checks the state of a Switch.

Actions

- Flip Switch Action - Changes the state of a Switch.

Triggers

- Switch Trigger - Fires when a Switch is in a specific state.

Traps

In addition to being able to be placed on Doors and Chests, Traps can be used by themselves in order to be sprung on the player in other scenarios such as proximity. To create a freestanding Trap, add a Trap Behavior to a State Machine:

1. Trap Behavior	
Target	Owner
Targets	1
Trap	Trap
Armed	<input checked="" type="checkbox"/>
Max Trigger Count	-1
Trigger Chance	0.5
Detection Difficulty	0
Detection Stat Name	
Disarm Difficulty	0
Disarm Stat Name	
Trigger Evaluation Requirement	Any
Condition Evaluation Requirement	Any

The Target should always point at the Trap that is being set. Armed determines if the Trap is active or not. Max Trigger Count can be set to -1 to make the Trap trigger any number of times. If it is set above 0, the Trap will only trigger that many times.

Trigger Chance can be used to set how likely the Trap is to go off. It should be in the range of 0 to 1. Detection Difficulty is how difficult the trap is to detect, and Detection Stat Name is the name of the Stat to use for detection. Similarly, Disarm Difficulty and Disarm Stat Name are used for disarming the Trap.

Triggers can be childed to the Trap and using the Trigger Evaluation Requirement, some number of them firing will set off the Trap. Conditions can also be childed to the Trap. These use the Condition Evaluation Requirement and are run after the Trap has triggered to determine if it goes off.

Several types of Actions can be childed to the Trap to perform different functions. Regular Actions will be run when the Trap goes off on a target. Else Actions will be run when the Trap is triggered, but the Conditions fail, or the Trigger Chance is not met. Stop Actions will be run if the Trap is disarmed successfully.

Conditions

- Detect Trap Condition - Attempt to detect a trap.

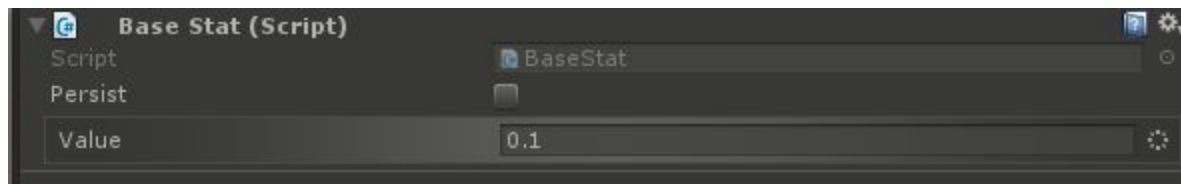
Actions

- Disarm Trap Action - Attempt to disarm a Trap.
- Set Trap Armed Action - Set the armed state of a Trap.

Stats

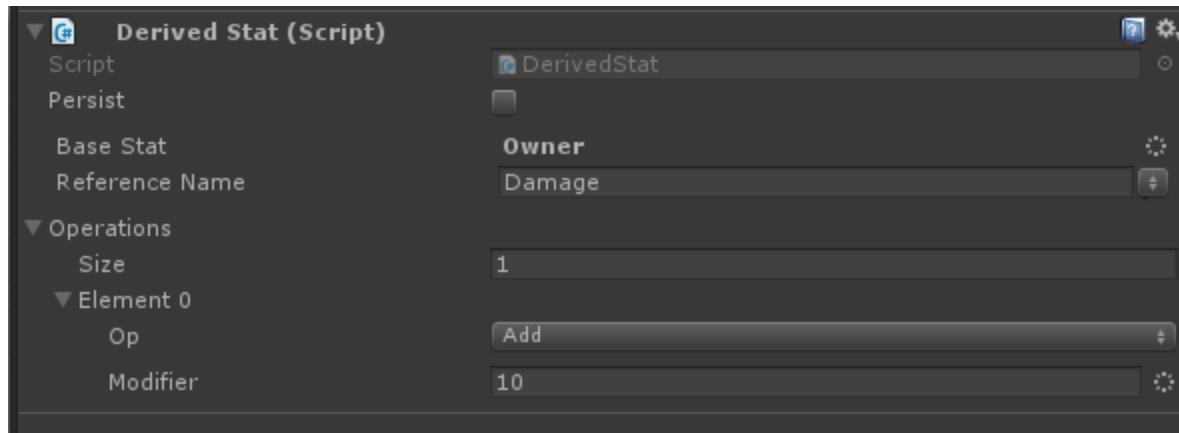
The Stats System in CAT RPG is fairly simple and is mainly based on the existing Value System in Base CAT. Stats are stored on a Character in a Value Holder. All Stats produce a Float Value type. There are several types of Stats: Base Stat, Derived Stat, Fuel Stat, and Stats provided by other systems such as Leveled Stats from the Progression System. All Stats can also have Stat Modifiers attached to them. These are typically used for Buffs and Debuffs.

Base Stat



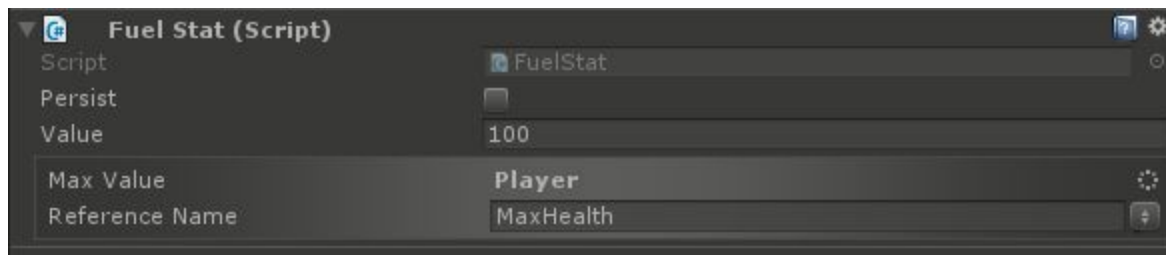
Base Stats look very much like Float Values except they can reference other Values or have their own. Unless referencing another Value, these Stats won't change unless a Stat Modifier is applied to them or they are explicitly set.

Derived Stat

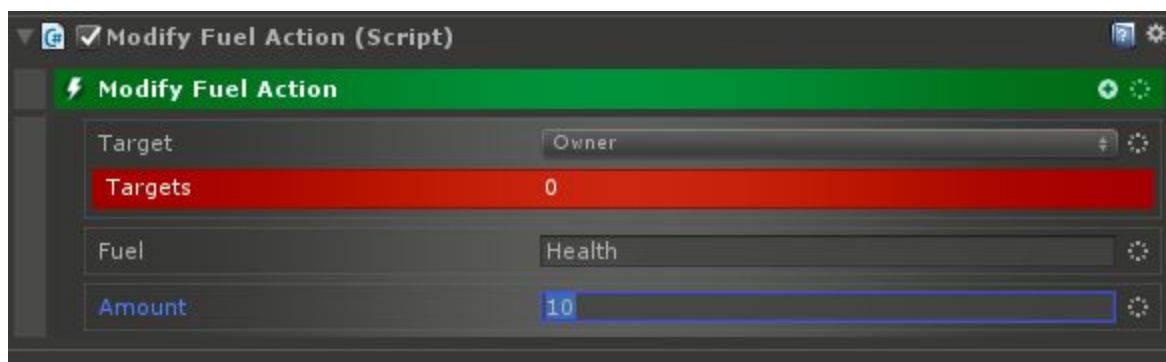


Derived Stats reference another Stat and then have a set of Operations that are performed on it. They will change depending on the Base Stat or any of the Modifiers. The Value of Derived Stats can not be directly set.

Fuel Stat



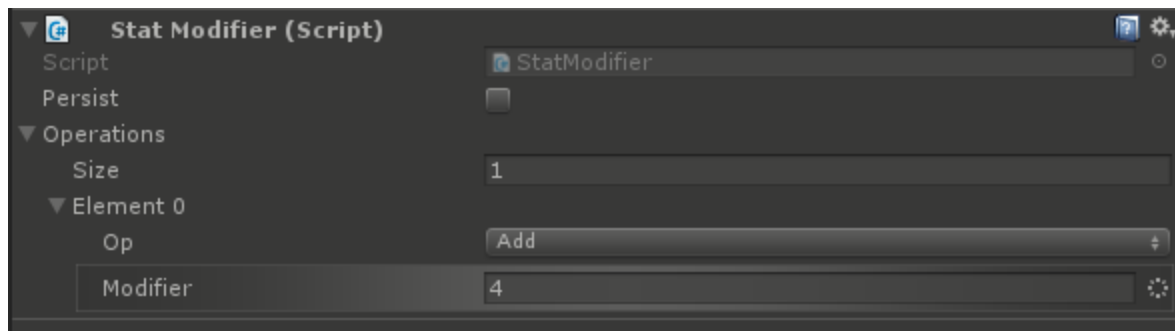
Fuel Stats are typically used for something with a Max Value that changes often. Some examples include Health and Mana. Typically, the Max Value will refer to another Stat. Fuel Stats can be set directly and / or modified using Stat Modifiers. The Modify Fuel Action can be used to modify the Fuel:



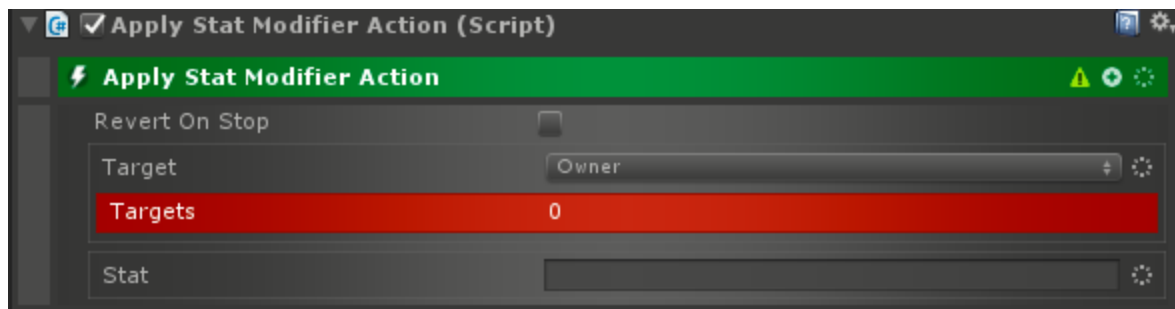
The Target should be set to the Character to modify the Fuel of, and the Fuel should be set to the name of the Fuel to modify. Amount should be the amount to change it by. Negative numbers will cause the Fuel to go down and positive numbers will cause it to go up.

Stat Modifiers

Stat Modifiers temporarily modify the value of a Stat using a list of Operations.

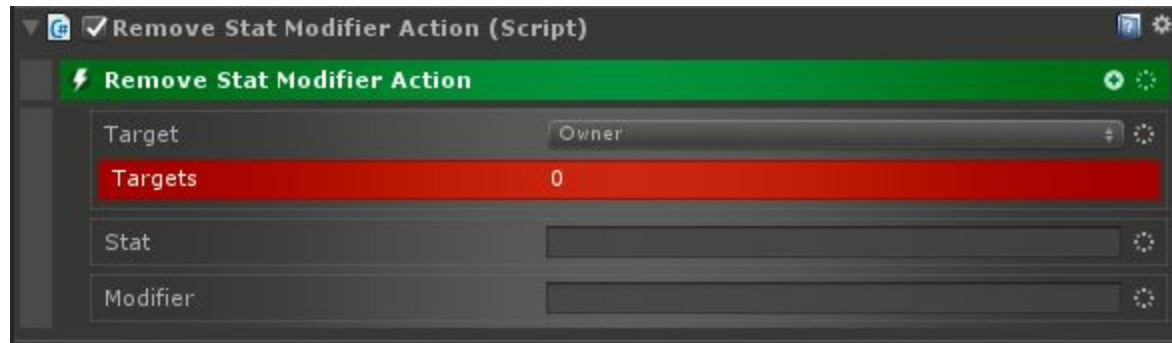


Stat Modifiers can be applied using the Apply Stat Modifier Action:



The Target should be set to the Character to modify a Stat on and the Stat should be set to the name of the Stat to modify. Stat Modifiers should be childed to the Action. If Revert On Stop is set, the Modifiers will be removed as soon as the Action is stopped.

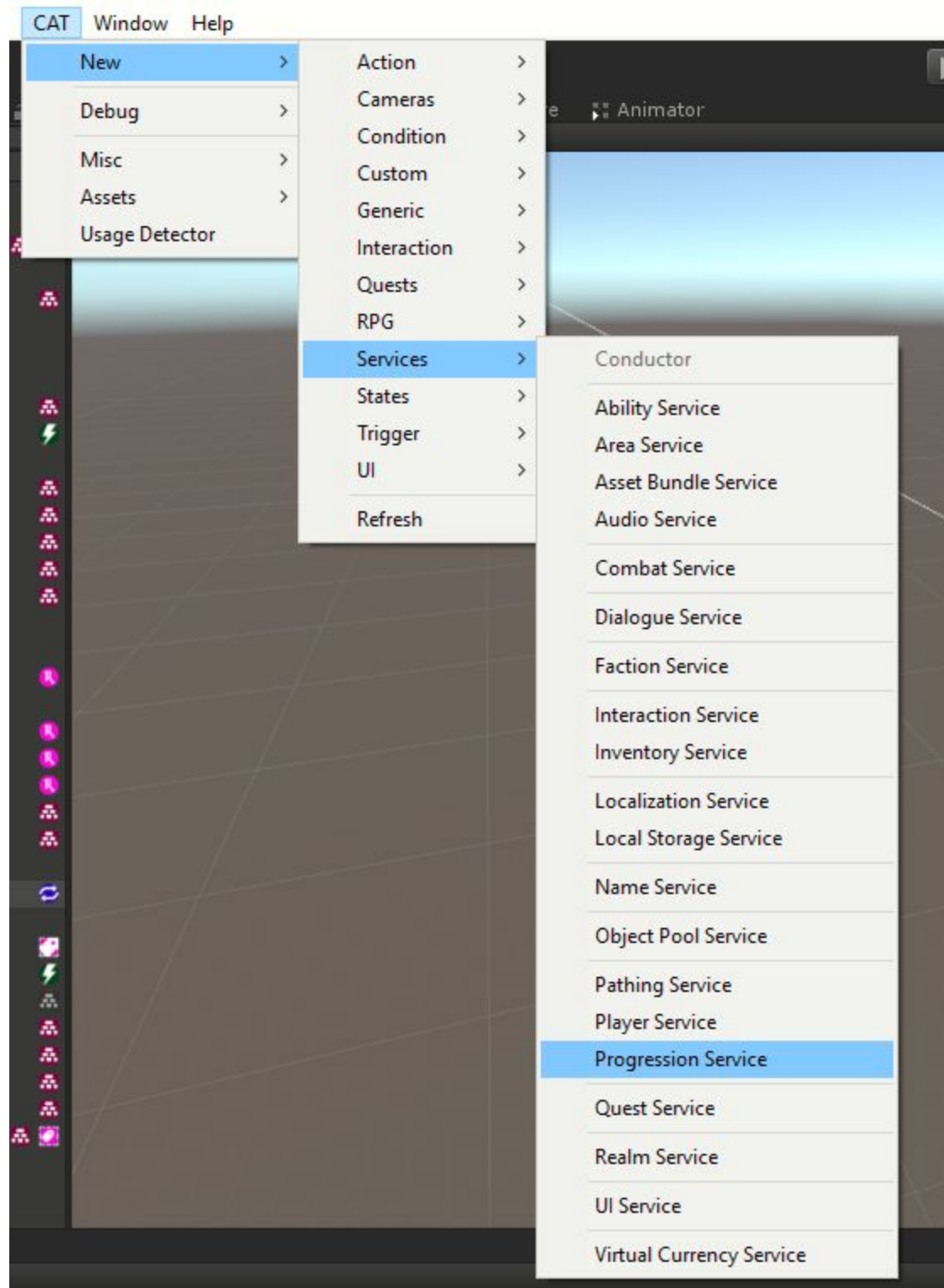
The other way to remove Stat Modifiers is through the Remove Stat Modifier Action:

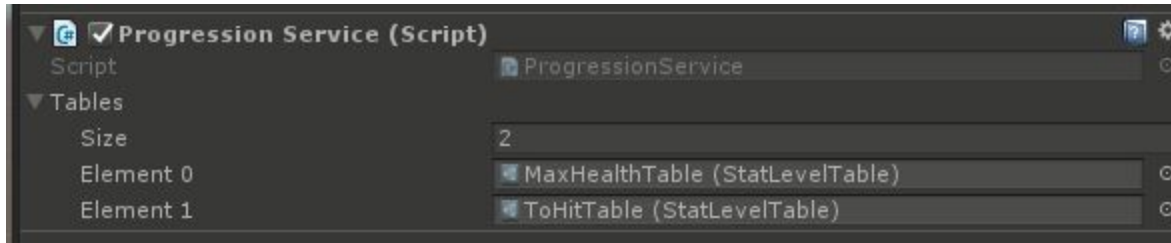


Target should be set to the Character to remove the Stat Modifier from, and Stat should be set to the name of the Stat. Modifier should be set to the name of the Stat Modifier to remove.

Progression

The Progression System handles Experience Points and Levels along with Skill Trees. To use this system, add a Progression Service:

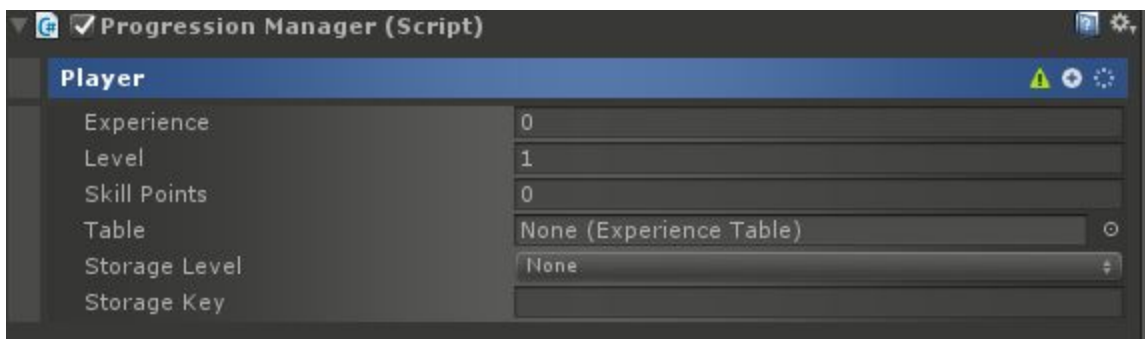




In a project using Asset Bundles, the Tables list can be left empty. Otherwise, any Stat Level Tables used in the game should be added here.

Progression Manager

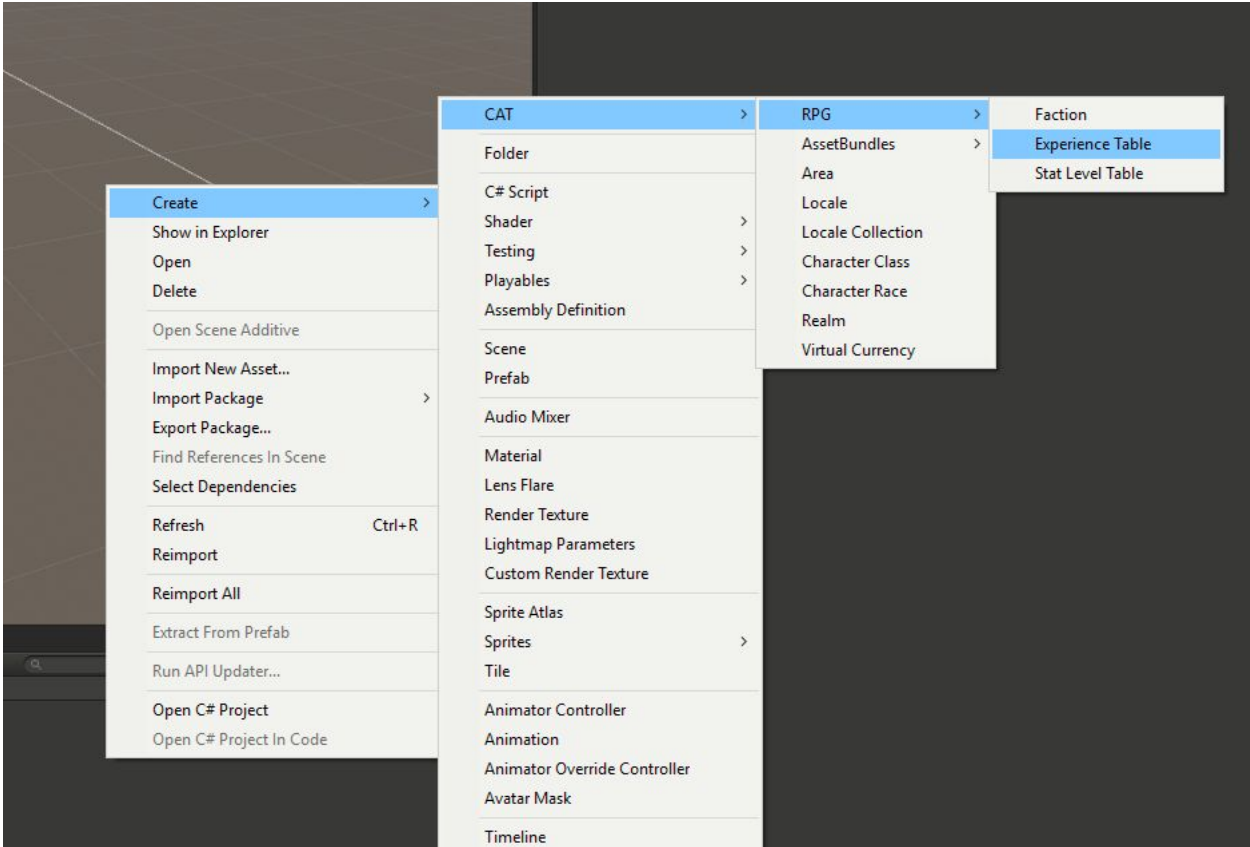
To add Progression to a Character, add the Progression Manager component:



Here, the current Experience, Level, and number of Skill Points can be directly set, but typically for players, the Storage Level and Key will be set (usually to Character) and then via the Persistence System, the Character's Progression will be saved. The Table property can be set to an Experience Table, or for the Player Character, it will automatically be set to an Experience Table attached to the Character.

Experience Tables

Experience Tables dictate the amount of Experience required to reach each level along with what happens when that occurs. To create a new one, use the menu in the Project Window:



Script	ExperienceTable
Levels	
Size	10
▼ Element 0	
Experience Required	0
Title	
String	(Localization Missing)
Actions	None (Action List)
Remove Actions	None (Action List)
▼ Element 1	
Experience Required	100
Title	
String	(Localization Missing)
Actions	LevelUp1 (ActionList)
Remove Actions	None (Action List)
▼ Element 2	
Experience Required	200
Title	
String	(Localization Missing)
Actions	LevelUp2 (ActionList)
Remove Actions	None (Action List)
▼ Element 3	
Experience Required	500
Title	
String	(Localization Missing)
Actions	LevelUp3AndUp (ActionList)
Remove Actions	None (Action List)
▶ Element 4	
▶ Element 5	
▶ Element 6	
▶ Element 7	
▶ Element 8	
▼ Element 9	
Experience Required	9000
Title	
String	(Localization Missing)
Actions	LevelUp3AndUp (ActionList)
Remove Actions	None (Action List)

The Experience Table contains a list of Levels. Each Level has an amount of Experience Required, a Title that will be prepended to the Character's name when they reach that level, and

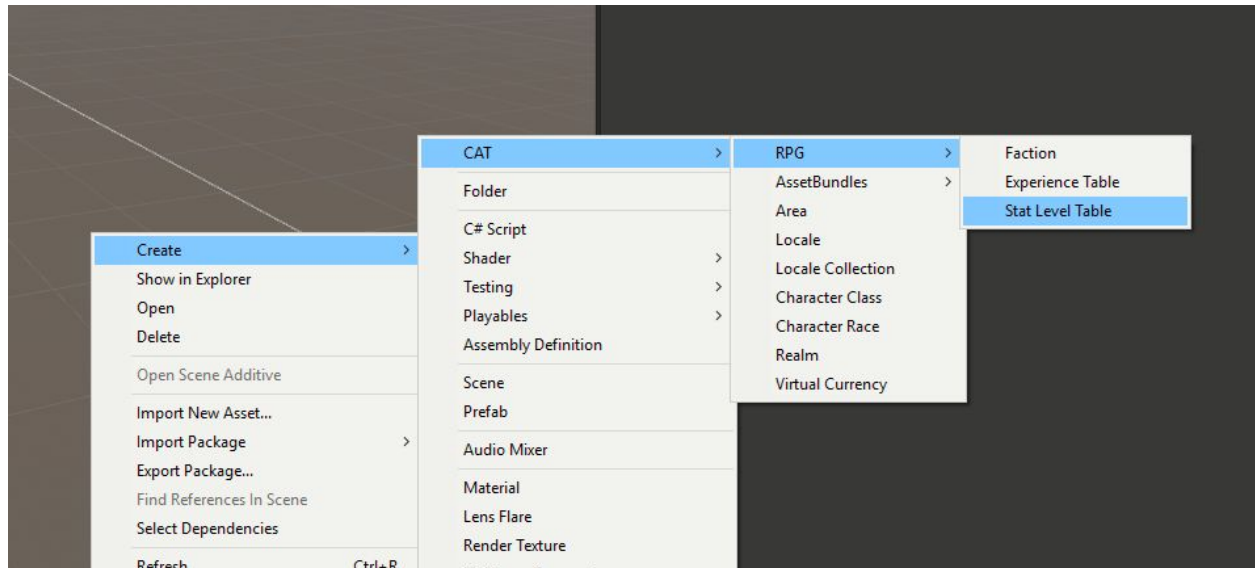
a set of Actions and Remove Actions. These will be executed when gaining or removing the level respectively. Typically, these will grant Abilities, Currency, or Skill Points. If the game does not have a mechanic to remove Levels from the Player, there is no need to define Remove Actions. A typical set of actions might look like this which resets the Player's Health and then grants them the Attack Ability:



Note that the actions contained here will only be run when the Level changes, so they should not be Continuous and should have permanent effects.

Stat Level Tables

Stat Level Tables are a way to have Stats change based on the Character's Level. They can be created by right click in the Project Window:

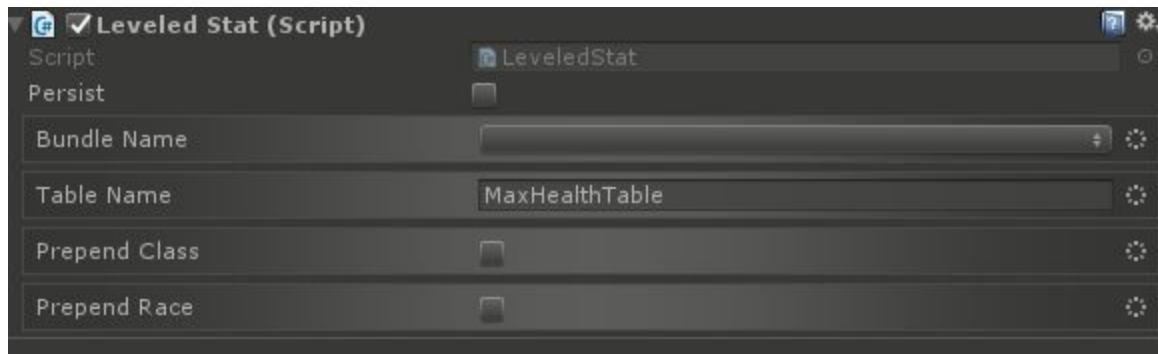


Script		StatLevelTable
▼ Levels		
Size	10	
Element 0	100	
Element 1	120	
Element 2	150	
Element 3	200	
Element 4	280	
Element 5	400	
Element 6	600	
Element 7	900	
Element 8	1500	
Element 9	3000	

They effectively contain a list of Levels with a Stat value for each. These are paired with a Leveled Stat to be effective.

Leveled Stats

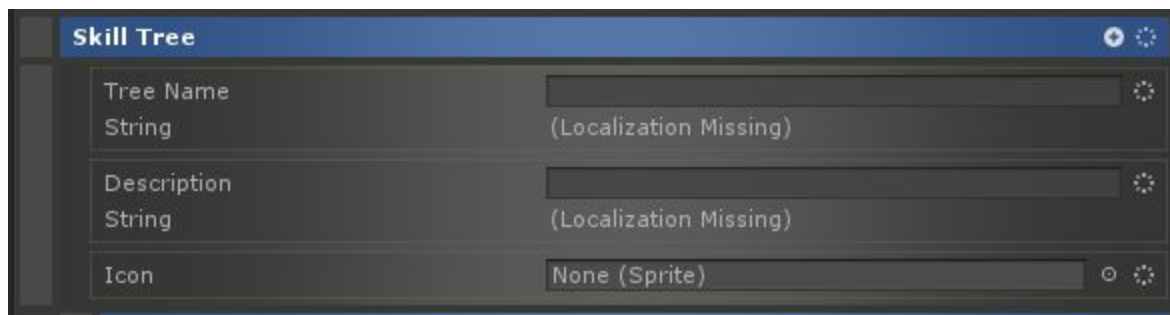
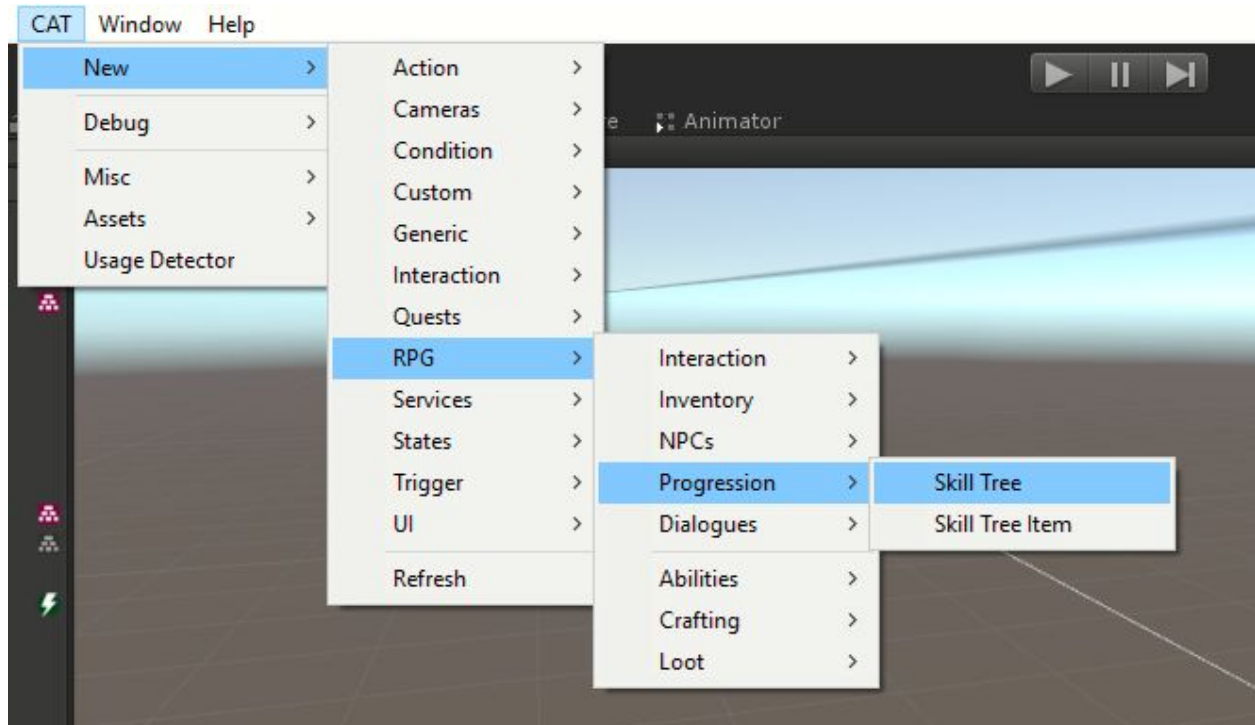
Leveled Stats are a type of Stat that references a Stat Level Table in order to determine their value. The Level of the Character they are attached to is used to determine what entry in the Stat Level Table to Use. These can be added to a Value Holder like any other Value or Stat.



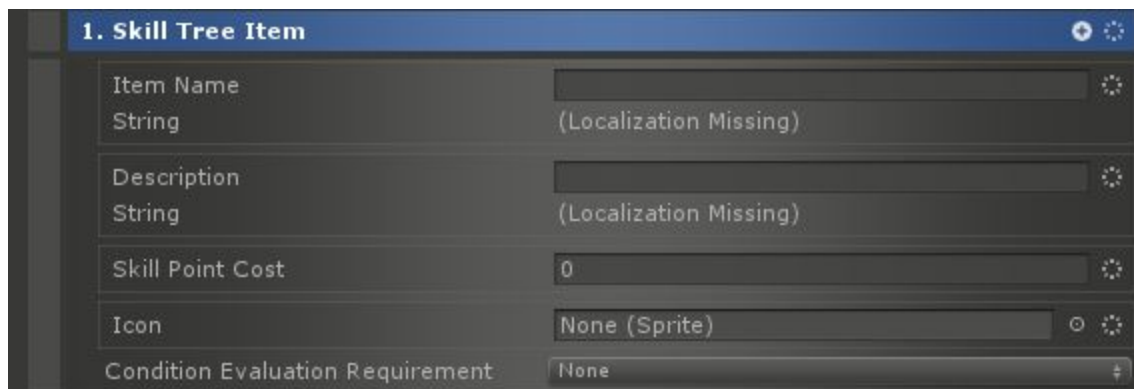
The Bundle Name and Table Name reference the Stat Level Table to be used. If not using Asset Bundles, leave the Bundle Name blank as in the example. Prepend Class and Prepend Race can be used to provide specific Tables for each Class or Race. This is done by prepending the name of the Class and / or Race to the Table Name.

Skill Trees

Skill Trees can be used to give the Player flexibility in customizing their Character through selecting items on a tree. To create a Skill Tree, use the CAT menu.



The parameters are pretty simple, just the Name, Icon, and Description. After this, Tree Items can be childed to it:



Again, the parameters include a Name, Icon, and Description. Additionally, there is a Skill Point Cost which sets how many Skill Points are required to purchase this item.

Additional Skill Tree Items can be childed to the Item, which will cause this Item to be a prerequisite to purchasing them.

Conditions can also be childed to the Item that will also dictate whether the Item can be purchased. The Condition Evaluation Requirement is used to determine how many of these conditions must be true in order for the Item to be purchasable.

Actions should be childed to the Item which like the Level Up Actions will initiate some permanent change on the Character. These Actions will only be run when the tree item is purchased.

Conditions

- Can Purchase Skill Tree Item Condition - Check if the target can purchase a skill tree item.
- Experience Condition - Checks if the target character has an amount of experience points.
- Has Skill Points Condition - Checks if the target character has an amount of skill points.
- Has Skill Tree Item Condition - Check if a character has a skill tree item.
- Level Condition - Checks the target character's level.

Actions

- Grant Experience Action - Grant experience to a target.
- Grant Level Action - Grant levels to a target.
- Grant Skill Points Action - Grant skill points to a target.
- Purchase Skill Tree Item Action - Purchase a skill tree item for a target.
- Reset Progression Action - Reset the progression of a target.
- Revoke Experience Action - Revoke experience from a target.
- Revoke Level Action - Revoke levels from a target.
- Revoke Skill Points Action - Revoke skill points from a target.

Triggers

- Can Purchase Skill Tree Item Trigger - Fire when the target can purchase a skill tree item.
- Experience Trigger - Fire when the target character has an amount of experience points.
- Has Skill Points Trigger - Fire when the target character has an amount of skill points.
- Has Skill Tree Item Trigger - Fire when a character has a skill tree item.
- Level Trigger - Fires depending on the target character's level.
- Level Up Trigger - Fires when the target levels up.

Behaviors

- Grant Experience On Death Behavior - Grant Experience to the killer when target Character dies.

Loot Types

- Experience Loot Type - Give Experience as Loot.
- Level Loot Type - Give Levels as Loot.
- Skill Point Loot Type - Give Skill Points as Loot.

Stats

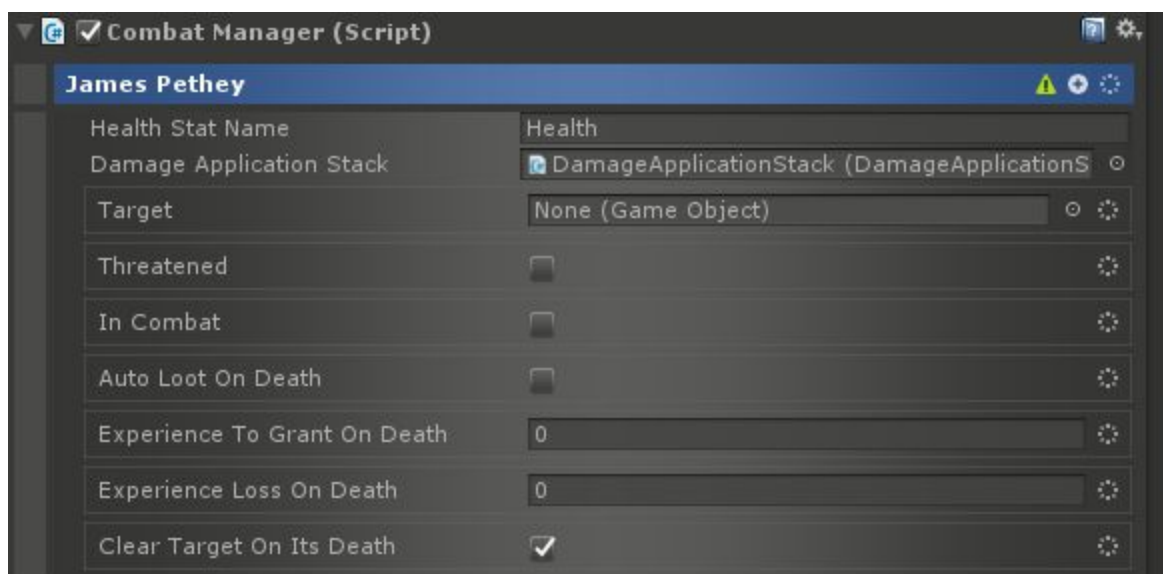
- Experience Stat - Always equal to the Character's Experience
- Leveled Stat - See Stat Level Table
- Level Stat - Always equal to the Character's Level
- Skill Points Stat - Always equal to the number of Skill Points the Character has.

Combat

The Combat System in CAT RPG is very flexible. At the core of it is a user-defined damage application stack which determines how damage is applied in the game. This includes handling accuracy, blocking, dodging, armor, etc. The Combat System heavily relies on the Stats System, which is required for its use.

Combat Manager

In order to make anything into a combatant, it must have the Combat Manager component added to it.



There are a number of options to define. Health Stat Name defaults to “Health” and as it implies is the name of the Stat used for the combatant’s health. This will typically be a Fuel Stat (see the Stat System for more information).

Damage Application Stack needs to point to a valid prefab with a Damage Application Stack component at the base level. This instructs the system on how to apply damage to this combatant.

Target is typically set during runtime, but can be set manually to force the combatant to target a specific other combatant.

Threatened and In Combat are also both typically set during runtime and represent whether the combatant is threatened by a foe or currently in combat respectively.

Auto Loot On Death can be enabled in order to automatically make any Loot attached to the Combatant via the Loot Manager to be generated as soon as the combatant dies.

Experience To Grant On Death can be set when used along with the Experience Service to grant an amount of experience to whoever kills this combatant.

Experience Loss On Death can be used with the Experience Service to reduce this combatant's experience when they die.

Clear Target On Its Death, when checked, will clear the target from any other character targeting this one when it dies.

Damage Application Stack

Damage Application Stack (Script)

Damage Application Stack

? 1. Hit Check Condition

Negate

Accuracy Stat Name

ToHit

Accuracy Operations

Prepend Damage Type To Accuracy

Prepend Attack Range Type To Accuracy

Dodge Stat Name

Dodge

Dodge Operations

Prepend Damage Type To Dodge

Prepend Attack Range Type To Dodge

Dodge Animation Parameter

Miss Animation Parameter

? 2. Block Condition

Negate

Block Stat Name

Block

Prepend Damage Type

Prepend Attack Range Type

Animation Target

Owner

Targets

0

Operations

Animation Parameter

⚡ 3. Reduce Damage Action

Reduction Stat Name

Armor

Prepend Damage Type

Prepend Attack Range Type

Is Percent

Random Reduction Stat Name

Apply Reduction To Fuel Name

Store Reduction Amount In

None

Operations

A Damage Application Stack has a list of Conditions and Actions which are executed or checked from top down. If a condition fails, the damage is avoided. All the Conditions and Actions are passed a set of local values. Most of these come directly from the Damage Action:



These include:

- damage: The amount of damage being done. This can be modified by setting this value in the Damage Application Stack.
- damageType: The type of damage (physical, electrical, etc)
- canMiss: Whether the attack might miss the target
- canDodge: Whether the target can dodge the attack
- canBlock: Whether the target can block the attack
- canReduce: Whether the target can reduce the damage of the attack (usually via armor)

- attackRangeType: Whether the attack was melee or ranged
- outputStat: The name of the stat to apply the damage to (i.e. health)

All of those are parameters to the Apply Damage Action except outputStat. That comes from a setting on the CombatManager of the target. However, if during Damage Application that value is changed, the damage will be applied to the new stat.

The other parameters to Apply Damage are:

- Min Damage Amount: The minimum amount of damage to apply. The damage roll uses this to determine the initial amount. It may be modified later.
- Max Damage Amount: The maximum amount of damage to apply. The damage roll uses this to determine the initial amount. It may be modified later.
- Operations: Mathematical operations to apply to the damage roll.
- Source: The attacker
- Target: The target to apply damage to.

Actions and Else Actions can be childed to Apply Damage. Regular Actions will be run when some amount of damage is applied. Else Actions will be run if no damage is applied (dodged, blocked, etc).

Special Actions And Conditions For The Damage Application Stack

1. Hit Check Condition

Negate ☐

Accuracy

Accuracy Stat Name: ToHit

► Accuracy Operations

Prepend Damage Type To Accuracy ☐

Prepend Attack Range Type To Accuracy ☐

Miss Animation Target: Owner

Targets 0

Miss Animation Parameter

Dodge

Dodge Stat Name: Dodge

► Dodge Operations

Prepend Damage Type To Dodge ☐

Prepend Attack Range Type To Dodge ☐

Dodge Animation Target: Owner

Targets 0

Dodge Animation Parameter

The Hit Check Condition does most of the work in a Damage Application Stack in determining if the attack is on target. It can roll against the attacker's accuracy, the target's dodge stat, or both. If either roll fails, it will cancel the attack. Operations can be applied to either stat in order to modify the target value. The rolls are always a number between 0.0 and 1.0. To ignore one of the rolls, leave the respective Stat field blank.

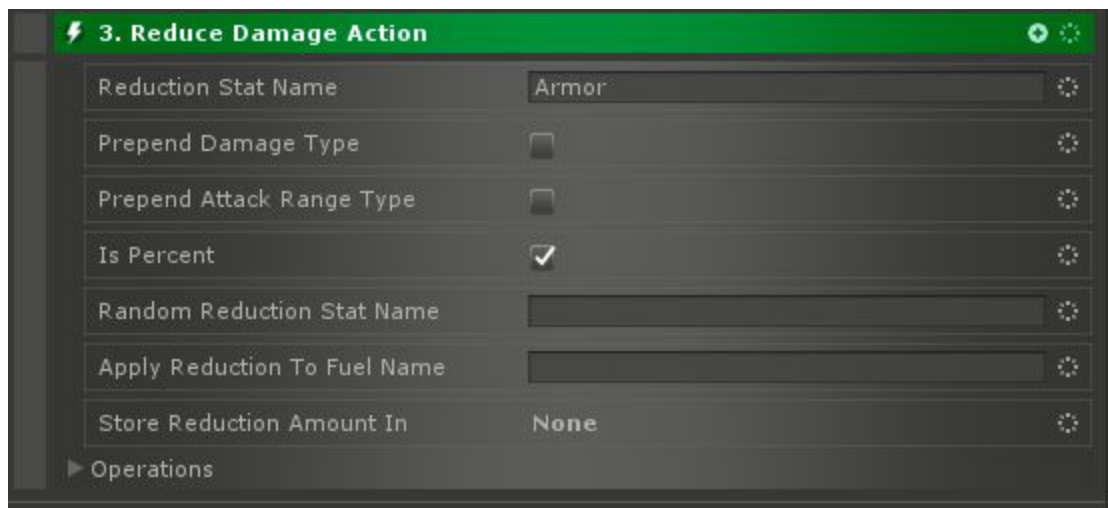
Using the checkboxes for each Stat, the name damage type or attack type (melee, ranged) can be prepended to the name of the Stat in order to use a specific Stat for that type of damage or attack.

Finally, animation parameters can be set depending on the outcome. If there's a miss, the Miss Animation Parameter is applied to the Miss Animation Target. Owner there is always the attacker and Targeted is always the defender.

If the defender dodges, then the Dodge Animation Parameter is applied to the Dodge Animation Target. Owner there is always the attacker and Targeted is always the defender.



Similarly to the Hit Check Condition, the Block Condition checks whether the defender blocks. The parameters work in exactly the same way.



The Reduce Damage Action can be used to reduce the amount of damage applied based on a Stat. For example, armor. The Stat name must be specified, and can have the damage type and/or attack type prepended to the name. If Is Percent is true, then the Stat is assumed to be a percent and will be multiplied with the damage to get the resulting modified damage.

If Random Reduction Stat Name is specified (and Prepend Damage Type and Attack Type apply to this), then after applying the reduction from the main stat, the damage will be further reduced by a random amount between 0 and the value of this Stat. Both Stats are optional, though for the Action to have any effect, one or both must be specified.

Apply Reduction To Fuel Name will cause the amount of damage reduced to be deducted from the given Fuel Stat name. Alternately, Store Reduction Amount In can be used to save the reduction amount to a value for later use.

All other Conditions and Actions can be used in the Damage Application Stack, but generally it will contain the ones outlined here.

Other Combat CATs

Conditions

- Combat Target Condition - This condition is true if the target(s) have a combat target set. Typically, this is used in an Effect Group to collect the target.
- Is Alive Condition - Checks if the target(s) are alive.

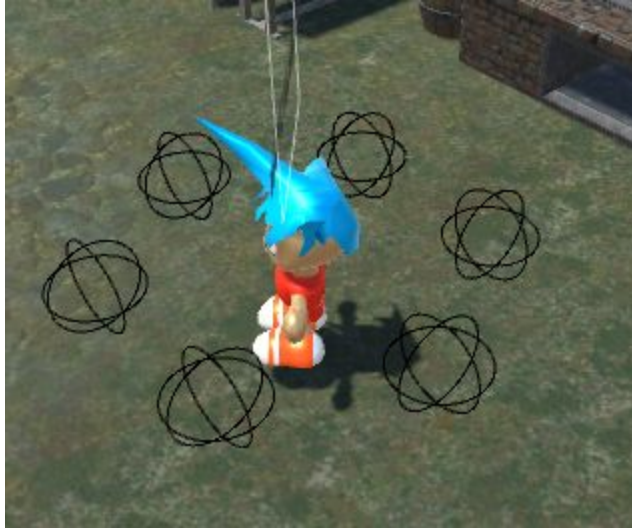
Actions

- Heal Action - Heals the target for a random value between Min Heal Amount and Max Heal Amount after applying operations.
- Resurrect Action - Reset the alive state on a combatant. This is important to do on the player if they are being brought back from the dead.
- Set Combat Target Action - Sets the combat target of a combatant.

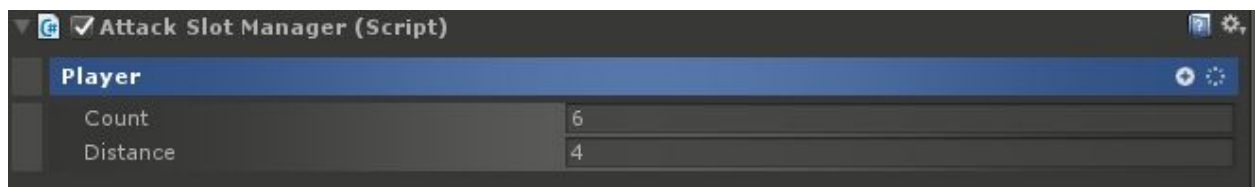
Triggers

- On Block Trigger - Fires when the target blocks an attack.
- On Death Trigger - Fires when the target dies.
- On Dodge Trigger - Fires when the target dodges an attack.
- On Heal Trigger - Fires when the target is healed.
- On Hit Trigger - Fires when the target is hit with an attack.
- On Miss Trigger - Fires when the target misses with an attack.

Attack Slots



When multiple enemies are attacking a target, they will appear more intelligent if they spread out evenly around the target. The Attack Slot System provides this feature. Each black sphere in the image above represents an attack slot. When an NPC tries to attack this character, they will pick an open slot and position themselves there in order to attack. To enable this behavior on a character, simply add the Attack Slot Manager component:



The Count parameter specifies how many slots to create and the Distance is how far away from the character they are. If using the NPC behaviors that come with CAT RPG, this is the only thing that needs to be done. There are, however, several CATs for more direct control of the system:

Conditions

- Has Available Attack Slot Condition - True if the target(s) have an available attack slot.

Actions

- Move To Attack Slot Action - Moves the targeted object(s) to their attack slot on the attack target.
- Release Attack Slot Action - Releases a previously reserved attack slot.
- Reserve Attack Slot Action - Reserve an attack slot on a target for an attacker.
- Set Navigation Destination To Attack Slot Action - Sets the destination of the targeted NavMeshAgent(s) to an attack slot on the attack target.

Triggers

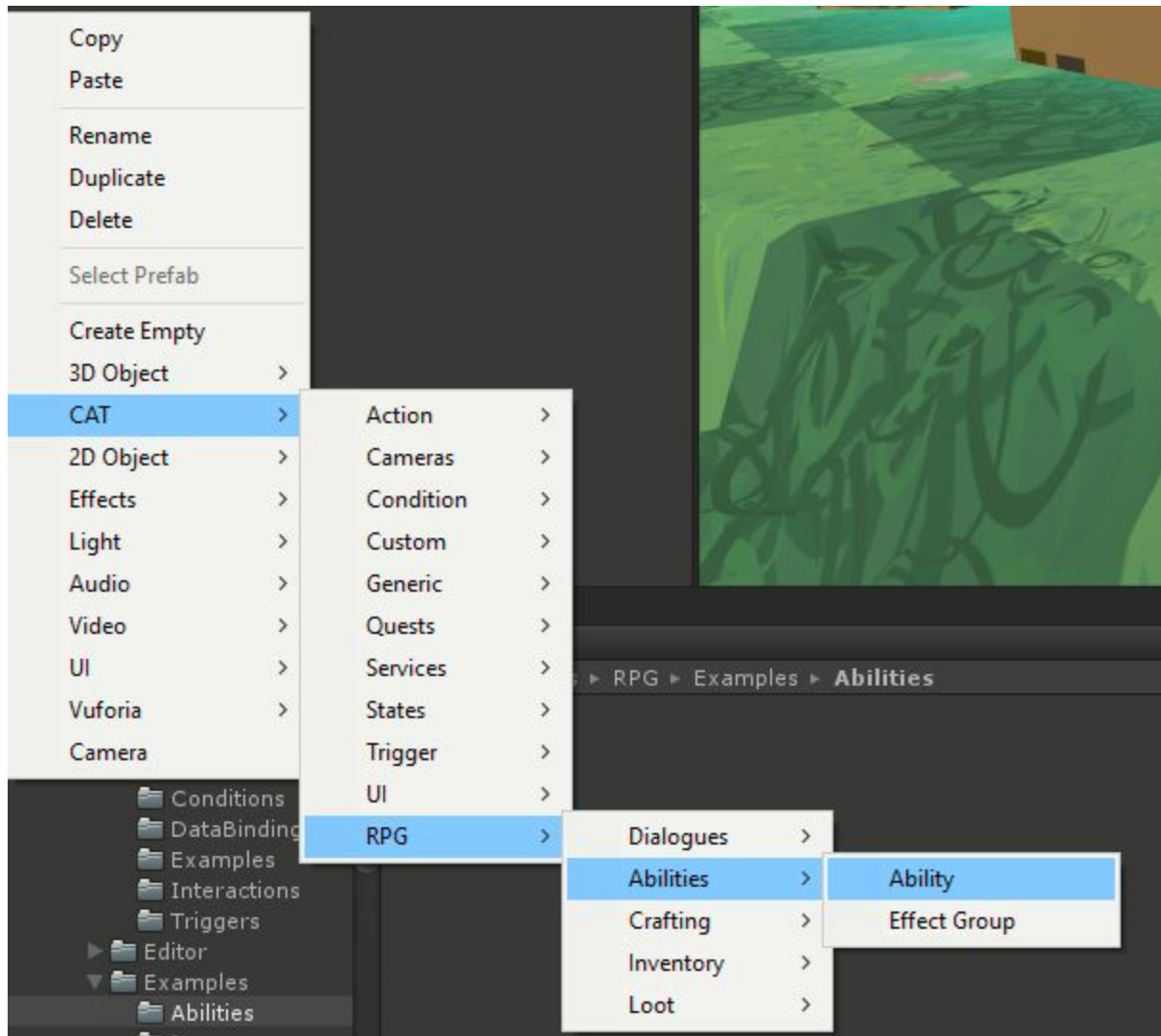
- Has Available Attack Slot Trigger - Fires when the target(s) have an available attack slot.

Abilities

To use Abilities, a character (player or NPC) must have an Ability Manager component attached to them. Available abilities are childed to the Game Object with the manager. However, typically, this should only be done for NPCs. Player Abilities can be stored as long as either the Game Object with the Ability Manager is saved (such as by a State Machine on the same Game Object Being saved or by setting the Storage Level and Storage Key). This allows the Progression system or other mechanisms to add new Abilities to the player which will be persisted.

Creating An Ability

To Create an Ability, you can use the CAT menu when right clicking in the hierarchy:



It is also accessible by the CAT menu at the top of the window. After creating the Ability Game Object, 3 new Game Objects are also created as children:



These each represent a stage the Ability goes through when it is cast. Actions can be childed to each. Here's what an Ability looks like in the Inspector:

Attack

Ability Name	Attack
String	(Localization Missing)
Description	Attack a target
String	(Localization Missing)
Ability Type	Normal
Min Range	1
Max Range	6
Triggers Global Cooldown	<input type="checkbox"/>
Icon	41
Fuel Costs	
Size	1
▼ Element 0	
Fuel Name	
Cost	0
► Operations	
1. Warmup Stage	
Duration	0
2. Release Stage	
Run Type	Serial
3. Cooldown Stage	
Duration	0

There's fields for Name and Description which are both Localized. Min and Max Range specify how close or far away the Caster needs to be from the Targets.

Ability Type can be either Normal or Passive. Passive Abilities are cast immediately on being granted and their effects should persist using Permanent Effect Groups.

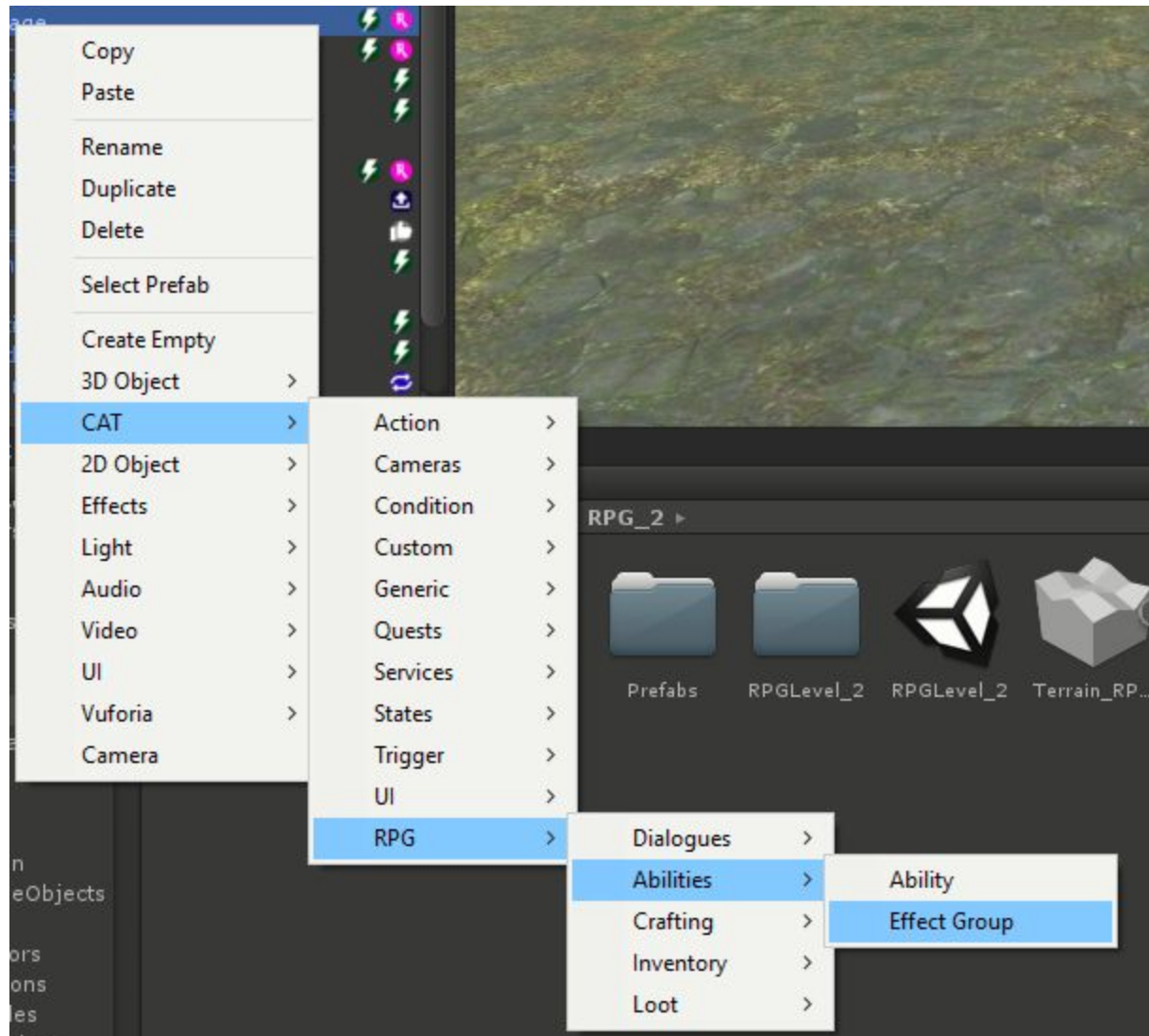
There is a per Caster Global Cooldown period which means that once any Ability using it is cast, no other Abilities may be cast until it has expired. If the Triggers Global Cooldown flag is set, then this Ability will use that mechanism.

Icon can be set to an icon for this Ability.

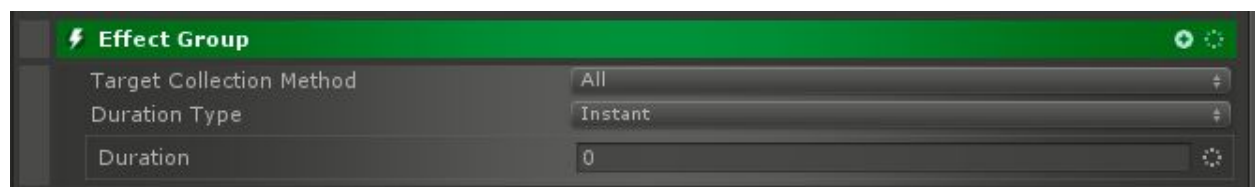
Fuel Costs define how much it costs to cast the Ability.

Effect Groups

Typically, under the ReleaseStage, one or more EffectGroups are placed. You can create these using the menu:



Effect Groups have several properties:



Effect Groups gather their targets from attached conditions. Any condition that sets targets can be used and childed to the Effect Group. When Target Collection Method is set to All, each Condition will be evaluated and all the gathered targets will be used. If it is set to Progressive, then the Conditions will be evaluated one at a time and the targets collected from one will be passed as targets to the next. The final Condition's returned targets will be used as the Ability's targets.

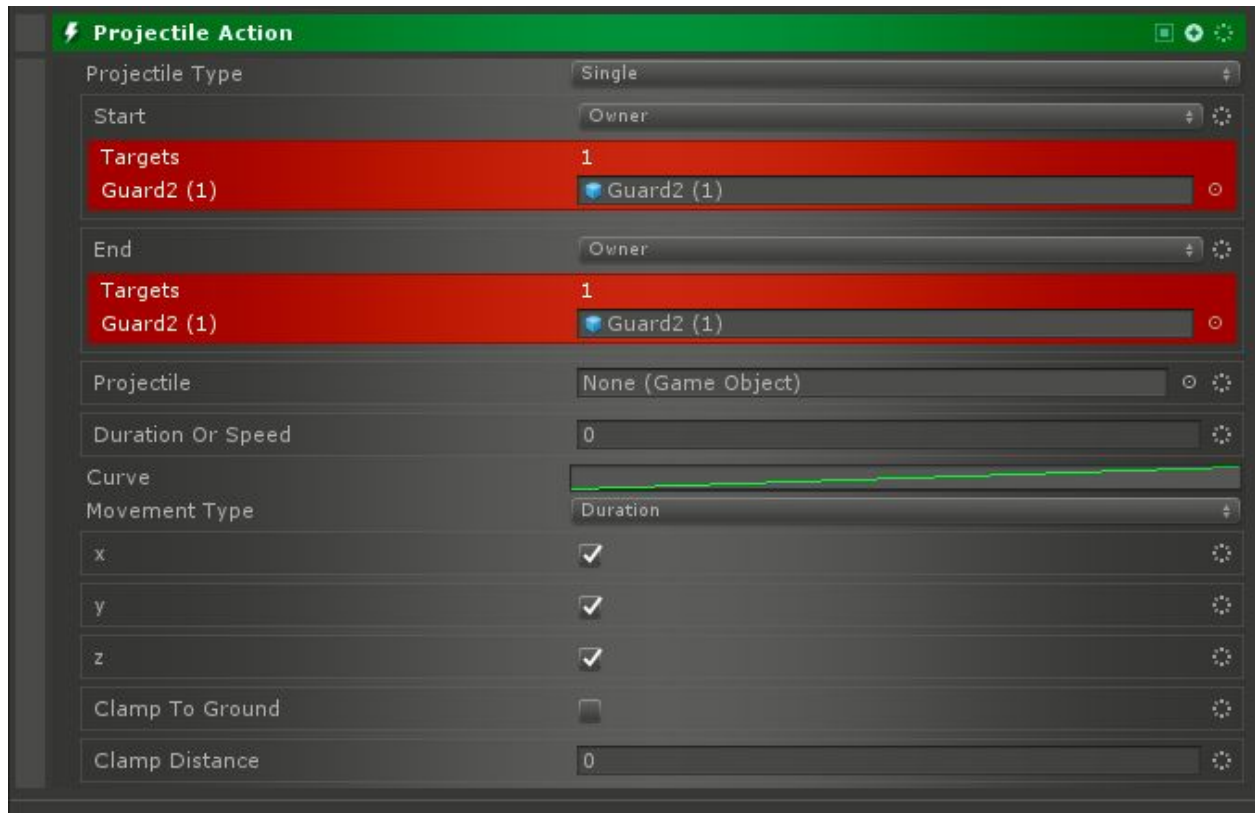
Duration Type is whether this Effect Group is Instant or has a specific duration. If it is set to Duration, then the Duration parameter specifies how long the Effect Group will be active on the target(s) for. If set to Permanent (typically only Passive Abilities will do this), then the Effect Group will stay on the target indefinitely.

Special Conditions For Effect Groups

There are two special Conditions that can be used for selecting targets in Effect Groups. The first is Set Ability Targets Condition. This can be used to directly set the targets of the Effect Group with a CATarget. Similarly, there's Set Default Ability Targets Condition. Use this at the end of a Progressive Target Collecting Effect Group in order to have a default target based on a CATarget if there are not already other targets picked up.

Special Actions For Effect Groups

The Projectile Action is meant specifically to be used in Effect Groups. It will run actions childed to it after showing a projectile going from the Start to End:



Projectile Type selects how the projectile behaves. Single means that a single projectile will go from the Start to the End. If there are multiple Start or End positions, then the average of each will be used. Multiple means that for each Start position, a projectile will be sent to each End position. Finally, MIRV means that a projectile will be sent from the Start to the average of the End positions and then will turn into multiple projectiles which will each move to an End position.

The Projectile field should be set to a Prefab for the projectile itself.

The rest of the parameters work exactly like they do in the Move Action.

Example Effect Group

Here's an example of an Effect Group:

Effect Group

Target Collection Method

All

Duration Type

Instant

Duration

0

1. Set Trigger Animation Param Action

Target

Owner

Path

Body

Targets

1

Body

Body

Parameter Name

attack

2. Damage Action

Source

Owner

Targets

1

Guard2 (1)

Guard2 (1)

Target

Targeted

Targets

0

Attack Range Type

Melee

Damage

Damage Type

physical

Min Damage Amount

60

Max Damage Amount

80

Operations

Size

0

Avoidance

Can Miss

☒

Can Block

☒

Can Dodge

☒

Can Reduce

☒

3. Ray Cast Condition

Negate

☐

Source

Owner

Targets

1

Guard2 (1)

Guard2 (1)

Collide Target

Targeted

Targets

0

Destination

Targeted

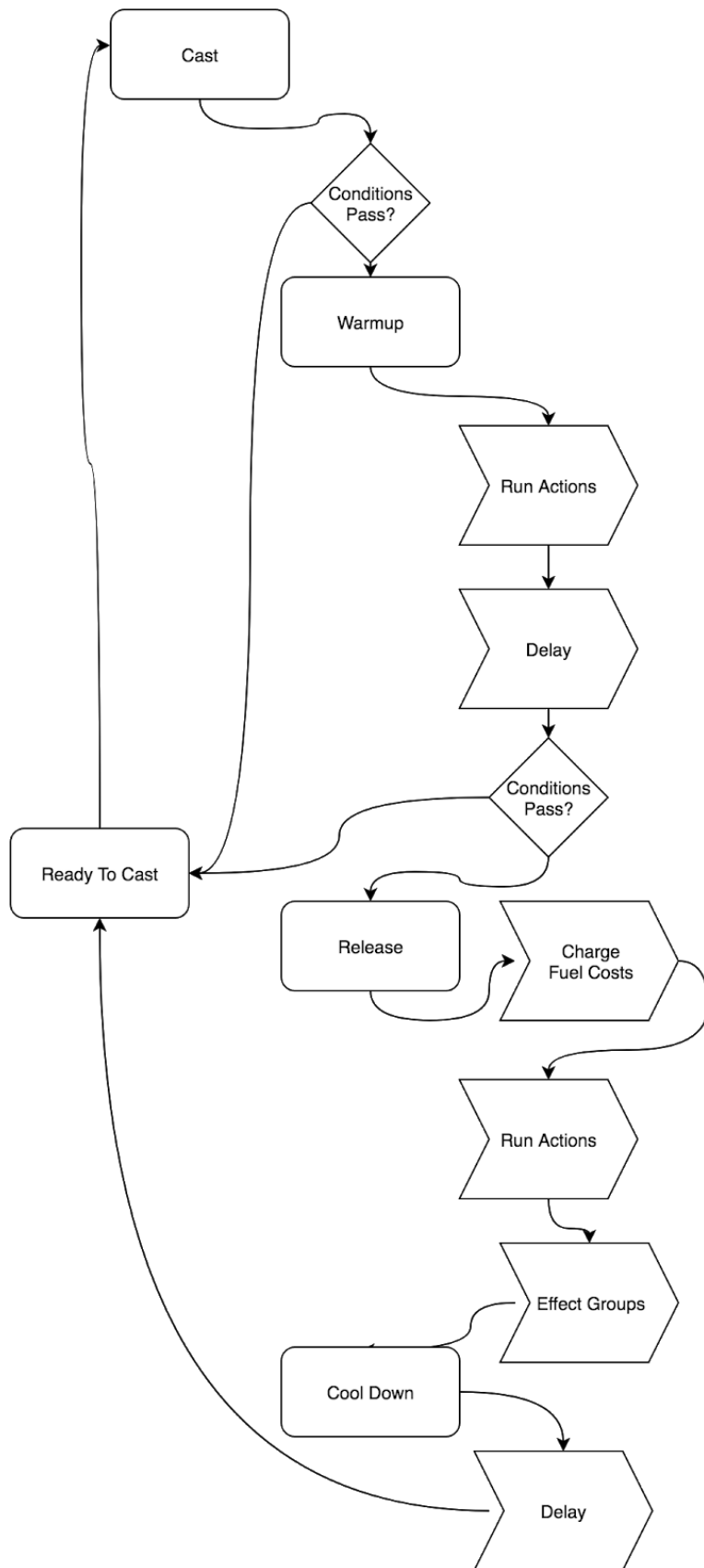
Targets

0

This is an Instant Effect Group that collects targets where there is a line of sight (via Ray Cast Condition) and it plays an attack animation and then deals damage to those targets. See the Combat System for more information on the Damage Action.

Ability Casting Flow

Here's a diagram of the flow that happens when casting an ability:



Actions

There are several Actions for the Ability System.

- Cast Ability Action - Immediately casts an Ability.
- Grant Ability Action - Grant an Ability to a caster. Can optionally be removed when the Action stops.
- Remove Ability Action - Remove an Ability from a caster. Optionally return it when the Action finishes.
- Run Ability Action - Immediately cast an Ability on a target. The Ability can either be childed to this Action, or the Ability Name field can be used to specify it.

DataBinding

The Ability System has a binding for UI Ability Button DataBind. This can be used on a button to cause the button to cast a specific ability. It will also disable the button if the Ability can't be cast at a given moment.

Triggers

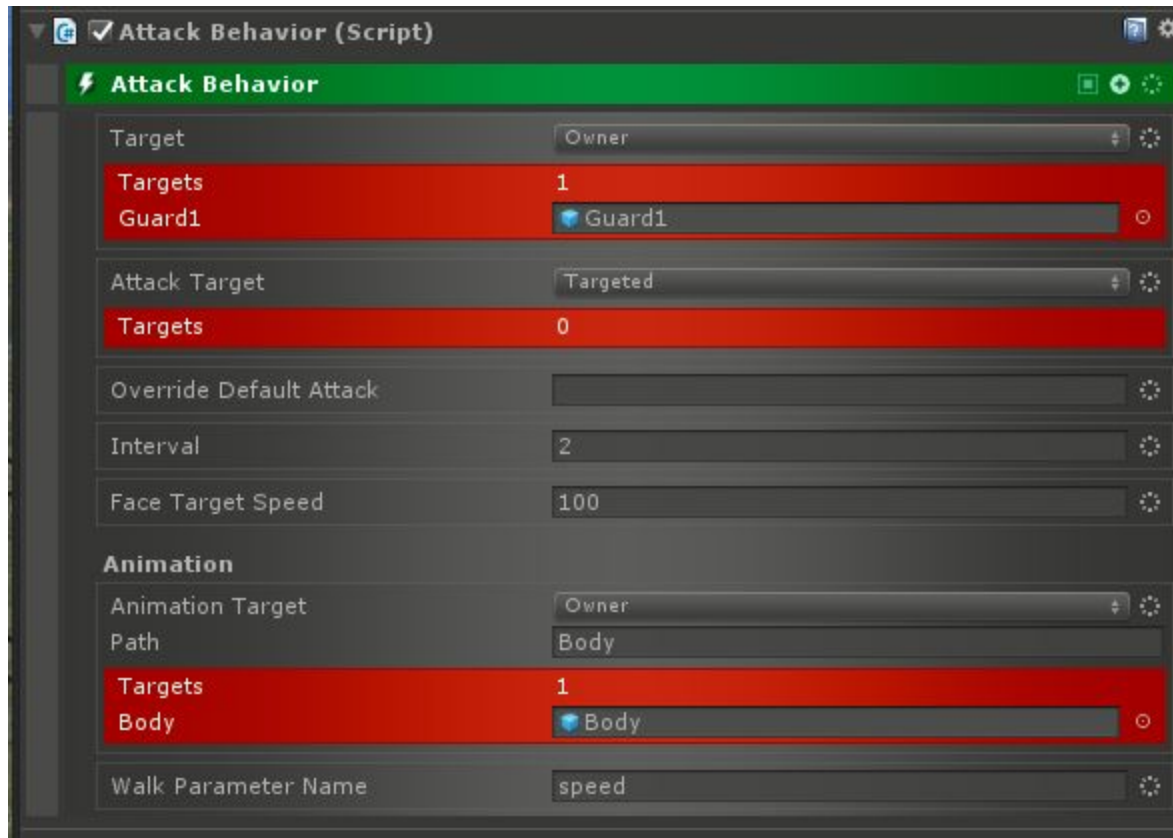
- Can Cast Ability Trigger - Fires when the specified Ability can be cast.
- Failed Cast Trigger - Listens on a caster for when an Ability fails.

NPCs

CAT RPG comes with several high level NPC Behaviors. These are intended to simplify building NPC AI and perform such actions as Attacking, Fleeing, Following, Patrolling, and Wandering. It also includes an Interaction to turn an NPC into a merchant.

Attack Behavior

For this behavior, an attacker will pursue a target and when in range of their default attack ability (or a specified other ability), they will attack until the target is dead.



A typical Attack Behavior will look like this. The Target specifies the attacker and should generally be Owner. The Attack Target specifies the Character to attack and will generally be Targeted. Override Default Attack can take the name of an Ability to use instead of the default Attack Ability. Interval should be set to the delay in seconds between attacks. Face Target Speed determines how fast the attacker will turn to face the target when attacking.

There are also animation settings such as the Animation Target which should point at the object which is to be animated. Note that this Behavior does not play any attack animations, but instead relies on the Ability to do that. The Walk parameter Name is an animation parameter to be used for walking.

Flee Behavior

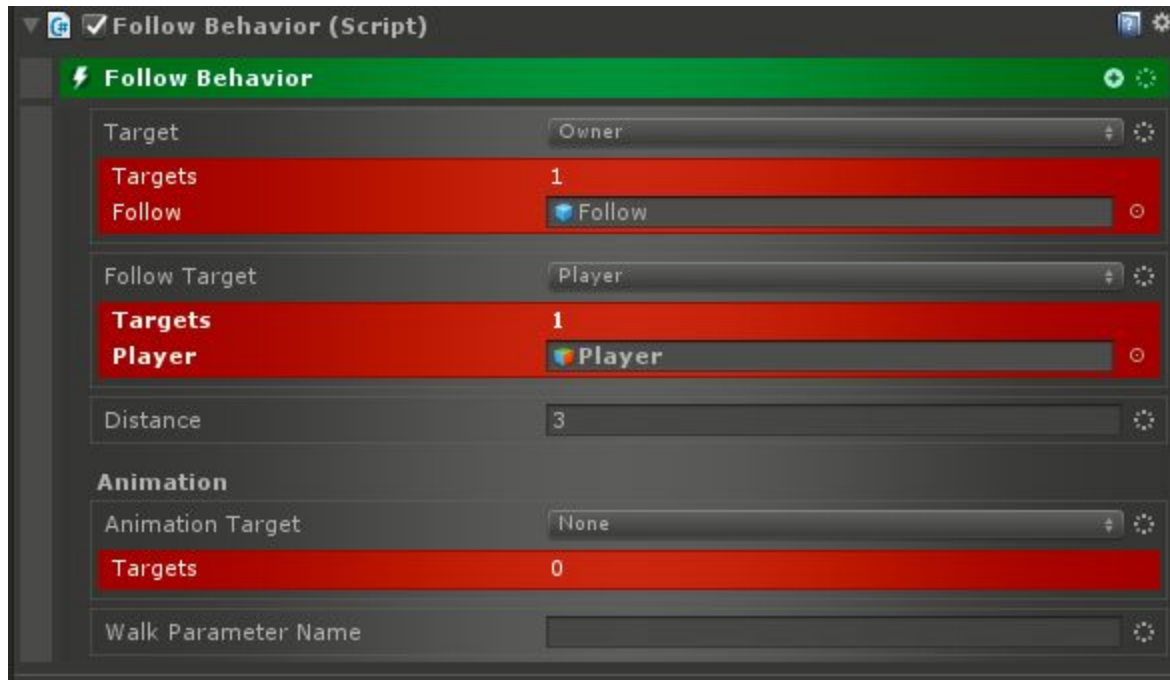
A Character running this Behavior will run away from a target until they get a specified distance away.



The parameters are fairly simple. The Target should usually point to Owner and is the Character that will be fleeing. Flee Target should be sent to the Character(s) that should run away from. Distance is the distance at which the Character is safe. Animation Target can be used to point to an object that will be animated. The Walk Parameter Name should be an animation parameter on the Animation Target to set when walking.

Follow Behavior

The Follow Behavior is the exact opposite of the Flee Behavior whereas the target will follow another Character until they get within a specified distance.



The Target should usually point to Owner and is the Character that will be following. Follow Target should be sent to the Character(s) that should be following. Distance is the distance at which the Character will approach to. Animation Target can be used to point to an object that will be animated. The Walk Parameter Name should be an animation parameter on the Animation Target to set when walking.

Over Head Indicator Behavior

This one is different from the others as it is not AI related. It will spawn the Character's default Interaction's prefab over their head. This should be used to show when NPCs have dialogue, quests, or are merchants for example.



The Target should be set to the Character that will display the indicators (usually Owner). Display Trigger determines when the indicators show. It can be on Proximity (when the Proximity Target gets within the Proximity Range), Mouse Over (when the mouse is over it), or Constant (it will always display).

Patrol Behavior

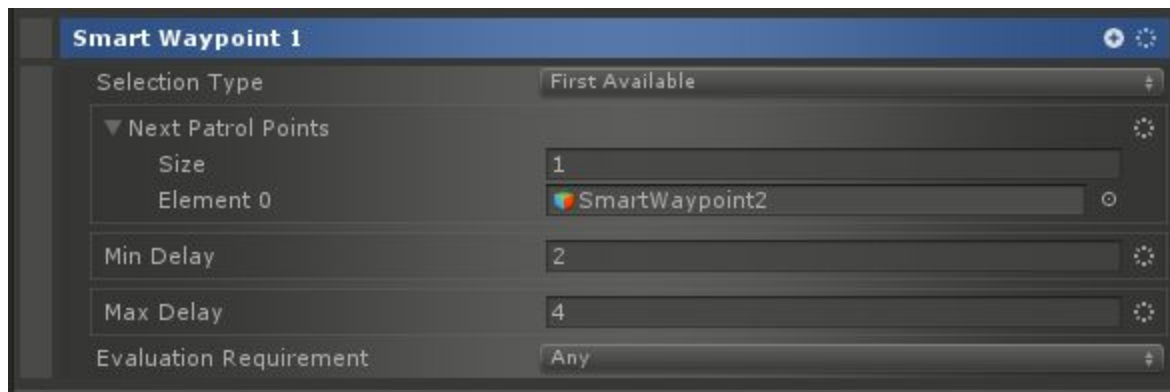
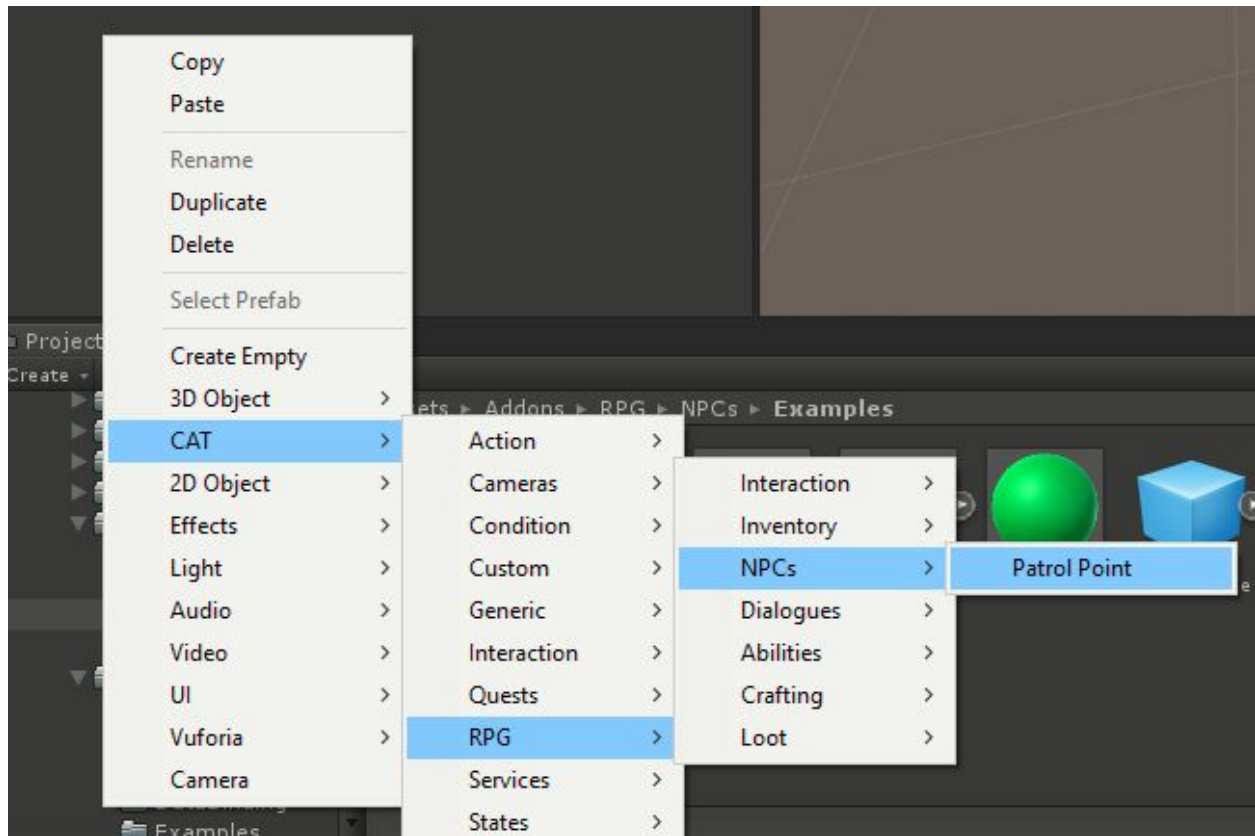
Using two or more Patrol Points, the Character will move between them in some manner. Patrol Points can be regular Game Object or they can be smart Patrol Points. In the latter case, there are options to make the Patrol Path be non-linear, or have different delays at each point.



The Target should be set to the Character that should Patrol. The First Patrol Point should point at the first Patrol Point if it is smart, or else (as pictured here), multiple Game Objects that will act as Patrol Points. Animation Target can be used to point to an object that will be animated. The Walk Parameter Name should be an animation parameter on the Animation Target to set when walking.

Patrol Point

Smart Patrol Points can be created to have better control over a Patrol Path:



They have several parameters. The Selection Type determines the method of selecting the next Patrol Point. Next Patrol Points holds a list of possible next destinations. Min and Max Delay determine in seconds how long the Character will stay at the Patrol Point.

Conditions can be childed to the Patrol Point. Using Evaluation Requirement to determine how many must pass, the Patrol Point will be conditionally available for patrolling.

Actions and Stop Actions can be childed to the Patrol Point as well. They are run when a Character arrives and leaves the Patrol Point respectively.

Wander Behavior

The Wander Behavior will cause a Character to walk around randomly. This can be done completely at random or to random Patrol Points.

The image shows a configuration panel for the 'Wander Behavior' in a game engine. The panel has a green header with a lightning bolt icon and the text 'Wander Behavior'. It contains several sections with various settings:

- Target:** A dropdown menu set to 'Owner'.
- Targets:** A red bar with the number '1'.
- Wanderer:** A dropdown menu set to 'Wanderer'.
- Patrol Points:** A section with a dropdown arrow and a 'Size' input field set to '0'.
- Min Distance Per Move:** An input field set to '2'.
- Max Distance Per Move:** An input field set to '10'.
- Use Home Position:** A checkbox that is unchecked.
- Home Position:** A dropdown menu set to 'Owner'.
- Targets:** A red bar with the number '1'.
- Wanderer:** A dropdown menu set to 'Wanderer'.
- Max Distance From Home:** An input field set to '0'.
- Min Delay:** An input field set to '2'.
- Max Delay:** An input field set to '5'.
- Is2D:** A checkbox that is unchecked.
- Animation:** A section with an 'Animation Target' dropdown menu set to 'None'.
- Targets:** A red bar with the number '0'.
- Walk Parameter Name:** An empty input field.

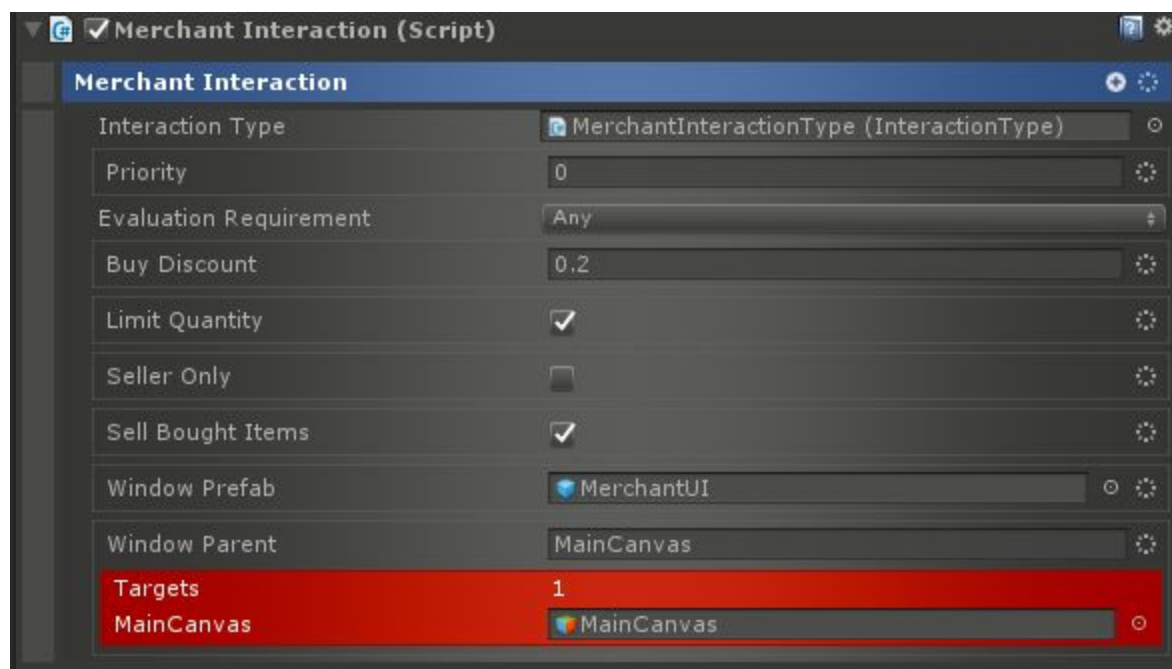
The Target should be set to the Character that should wander. Patrol Points can be set to regular Game Object or smart Patrol Point objects. If none are specified, the Target will wander to random points. Without Patrol Points, Min and Max Distance Per Move determine how far the Target will move each time.

A Home Position can be set along with a Max Distance From Home to keep the Target wandering close to an area. Min and Max Delay sets how long to linger at each stop. Is 2D should be set for 2D games. Animation Target can be used to point to an object that will be

animated. The Walk Parameter Name should be an animation parameter on the Animation Target to set when walking.

Merchant Interaction

This Interaction can be used to turn a Character into a Merchant or shop keeper. When the player interacts with the Character, a shop window will appear, and it will be possible to buy items the Merchant has in their inventory. The Merchant can also be set up to allow the player to sell items.



If Seller Only is unchecked, Buy Discount can be used to make the Merchant purchase items from the player at a discount. The Limit Quantity setting allows the Merchant to have a limited stock (as specified by the quantity of items in their inventory). When an item is bought, it will be removed from the inventory. Seller Only can be used to make the Merchant only sell items and not buy them. If Sell Bought Items is checked, the Merchant will sell any items they purchase from the player.

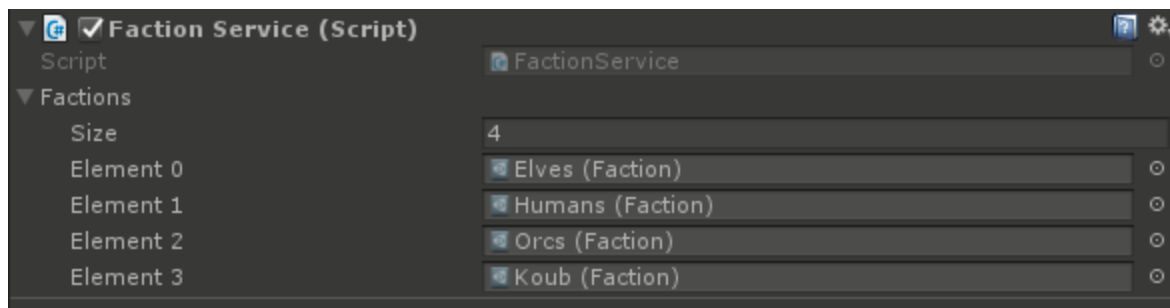
Window Prefab should be set to the window UI to use for the Merchant, and Window Parent should specify the parent object of that window when creating it.

Actions

- Buy Item From Merchant Action - Target will purchase item from merchant.
- Sell Item To Merchant Action - Target will sell item to merchant.

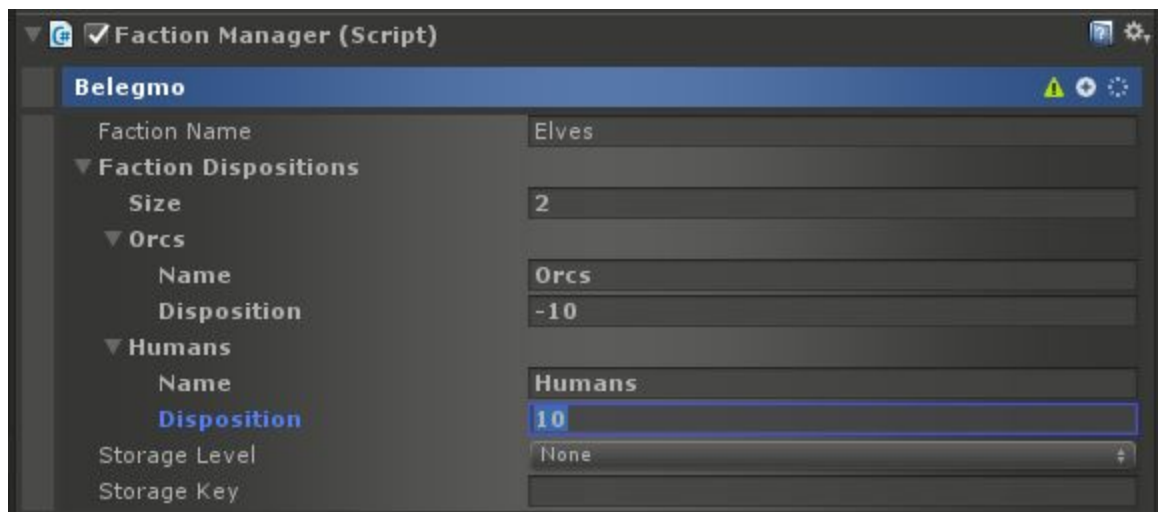
Factions

The Faction System can be used to determine if two Characters are friends, enemies, or indifferent. To use the Faction System, add a Faction Service to a scene with a Conductor.



At present, the Faction Service requires linking all available Factions in the game.

Once the Faction Service is added, a Faction Manager component can be placed on Characters or the Player:



The first parameter is for the name of the Faction to apply to this Character. This should match the names in the Faction Service.

Faction Dispositions can be manually added to have this character's initial disposition to other factions be set ahead of time. The disposition is an integer usually between -10 (hated) and 10 (loved).

Storage Level and Storage Key can be used to persist this Character's Faction information.

Conditions

- Faction Disposition Condition - Determines the disposition of the target to a specified faction and compares it to the input.
- Is Enemy Condition - Determine if a character is an enemy of the target based on the threshold.
- Is Friend Condition - Determine if a character is an friend of the target based on the threshold.

Actions

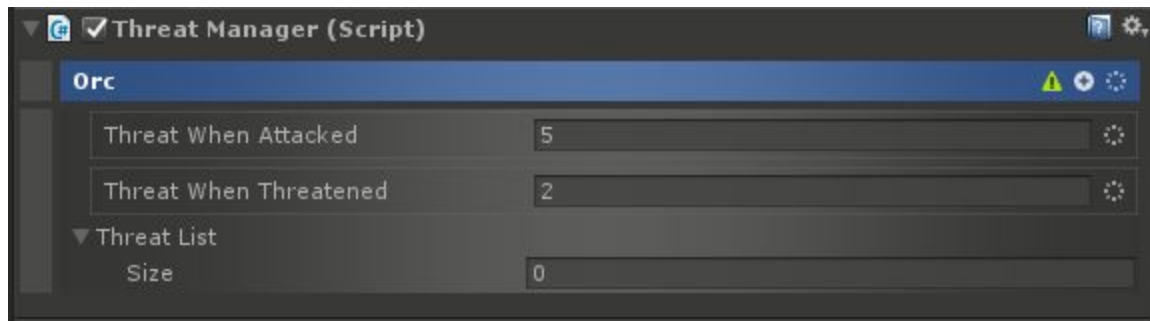
- Modify Faction Action - Modifies the disposition of a target towards a faction.
- Set Faction Action - Sets the disposition of a target towards a faction.

Triggers

- Faction Disposition Trigger - Fires when disposition of the target to a specified faction and compares it to the input.
- Is Enemy Trigger - Fires when a character is an enemy of the target based on the threshold.
- Is Friend Trigger - Fires when a character is an friend of the target based on the threshold.

Threat List



The Threat List System allows NPCs to keep track of which Characters are threatening them in order to determine how to respond and who to target. In order to use the system, add a Threat Manager to an NPC.





The Threat When Attacked is how much Threat to add to a target when they are attacking the NPC, and Threat When Threatened is how much Threat to add to a target when they just Threaten the NPC.


In order to act on the Threat, the Threat Sensing Behavior should be used. This Behavior will cause the NPC to change states to attack, flee, threaten, etc depending on what is threatening them. It emulates various senses on the NPC such as hearing and sight.


⚡ Threat Sensing Behavior


Target Owner  


Targets 1

Orc  Orc 


Base Threat Mod 10 


Check Frequency 0.5 


Layer Mask Everything  

Debug 


Levels


Threatened At Level 2 


Attack At Level 5 


Flee At Level 20 


Vision

Enable Vision ☒ 



Vision Range 30 


Field Of View 120 


Vision Threat Mod 1 


Visual Threat Fade Time 5 

Hearing



Enable Hearing  


Hearing Range 0 


Hearing Threat Mod 0 


Hearing Threat Fade Time 0 

Proximity

Enable Proximity  


Proximity Range 3 

Proximity Threat Mod 1 


Proximity Threat Fade Time 5 

States

Default State Name Wander 

Threaten State Name 

Attack State Name Attacking 

Flee State Name 

The Target should be set to the NPC to enable Threat Sensing on. Base Threat Mod is a modifier (multiplier) to all Threat that this Behavior processes. Check Frequency determines how often (in seconds) that the senses are checked. Layer Mask determines what Physics Layers are used to detect Threats. Debug will enable some additional debugging to the log.

Threatened At Level determines the minimum Threat to be Threatened. Attack At Level determines the minimum Threat for the NPC to attack whatever is threatening it. Flee At Level defines the minimum Threat at which the NPC will flee the target.

Vision can be enabled with the Enable Vision option. Vision Range specifies how far the NPC can see. Field of View defines the angle in degrees that the NPC can see. Vision Threat Mod is a multiplier applied to Threat that comes in through Vision. Visual Threat Fade time defines how long after no longer seeing a Threat that they are removed from the Threat List.

Hearing can be enabled with the Enable Hearing option. Any sound made via the Play Sound Action will be heard. Hearing Range defines how far the NPC can hear. Hearing Threat Mod is a multiplier on any Threat that comes in through Hearing. Hearing Threat Fade Time determines how long after no longer hearing a Threat that they are removed from the Threat List.

Enable Proximity will enable Proximity sensing. Proximity Range defines how close Characters must be to be sensed in this way. Proximity Threat Mod is a multiplier to any Threat that comes in through Proximity. Proximity Threat Fade time sets how long after leaving proximity that a Threat is removed from the list.

Default State Name is the name of the State that will be used when there are no active Threats. Threaten State Name is the name of the State that will be used when threatening a target. Attack State Name is the name of the State that will be used to attack a target. Flee State Name is the name of a State that will be used when fleeing a target. If left blank, these States will be ignored.

Conditions

- Is Threatened Condition - Checks how threatened a target is by a source.

Actions

- Add Threat Action - Adds threat from a source to a target.
- Get Biggest Threat Action - Sets a value to the biggest threat of a target.

- Get Threat List Action - Gets the entire threat list and stores it in a value.
- Remove Threat Action - Removes threat from a source from a target.
- Set Threat Action - Directly set the amount of threat from a source.
- Target Biggest Threat Action - Sets the target to the biggest threat of a target.

Triggers

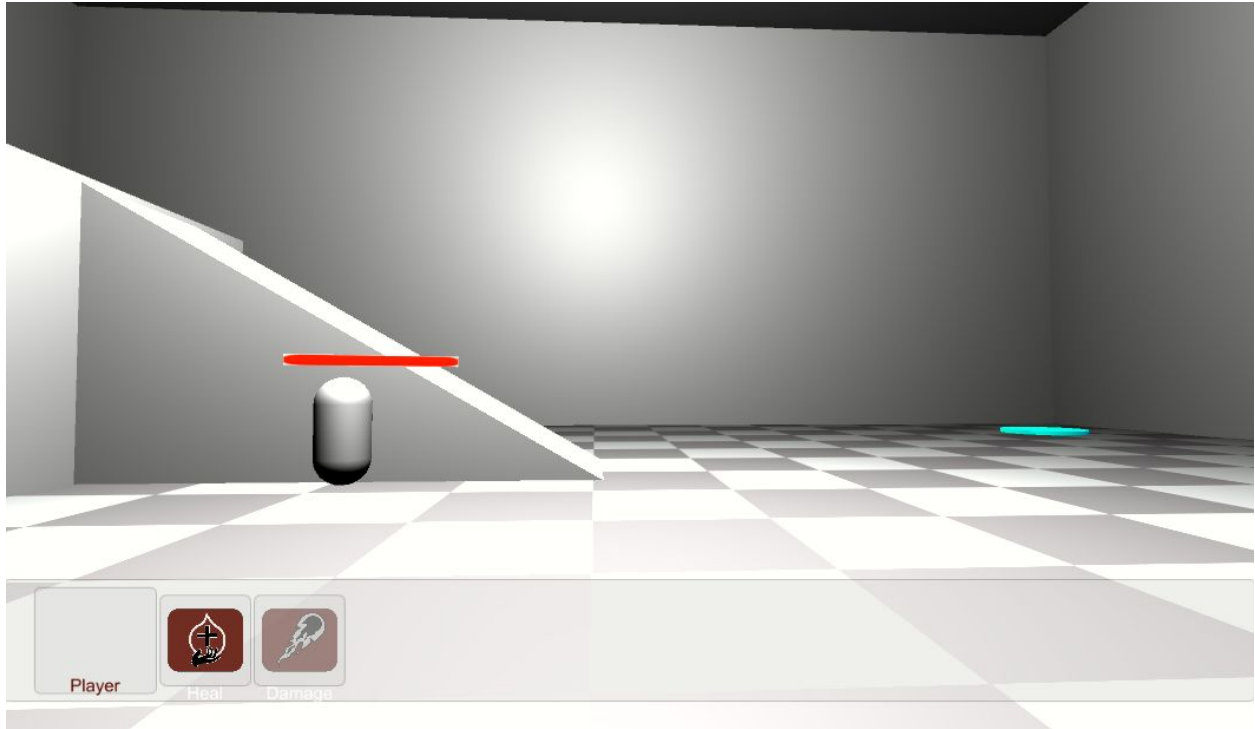
- Is Threatened Trigger - Fires depending on how threatened a target is by a source.

Examples

There are several examples located in the CAT RPG Example folder. Most notable is the Complete Game example which is a playable (but short) RPG which demonstrates almost all of the features.

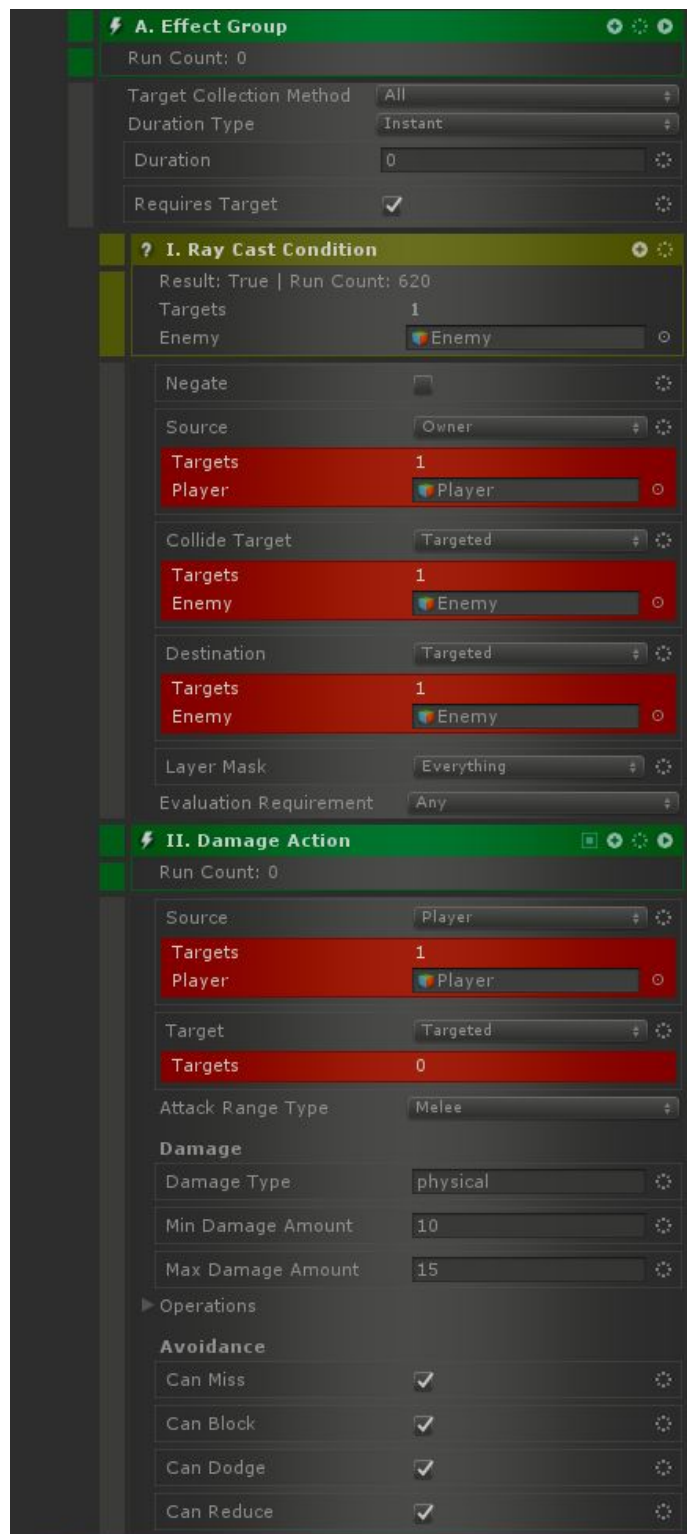
Abilities

Scene: RPG/Examples/Abilities/AbilityGym.unity



This example shows off two basic abilities which are displayed in the bar at the bottom of the screen. Click on the enemy to target them and then click the Abilities to test them out.

The Abilities can be found under the Player Game Object as Heal and Damage.



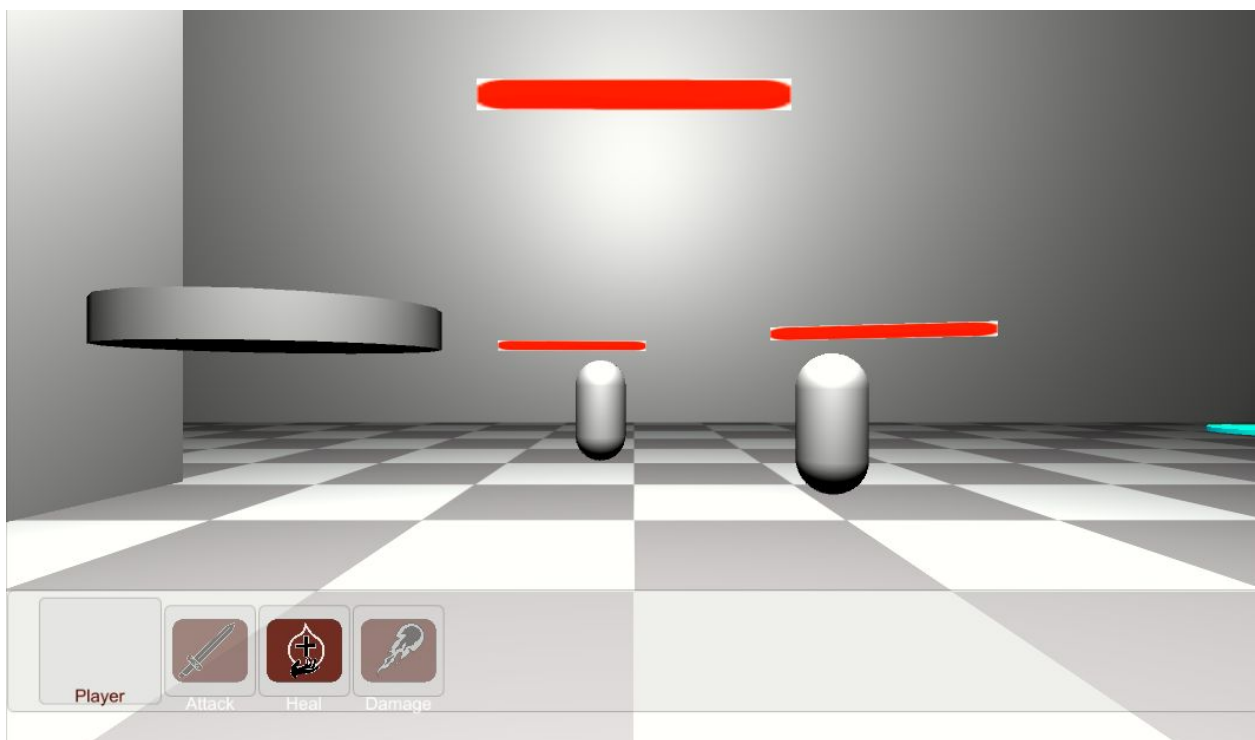
The Damage Ability shown above has one Effect Group. It targets the current Combat Target and ensures that there is line of sight to it via a Ray Cast Condition. If that succeeds, then the

Damage Action will cause between 10 and 15 damage potentially, though the attack may miss and the target can block, dodge, or reduce the damage.

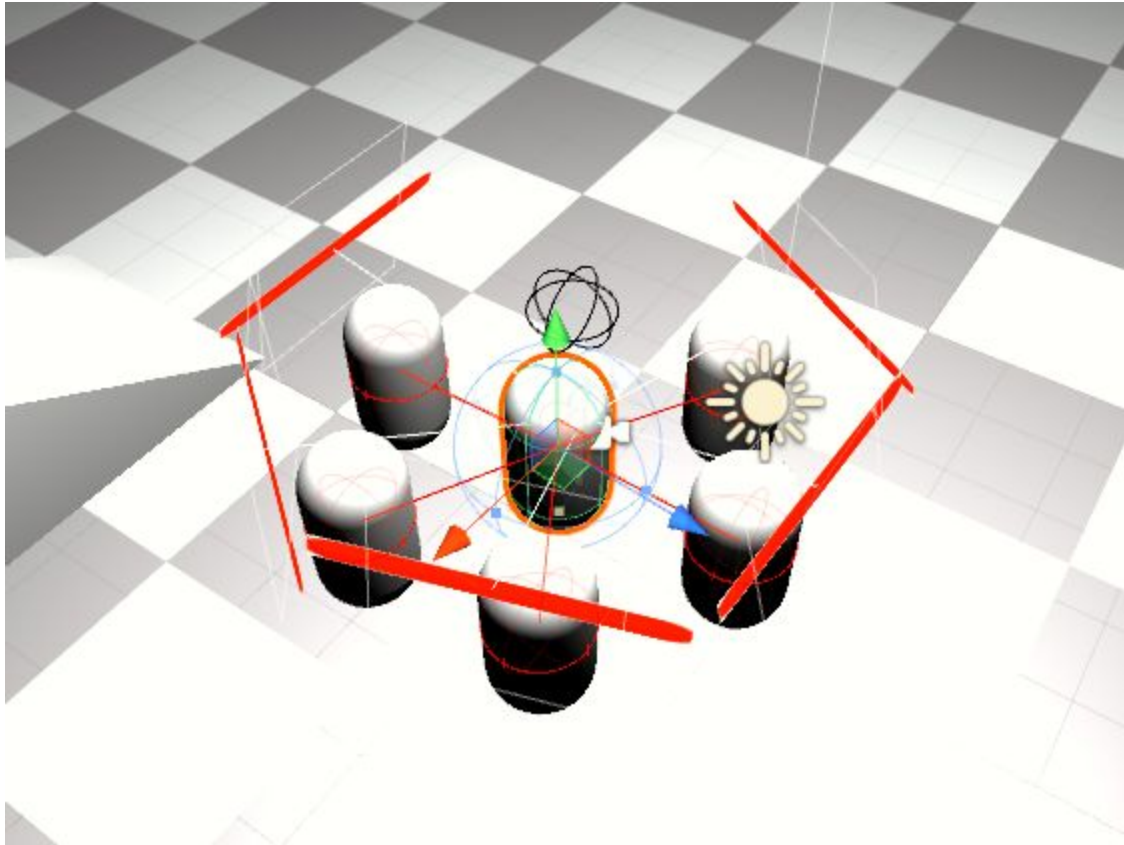
To move the player, use WASD. Hold down the left mouse button and move the mouse to look and change direction.

Attack Slots

Scene: RPG/Examples/AttackSlots/AttackSlotsExample.unity

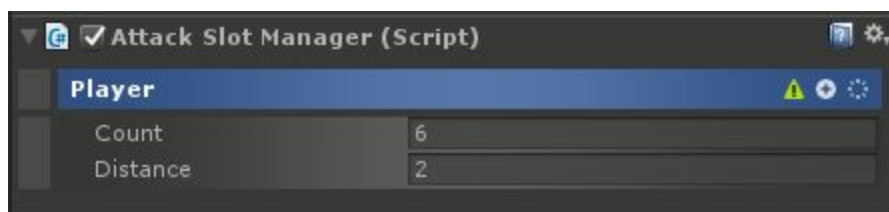


In this example, there are 4 enemies which will all pick an attack slot and attack the player. They should all line up in a ring:



To show the attack slots in the Scene window, select the Player Game Object. During runtime, reserved slots will show in red. The AttackBehavior on each enemy in the Alive State automatically uses Attack Slots if available and if they are appropriate to the attack ability.

To make this work, the Player has an Attack Slot Manager:



To move the player, use WASD. Hold down the left mouse button and move the mouse to look and change direction.

Complete Game

Scene: RPG/Examples/CompleteGame/CompleteGame.unity

This is the most complex example. It shows a sample level from an RPG and makes use of almost every system in CAT RPG (and many in base CAT).



To move the player, use WASD. Hold down the left mouse button and move the mouse to look and change direction. Hold Shift to run. Upon starting the game, the player has no abilities and therefore no way to attack. To start the quest, approach and click on Belegmo pictured below. After achieving second level, attack by either clicking the attack ability icon (shows up where the 6 is below) or by holding the mouse down on a target. Only enemies can be attacked.

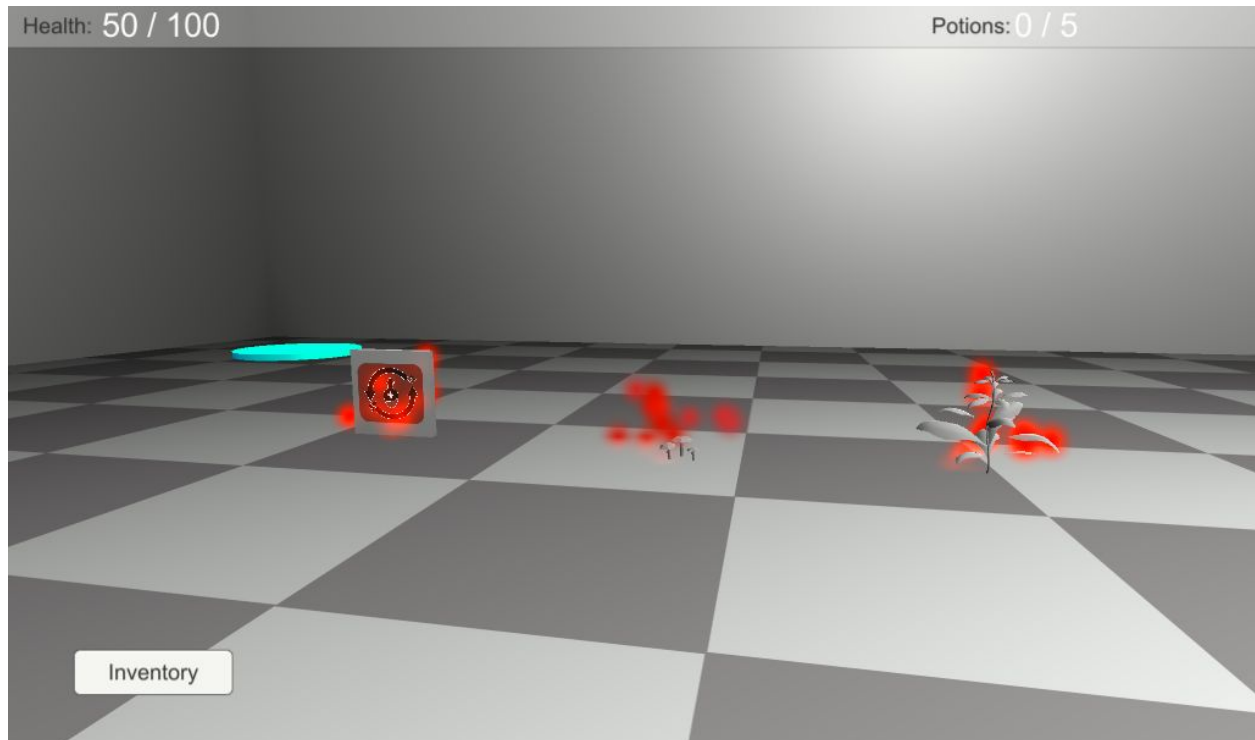


1. Health (red), Mana (blue)
2. Currency
3. Current top quest and top quest step. Click for Quest Log.
4. Click to open Inventory window.
5. Level and experience.
6. Abilities will show here. Click to use.

Progress will be saved, but it is possible to reset it using the CAT -> Misc -> Clear Local Storage menu option.

Crafting

Scene: RPG/Examples/Crafting/CraftingExample.unity



This example shows basic crafting. To test, click on the two ingredients (pictured on the right above) and the recipe (on the left) to collect them. Then click the Inventory button and select the Recipe and click Use:

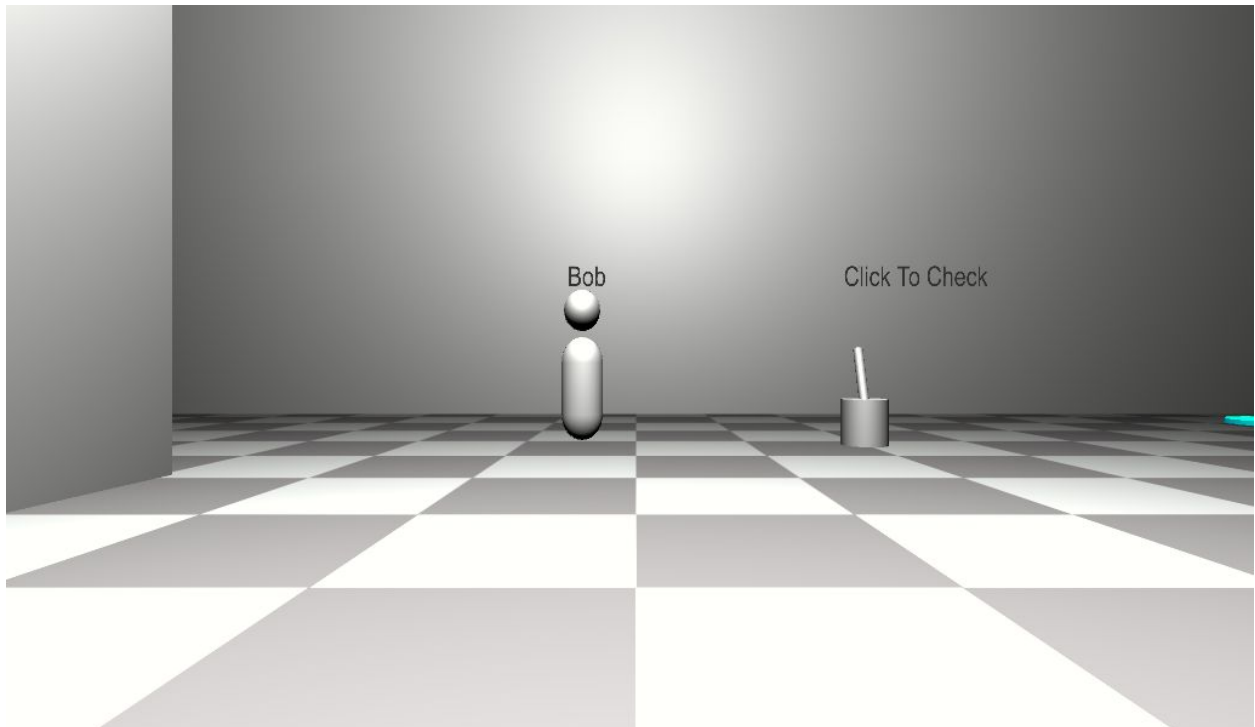


Note that in order to interact with things to pick them up, you must be close to them.

To move the player, use WASD. Hold down the left mouse button and move the mouse to look and change direction.

Dialogue

Scene: RPG/Examples/Dialogue/DialogueExample.unity



In this small example, click on Bob to show his Dialogue. Then, click on the switch to the right in order to display the results of several Conditions about which Dialogue Lines were shown. To reset, use the CAT -> Misc -> Clear Local Storage menu option.

Bob has a Dialogue Manager and a Dialogue:

▼

✓ Dialogue Manager (Script)

?

⚙

Bob

⊕

⚙

Override Display Type

None

⬇

Override Speech Bubble Pr

None (Game Object)

⊙

⚙

Override Dialog Window

None (Game Object)

⊙

⚙

Accumulate Lines

☐

Character Portrait

None (Game Object)

⊙

⚙

Character Name

Bob

⚙

String

(Localization Missing)

1. Bob Dialogue

⊕

⚙

Playback Type

First Only

⬇

Override Display Type

None

⬇

Override Speech Bubble Pr

None (Game Object)

⊙

⚙

Override Dialog Window

None (Game Object)

⊙

⚙

Storage Level

Character

⬇

Dialogue ID

BobDialogue

Restart On Exit

☒

⚙

Evaluation Requirement

Any

⬇

A. Bob Greeting

⊕

⚙

Line

Hey, I'm Bob!

⚙

String

(Localization Missing)

Override Display Type

None

⬇

Delay

0

⚙

Next Line Name

⚙

Override Speech Bubble Pr

None (Game Object)

⊙

⚙

Override Dialog Window

None (Game Object)

⊙

⚙

Voiceover

None (Audio Clip)

⊙

▶ Audio Source Options

Override Character Portrait

None (Game Object)

⊙

⚙

Override Character Name

⚙

String

(Localization Missing)

I. Bob Greeting Response

⊕

⚙

Line

Cool!

⚙

String

(Localization Missing)

Next Line Name

Question

⚙

Exits Dialogue

☐

⚙

Bob also has a Dialogue Interaction Behavior which will pop up the Dialog when he is clicked on. Additionally, he has several Triggered Actions which enable the messages that show up when clicking on the switch.

1. Dialogue Interaction Behavior

Target: Owner

Targets: 1

Bob: Bob

2. Triggered Action

A. Speak With Trigger

Negate: ☐

Check On Start: ☒

Fire Once: ☐

Target: Owner

Targets: 1

Bob: Bob

B. Set Enabled Action

Target: Owner

Path: BobCanvas/TalkedTo

Targets: 1

TalkedTo: TalkedTo

Is Enabled: ☒

Revert On Stop: ☒

3. Triggered Action

A. Has Dialogue Trigger

Negate: ☐

Check On Start: ☒

Fire Once: ☐

Target: Owner

Targets: 1

Bob: Bob

Dialogue ID: BobDialogue

B. Set Enabled Action

Target: Owner

Path: BobCanvas/HasDialogue

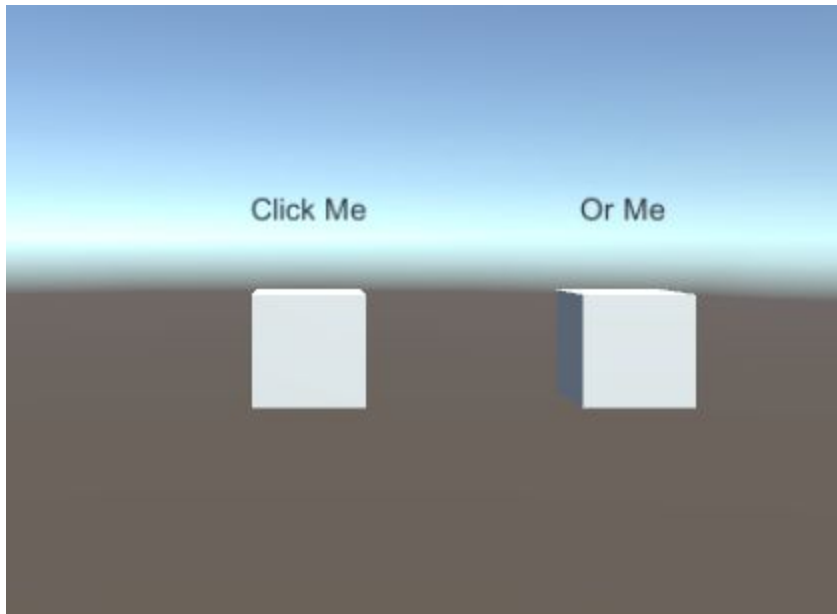
Targets: 1

HasDialogue: HasDialogue

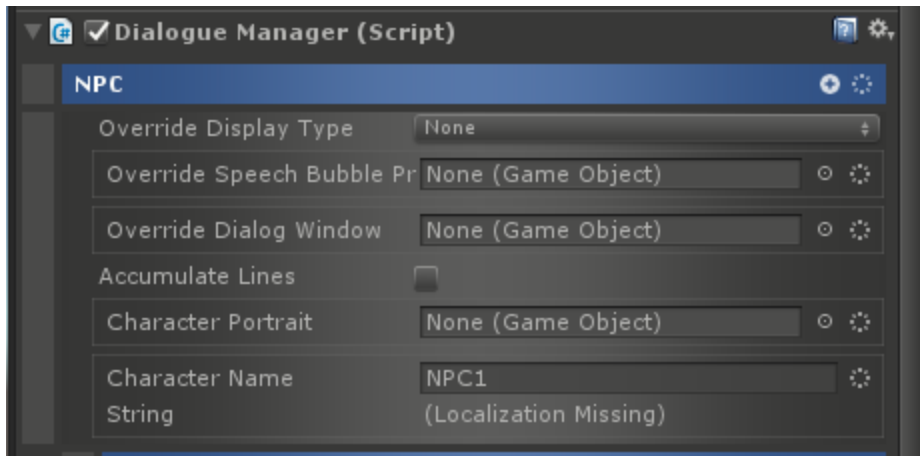
Is Enabled: ☒

Revert On Stop: ☒

Scene: RPG/Examples/Dialogue/DialogueSimple.unity



This is an even simpler Dialogue example. Click on either cube to bring up its dialogue. Each cube has a Dialogue Manger:



And a dialogue:

Dialogue (Script)

Dialogue

Playback Type

First Only

Override Display Type

Windowed

Override Speech Bubble Pr

None (Game Object)

Override Dialog Window

DialogueWindow

Storage Level

Character

Dialogue ID

testID10

Restart On Exit

☒

Evaluation Requirement

Any

1. Line 1

Line

Hey how are you?

String

(Localization Missing)

Override Display Type

None

Delay

0

Next Line Name

Override Speech Bubble Pr

None (Game Object)

Override Dialog Window

None (Game Object)

Voiceover

None (Audio Clip)

Audio Source Options

Override Character Portrait

None (Game Object)

Override Character Name

String

(Localization Missing)

A. Reaction 1

Line

Good!

String

(Localization Missing)

Next Line Name

Line2

Exits Dialogue

☐

B. Reaction 2

Line

Bad!

String

(Localization Missing)

Next Line Name

Line3

Exits Dialogue

☐

Factions

Scene: RPG/Examples/Factions/FactionGym.unity



This example shows two opposed Factions, Orcs and Elves. It spawns one of each and directs them to walk to the center of the level where they will see each other and start to fight since they are enemies. The player can also attack or heal either Faction.

To move the player, use WASD. Hold down the left mouse button and move the mouse to look and change direction. To attack, select a target with the mouse and then use one of the ability buttons at the bottom.

In the same folder as the scene are the Faction definitions `Elves.asset`, `Humans.asset` (the player), and `Orcs.asset`.

Elves

Open

Script

Faction

Faction Name

Elves

String

(Localization Missing)

Description

String

(Localization Missing)

On Attack Modifier

-2

On Kill Modifier

-5

On Heal Modifier

2

Faction Dispositions

Size

2

Orcs

Name

Orcs

Disposition

-10

Humans

Name

Humans

Disposition

10

Orcs

Open

Script

Faction

Faction Name

Orcs

String

(Localization Missing)

Description

String

(Localization Missing)

On Attack Modifier

-2

On Kill Modifier

-5

On Heal Modifier

2

Faction Dispositions

Size

2

Elves

Name

Elves

Disposition

-10

Humans

Name

Humans

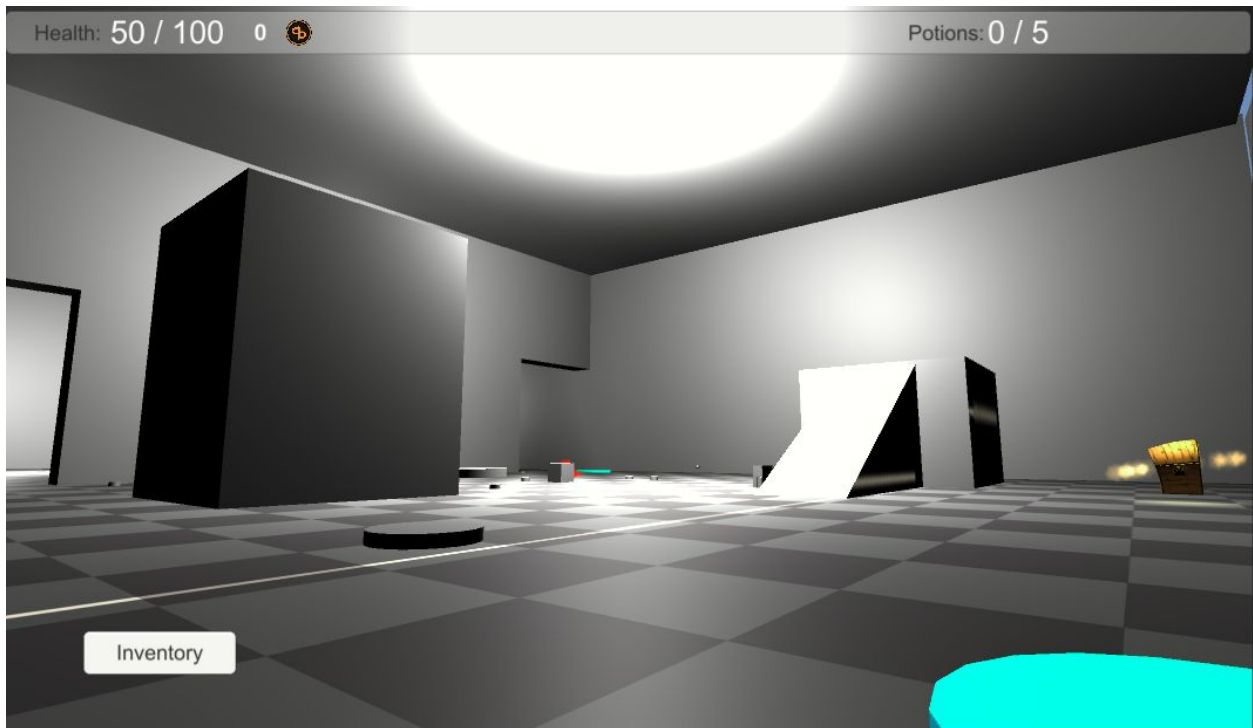
Disposition

0

Note that Orcs and Elves have a Disposition of -10 to each other.

Interactive Objects

Scene: RPG/Examples/InteractiveObjects/InteractiveObjectsExample.unity

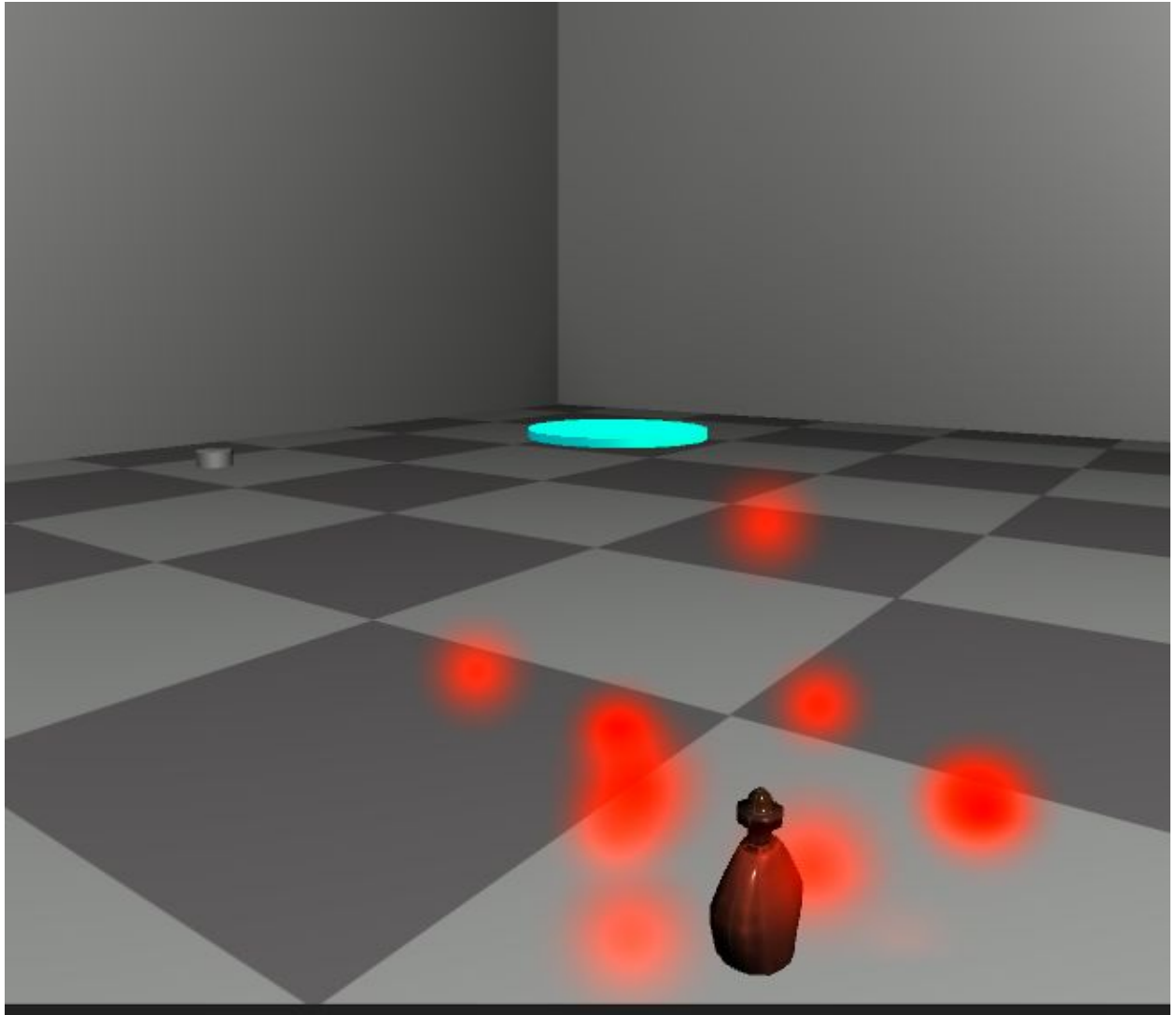


This example includes many different Interactive Object types including Doors, Chests, Switches, Destructibles, Traps, and Moving Platforms.

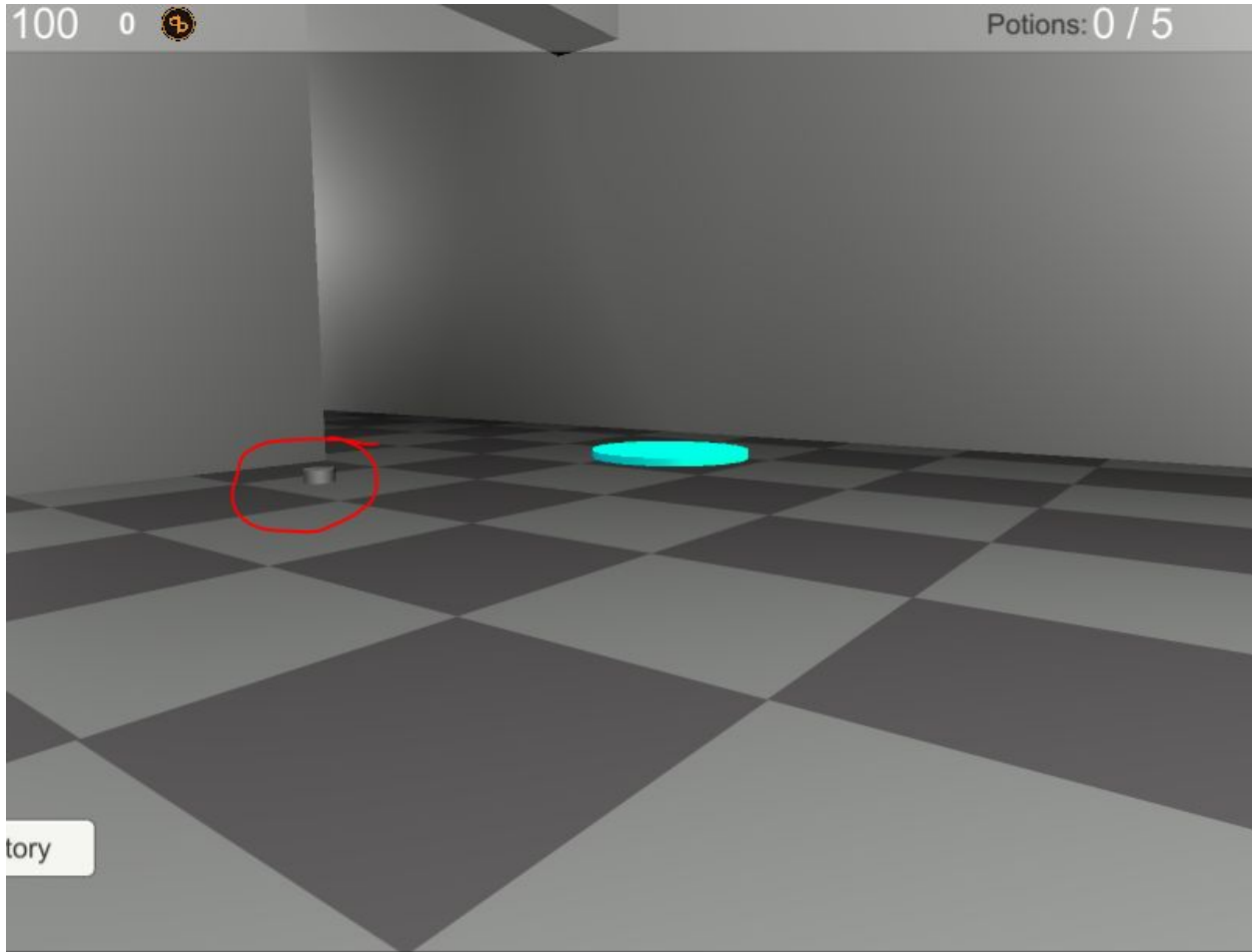
To move the player, use WASD. Hold down the left mouse button and move the mouse to look and change direction.

Doors

There are two Doors on the back wall. The one on the right (center of the image above) is locked. The key is the potion bottle shown here:

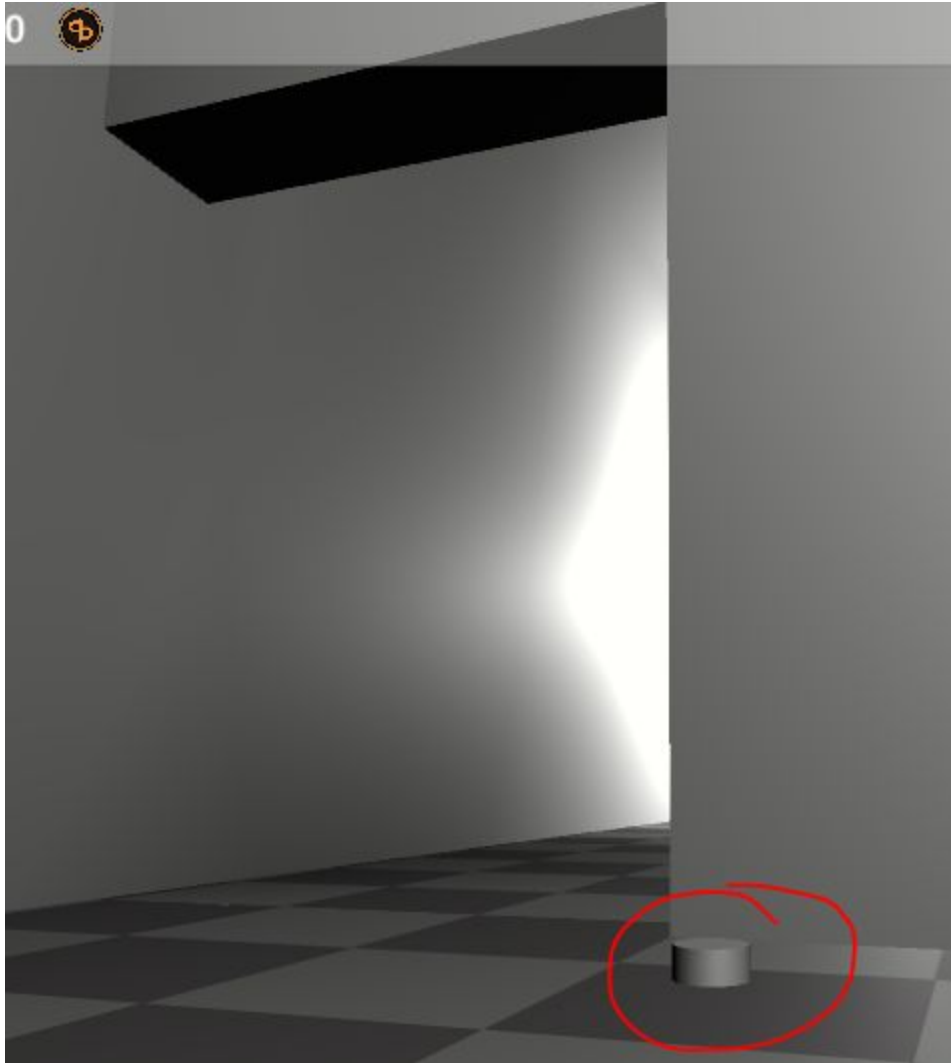


Click on it to pick it up and then click on the door to open it.



Alternately, there is a switch in front of the door that can be used to unlock it. Click the Switch and then click the door to open it even without a potion. The Switch is push on, push off. This means that the first click will turn it on and unlock the Door. The second click will turn it off and re-lock the door.

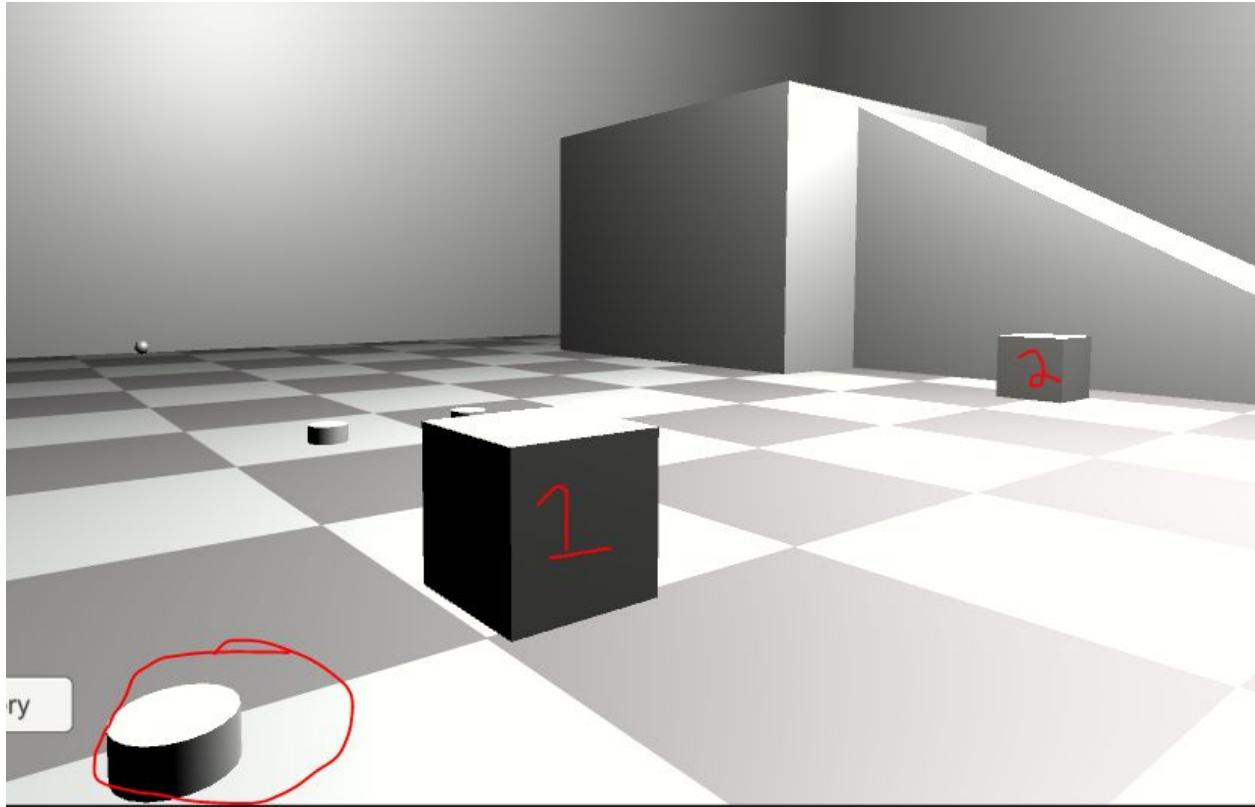
The other Door is to the left in the image above. It is not locked and can be opened by clicking on it or by using the Switch in front of it.



This Switch is also push on/push off and will directly control whether the Door is open (on) or closed (off).

Destructibles

There are two Destructibles in the example.



Destructible #1 in the image above is controlled by the circled Switch. Click the Switch to destroy the cube. This Switch is momentary, so it will push on but will immediately turn off. Destructible #2 will be destroyed if the player comes too close.

Chest

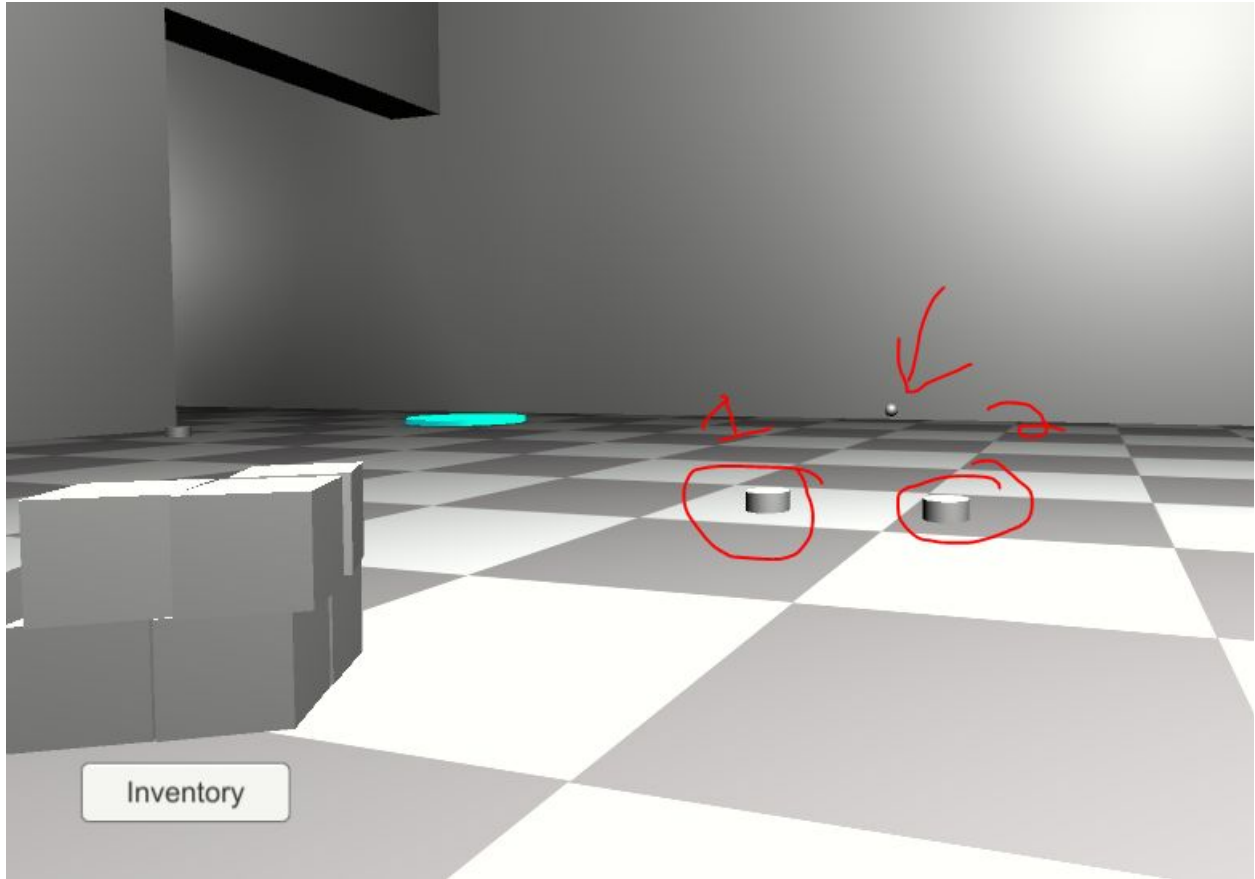
There is a Chest in the corner of the level.



Click to open and loot. It can be clicked again to close.

Spawners

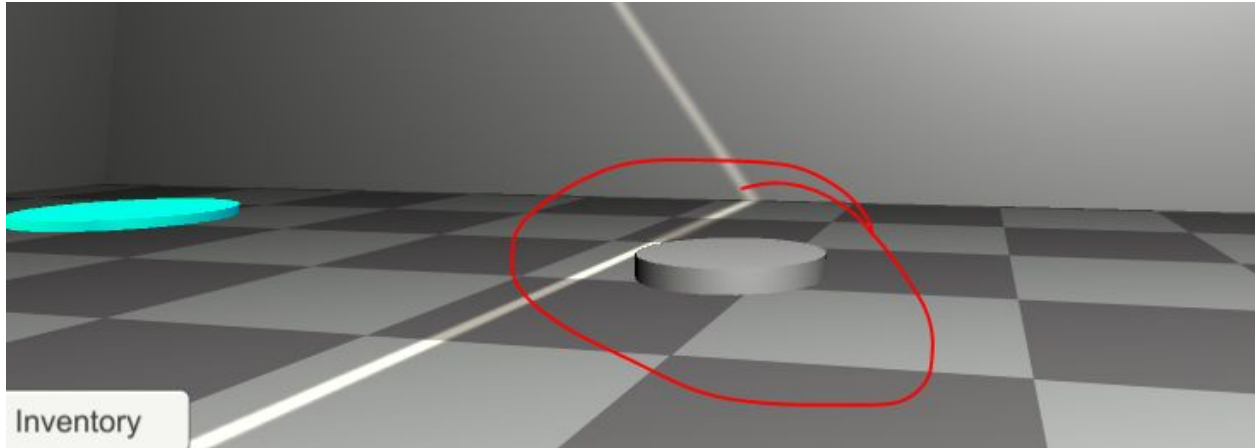
In addition to the other Switches mentioned, there are a pair of Switches that control a Spawn Camp.



Switch #1 above will instruct the Spawn Camp to unpause and start spawning. It is push on/push off, so when off, the Spawn Camp will be paused again. Switch #2 will remove all the spawned characters and is momentary. The Spawn Camp is denoted by the small ball with an arrow pointing at it in the image above.

Trap

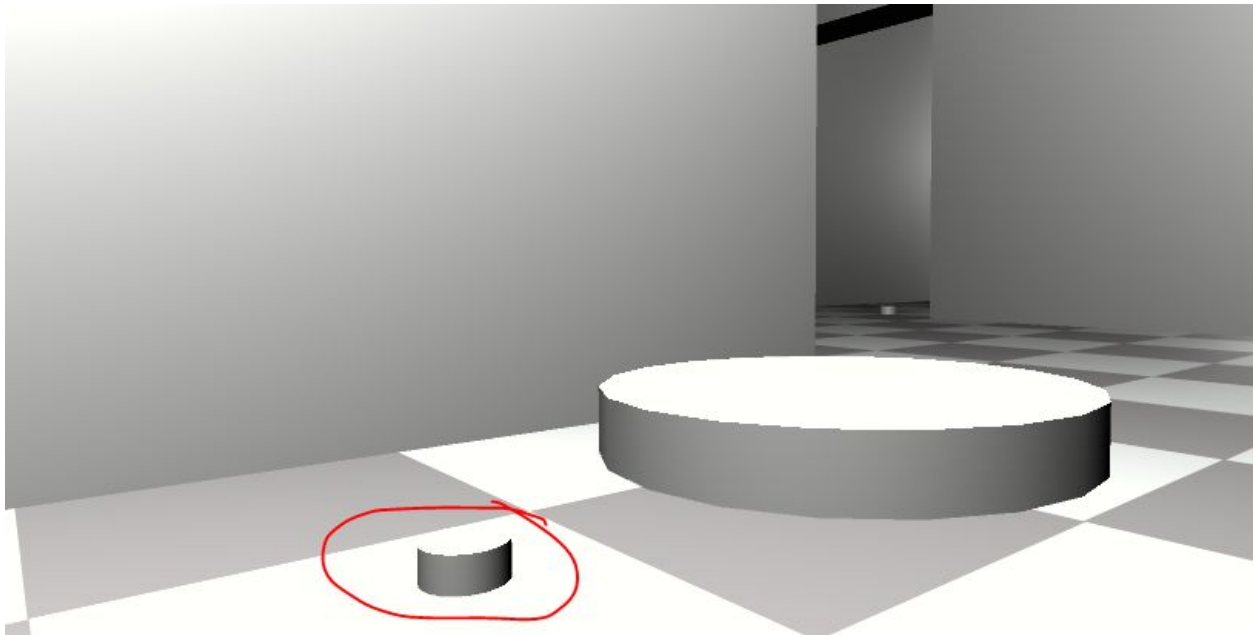
There is a Trap in the level denoted by this disc:



If the player steps on it, they will take damage.

Moving Platform

There is also a Moving Platform in the level.

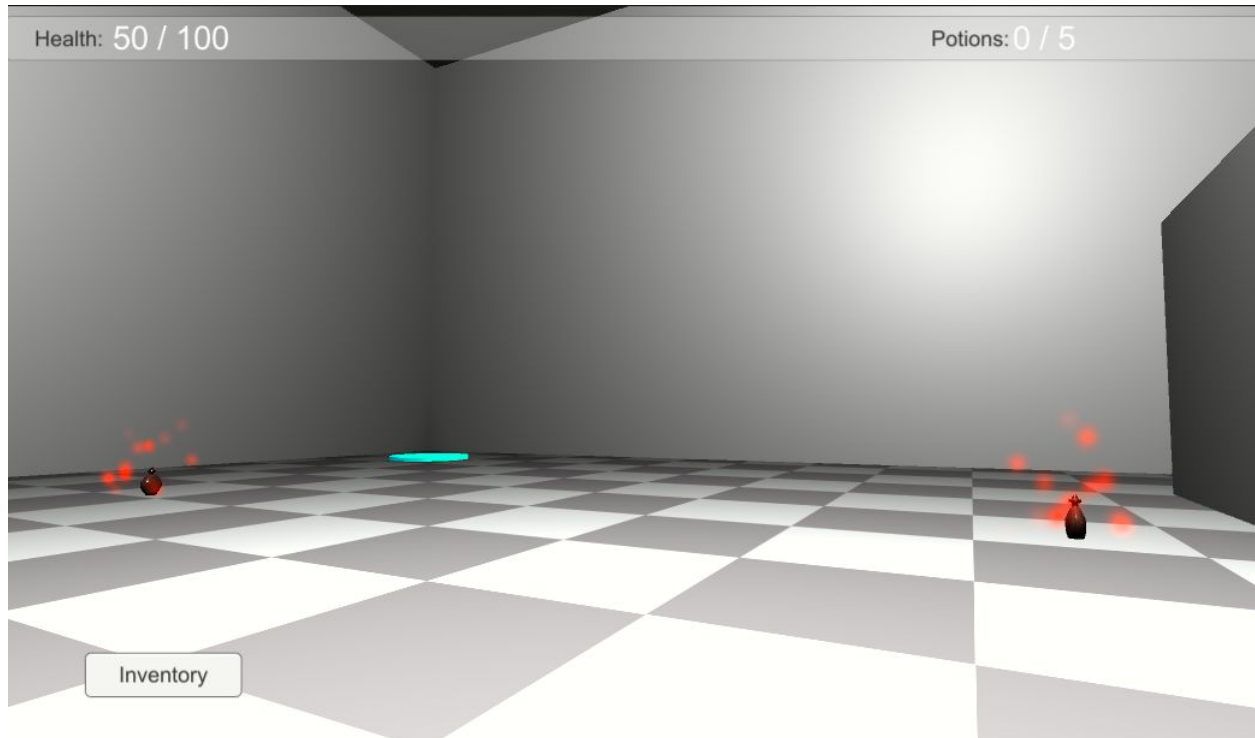


It can be turned on or off by the circled Switch which is push on/push off.

Inventory

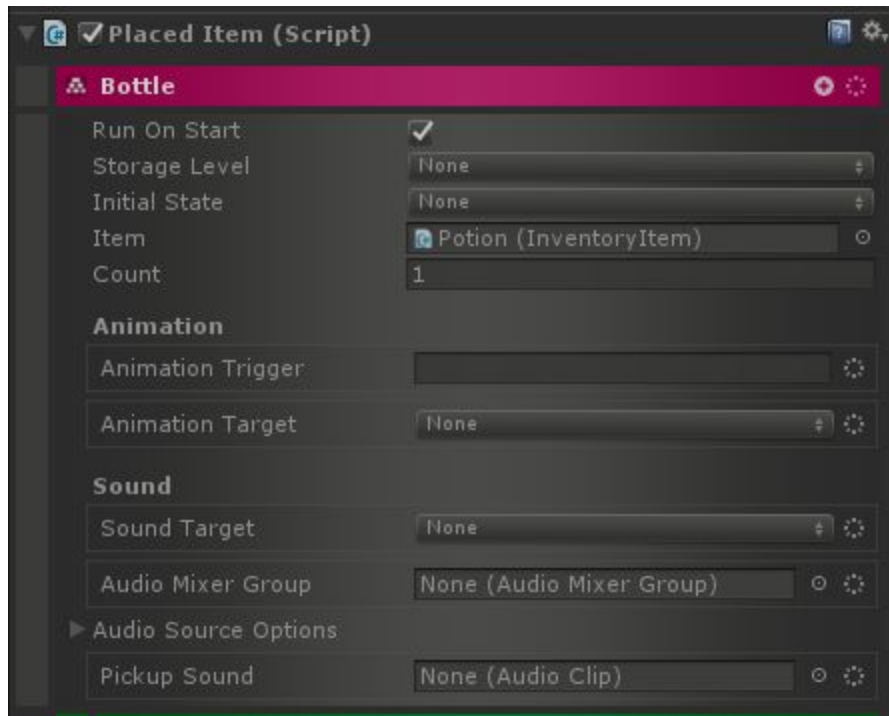
Scene: RPG/Examples/Inventory/InventoryExample.unity

In the Inventory example, there are 5 potions hidden throughout the level. Once all are obtained, the count will flash.

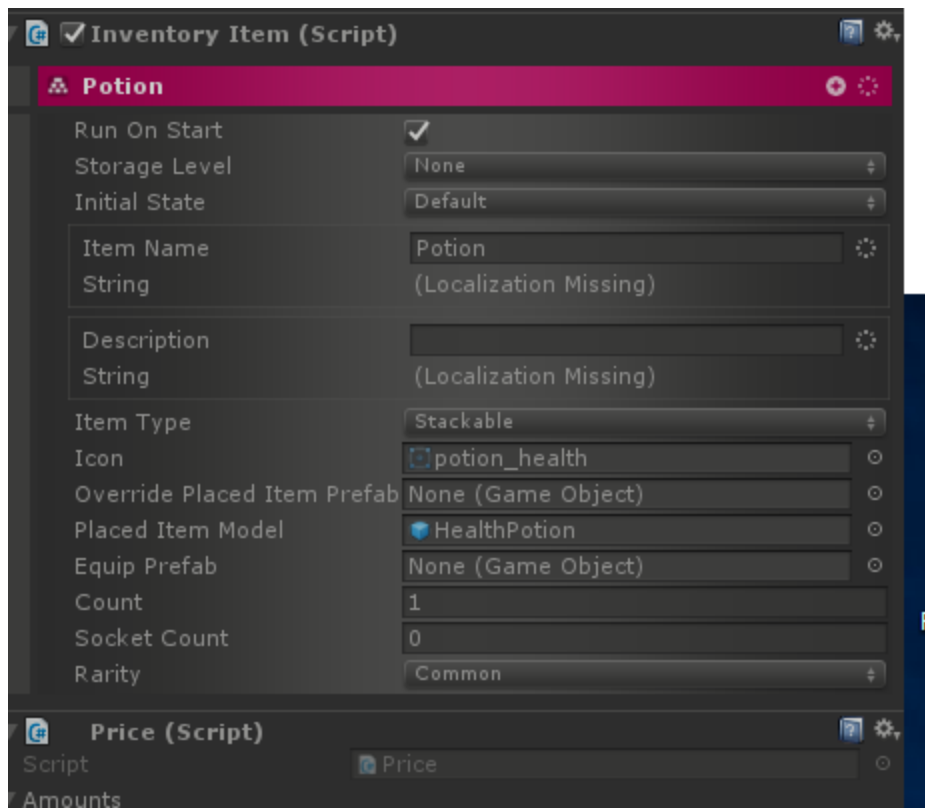


To move the player, use WASD. Hold down the left mouse button and move the mouse to look and change direction.

Each potion in the level is a Placed Item:



When picking up, it becomes the Potion item in inventory:



Using the Inventory button in the corner of the screen will cause the Inventory Window to pop up. Items in the window can be clicked on to bring up a sub-menu which allows dropping or using them. Using the potions will restore health.

Under Canvas/HUD/PotionsCount, which is a State Machine, there are two States: Default and Pulse.

The Default State has a Have Item Quantity Trigger which checks for the player to have 5 potions and then has a Change State Action to change to Pulse. The Pulse State has some nested Action Lists which animate the color and scale of the text attached to the PotionsCount Game Object.

Loot

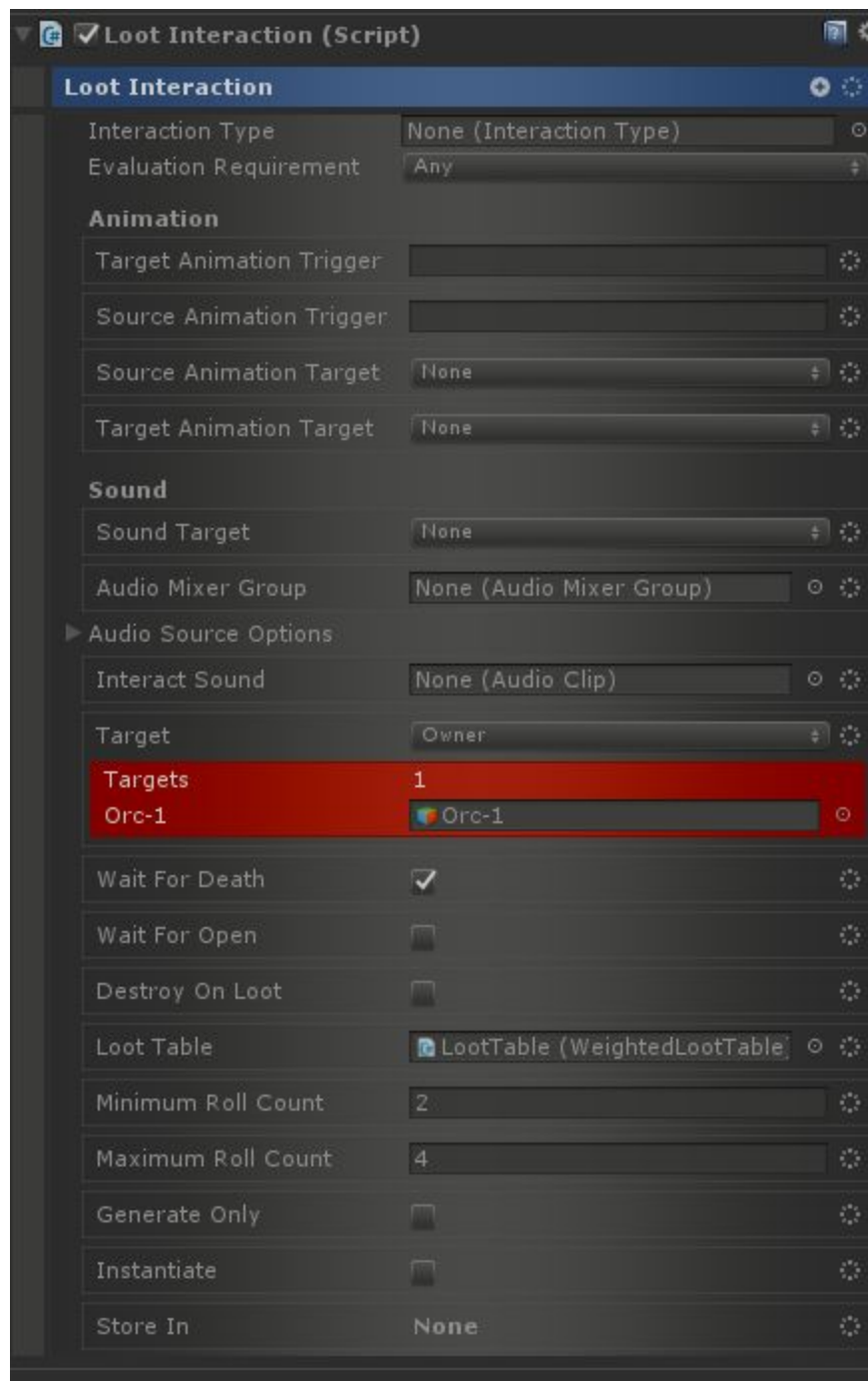
Scene: RPG/Examples/Loot/LootGym.unity



The Loot Gym spawns Orcs which move to the center of the area and the player can click on them to target them and then use Abilities to kill them. Once dead, they will spawn Loot which can be collected.

To move the player, use WASD. Hold down the left mouse button and move the mouse to look and change direction.

The Orcs have a Loot Interaction which is set to Wait For Death:



It picks 2 - 4 items from this Loot Table:

Weighted Loot Table (Script)

Loot Table

1. Currency

Weight

4

Icon

Cherry

Pickup Item Prefab

None (Game Object)

Evaluation Requirement

Any

Currency Name

Cherries

Min Amount

10

Max Amount

10

2. Bottle

Weight

5

Icon

potion_health

Pickup Item Prefab

Bottle

Evaluation Requirement

Any

Target

Owner

Targets

0

Spawn At Position

☒

Position

Owner

Offset

X 0Y 2Z 0

Randomize Offset

☐

Targets

0

Bundle Name

Item Name

Potion

Count

1

Place On Ground

☐

Min Attachments

0

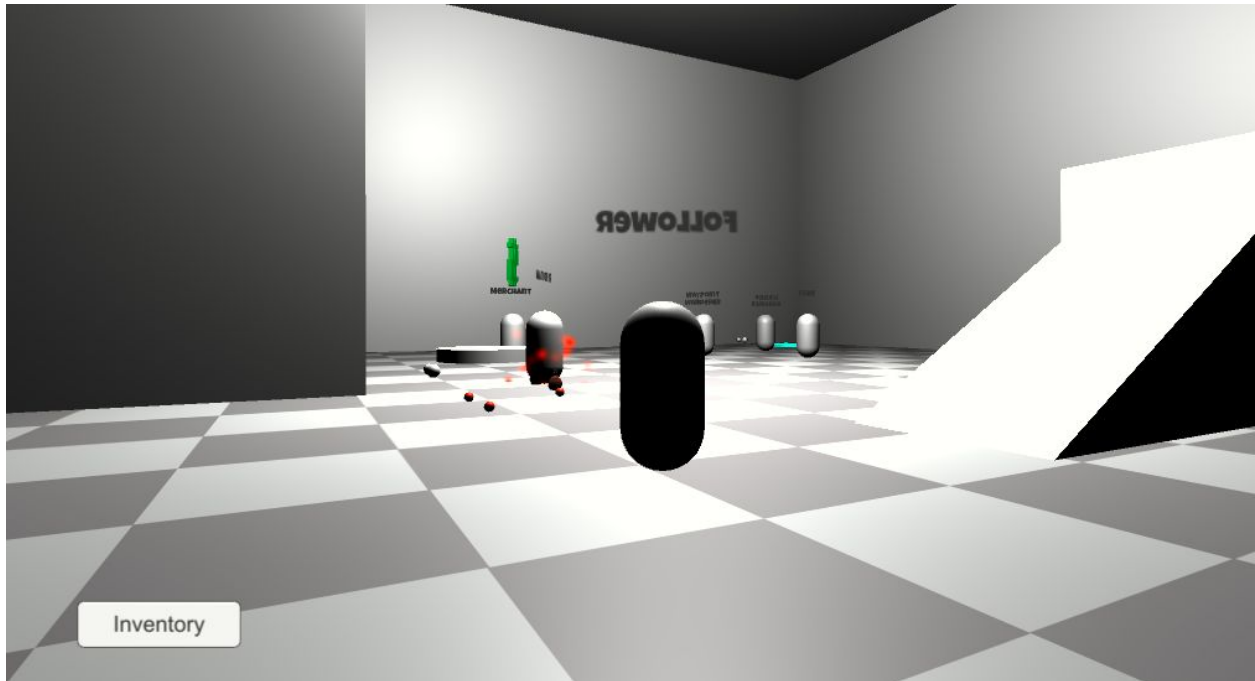
Max Attachments

0

Attachments

NPCs

Scene: RPG/Examples/NPCs/NPCBehaviorGym.unity

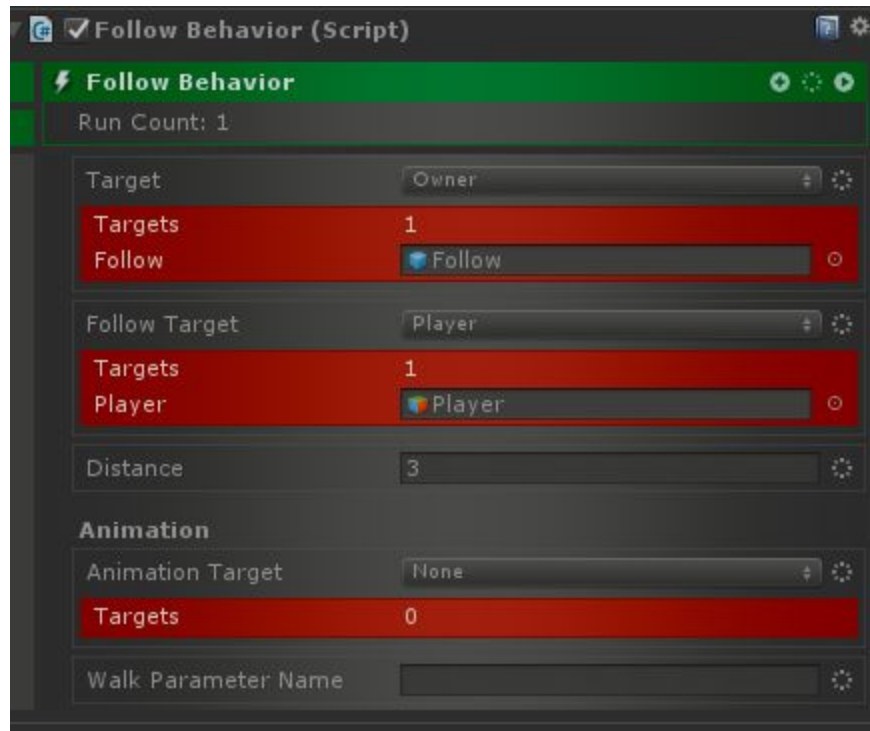


In this gym, there are examples of many of the NPC behaviors available in CAT RPG Builder including Followers, various Wanderers, Flee, and Merchant. Each NPC has a label above its head saying what it is doing.

To move the player, use WASD. Hold down the left mouse button and move the mouse to look and change direction.

Follower

The Follower as the name implies is set to follow the player up to a certain distance using the Follow Behavior:



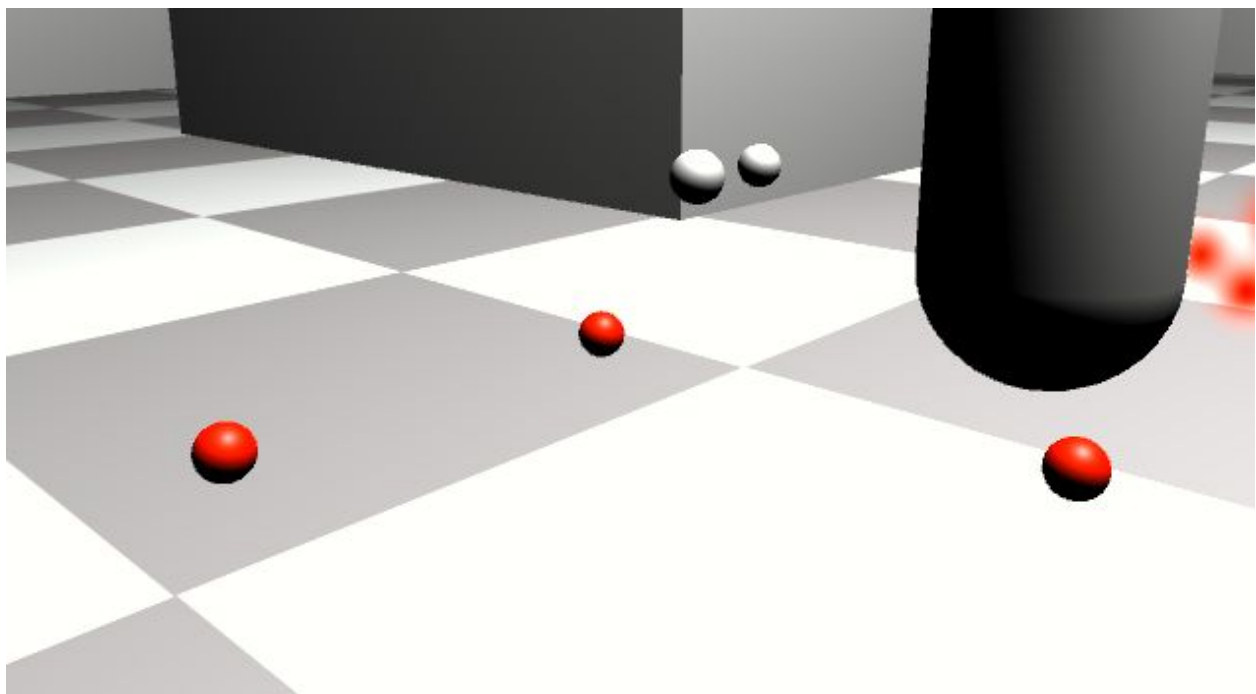
Flee

Similarly, the Flee NPC uses the Flee Behavior to run away from the player to a certain distance:



Merchant

There's also a Merchant. To buy items, collect the red balls which are currency:



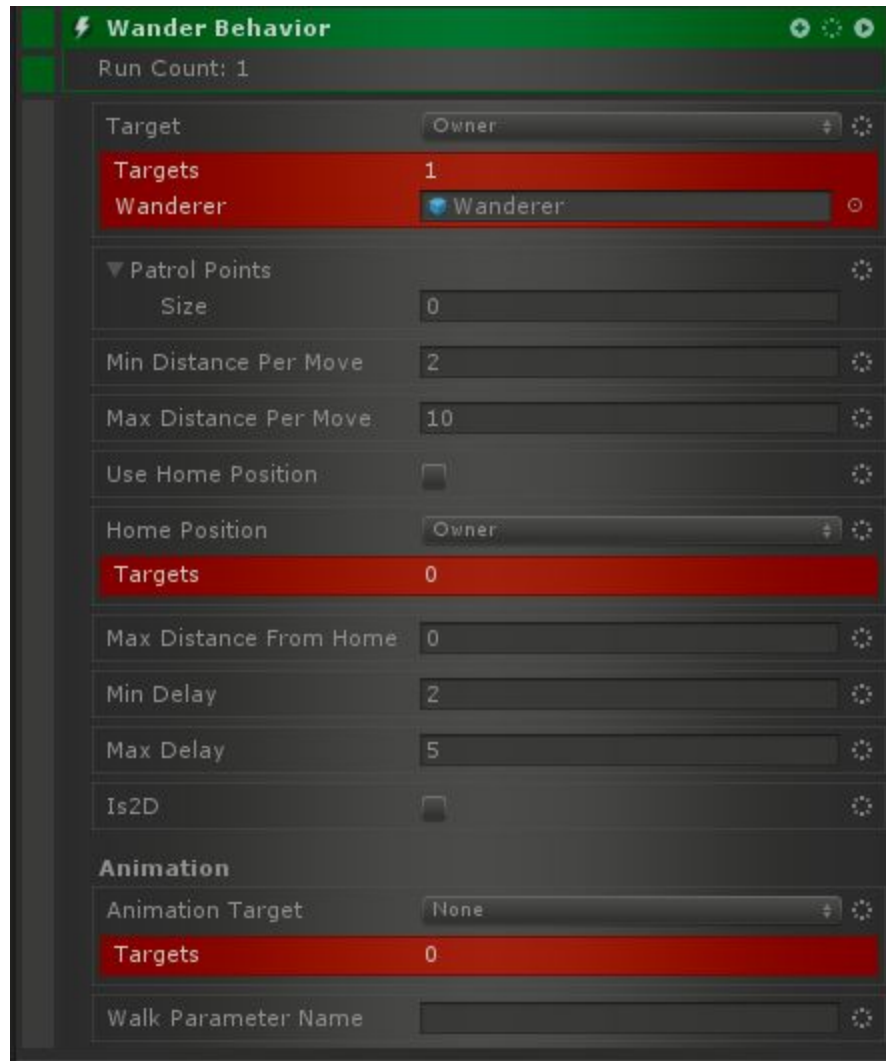
Approach and click on the Merchant to interact:



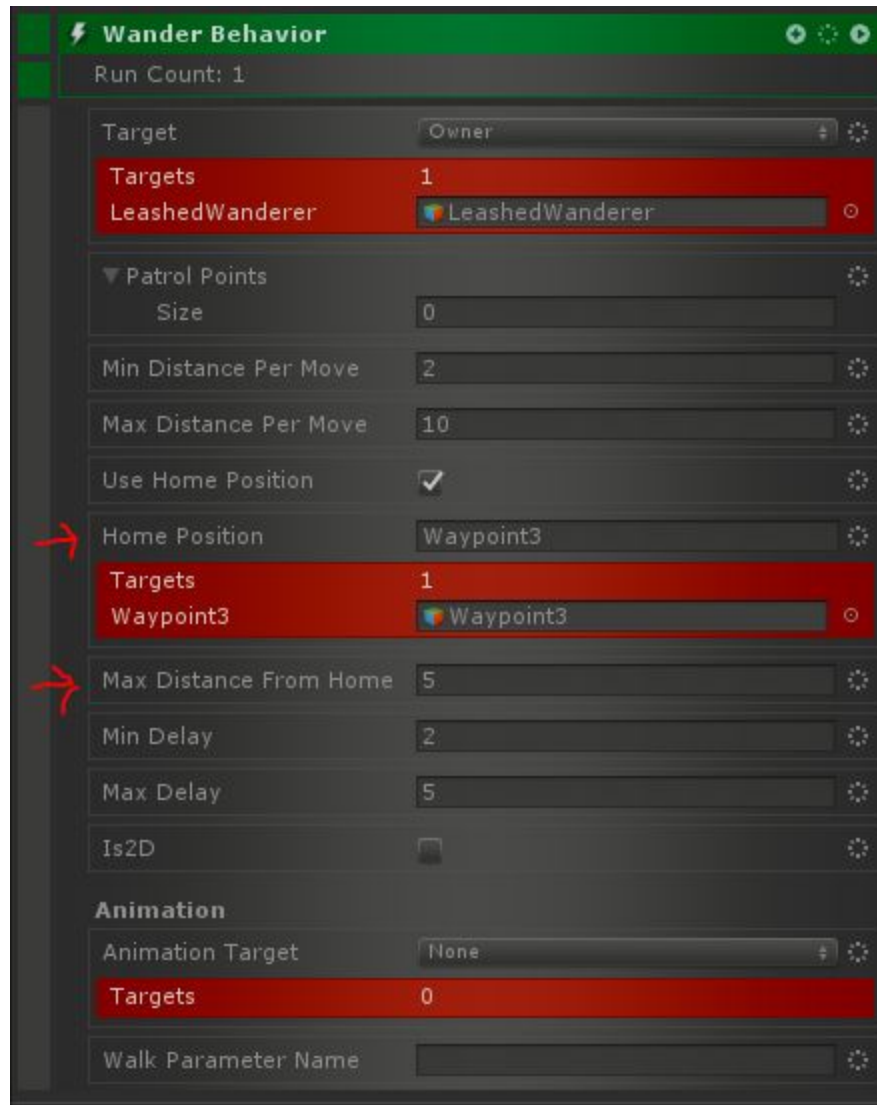
Click on items in the top window to buy them and on the bottom window to sell.

Wanderers

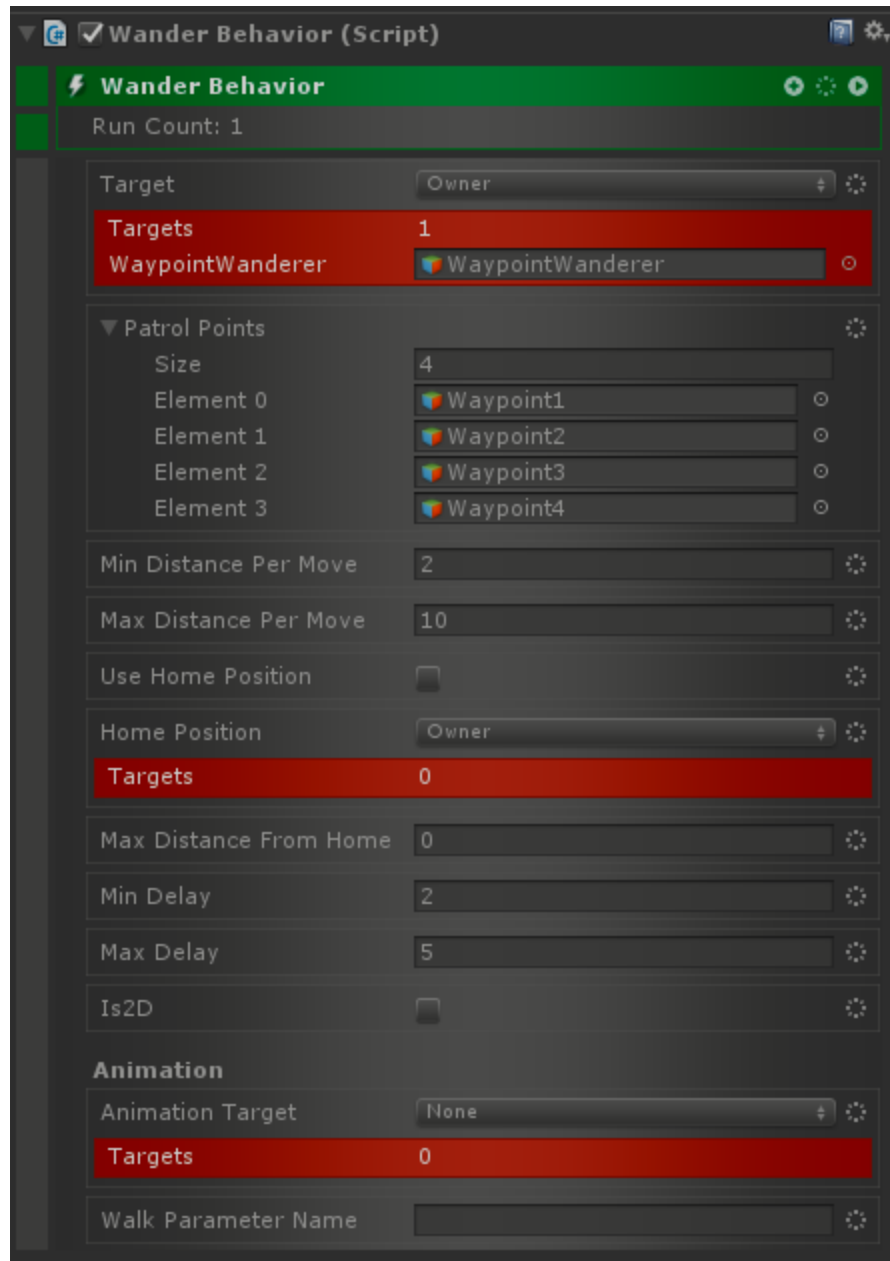
There are 3 Wanderer NPCs showing different variants of the behavior. The first is just a regular Wanderer. This NPC will just wander aimlessly anywhere it can get to.



Then, there's the Leashed Wanderer which acts the same way except it will never stray too far from its home point:



Finally, there's the Waypoint Wanderer which moves from Waypoint to Waypoint randomly. This is different from the Patrol Behavior because it randomly selects waypoints.



Patrol

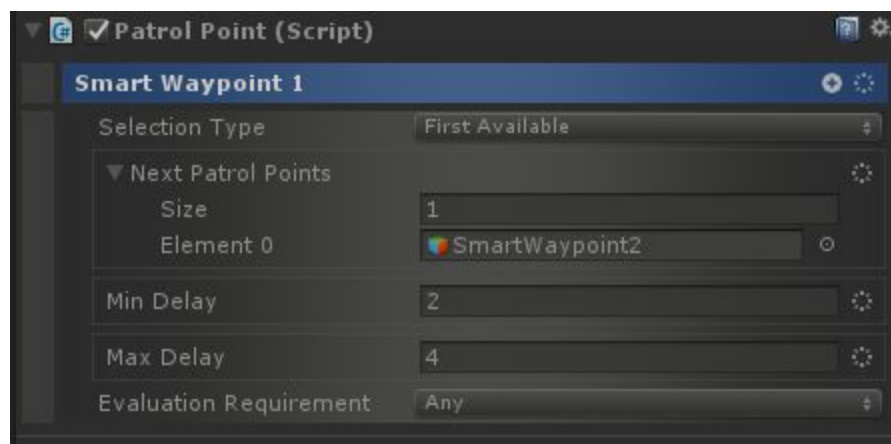
There are two Patrol NPCs one of them uses normal Game Object based Patrol Points and the other uses Game Objects with Smart Waypoints attached. The simpler one will just visit each Waypoint in order.



While the Smart Waypoint one will make decisions at some of the Waypoints:



It will start at the first Smart Waypoint and then decide where to go next.



The third Smart Waypoint will only be used if the Player is nearby:



Progression

Scene: RPG/Examples/Progression/ProgressionGym.unity



In this Gym, Orcs are spawned which the player can attack and kill. Killing an Orc grants experience and the player will level up. The current level and experience are displayed at the top of the screen as shown above and marked in red.

The player starts with only one ability but gains a second at level 2.

To move the player, use WASD. Hold down the left mouse button and move the mouse to look and change direction.

Threat List

Scene: RPG/Examples/ThreatList/ThreatListGym.unity



Similar to the Factions Gym, the Threat List Gym shows off the Threat List and Threat Sensing Behavior. Orcs are spawned on one side and Elves on the other and they meet in the middle to fight. The player can also attack or heal either side.

To move the player, use WASD. Hold down the left mouse button and move the mouse to look and change direction.

⚡ Threat Sensing Behavior

Run Count: 1

TargetOwner

Targets1

Orc-0Orc-0

Base Threat Mod10

Check Frequency0.5

Layer MaskEverything

Debug☐

Levels

Threatened At Level2

Attack At Level5

Flee At Level20

Vision

Enable Vision☒

Vision Range30

Field Of View120

Vision Threat Mod1

Visual Threat Fade Time5

Hearing

Enable Hearing☐

Hearing Range0

Hearing Threat Mod0

Hearing Threat Fade Time0

Proximity

Enable Proximity☐

Proximity Range3

Proximity Threat Mod1

Proximity Threat Fade Time5

States

Default State NameWander

Threaten State Name

Attack State NameAttacking

Flee State Name

The Threat Sensing Behavior controls what the NPCs are doing at any given moment and moves them to different states such as Wander or Attacking.