

Day 7 (Yarn Spinner Jam)

[Download this package](#)

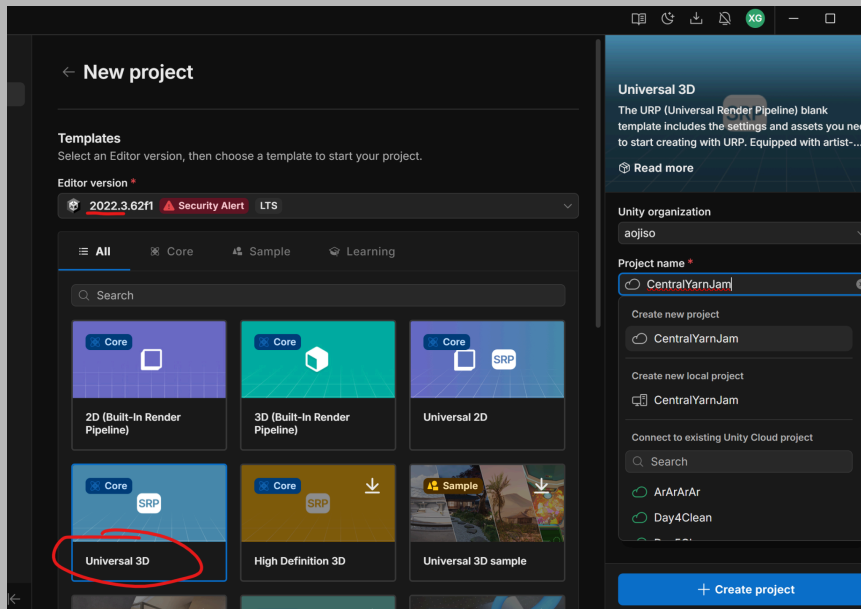
Overview

Yarn Spinner is a powerful and flexible dialogue system that is frequently used in Unity. It's been used in some celebrated indie games including [Night in the Woods](#) and [Dredge](#).

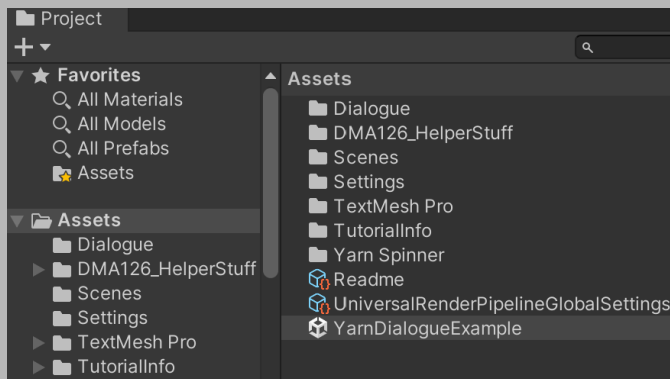


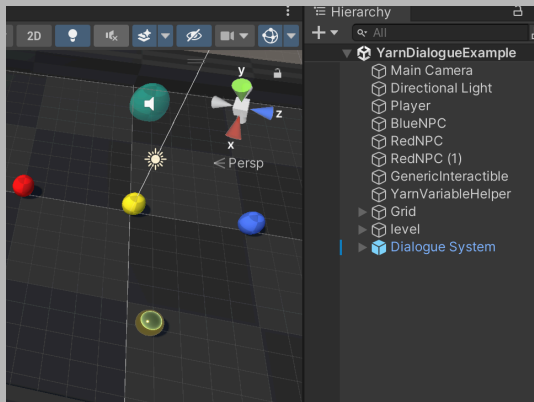
Setup

1. Create a new Unity Project, choosing Universal 3D as the template.

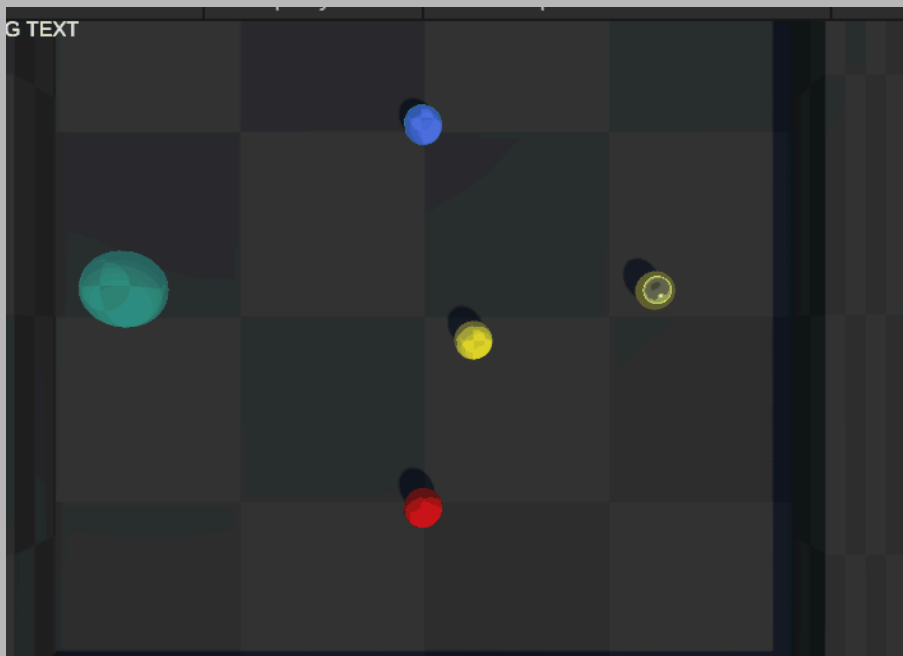


2. Once your project is open, download, and import [this package](#), which includes a simpler easier to work with example scene, though I recommend checking out the official examples as well.
3. Double-click to open scene [Assets/YarnDialogueExample](#)





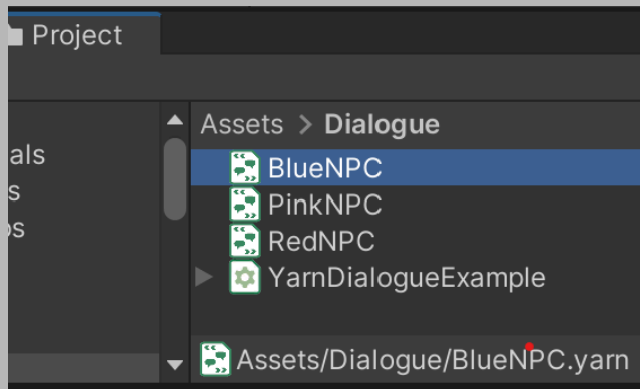
4. Hit play, and move with the arrow keys or WASD, and press space to interact with objects.



Yarn Files

We write the specific dialogue/text for our game in special *.yarn* text files. In these we're able to specify a dialogue tree structure, conditional text (and more)

For yarn file syntax, the [official guide](#) is very good. (start with the 'beginner's guide', as recommended there)



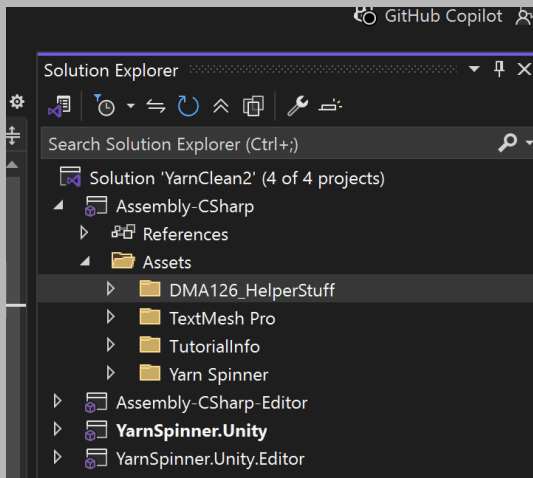
```
BlueNPC.yarn  X
1      title: BlueNPC
2      ---
3      <<if visited(BlueNPC) is true>>
4      Well, if it isn't my number 1 fan!
5      <<endif>>
6      I'm the blue one.
7      ===
8
9      title: BlueNPC_praised
10     ---
11     The one and only!
12     ===
13
14     title: BlueNPC_denigrated
15     ---
16     Hey! Not literally a million. I'm one in a million anyhow
17     ===
```

(This document goes into more detail below)

Easier Editing of Yarn Files in Visual Studio

Mark yarn files visible in Visual Studio

To make yarn files selectable from within Visual studio, you can show hidden files by clicking the 3-rectangle button in the solution explorer (see gif below). You may need to hit '↻' (refresh) if after you've added new files.



Visual Studio Code Extension (personal computers only, probably)

If you're working on a "personal computer" (not one of the lab PCs)

There is an extension with nicer editing within Visual Studio Code (not plain Visual Studio unfortunately)

[Setup instructions are here](#)

(It will probably not work on the lab computers.)

Yarn File Syntax

Nodes

Example: see RedNPC

Each yarn file contains at least 1 node, a named sequence of text. Here is a yarn file with exactly 1 node.

```
title: MyDialogueNode
---
My dialogue text.
This light blue text will be displayed.
The magenta title above can be referred to with jumps and detours (see later)
===
```

Files can contain multiple nodes:

```
title: MyDialogueNode_1
---
Hello! (this text is from node 1)
```

```
===
```

```
title: MyDialogueNode_2
```

```
---
```

```
Hello! (this text is from node 2)
```

```
===
```

NOTE:

Node names can only contain letters, numbers and underscores, and cannot start with numbers.

Comments

You can add c-style comments e.g. `//comment`

```
title: MyDialogueNode
```

```
---
```

```
//Like this!
```

```
Hello!
```

```
===
```

Options

Example: see RedNPC

You can add player choices into the dialog with: `'-> choice text'`

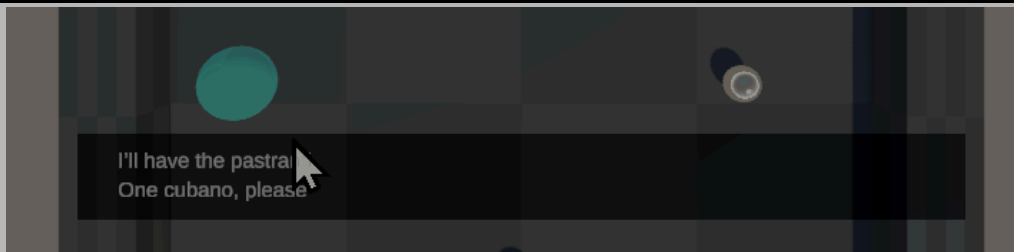
To display some response text with a chosen option, just add and indent after the choice line:

```
-> I'll have the pastrami
```

```
    Coming right up!
```

```
-> One cubano, please
```

```
    No cubanos today, sorry.
```



Jumps/Detours

Example: see RedNPC

You can jump to a particular node by name as follows. (the node can be defined even in another yarn file)
Any text after a jump will be skipped. (e.g. the 'take care won't happen if the 'tell me everything' option is chosen)

```
-> I don't have time to listen to your story
    OK, then leave!
-> I'm sorry, I'm in a bit of hurry
    Ah, sorry to trouble you.
-> Fine, tell me everything
    <<jump TheWholeStory>>
Take care!
```

```
title: TheWholeStory
---
Well, it all started the day I forgot take my ID card, and went to take out the trash.
You see.. Back in those days....
===
```

Alternatively, the detour command can be used if you want to don't want to skip subsequent text

```
-> The sun is very bright today.
    It sure is.
-> That's a strange shaped cloud, isn't it?
    That reminds me of a story.
    <<detour LongStoryAboutClouds>>
Well, I need to be going now.
```

With the detour here, no matter which option you pick, "Well, I need to be going" will always be displayed

Conditionals

Example: see BlueNPC. Has special dialogue if you collect the gold.

You can conditionally display, or exclude text with if statements, similar to other programming languages:

```
<<if visited("BlueNPC") == true>>
You've talked to me at least once before!
<<else>>
This is the first time we're speaking!
<<endif>>
```

To check if a node has been accessed/displayed already you can use `visited("NodeName")`

You can also use `visited_count("NodeName")` to check for a specific number of visits.

```
<<if visited_count("BlueNPC") == 0>>
We're speaking for the first time
<<elseif visited_count("BlueNPC") == 1>>
We're speaking for the 2nd time
<<elseif visited_count("BlueNPC") == 2>>
We're speaking for the 3rd time
<<else>>
We've spoken many times already.
<<endif>>
```

!!! Conditional Options !!!

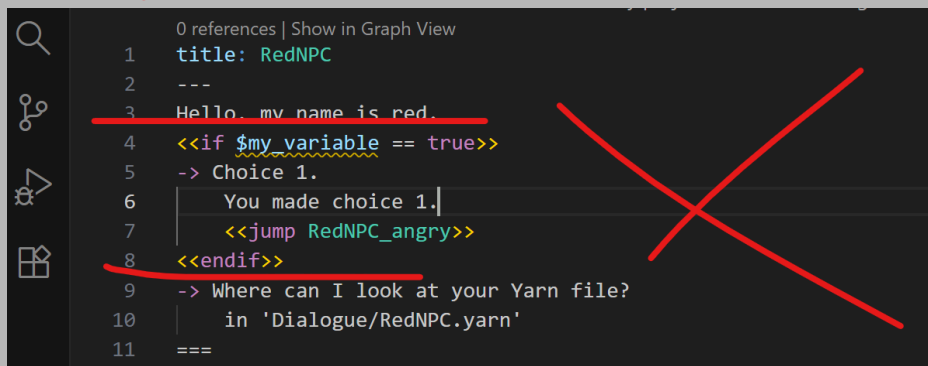
If you want to show/hide particular options (e.g. only offer a secret menu item if the player has collected a member-badge) wrapping the option in 'if' statements like you can with regular text will **NOT** work.

Instead, simply put an 'if' with the condition *AFTER* the choice on the same line:

```
-> put a sticker on the screen <<if $sticker_on == false>>
```

This is from GreenNPC, and only shows different options depending on whether the big sticker is showing on screen.

The following will not work!



```
0 references | Show in Graph View
1 title: RedNPC
2 ---
3 Hello, my name is red.
4 <<if $my_variable == true>>
5 -> Choice 1.
6 | You made choice 1.
7 | <<jump RedNPC_angry>>
8 <<endif>>
9 -> Where can I look at your Yarn file?
10 | in 'Dialogue/RedNPC.yarn'
11 ===
```

Yarn Variables

Please check Yarn Spinner's [official documentation](#) for further info on variables.

How to get/set variables from within a yarn script

Example Usage: WhiteNPC lets you set their name, and reports it back to you.

Variables always start with a '\$' and cannot contain spaces or start with a number.

To set a variable, use a command like the following

```
<<set $sticker_on to true>>
```

To insert a variable into text, simply enclose in curly braces.

```
I guess my name was {$white_npc_name}.
```

How to get/set Yarn variables from within a Unity C# script

To set a yarn variable from a C# script...

You can set numerical values (float), text (string), or true/false (bool)

```
FindObjectOfType<InMemoryVariableStorage>().SetValue("$var_name", 1.5f);
```

```
FindObjectOfType<InMemoryVariableStorage>().SetValue("$var_name", "some text");
```

```
FindObjectOfType<InMemoryVariableStorage>().SetValue("$var_name", true);
```

To get Yarn variable's value from within a Unity C# script

For bools, floats, and string values respectively:

```
bool boolValue = false;  
FindObjectOfType<InMemoryVariableStorage>().TryGetValue("$var_name", out boolValue);
```

```
float floatValue = 0;  
FindObjectOfType<InMemoryVariableStorage>().TryGetValue("$var_name", out floatValue);
```

```
string stringValue = "";  
FindObjectOfType<InMemoryVariableStorage>().TryGetValue("$var_name", out stringValue);
```

You can also put one of 'TryGetValue' inside an if statement if you want to know whether variable is defined or not

```
string stringValue = "";
if (FindObjectOfType<InMemoryVariableStorage>().TryGetValue("$var_name", out stringValue))
{
    //$var_name has been set at least once
}
else
{
    //$var_name is not yet set
}
```

Inline Commands

Example: GreenNPC uses multiple commands to alter the game world.

For the jam today, we've provided a drag and drop solution for executing changes in Unity from within a yarn script, but scripting yourself is pretty straightforward. Check the [official documentation](#) for details.

'wait' command

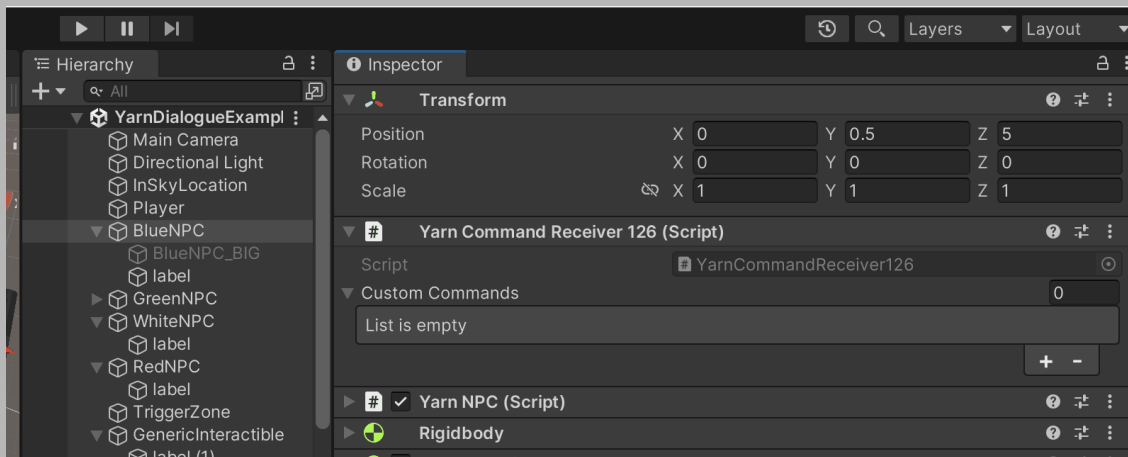
You can pause the display of text temporarily like so:

```
<<wait 2>>
```

Where '2' can be the number of seconds you want to pause.

Adding a callable command

1. Add a [YarnCommandReceiver126](#) to the object you want to receive the command.
2. Click the '+' to add a custom command
3. Set its Command ID to something short, but descriptive (do not use spaces, instead use "camelCase" or "under_score")
4. Populate the Action with what you want to happen. (you can act on any object in the scene!)



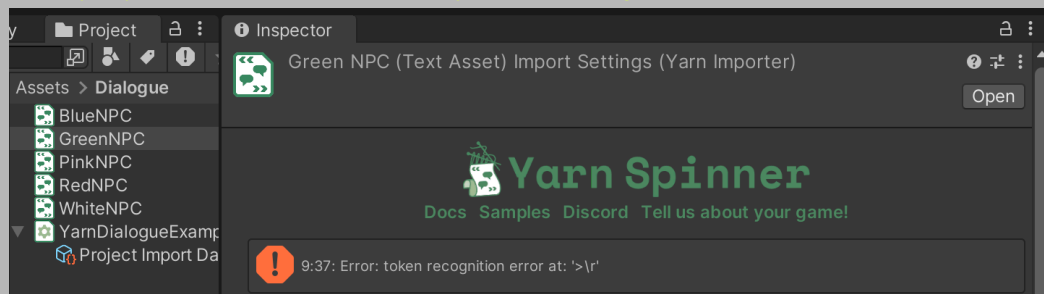
Calling a command

To call a command, use a line like the following. Where 'BlueNPC' is the name of the game object with the command, and 'makeBig' is the command you want to execute.

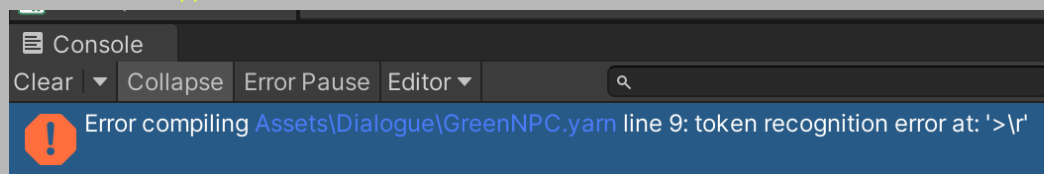
```
<<DoCustomCommand126 BlueNPC makeBig>>
```

Troubleshooting

- Select your yarn script and see if there's any helpful message in the inspector



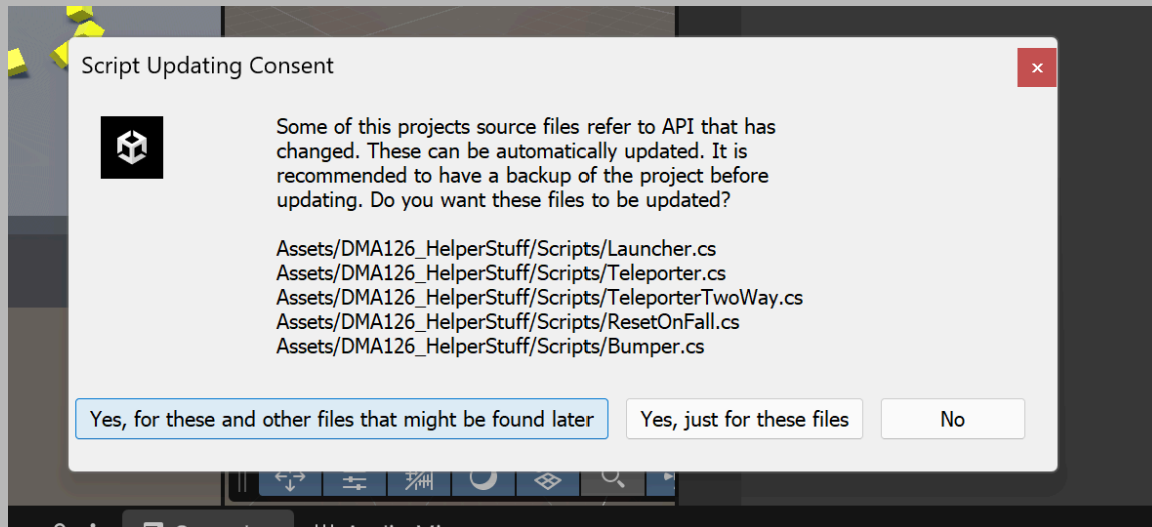
Errors will also appear in the console:



- Special characters and spaces can cause problems in variable and command names.
- If statements, commands need enclosed in double '<<' '>>'

FOR ANYONE USING UNITY 6

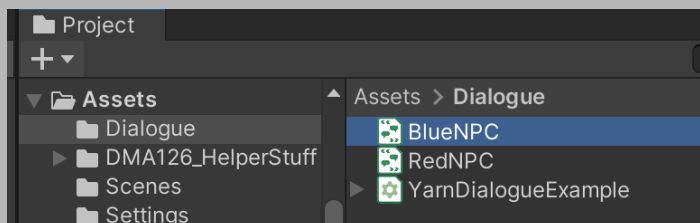
Please choose 'Yes, for these...' when this menu comes up.



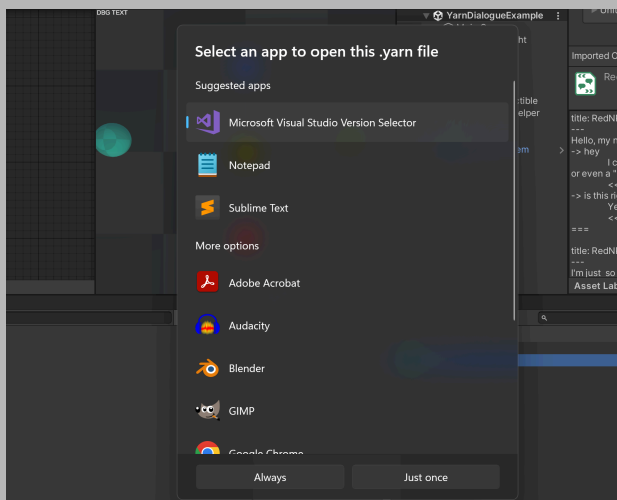
Exercises

Excercise 1: Modify an existing NPC's dialogue

- Double click on Dialogue/Blue Npc



- Open in a Visual Studio, or whatever text editor you prefer.



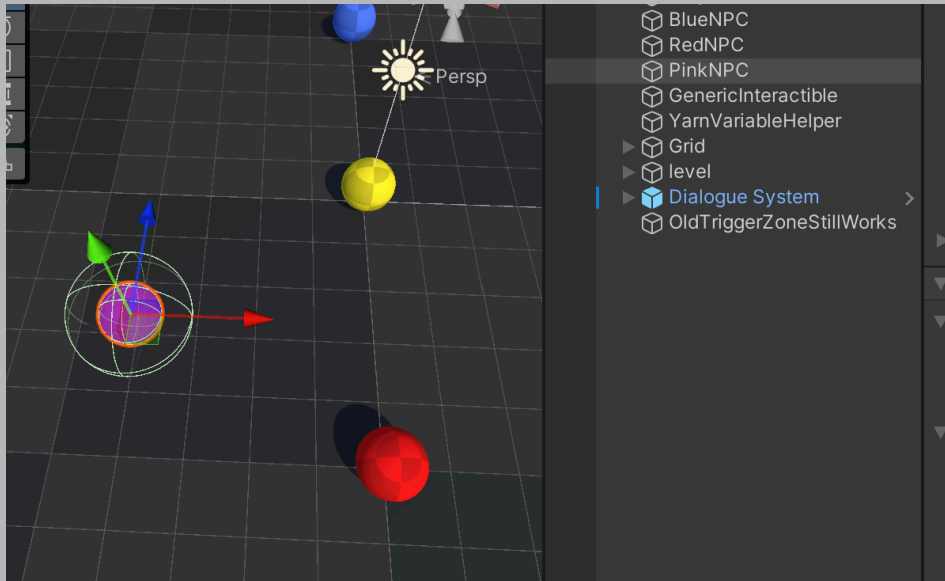
**there is a fancier plug-in for Visual Studio Code specifically for editing these files, but the instructor will stick to a plain text editor for today]*

- Try altering the BlueNPC's text to include some choices and jumps to other nodes

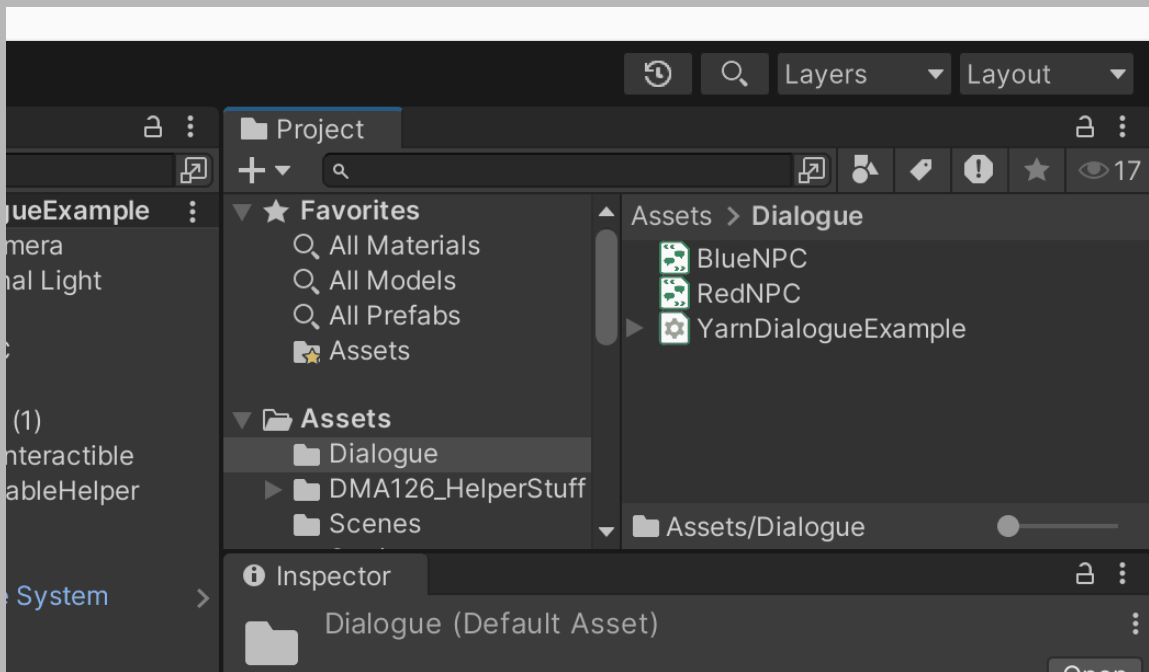
```
BlueNPC.yarn
1      title: BlueNPC
2      ---
3      I'm the blue one.
4
5      -> What, THE blue one???
6          <<jump BlueNPC_praised>>
7      -> Eh, I've seen a million blue ones.
8          <<jump BlueNPC_denigrated>>
9      ===
10
11     title: BlueNPC_praised
12     ---
13     The one and only!
14     ===
15
16     title: BlueNPC_denigrated
17     ---
18     Hey! Not literally a million. I'm one in a million anyhow
19     ===
```

Exercise 2: Create a new NPC

- Duplicate RedNPC, move and assign a different material



- In the 'Dialogue' folder, right click and choose Create > Yarn Spinner > Yarn Script, and give a name for your new NPC



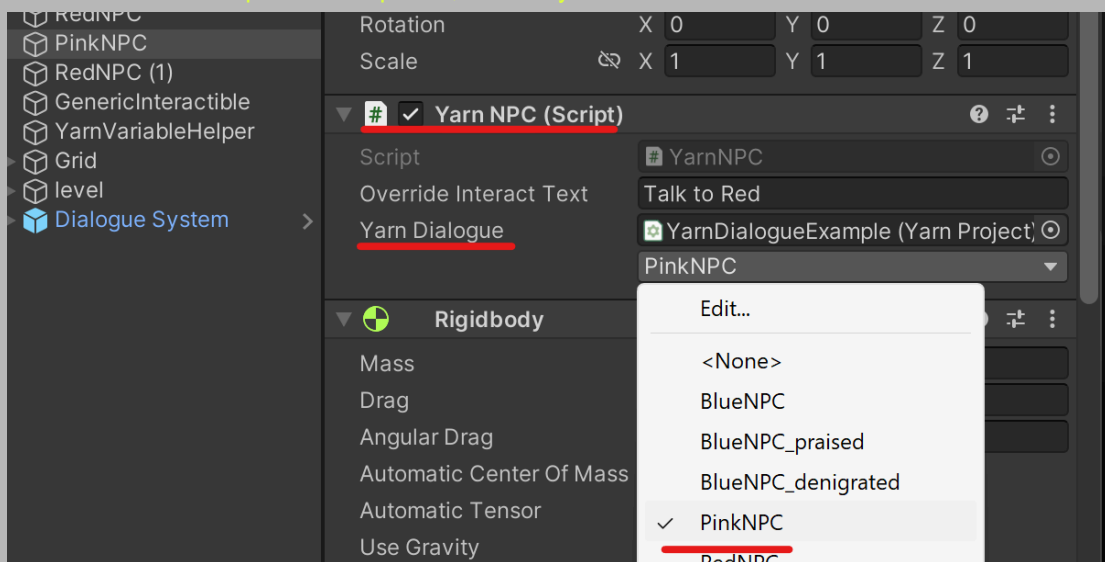
- open the yarn script and, and write some dialogue for your new NPC (don't forget to save!)

```

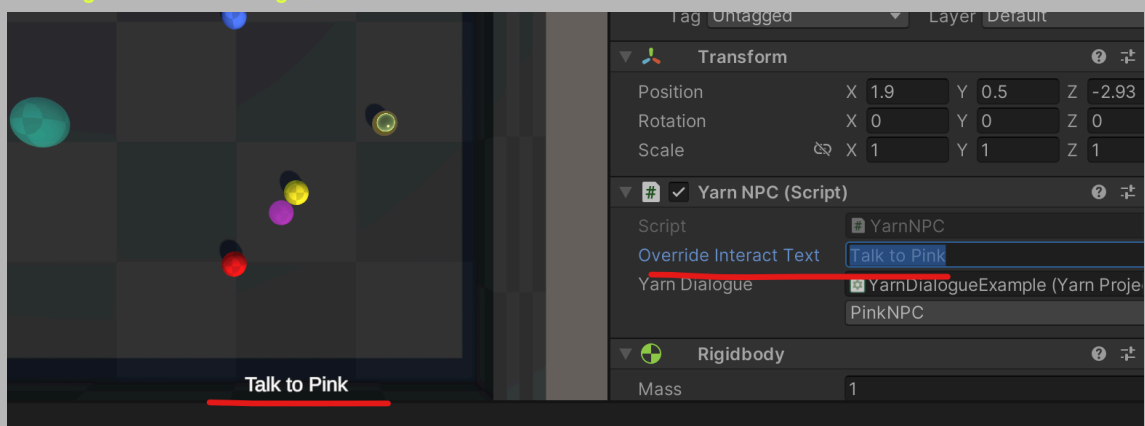
1  title: PinkNPC
2  ---
3  I'm pink, but not by choice.
4  -> I'm the player, but I feel like an NPC
5  -> What's wrong with pink?
6  What? sorry, I spaced out and wasn't listening. Try someone else.
7  ===
8

```

- On the YarnNPC script in the inspector, choose you new node title.



- You might want to change, or make blank the 'Override Interact Text'



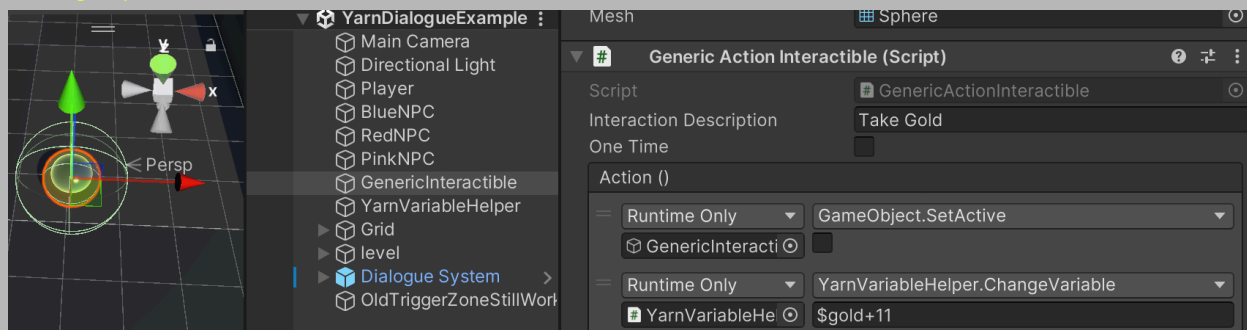
- Hit play, and make sure your new NPC says the correct dialogue.

Other Scripts

Generic Action Interactable

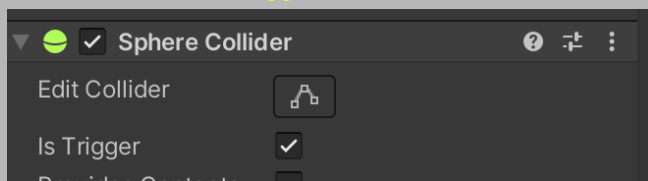
GenericActionInteractable can be attached to an object with a trigger collider to perform generic actions, similar to the TriggerZone.

The gold looking sphere pictured here is an example of a way you can implement a collectible item (in this case, gold)



How to use:

- Create a sphere (or other shape)
- Set its collider to 'Is Trigger'



- Add a **GenericActionInteractable** script
- Set up its action to do your desired command (instructions similar to those for a [trigger zone](#) or a [UI button](#))

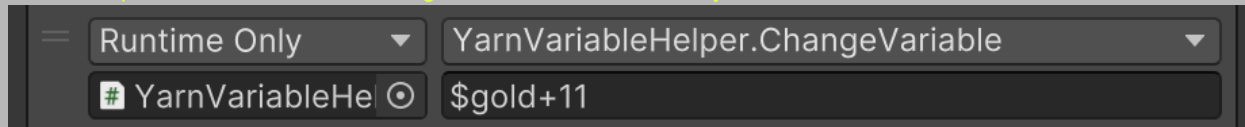
YarnVariableHelper.cs

To change a yarn variable via a trigger zone, UI button, or Generic Action Interactable, use the YarnVariableHelper game object in the scene, and call the 'ChangeVariable' function

In the textbox, simply put something like `$variable_name+number`

1. the name of the variable you want to modify (prefixed with a \$),
2. '+' to increment, '-' to decrement, or '=' to set
3. The increment, decrement or set value

The example adds '11' to variable '\$gold', which is checked by the blue NPC



The helper scripts from roll-a-ball

The scripts in [DMA126 HelperStuff](#) should generally work as they did in roll-a-ball, except for Bumper, ForceField, and Launcher, but you should focus on dialogue for the jam today.