Documentation

Team

Chris Gregan
Founder & Developer

John OKane
Founder & Developer

Ronan Pearce
Founder & Developer

Contact

Inquiries: chris@fungusgames.com

Twitter: twitter.com/fungus_games

Facebook: facebook.com/fungusgames

Web: fungusgames.com

Welcome

Fungus is a free open source tool for making storytelling games in Unity 3D. This documentation will help you learn how to use the storytelling features in Fungus & Unity 3D.

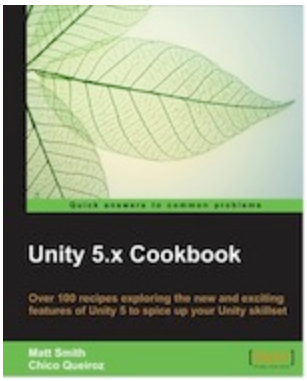
Features

Fungus allows you to easily add storytelling features to your Unity games with no coding, via an intuitive visual scripting system. Join a global community of artists & writers making amazing games.

- Flowchart-based character dialogue
- Suitable for visual novels, RPGs, hidden object and puzzle Games and interactive fiction.
- Internationalization of your character dialogue.
- Easy control of sprites, camera and audio to help tell your story
- Works for both 2D and 3D Unity games
- Integrates easily with other Unity code and simple to extend.
- 100% free and open source

Credits

Dr. Matt Smith has kindly contributed this documentation to the Fungus project. If you're new to Unity development then a great source of information, ideas and sample projects is the Unity 5.x Cookbook, which Matt and co-author Chico Queiroz have re-written from scratch for Unity 5 (published by Packt Publishing):



Matt is a computing academic, specialising in interactive multimedia systems, including computer games, web applications and mobile apps. He is the main maintainer of the Fungus documentation and is Senior Lecturer in Computing at the Institute of Technology (ITB), Blanchardstown, Dublin, Ireland.

Get Support

If you are having an issue with Fungus, please let us know on the forum and we'll do our best to help.

Contribute

We welcome all sorts of contributions to the project. You can report bugs, suggest new features or even contribute source code changes.

- Report bugs: GitHub issue tracker
- Feature roadmap: Trello roadmap
- Source Code: GitHub code repository

License

The Fungus software is licensed under the Open Source MIT License.
The Fungus documentation is licensed under the Creative Commons Attribution-NonCommercial License.

Quickstart

Follow these steps to get up and started with Fungus quickly. Then learn more about what Fungus can do and how to do it from the other documentation pages and videos.

Installing Unity

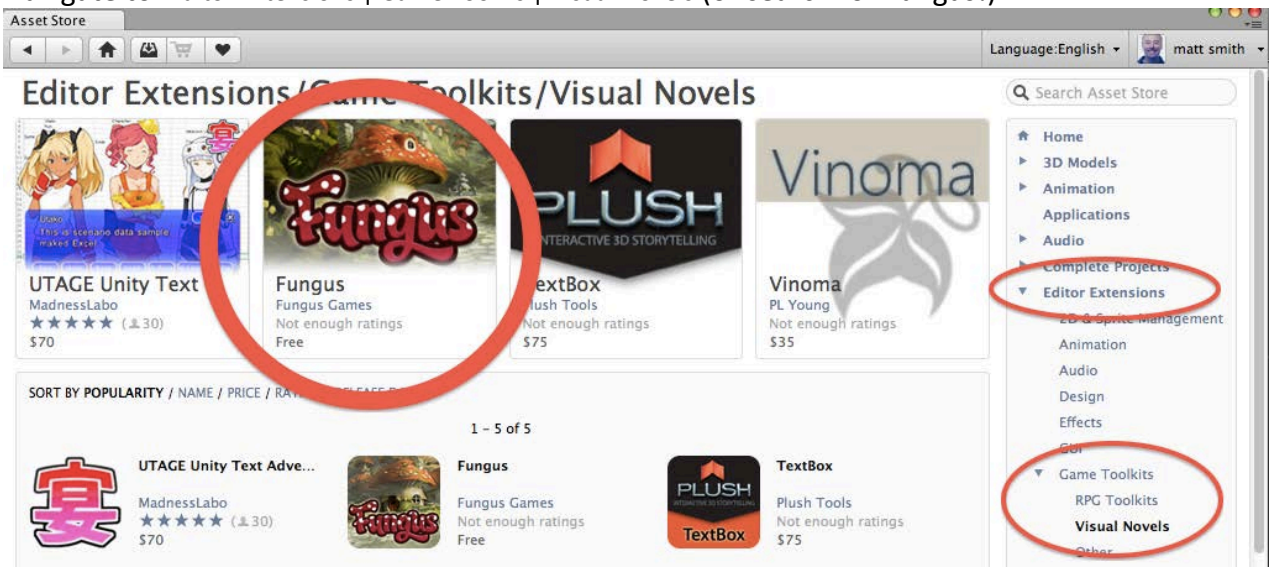
Download and install the latest version of Unity from the Unity3D.com website.

Note, the download page is usually located at: Unity3D.com/get-unity

Installing Fungus - from Unity Asset store

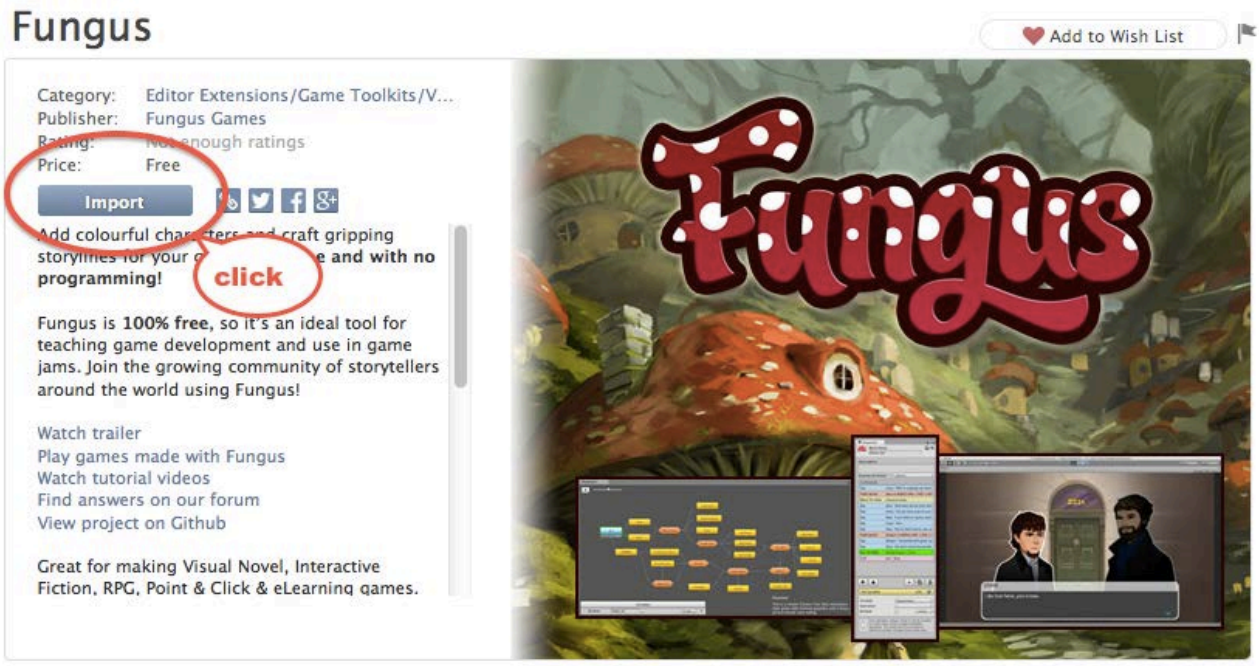
Follow these steps to quickly install Fungus for a Unity project via the Unity Asset Store:

1. Create a new Unity 2D project.
2. Open the Asset Store window.
3. Navigate to: Editor Extensions | Game Toolkis | Visual Novels (or search for Fungus!).



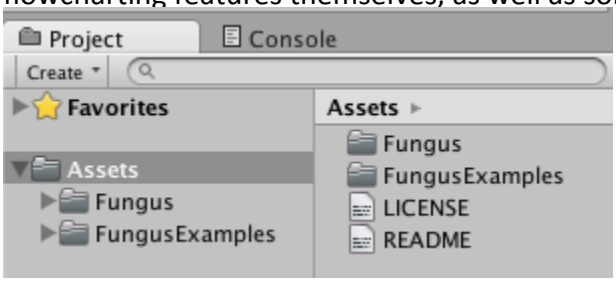
4. On the Fungus details page click the Import button,
   (the first time you download Fungus you'll be asked to agree to the MIT open source

licence):



5. Choose to import all contents from this package.

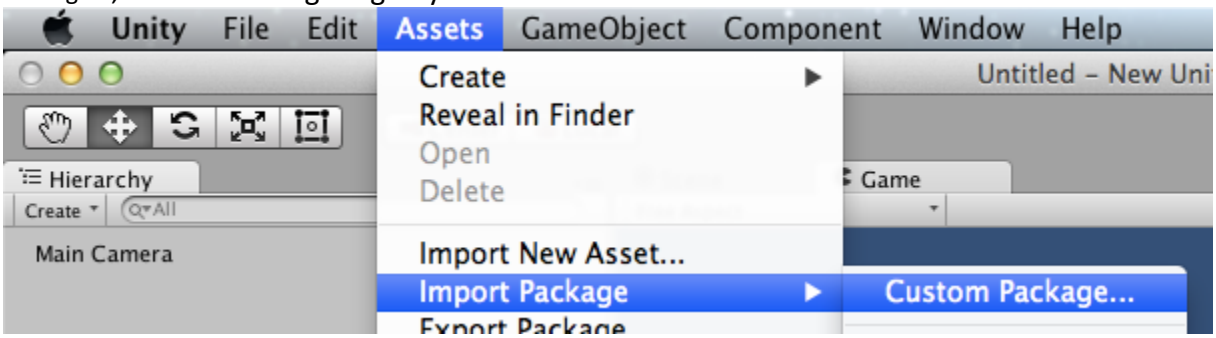
You should now see a Fungus folder in your Project panel, containing both the Fungus flowcharting features themselves, as well as some sample scenes to get you started:

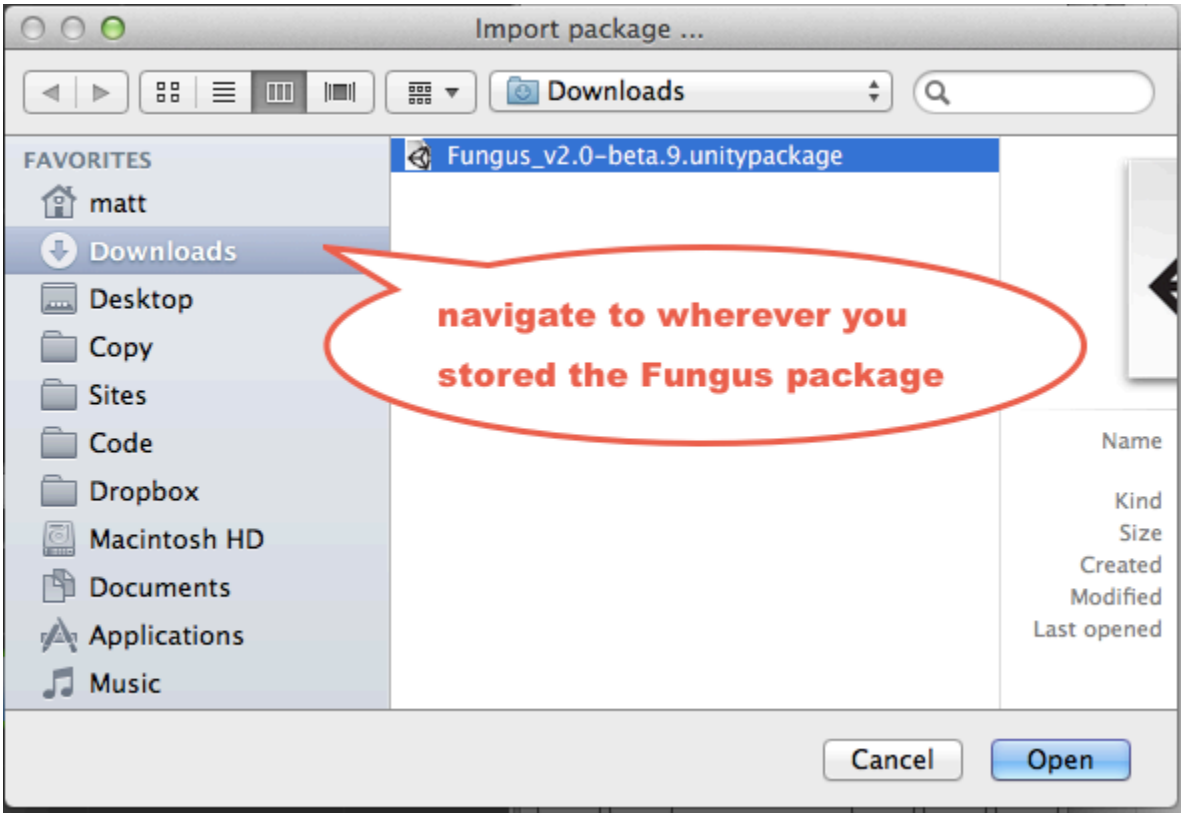


Installing Fungus - from Fungus package

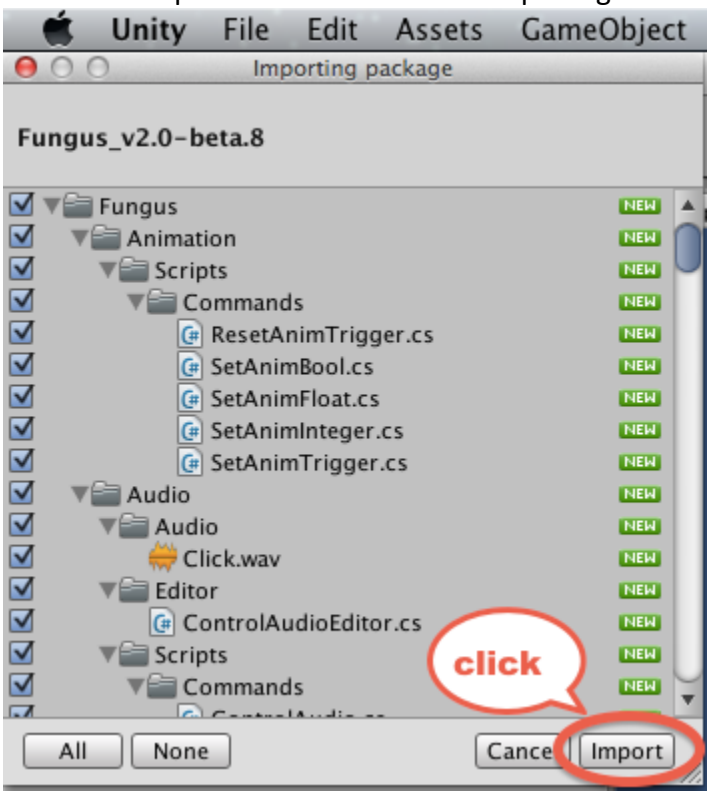
Follow these steps to quickly install Fungus for a Unity project:

1. Download the latest version of the Fungus unitypackage from the FungusGames.com website.
2. Create a new Unity 2D project.
3. Import the Fungus unitypackage by choosing menu: Assets | Import Package | Custom Package..., and then navigating to your downloaded file location.

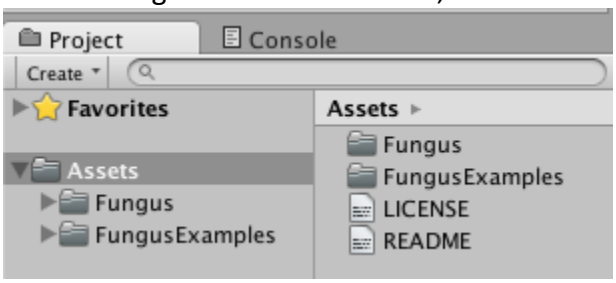




4. Choose to import all contents from this package.



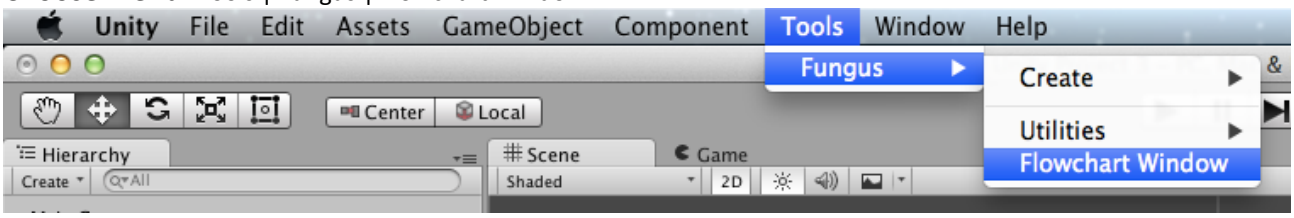
You should now see a Fungus folder in your Project panel, containing both the Fungus flowcharting features themselves, as well as some sample scenes to get you started:



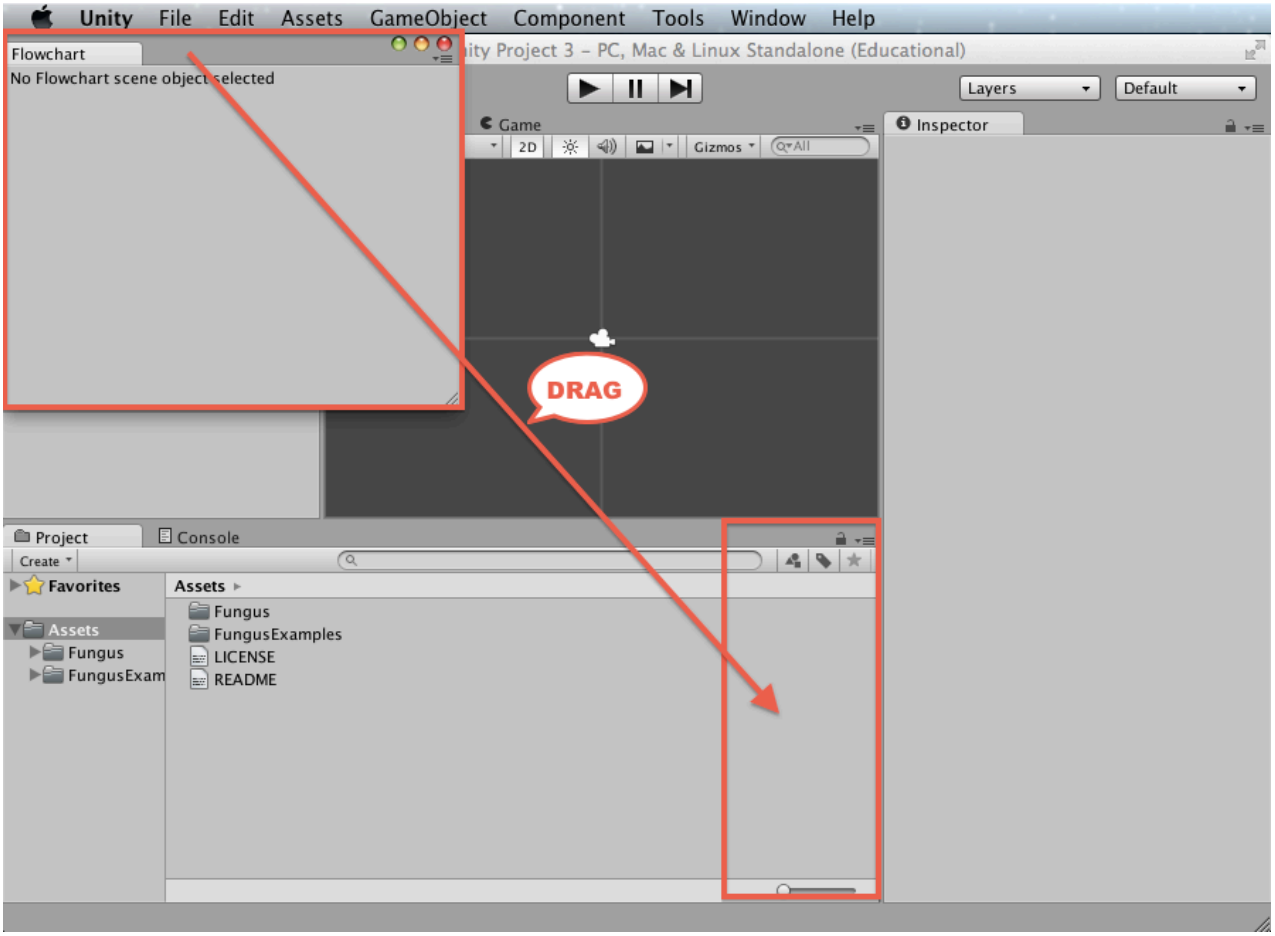
Opening and docking the Flowchart window

You'll need the Fungus Flowchart window when working with Fungus. Open and dock this window somewhere handy by following these steps:

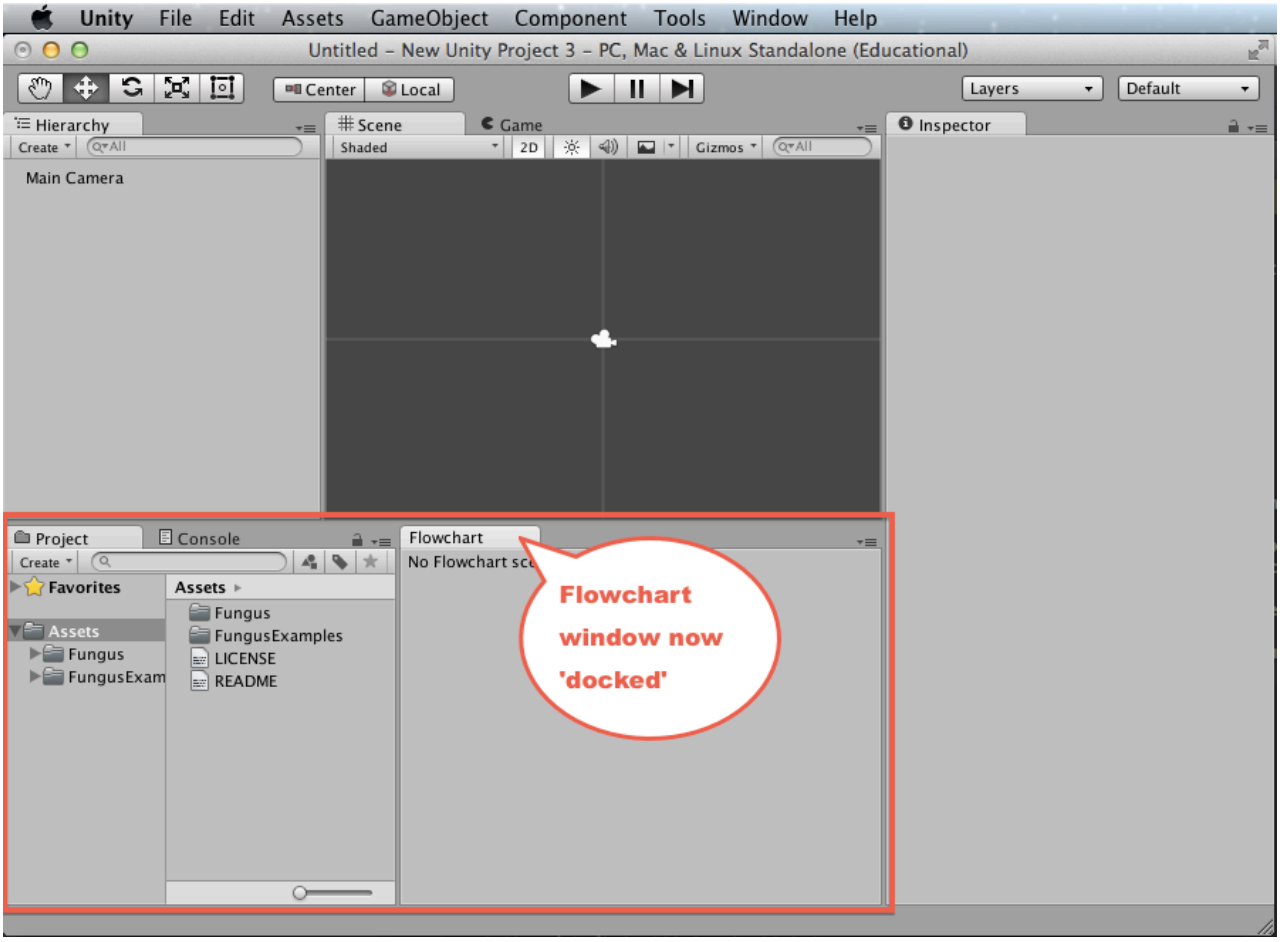
1. Choose menu: Tools | Fungus | Flowchart Window



2. Drag-and-drop the Flowchart window to the location you wish to dock it:



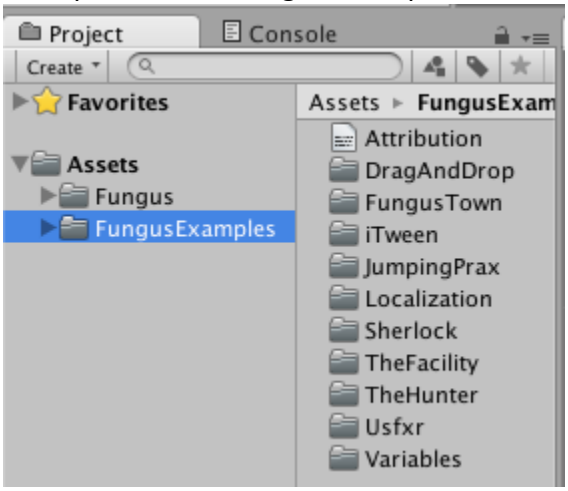
3. The Flowchart window is now docked and part of your Unity window layout:



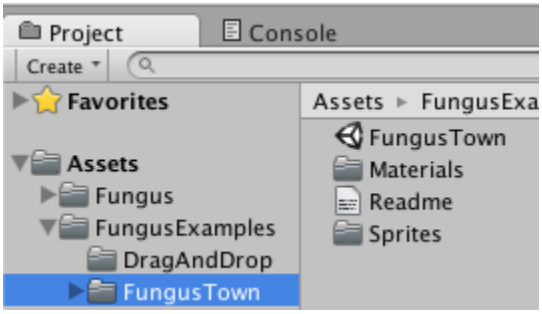
Finding the example folders and scene files

Two folders are created when you install Fungus, the Fungus features themslves (in folder 'Fungus') and a set of examples (in folder 'FungusExamples').

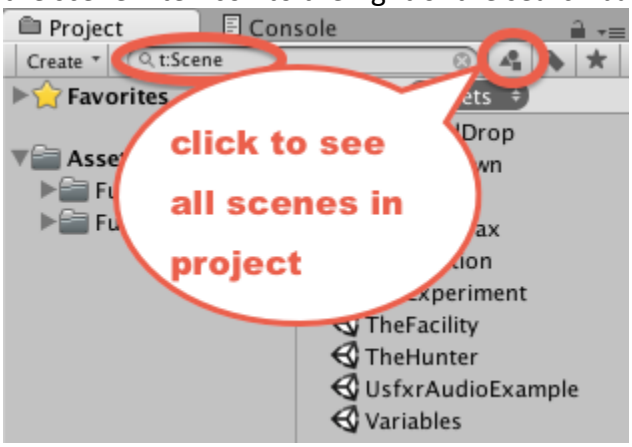
Examples include Drag and Drop, Sherlock and Fungus Town:



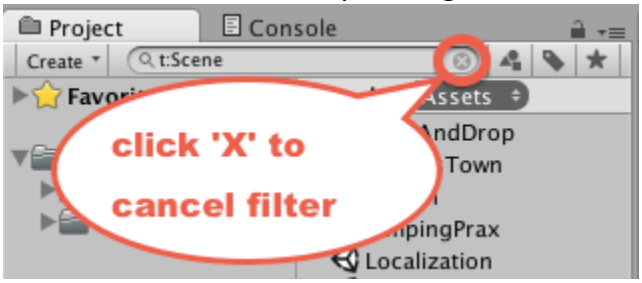
You can use the left-hand side of the Unity Project window to explore each example folder:



Alternatively, you can 'filter' the Project view to show all scenes (and no other files) by clicking the scene filter icon to the right of the search bar:

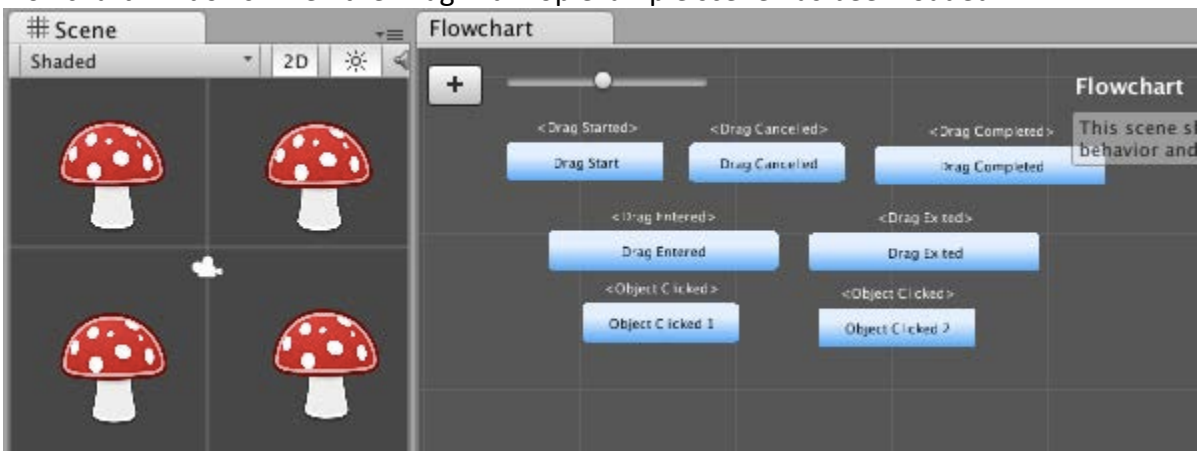


You can cancel the filter by clicking the 'x' in the search bar:

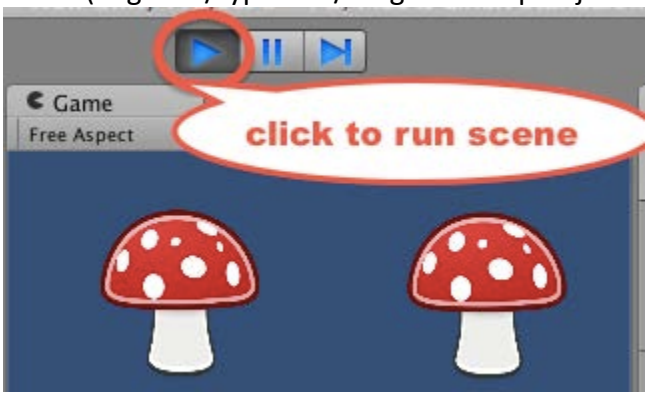


Loading and playing the example scenes

To **load** an example scene, double click the desired example's scene object in the Project window, and the scene should load. For example, this screenshot shows the scene and Flowchart windows when the DragAndDrop example scene has been loaded:



To **run** the currently loaded scene (i.e. to enter **Play-mode**), click the Unity 'play' triangle button at the center top of the Unity application window, and then do whatever makes sense in that scene (e..g click/type text/drag-and-drop objects etc.!):



Note: you click the 'play' button a second time to end **Play-mode**.

Changes made during playmode don't persist

As with all Unity projects, you can **change** the properties of gameObjects while a scene is running, but these changes are 'ephemeral' - they only last while the scene is running. As soon as you end play mode the properties of all objects in the Hierarchy will revert to those saved in the Scene file.

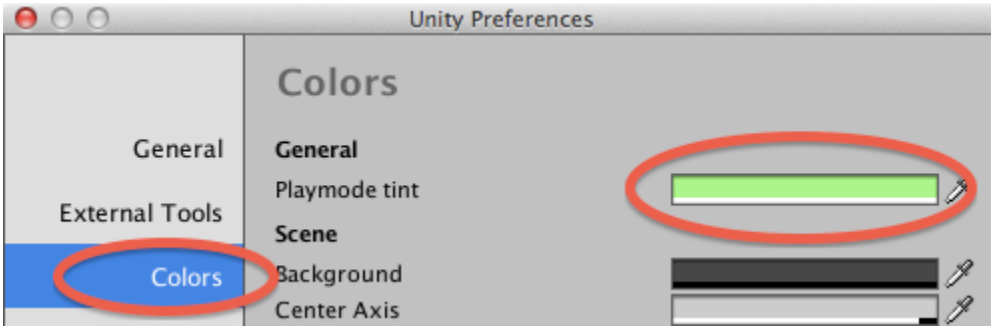
This makes it easy to 'tweak' values of objects in **Play-mode**, and then when the desired behaviour is achieved, those values can be set for the saved scene properties.

Values set when Unity is in **Edit-mode** will be saved when you saved your scene (CTRL-S / Command-S, or menu: File | Save Scene).

Change your preferences to highlight Play-mode

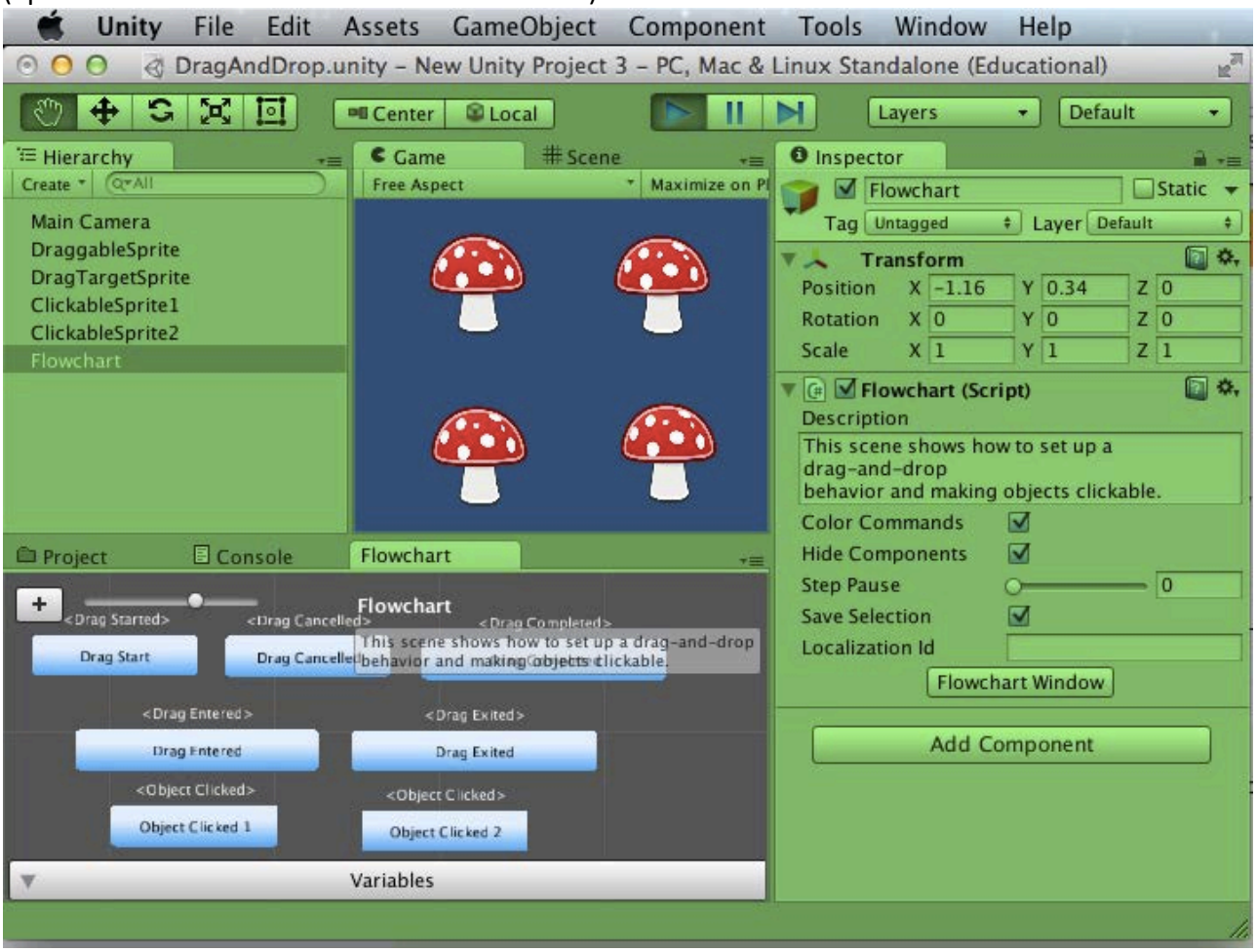
Sometimes we can forget we are in Unity **Play-mode**, and then make changes to Hierarchy gameObject values that are then 'fogotton' when we do stop playing the scene. A good way to avoid this problem is to to set a 'tint' to the Unity editor to make it visually very clear to us when we are in **Play-mode**. To add a tint to **Play-mode** do the following:

1. Open the Unity preferences dialog by choosing menu: File | Preferences ...
2. Select the Colors preferences, and choose a light colored tint (we chose a light green in this case):



3. Close the dialog (changes are saved automatically).

4. When you next enter **Play-mode** you'll see most of the Unity Editor windows turn green (apart from the Game and Flowchart windows):

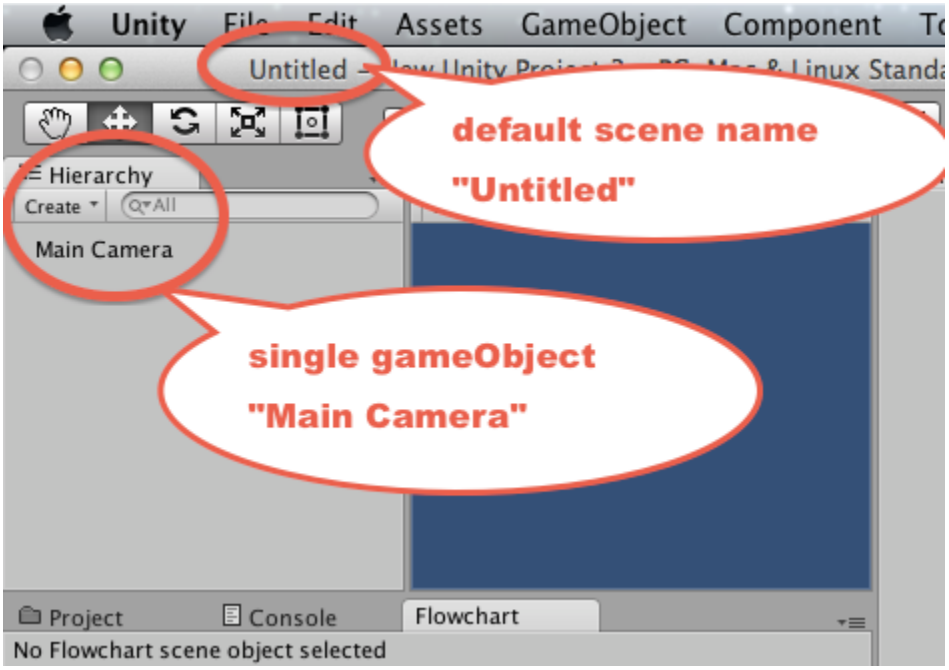


Creating, naming and saving a new scene from scratch

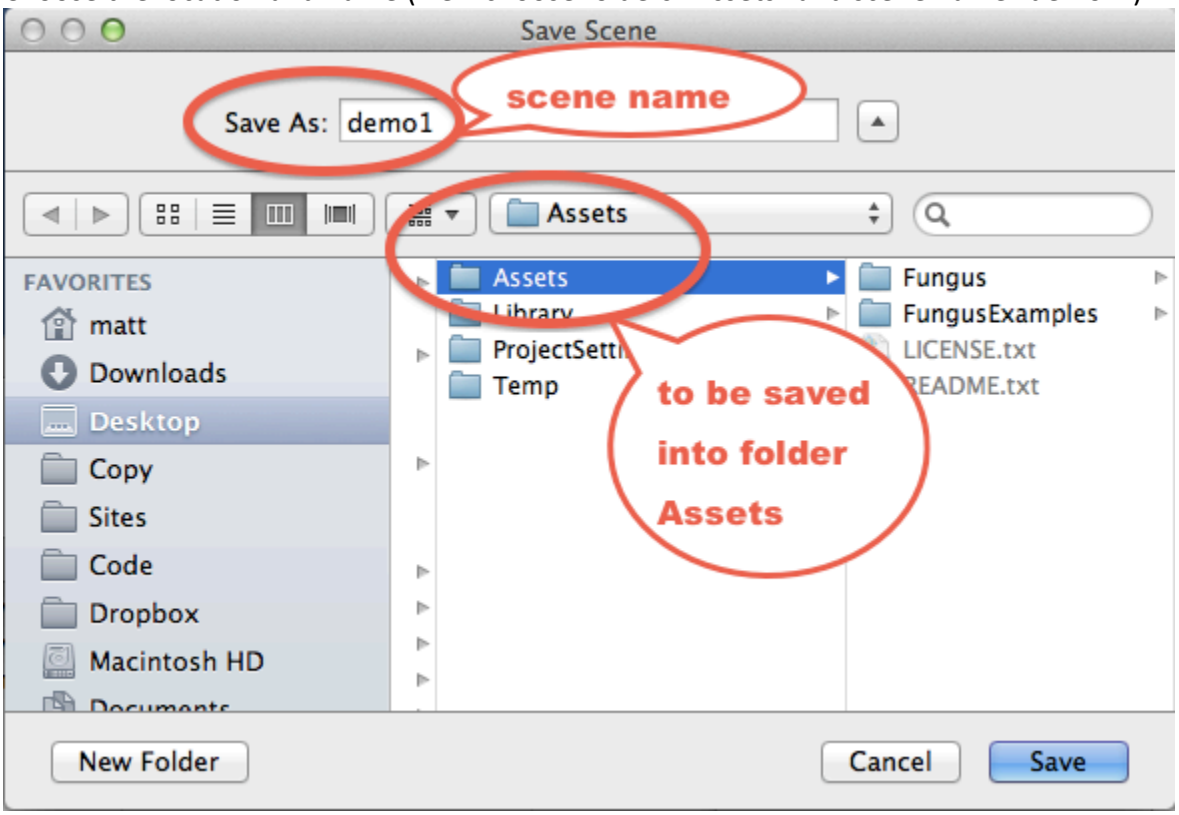
To create a new scene in Unity do the following:

1. Choose menu: File | New Scene
2. Note: if you have any unsaved changes for the current scene you need to either save or abandon them before a new scene can be created.
3. You should now have a shiny new scene, with a Hierarchy containing just one gameObject, a Main Camera. The new scene will have been give the default name

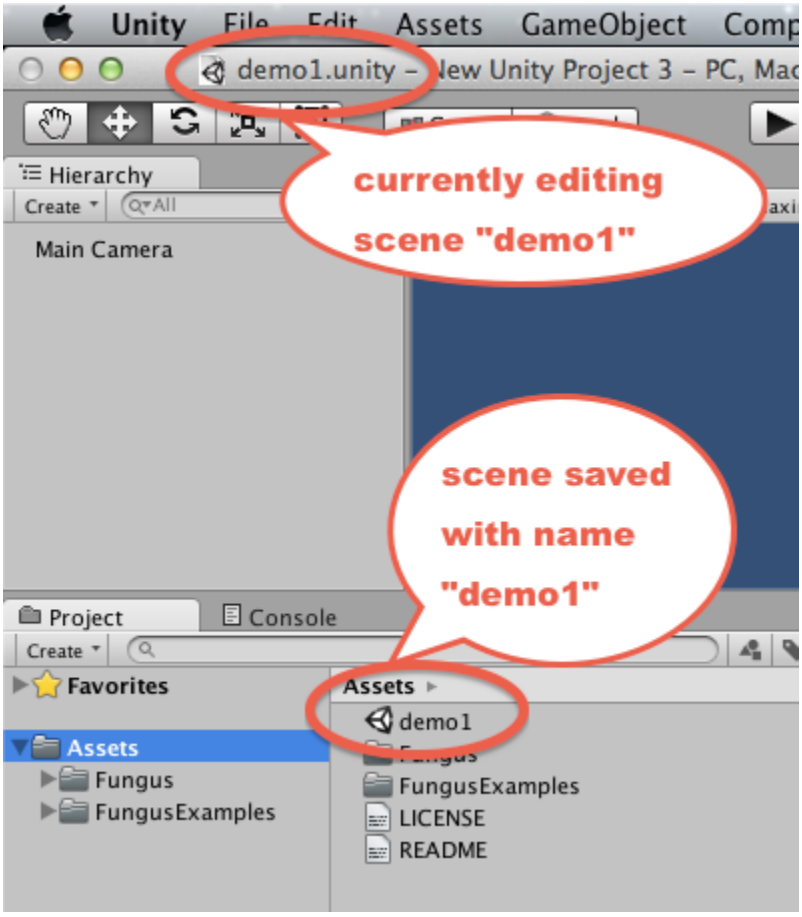
"Untitled", which you can see in the title of the Application window:



4. Good practice is to save your scene (in the right place, with the right name), before creating your work in the scene. Let's save this scene in the root of our project "Assets" folder, naming it "demo1". First choose menu: File | Save Scene As...
5. Choose the location and name (we'll choose folders "Assets" and scene name "demo1"):



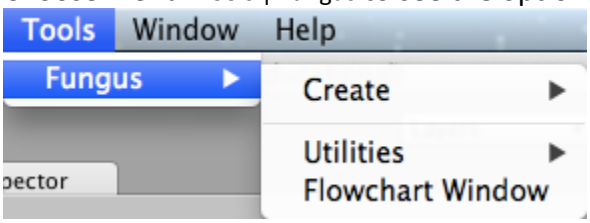
6. Once you have successfully saved the scene you should now see the new scene file "demo1" in your Assets folder in the Project window, and you should also see in the Application window title that you are currently editing the scene named "demo1":



Menu: Tools | Fungus

The core Fungus operations are available from the Unity Tools menu.

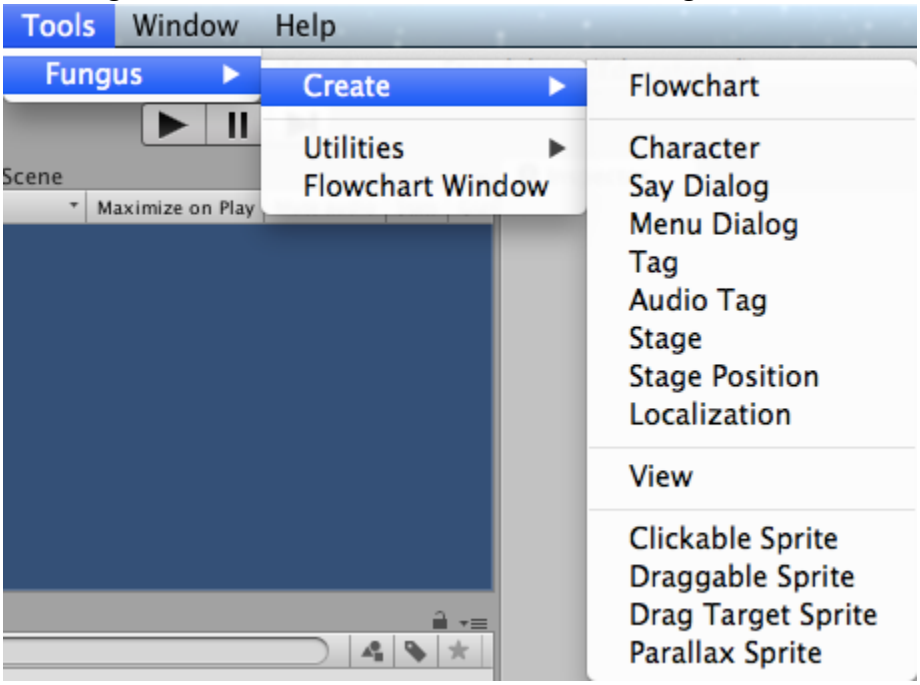
Choose menu: Tools | Fungus to see the options available:



As can be seen, there are 2 submenus, Create and Utilities, plus the Flowchart Window action (which reveals the window if already open, or opens a new window if the Flowchart window was not previously opened).

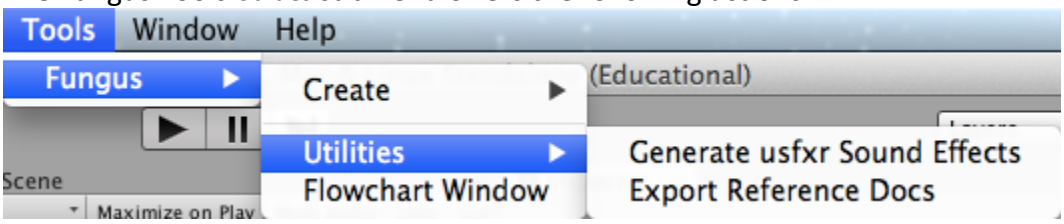
Menu: Tools | Fungus | Create

The Fungus Tools Create submenu offers the following actions:



Menu: Tools | Fungus | Utilities

The Fungus Tools Utilties submenu offers the following actions:

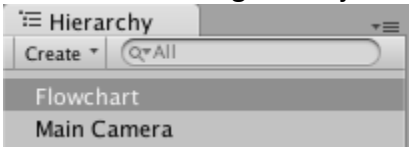


Create a Flowchart

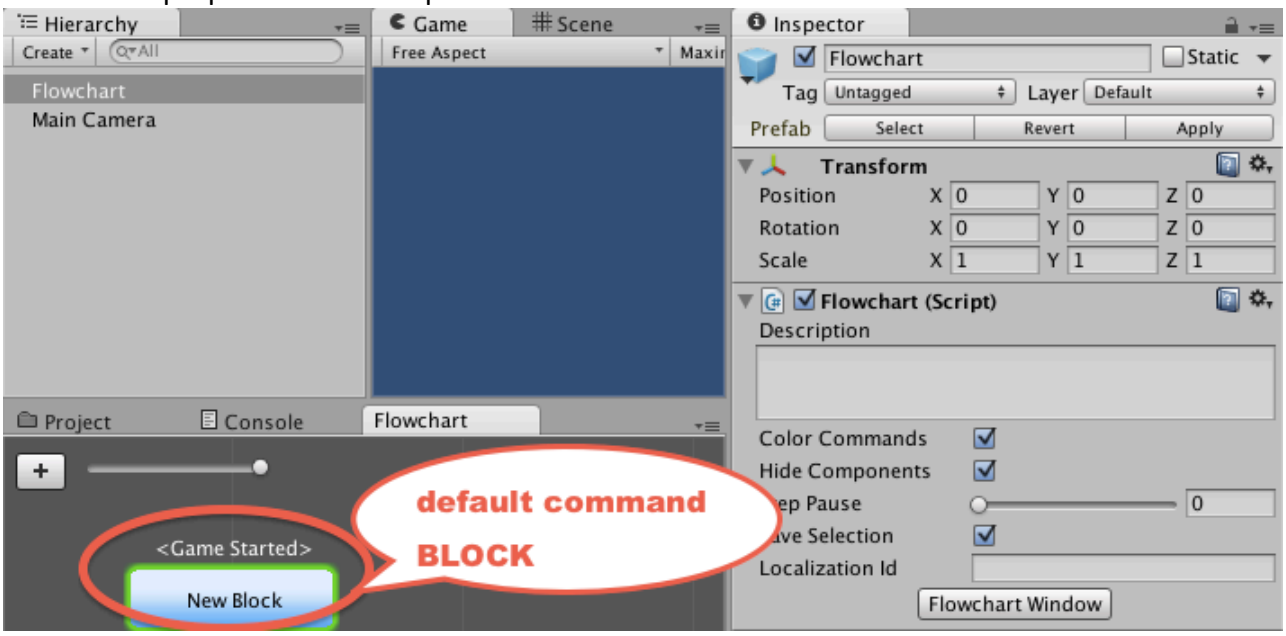
To create a Fungus Flowchart do the following:

1. Choose menu: Tools | Fungus | Create Flowchart

2. A new **Flowchart** gameObject should appear in the Hierarchy window.



3. Select the **Flowchart** gameObject in the Hierarchy window, and you'll see the **Flowchart's** properties in the Inspector Window:

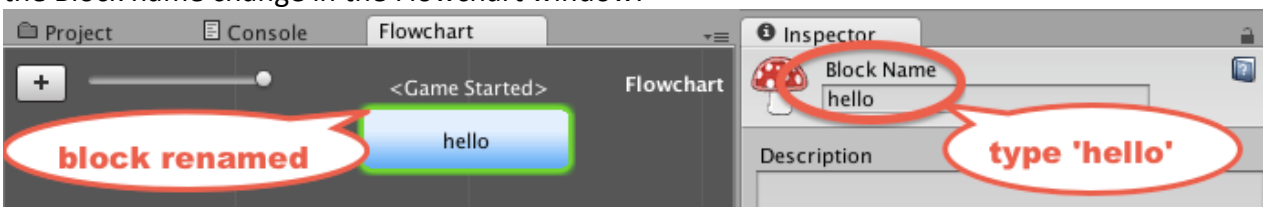


4. If you have not already displayed the Flowchart Window, you can do so by clicking the Flowchart Window button in the Inspector.
5. As you can see, when a new Flowchat is created a single command Block named "New Block" is automatically created, with the Event handler "Game Started" (so it will start executing Fungus commands as soon as the scene goes into **Play Mode**).

Flowchart Block property viewing and editing

Let's change the name of the default command Block of a new Flowchart in the Flowchart window to "hello". Do the following:

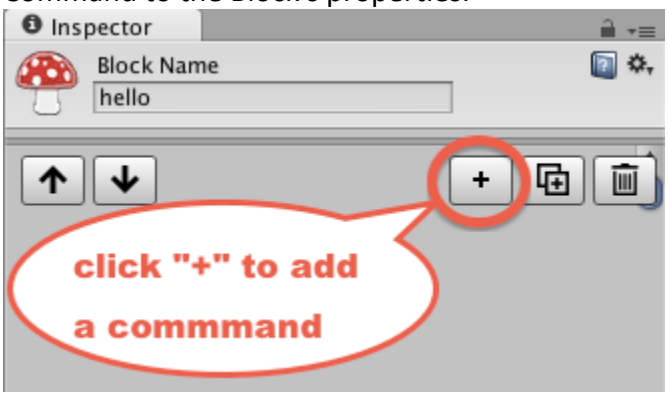
1. Create a new Fungus Flowchart (if you haven't already done so).
2. Click to select the Block in the Flowchart window (when multiple blocks are present, the selected one gets a green highlight border).
3. In the Inspector change the text for the Block Name property to "hello". You should see the Block name change in the Flowchart window:



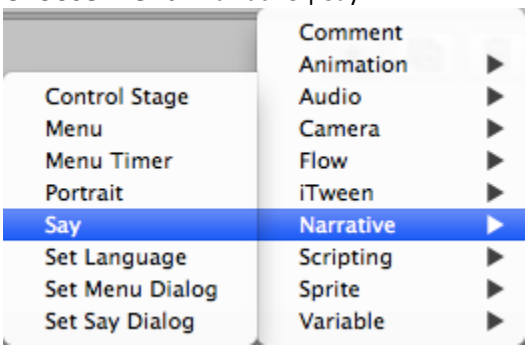
Add a Say command

To add a "Say" command to a Block do the following:

1. (setup) Create a new scene, add a Fungus Flowchart to the scene.
2. Ensure the Block is selected, and you can see its properties in the Inspector, and ensure the name of the Block is "hello".
3. Click the Plus button in the bottom half of the Inspector window, to add a new Command to the Block's properties:

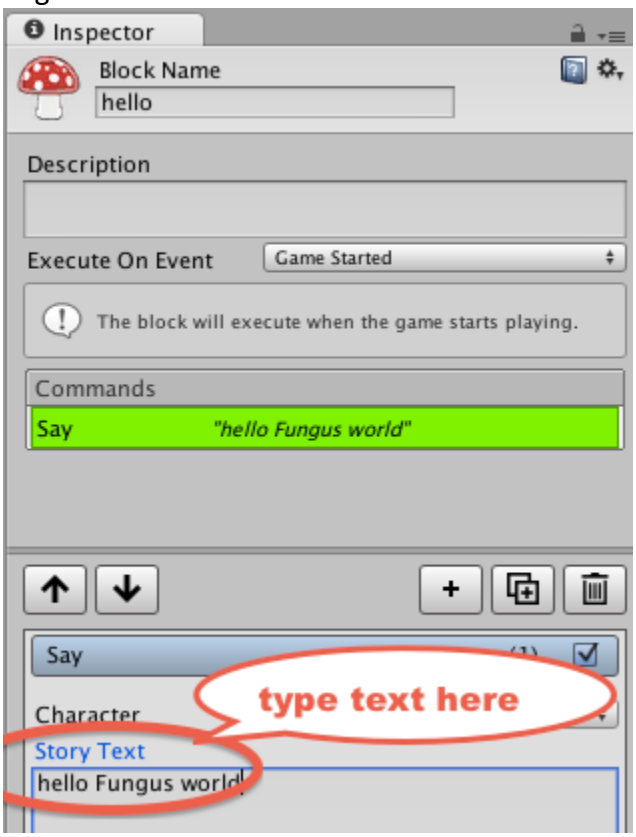


4. Choose menu: Narrative | Say:



5. Since this Block only has one Command, that command is automatically selected (shown with a green highlight).

6. In the "Story Text" textbox in the bottom half of the Inspector window type in "hello Fugus world":



7. Run the scene, and see Fungus create a dialog window, and output the text contents of your Say command:

Next Steps

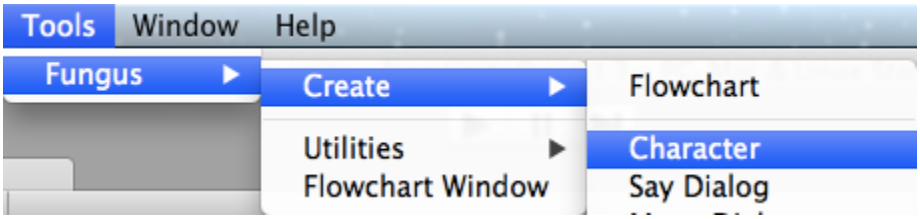
Having got up and running, here are some next steps to get to know what you can do with Fungus.

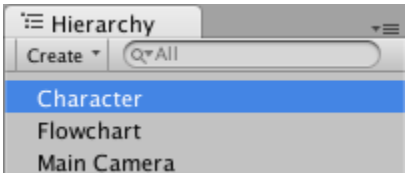
Adding Characters, for use in Say commands

We can associated words spoken by the Say Command with a particular Character. Consider the following Tom and Jerry scene:

[Tom] Where is that mouse?
[Jerry] Where is that cat?
[Tom] Aha...
[Jerry] Arrrrgggggggg!!!!!!!

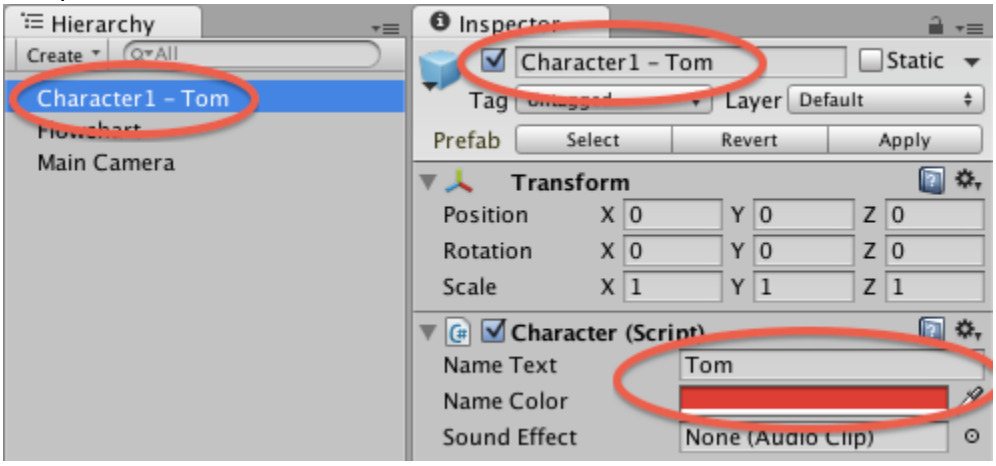
To implement the above in Fungus we need to create and name two Characters. Do the following:

1. (setup) Create a new scene, add a Fungus Flowchart to the scene, and select the Block in the Flowchart.
2. Rename the Flowchart Block "cat and mouse".
3. Choose menu:
   Tools | Fungus | Create | Character:
   

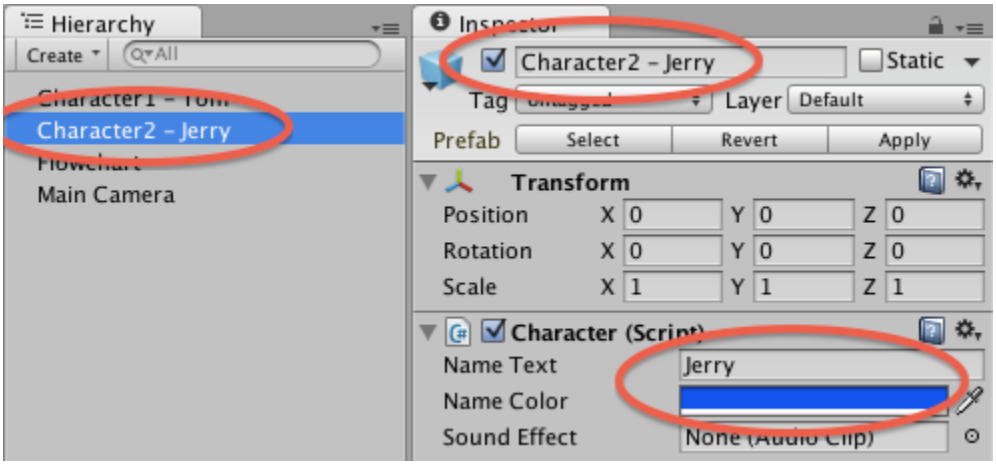
4. You should now see a new gameObject "Character" in the Hierarchy window, named Character.
   

5. Ensure gameObject "Character" is selected, and edit its properties in the Inspector. Rename the gameObject to "Character1 - Tom", then in its Character (Script)

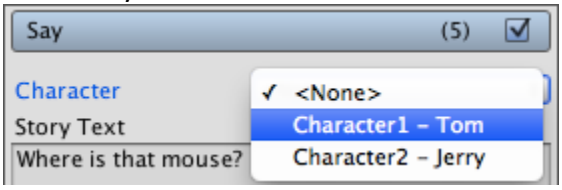
component set the Name Text to "Tom" and the Name Color to red:



6. Repeat the previous two steps to create a second character "Character2 - Jerry", then in its Character (Script) component set the Name Text to "Jerry" and the Name Color to blue:



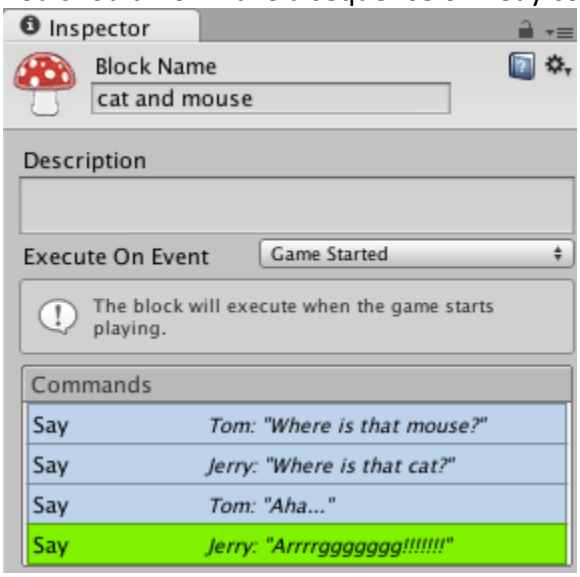
7. Now we have our two character gameObjects, we can assign them to any Say commands as appropriate.
8. Create a Say Command for Tom, with text "Where is that mouse?", setting the Character of this Say command to "Character1 - Tom":



9. Repeat the above step for the 3 remaining statements, for:
    - Jerry "Where is that cat?"
    - Tom "Aha..."

- Jerry "Arrrrgggggggg!!!!!!!"
  Assigning the appropriate Character for each Say Command from the menu of Character gameObjects in the Hiearchy.

10. You should now have a sequence of 4 Say commands in your Block:



11. When you run the scene you should see a sequence of statements, clearly showing who is saying what - both the character name is given, and also that name is coloured according to the properties we set for the character gameObjects (red for Tom, and blue

for Jerry):

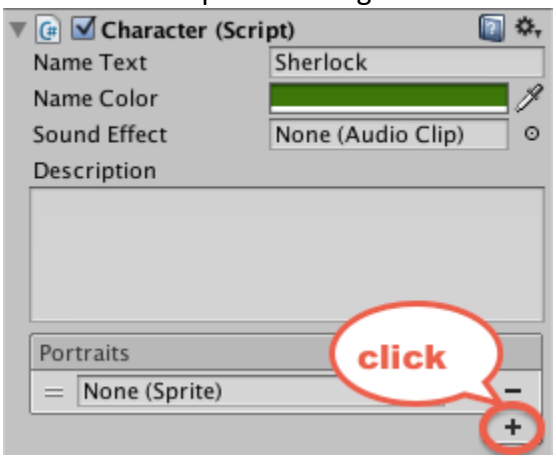
Listing portrait image(s) for use by Characters

If you add one or more portrait images to a character, then each Say command for that character can define which of those portrait images should be displayed, alongside the (colored) name of the Character.

To add portrait images to a character do the following:

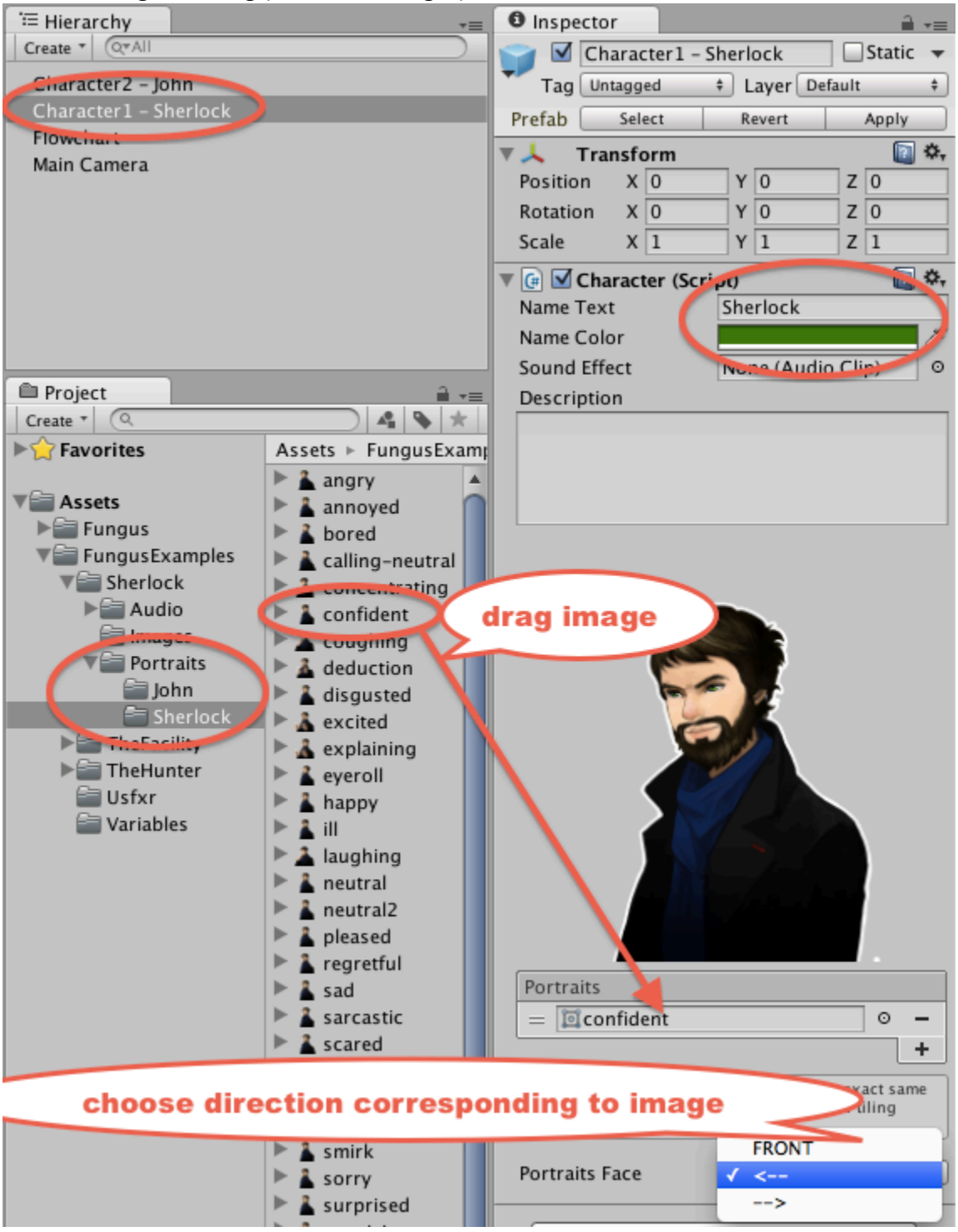
1. (setup) Create a new scene, add a Fungus Flowchart to the scene, and select the Block in the Flowchart.
2. Rename the Flowchart Block "The case of the missing violin".
3. Create a character, and in the Inspector give your character Name Text (we chose "Sherlock") and a name color.

4. Now in the Inspector click the Add Portrait button (the plus-sign "+"), to get a 'slot' into which to add a portrait image:



5. Drag the appropriate image into your new portrait image slot (in this screenshot we used the 'condident' image from the Sherlock example project). Also set the direction

that the image is facing (left / front / right):

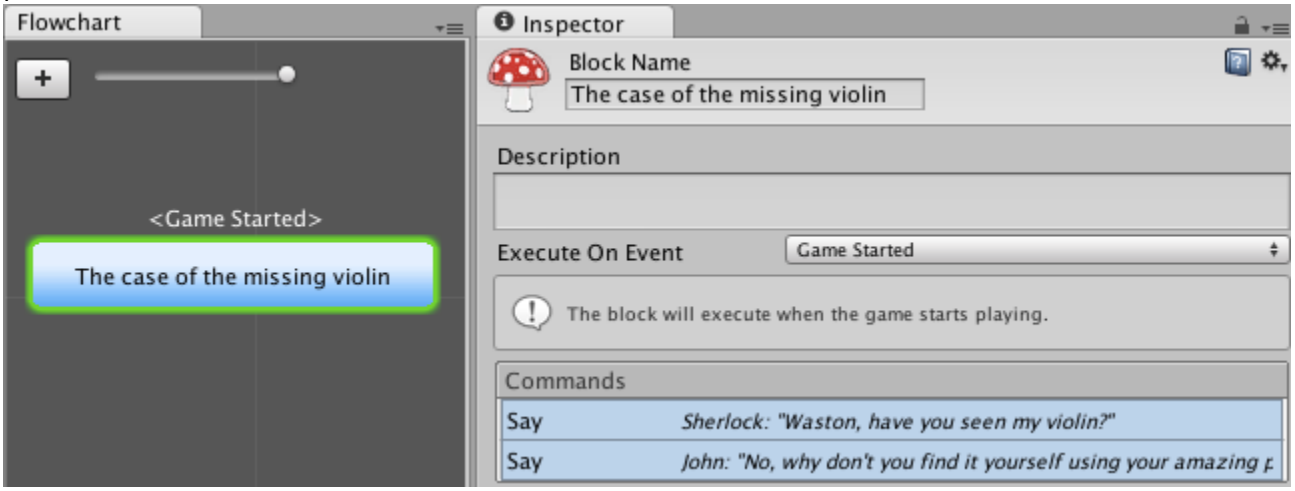


6. Create a second character (e.g. John, using Name Color blue, and portrait image 'annoyed').
7. Now select your Block in the Fungus Flowchart, so you can add some Commands to be executed...

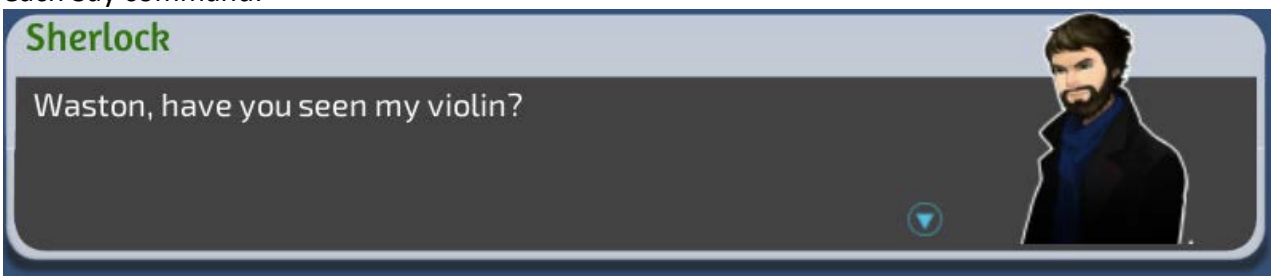
8. Create a Say command, for your Sherlock Character, saying "Watson, have you seen my violin?" and choosing portrait 'confident' (since this is the only we added to the Character):

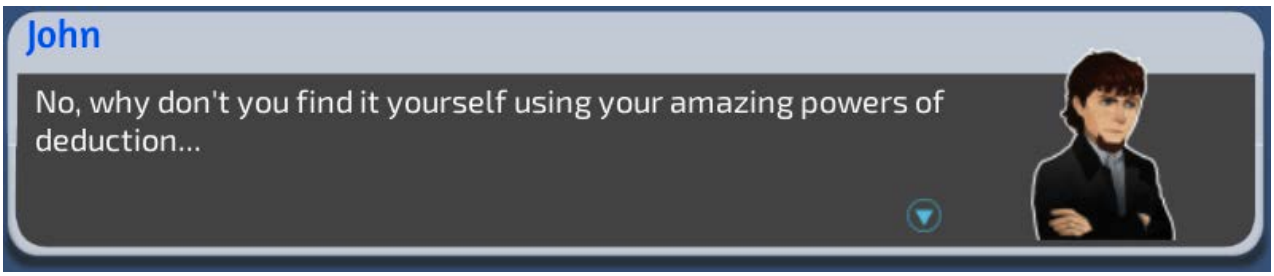
9. Add a second Say command, this time for Character John, saying "No, why don't you find it yourself using your amazing powers of deduction..." and choosing the 'annoyed'

portrait for John.



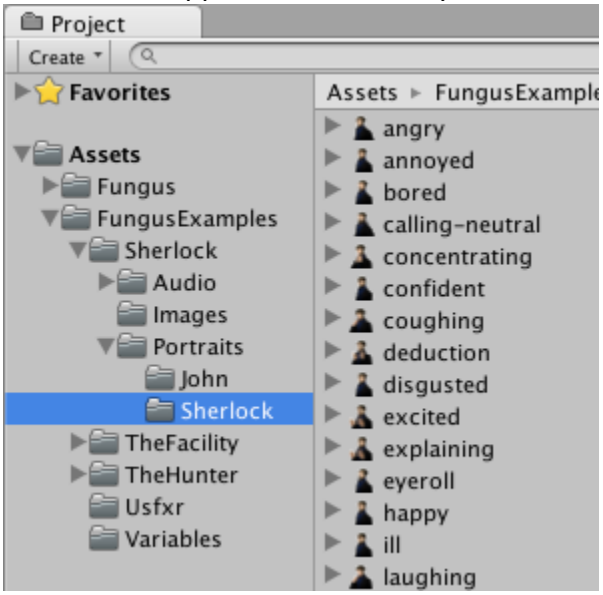
10. When you run the scene you should see a sequence of statements, clearly showing who is saying both with (colored) name text AND also the portrait image you selected for each Say command:





As you can see in some of the Fungus Example projects, many games will have a wide range of different portrait images for each character, to allow a full range of visual expression of

emotion to support the text of Say commands:



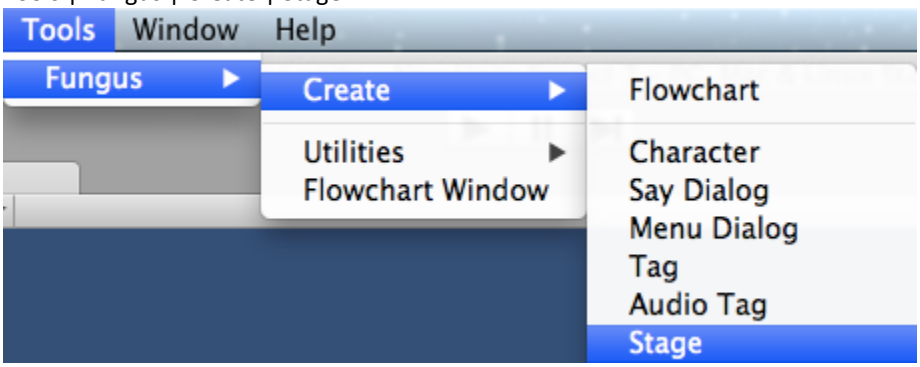
Add a Stage

Portrait images can be used in two ways in Fungus.

- They can be shown as part of the **Say** commands in the Say Dialog.
- Alternatively Portraits can be displayed and moved around the screen inside Fungus **Stages**, using the Portrait Command.

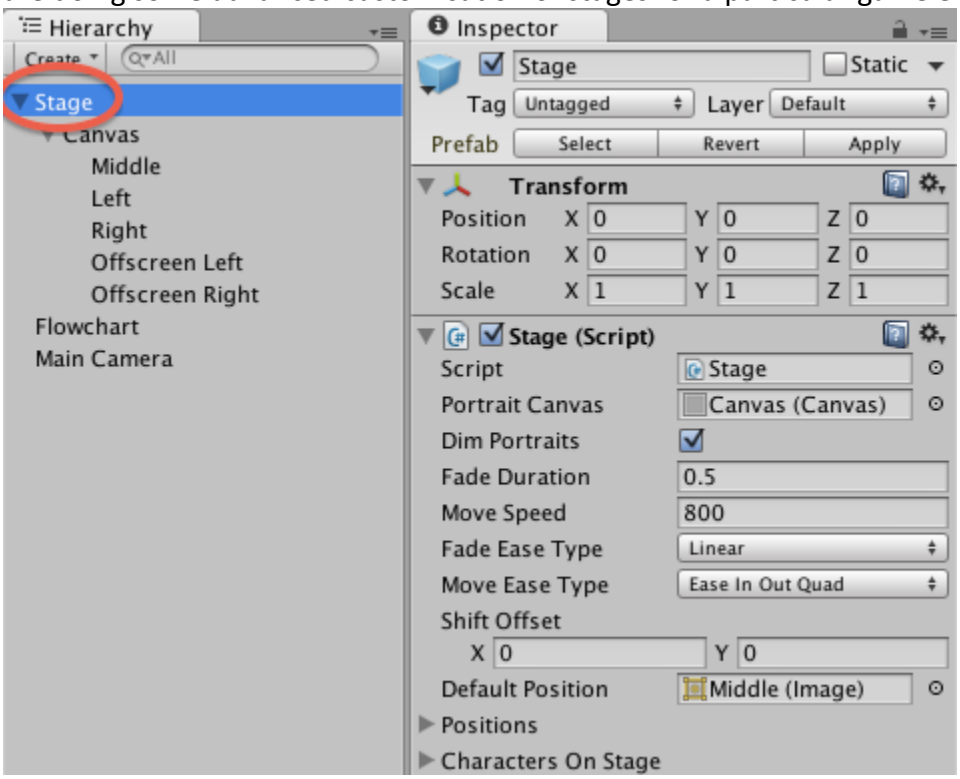
Create a simple stage that covers the whole game Window as follows:

1. (setup) Create a new scene, add a Fungus Flowchart to the scene, and select the Block in the Flowchart.
2. Rename the Flowchart Block "stage demo".
3. Create a Fungus Stage gameObject in the scene by choosing menu:
   Tools | Fungus | Create | Stage:



4. You should now see a new gameObject "Stage" added to the scene Hierarchy.

5. If you select it you will see its properties in the Inspector. We can leave the default settings, since these are for the stage to cover the whole Game window. There are some child gameObjects inside the Stage, but you don't need to worry about these unless you are doing some advanced customisation of stages for a particular game effect.



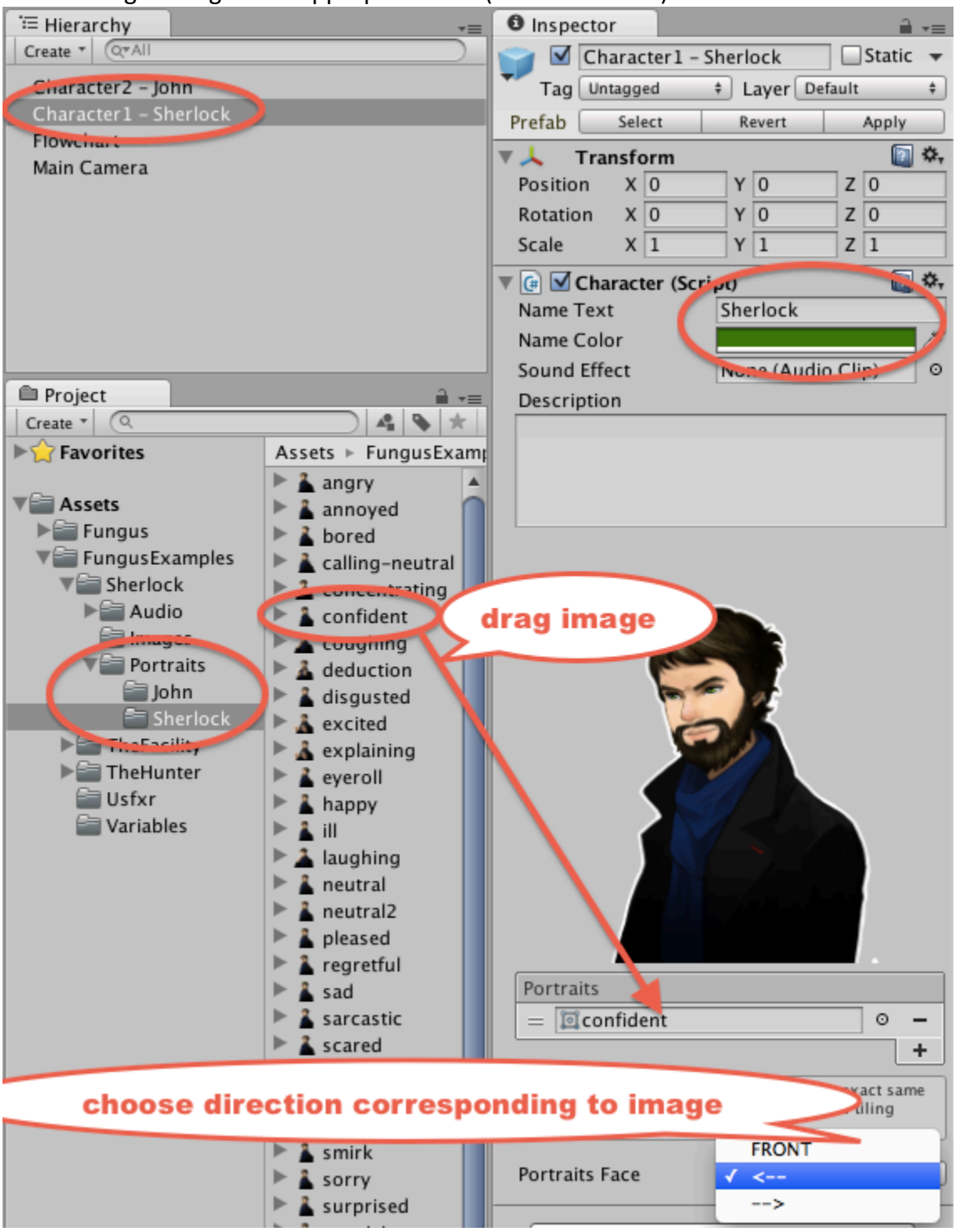
Now you have added a Fungus Stage to your scene, you will be able to make large Portrait images appear / move in-out of the screen using the **Portrait** Command in Fungus Flowchart Blocks...

Displaying Portrait images on stages with the Portrait command

Once you have a Fungus Stage, and a character then you can instruct Fungus to display / move onscreen the Character Portrait images. To make character images appear as part of a scene do the following:

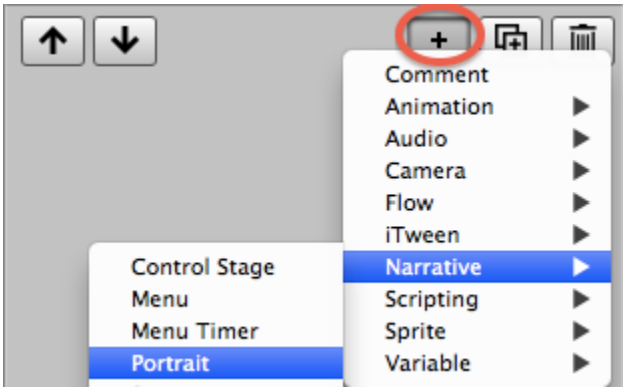
1. (setup) Create a new scene, add a Fungus Flowchart to the scene, and select the Block in the Flowchart.
2. Rename the Flowchart Block "sherlock enters dramatically".
3. Create a Fungus Stage gameObject in the scene by choosing menu:
   Tools | Fungus | Create | Stage.
4. Create a new character, name the gameObject "Character1 - Sherlock", set the Name Text to "sherlock" and the Name Color to green. Add to this character a portrait (we

used the sherlock-confident image from the Fungus Example project "Sherlock"). And set the image facing to the appropriate side (in our case: left):

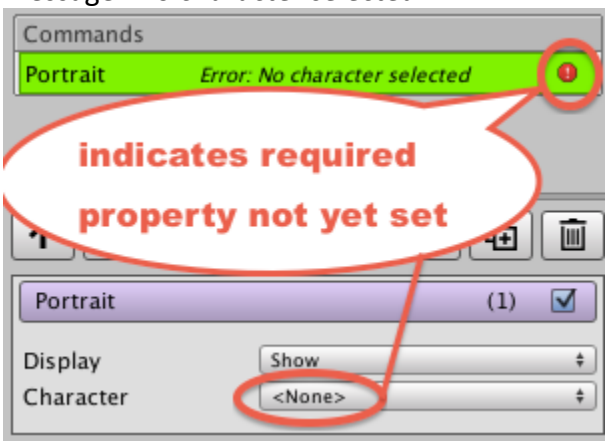


5. Add a Portrait Command by clicking the Add Command button (the plus-sign "+"), then choosing menu:

Narrative | Portrait:

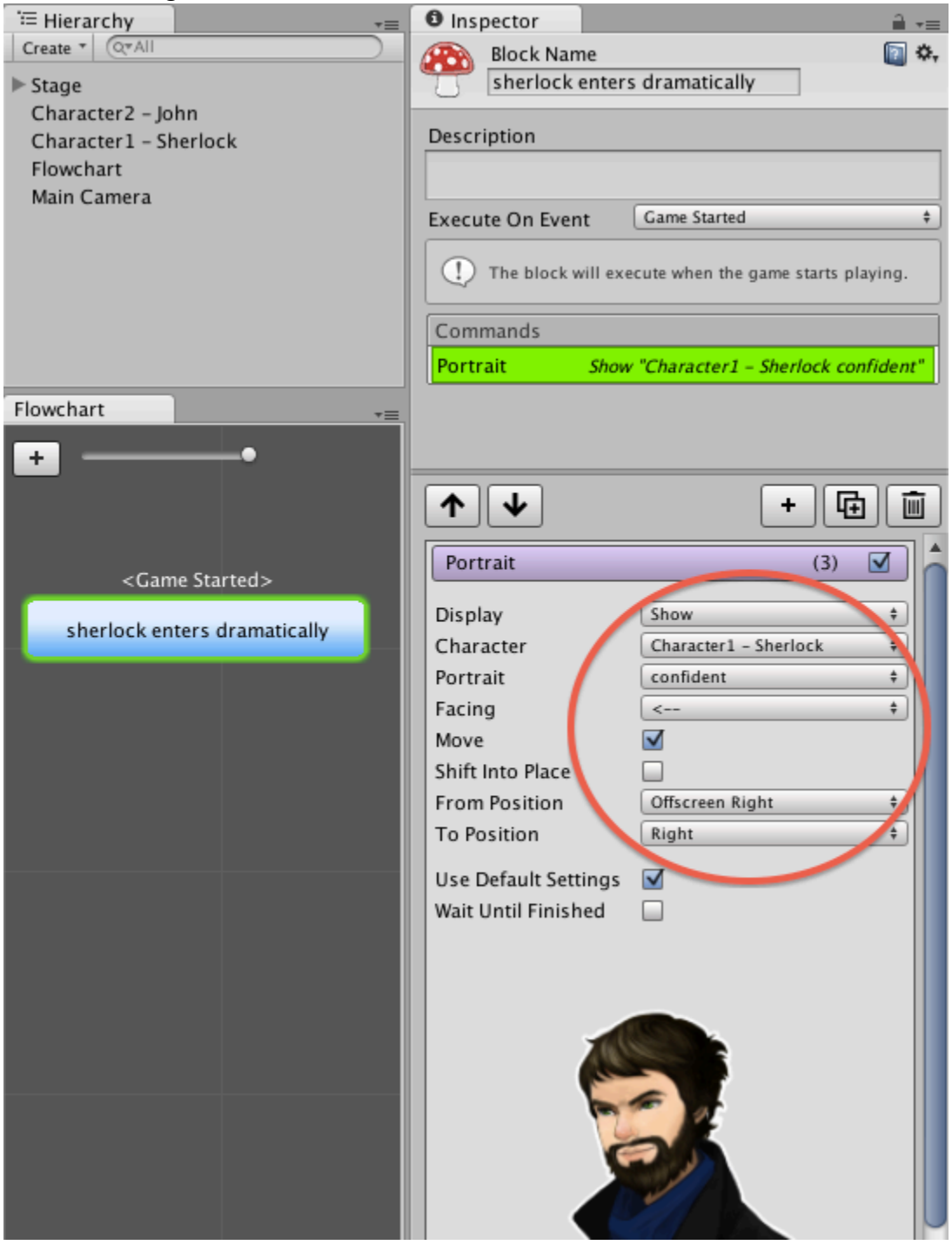


6. You will now see your new Portrait command in the top half of the Inspector, and its Command properties in the bottom half of the Inspector. Note the red exclamation mark at the right of the highlighted (green) Command row - this indicated when a command has one or more required properties that have not been set. We see the error message *"No character selected"*:



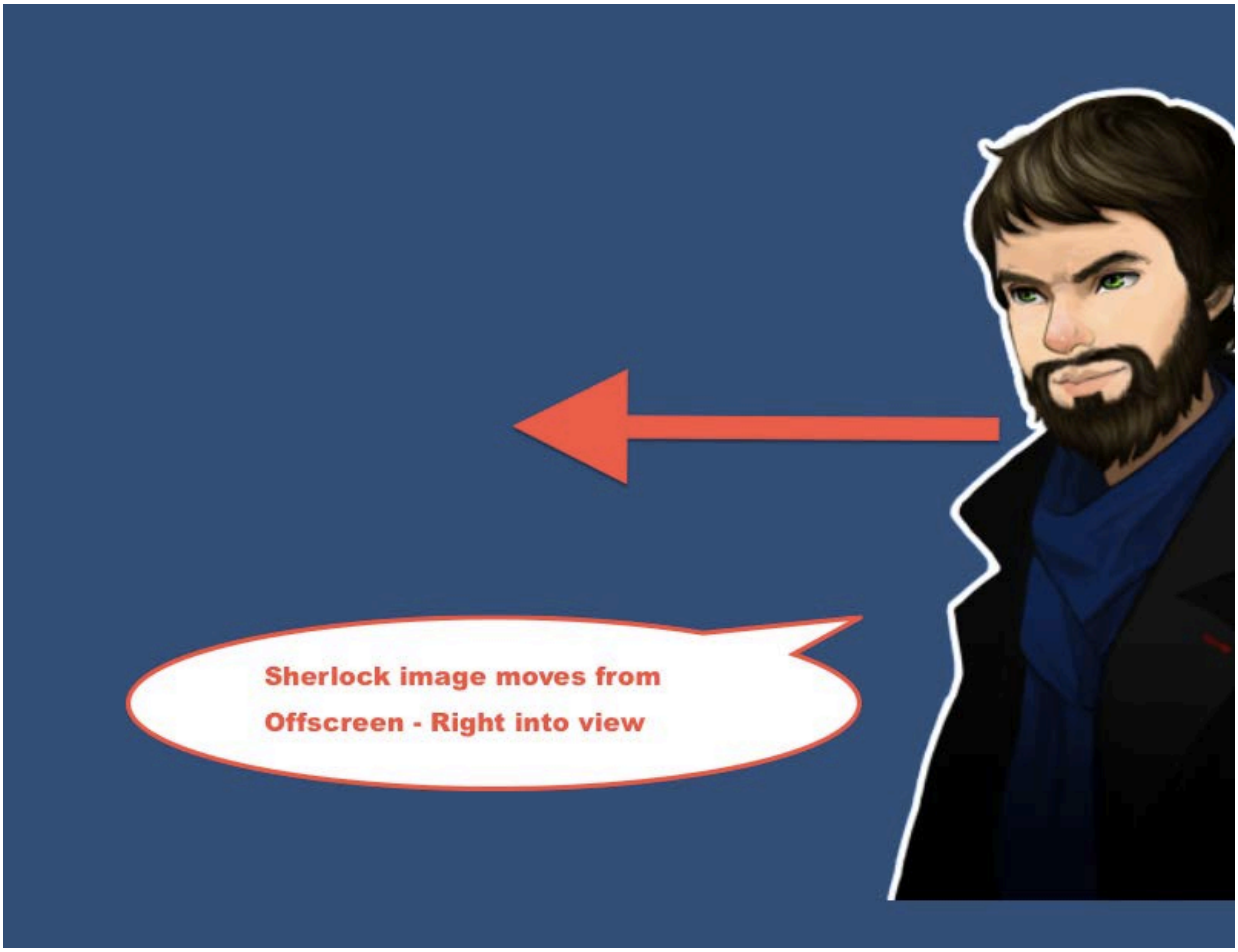
7. Set the portrait's character to "Character1 - sherlock", and set the following properties:
    - Portrait: confident
    - Facing: <-- (left)
    - Move: Yes (check the checkbox)
    - From Position: Offscreen Right

- To Position: Right



8. When you run the scene, the Sherlock portrait image should move into view having started from Offscreen - Right. The image stops when it gets to about a third the way

onto the screen:

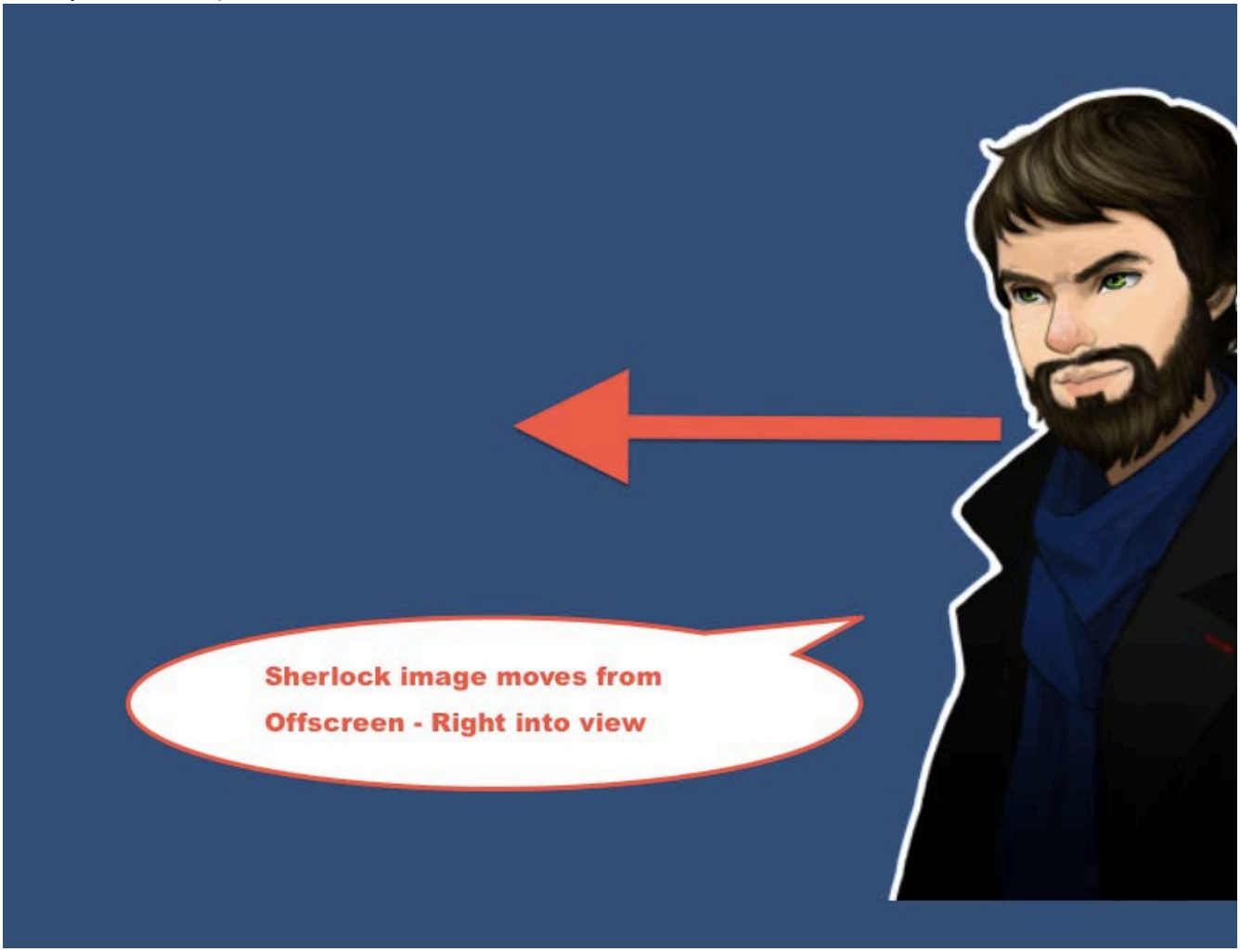


Note, a common Command flow sequence is:

- to have a character enter on screen (Portrait command),
- then have that character say something (Say command),
- then have another character enter the screen (Portrait command),
- and then that second character says something (Say command).

Here is just such a sequence for the "Case of the missing violin" two-sentence scenario explored in the recipe to learning how to create Fungus Characters (recipe: Listing portrait image(s) for

use by Characters):



Here we see the Play Mode user experience of the output of running such a workflow:

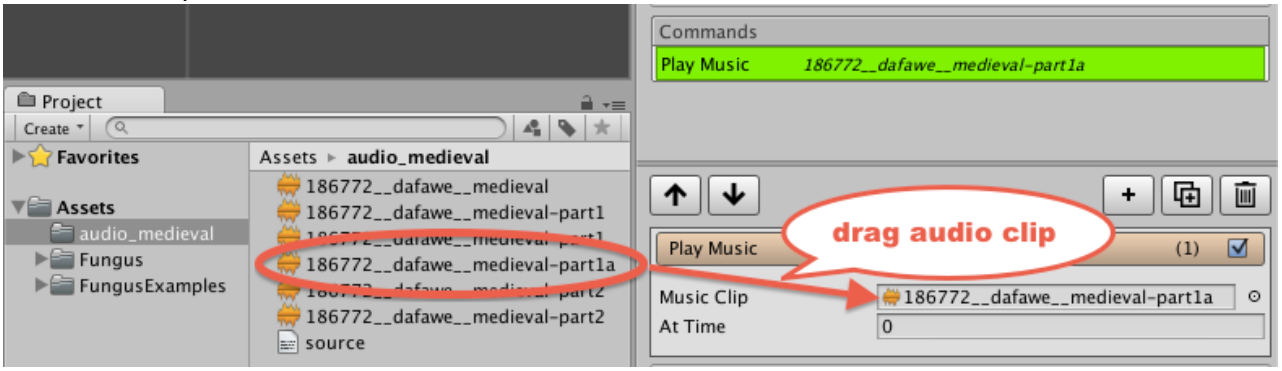
We can also see that the character that is Say'ing something, has a WHITE-outline around its Stage portrait, to visually reinforce to the user which character is speaking at any point in time...

Play some music

Music sound clips loop, so they are restarted once they have finished playing. Often the first Command in a Block is a **Play Music** Command. Add music to a Block as follows:

1. (setup) Create a new scene, add a Fungus Flowchart to the scene, and select the Block in the Flowchart.
2. Add a Play Music Command to the current Block by clicking the Add Command (plus-sign "+" button) in the Inspector, and then choosing menu: Audio | Play Music.
3. Ensure the Play Music command is selected (green highlight) in the top of the Inspector, and then drag the desired music clip file into the "Music Clip" property in the bottom

half of the Inspector:



4. Change the volume as desired
   (the default is 1, values are between 0.0 and 1.0, representing percentages of volume from 0% - 100%).
5. Play your scene - the music clip should play, and keep looping.

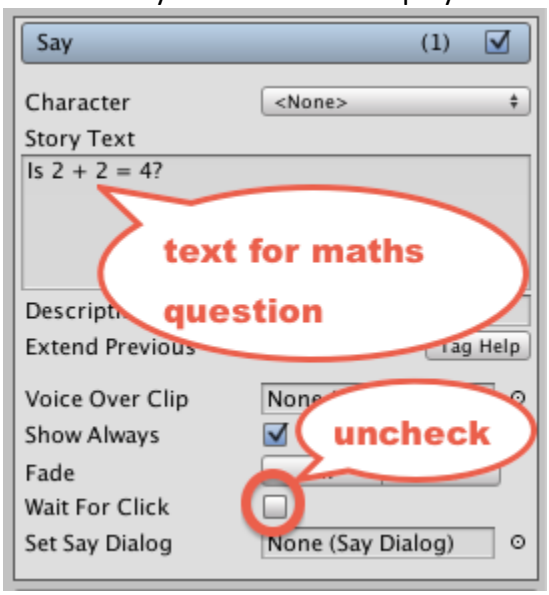
NOTE: If you wish to start playing the music clip from a known time-point (rather than from the beginning), then enter the desired time-point in the Inspector property "At Time" for your Play Music command.

Add menu commands to branch to other blocks

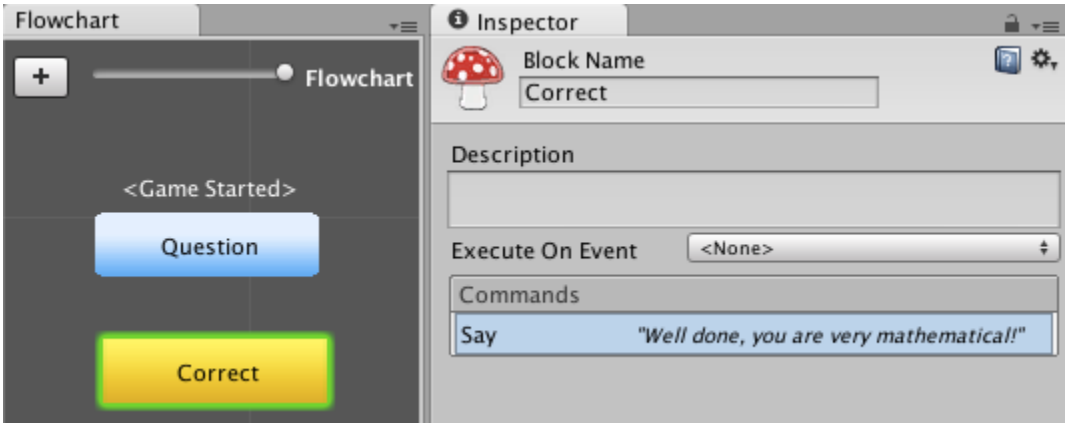
Let's use a Say command above to ask a tricky mathematical question, and demonstrate the Menu command by offering the user a choice been "correct' and "incorrect" answers. Menu commands transfer control to another block - so we'll need to add 2 new blocks to correspond to the 2 answers. Do the following:

1. (setup) Create a new scene, add a Fungus Flowchart to the scene, and select the Block in the Flowchart.
2. Rename the Block in the Flowchart to "Question".
3. Create a Say command, with **Story Text** to ask the question: "Is 2 + 2?".

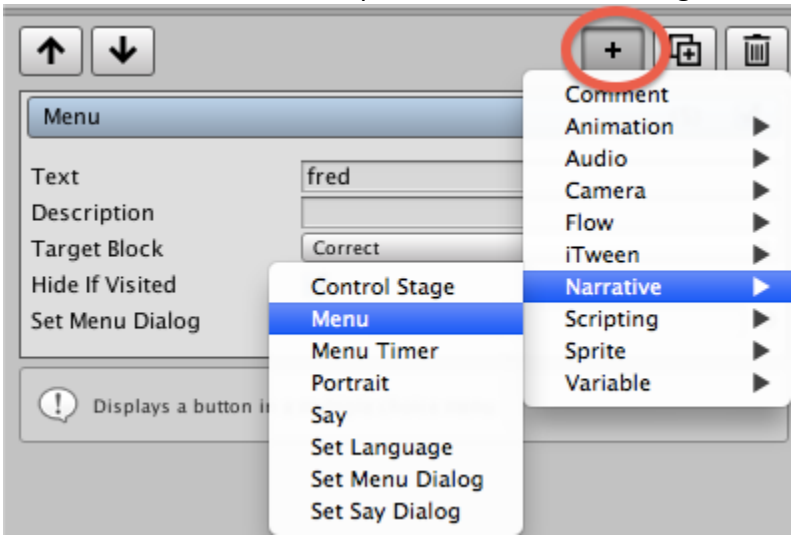
4. Uncheck the "Wait For Click" checkbox (this is so we see the menu options immediately after the Say command has displayed the question):



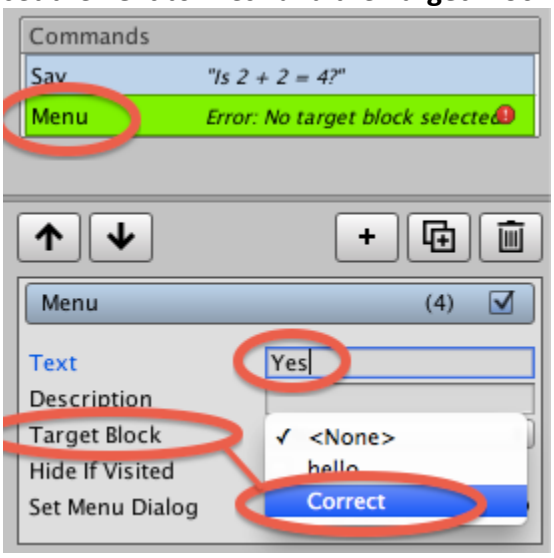
5. Create a new Block, named "Correct" which contains a **Say** command with the text "Well done, you are very mathematical!". Click the plus-sign button in the Flowchart window to add a new Block to the Flowchart, rename it "Correct" and then add that Say command:



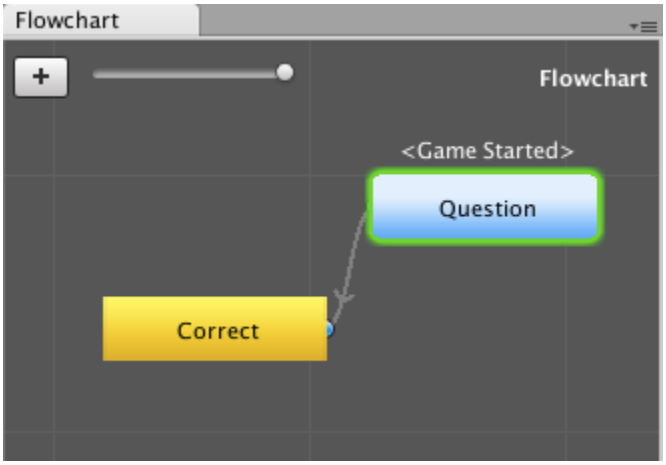
6. Select the "Question" block, and add a Menu command by clicking the plus-sign add Command button in the Inspector and then choosing menu: Narrative | Menu.



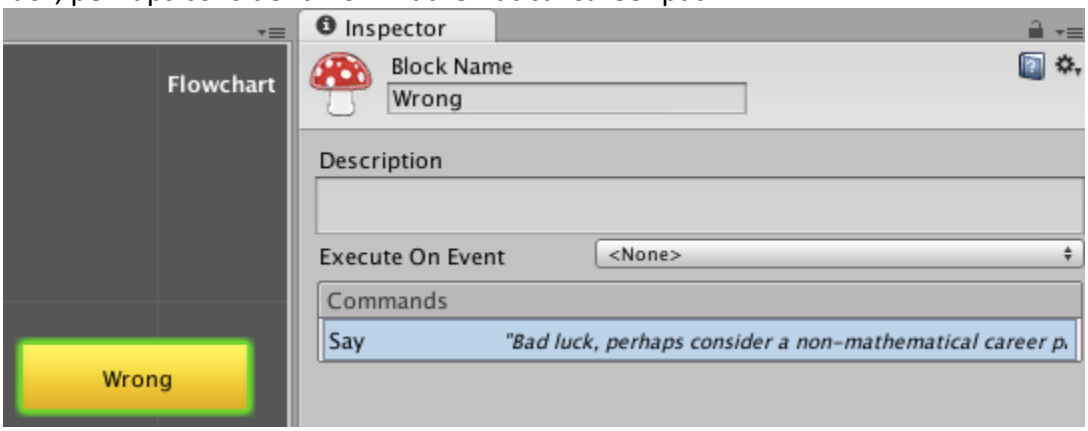
7. With this new Menu command selected (green) in the top half of the Inspector window, set the **Text** to "Yes" and the **Target Block** to your new "Correct" block:



8. You should now see how the 'flow' of commands can change from Block "hello" to Block "Correct" in the Flowchart window:

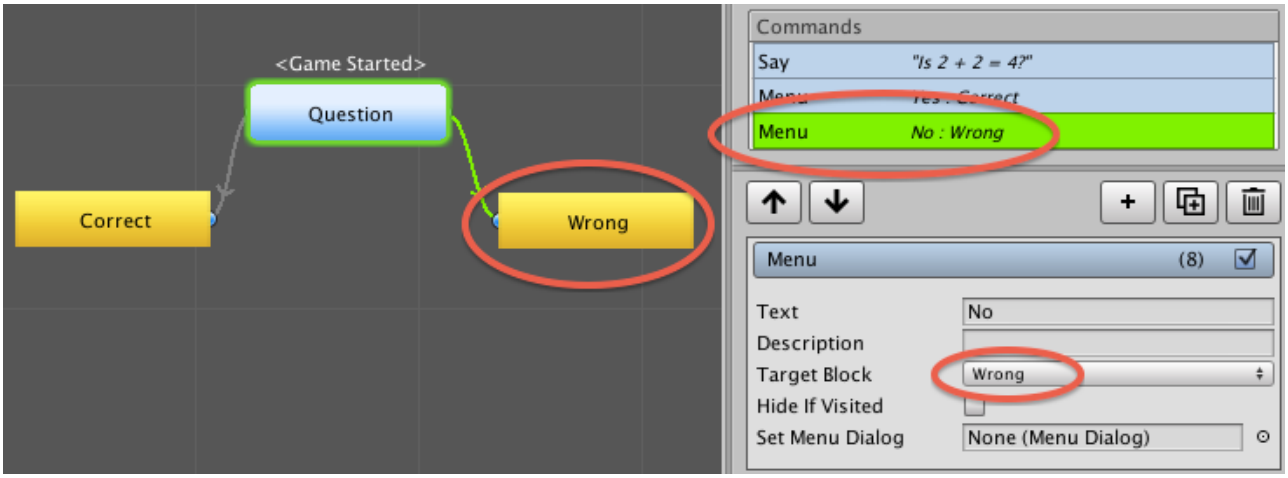


9. Add a second new Block named "Wrong", containing a Say command with text "Bad luck, perhaps consider a non-mathematical career path..."

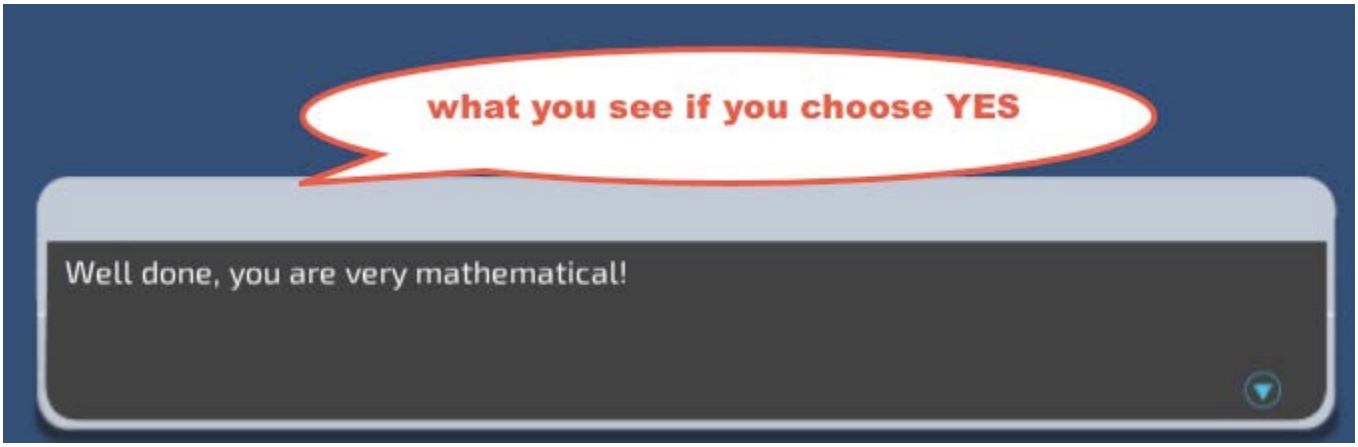


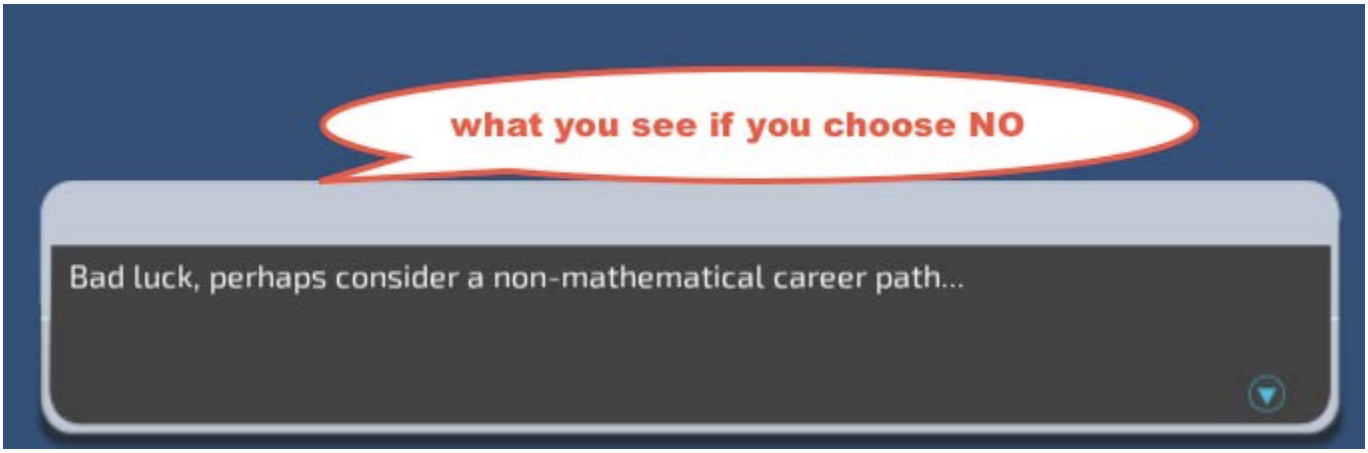
10. Now we need to add another Menu command to our "hello" block, offering the user the "No" answer to our maths question, and passing control to Block "Wrong" if they disagree that 2 + 2 = 4. Select the "hello" block, and add a Menu command. With this new Menu command selected (green) in the top half of the Inspector window, set the **Text** to "No" and the **Target Block** to your new "Wrong" block.
11. You should now see in the Flowchart window how block "hello" can pass control to either block "Correct" or Block "Wrong" - depending on which menu answer the user

selects.



12. Run the scene, and you should see the Say question appear at the bottom of the screen, and also the two Menu buttons "Yes" and "No" in the middle of the screen. Clicking "Yes" then runs the "Correct" Block's commands, and clicking "No" runs the "Wrong" block's commands:



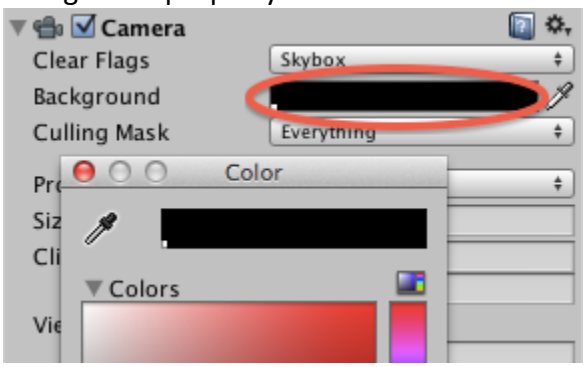


Change Camera background colour

Unity cameras determine what the user sees when a scene is running. When nothing is present in all or part of the camera's rectangle a solid "Background" colour is displayed. Unity cameras have a default Background of a medium dark blue colour. You can change this as follows:

1. (setup) Create a new 2D scene, unless you already have a scene with which to work.
2. Select the Main Camera in the Hierarchy.

3. In the Inspector for the Camera component, click and choose a different value for the Background property - often black works well.



4. Now when any part of the camera rectangle (frustrum) shows no gameOjects then your custom Background colour will be what the user sees.

Add a background sprite

To add any sprite image file from your Unity Project folder into the current scene, simply drag a reference to the sprite image file from the Project window onto the Scene window, and rotate / resize desired. The sprite will appear as a new gameObject (with same name as Sprite Project

image file) in the Hierarchy window:

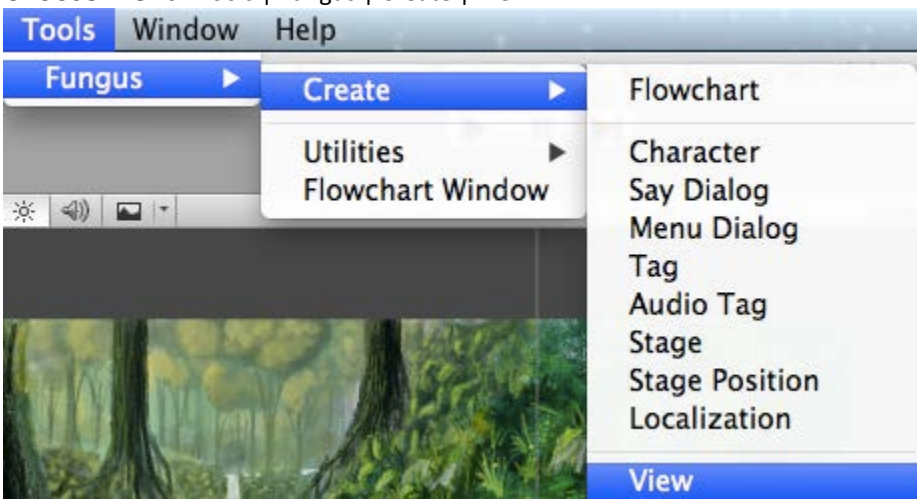
NOTE: You may not be able to see the sprite, because what we see depends on the current settings for the camera. What the camera shows, how it moves etc. can be controlled by Fungus Views and Commmands relating to Views.

Adding and customising a view

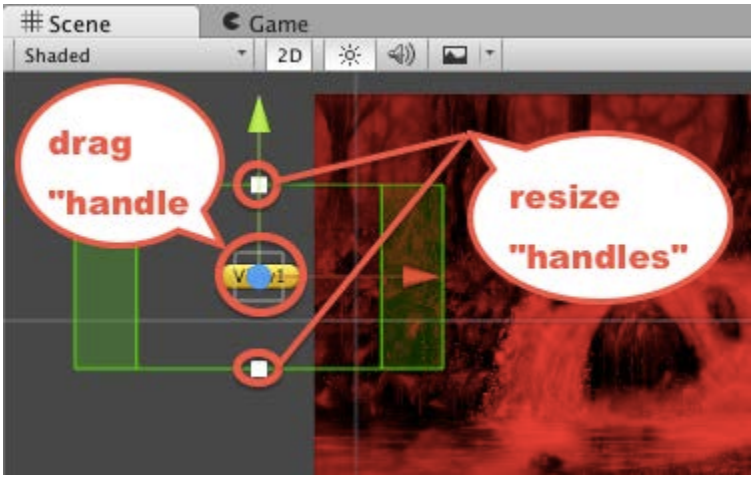
What the main camera of a scene displays to the user, and how it moves etc. can be controlled by Fungus Views and Fungus Commmands relating to Views. A Fungus View is a special gameObject in the Hierarchy, it appears as a green outlined inner rectangle, with two filled green rectangles on the left and the right. The ratio of the outlined inner rectangle is 4:3. The ratio of the outer rectangle (which includes the two filled green left and right rectangles) is 16:9. These two ratios cover almost every common phone, tablet and computer screen width-to-height ratio. So arranging the view so that a background Sprite image looks good for both inner- and outer- rectangles of a view, pretty much ensures your game will look good on any device. Setting the background color of the camera to something like black also means on the rare device that has an odd ratio showing content outside of the view outer rectangle, the game should still look perfectly acceptable.

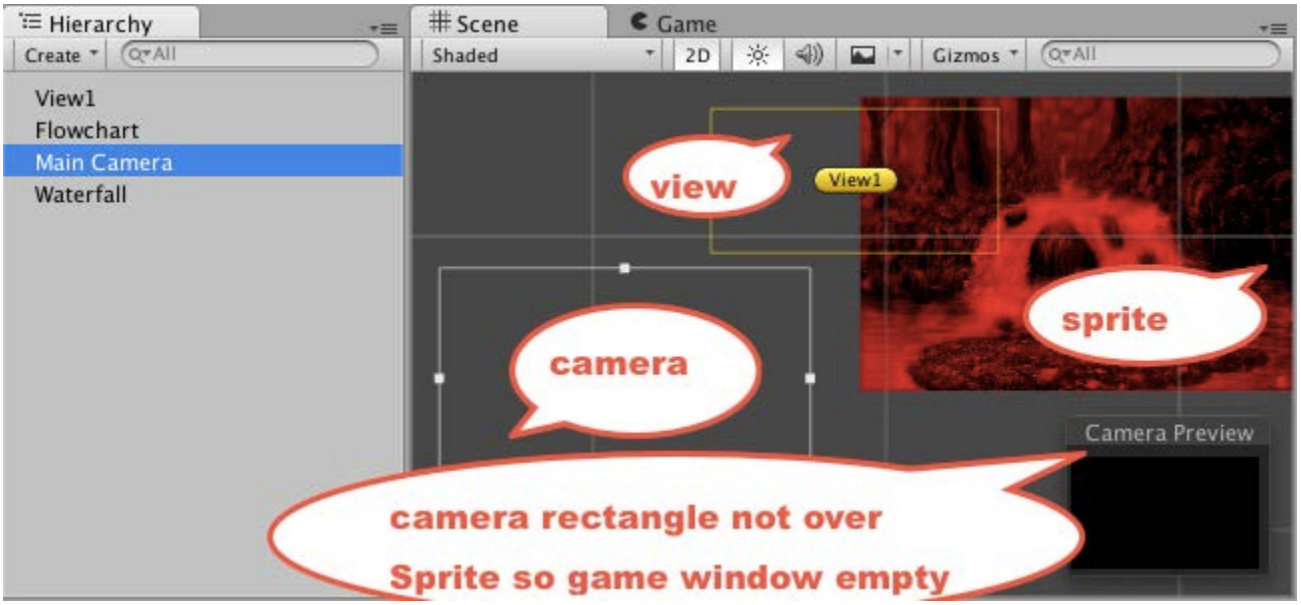
To add a view to the current scene do the following:

1. (setup) Create / Edit a scene that has a Sprite background image gameObject
2. Choose menu: Tools | Fungus | Create | View:

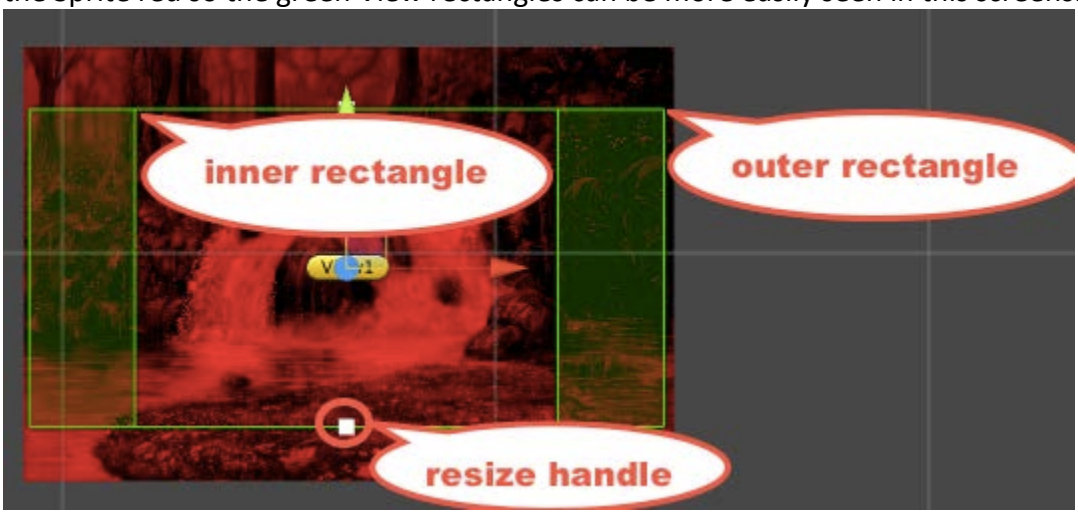


3. Rename this View as "View1".
4. Use the two white squares to resize the view (it maintains its proportions). Use the center square outline, or vertical and horizontal arrows to move the View around the Scene window.





5. Ensure the View is selected in the Hierarchy, then position the view so that it is approximately centered on your background sprite image
6. Resize (and if necessary reposition) the View to be as big as possible, but ensuring that its outer rectangle stays within the bounds of the background sprite. (Note we've tinted the Sprite red so the green View rectangles can be more easily seen in this screenshot):



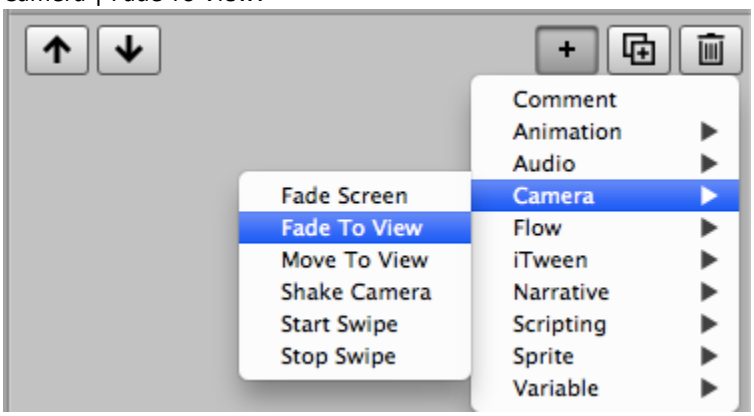
7. Note: You can also rotate the view with the Unity Rotate tool

NOTE: Utnil you add a "Fade To View" Fungus command, you still may not see the Sprite in the Game window when the scene plays, since the Main Camera has not been oriented to resize and align with the view.

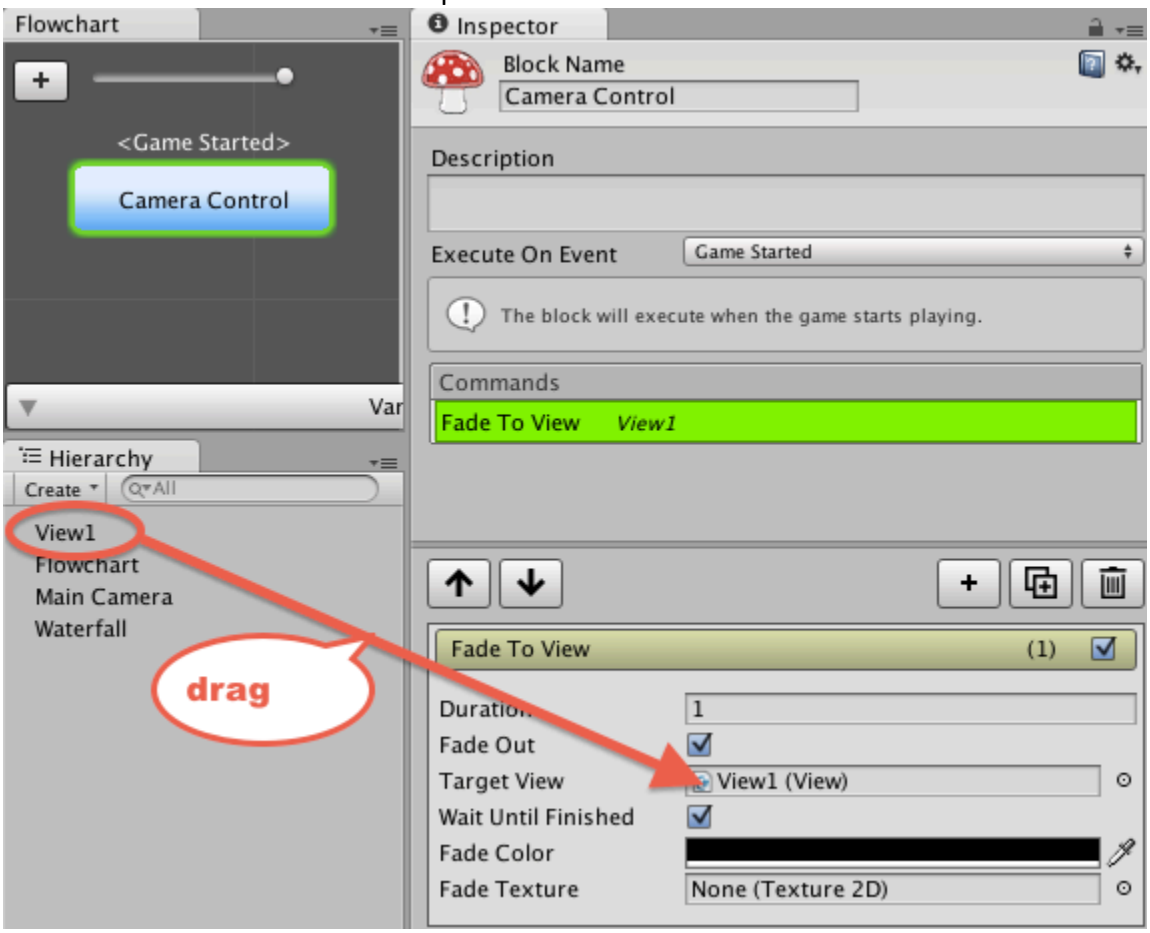
Add a Fade To View command

Once you have a Scene that contains some background Sprites and Fungus Views, you are ready to use the Fungus camera related Commands to control what the user sees. The simplest camera control is to make the Game window fade from a solid colour to the Main Camera being sized, positioned (and if necessary rotated) to show a specified Fungus View. Do the following:

1. (setup) Create / being editing a Scene containing a background Sprite image, and a Fungus View that has been positioned to show all / some of the Sprite.
2. In the Fungus Flowchart rename the Block "Camera Control".
3. Add a new "Fade to View" Command to the Block. First click the Plus button in the bottom half of the Inspector window, to add a new Command, then choose menu: Camera | Fade To View:

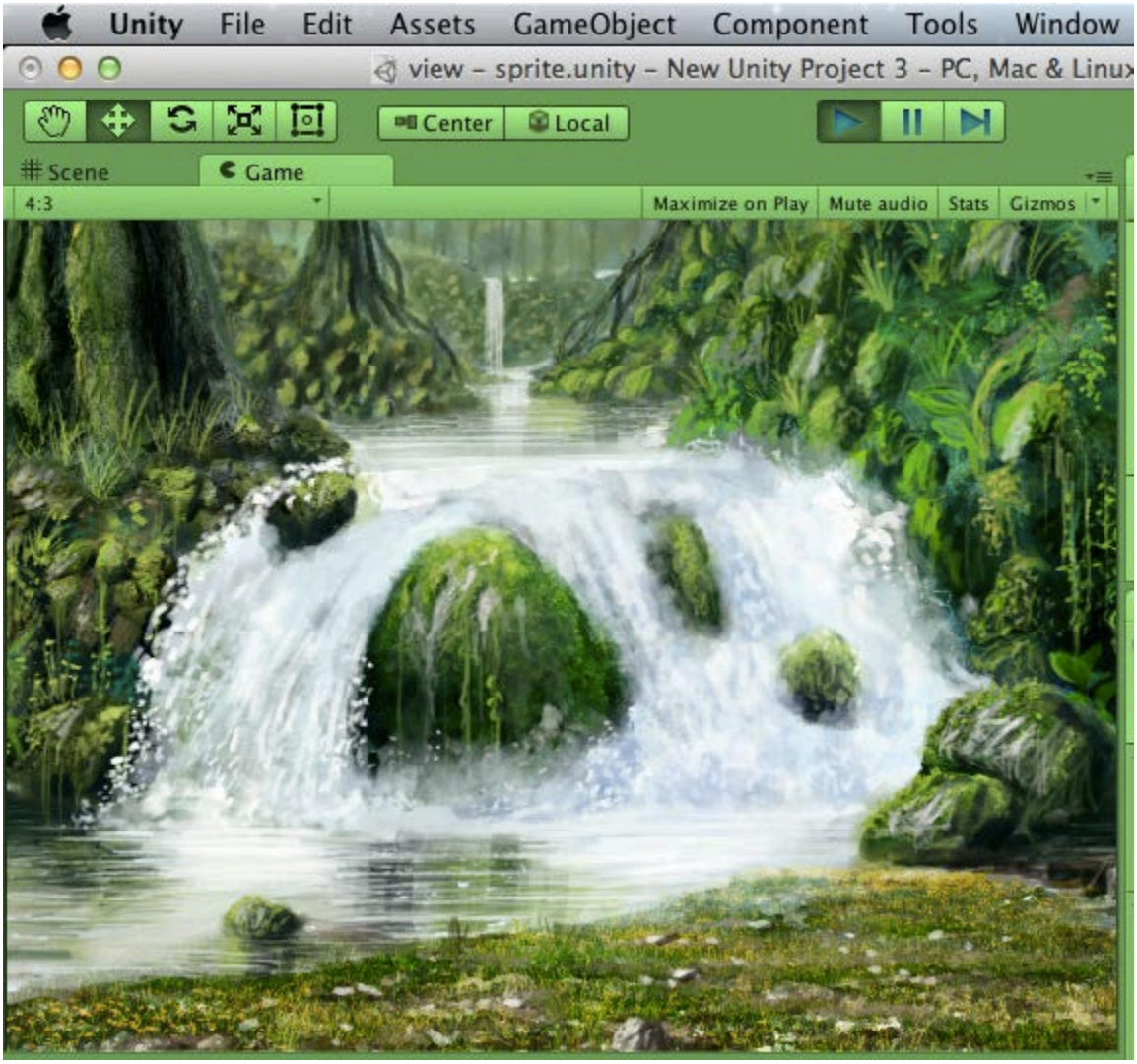


4. Now Drag "View1" from the Hierarchy window into the "Target View" property of the Fade to View Command in the Inspector:



5. (We'll keep the defaults of 1 second and fade From Color of black).
6. When you run the Scene the Game window should start off solid black, and then slowly the background Sprite image within the View rectangle should fade into view.

7. Now Drag "View1" from the Hierarchy window into the



Audio in Fungus games

Almost every game benefits from some sound! Often we categories audio clips into three kinds:

1. Music
2. Sound effects
3. Speech

Fungus provides straightforward ways to include all 3 kinds of audio clip in your game, using the techniques presented here.

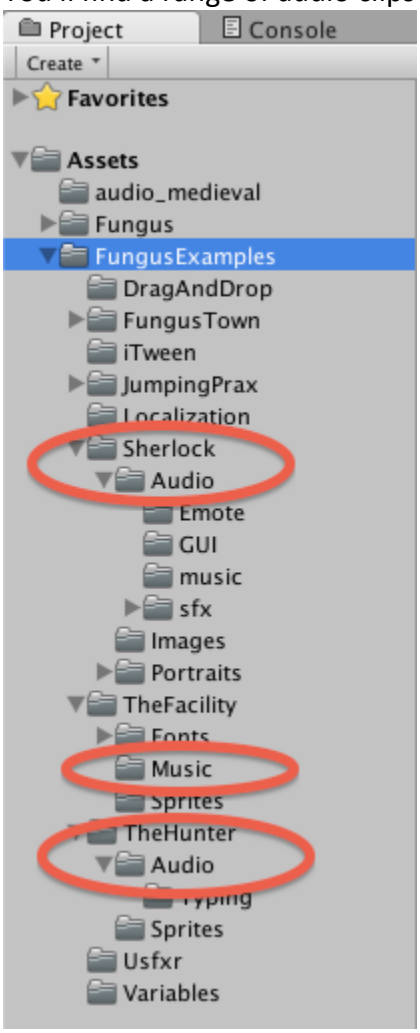
Sources of free to use audio clips and music

Before you can **add** audio clips to a game you need to get some audio clips. Here are some sources of audio clips to use when learning about audio in Fungus, in case you don't have some of your own to hand.

The following are some good places online to fine music and sound effects for games. Some are free for any use (including commerical), some are just free for personal use. As always, check the licence of media assets before using them for any commerical products ...

- Freesound.org
- lots of creative commons and royalty free sounds at SoundBible.com
- a great list of audio sources in peoples answers to questions at Answers.unity3d and StackOverflow.com
- mixture of free and paid music sources at PixelProspector.com

You'll find a range of audio clips included inside the Fungus Examples folders:



Adding audio assets to your project

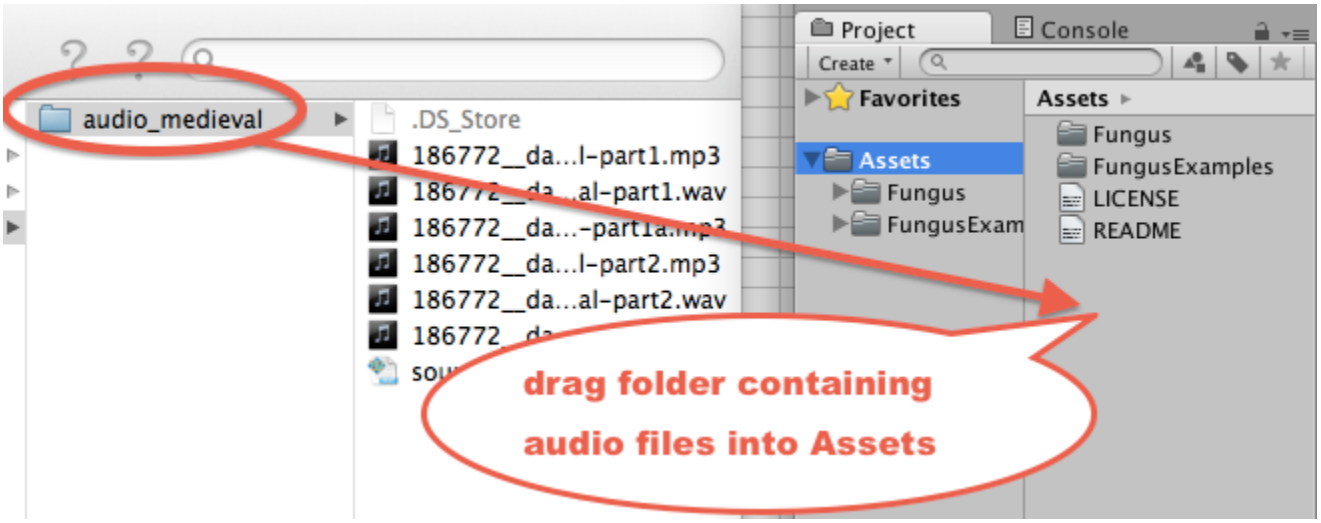
Once you have some audio clips on your computer, you need to import them into your Unity project.

Method 1 (menu)

You can do this one clip at a time, by choosing menu: Assets | Import New Asset... and navigating to and selecting each clip.

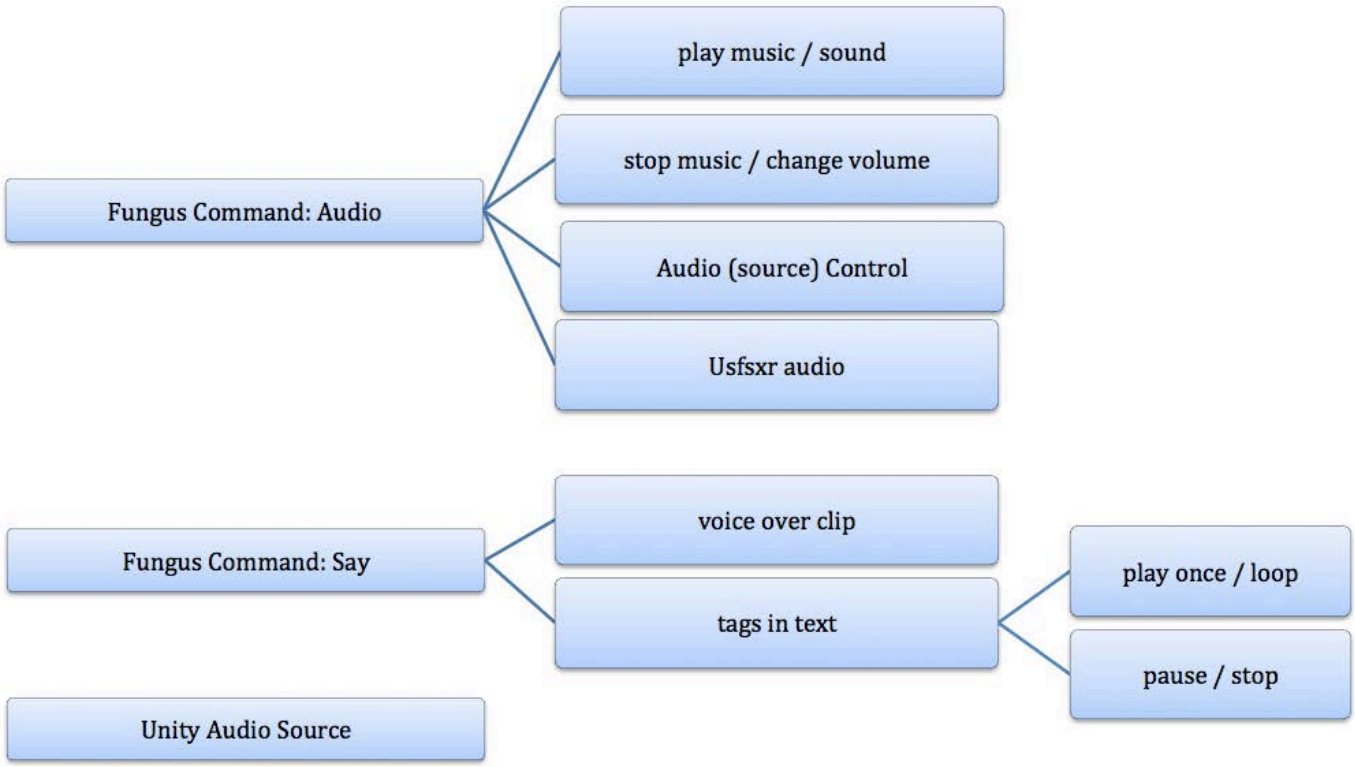
Method 2 (drag-drop)

Alternatively you can **drag** files or entire folders into your Unity Project window, and Unity will make a copy of, and then import the dragged files:



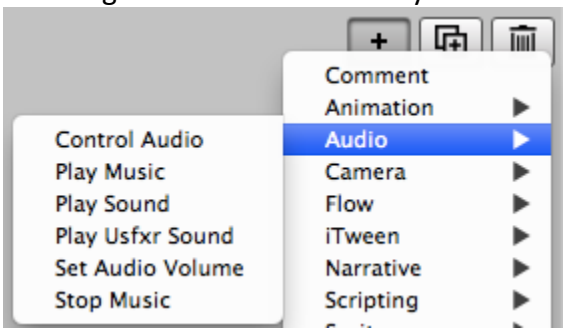
Three ways to work with audio in Fungus games

There are 3 main ways to work with audio in Fungus games. These are the Audio commands, the Say command, and gameObjects containing Unity Audio Source components. All three are discussed below:

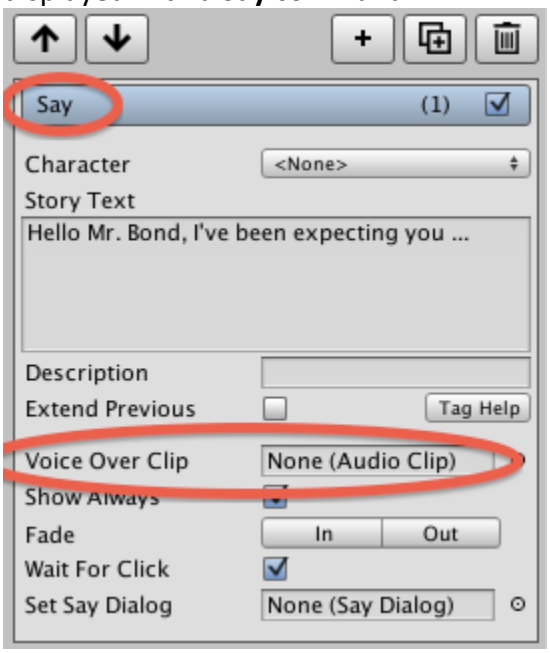


List of Fungus audio commands

The range of audio **Commands** you can add to a Block are as follows:



Also you can declare an audio clip that contains the speech voiceover to correspond to text displayed with a **Say** command:

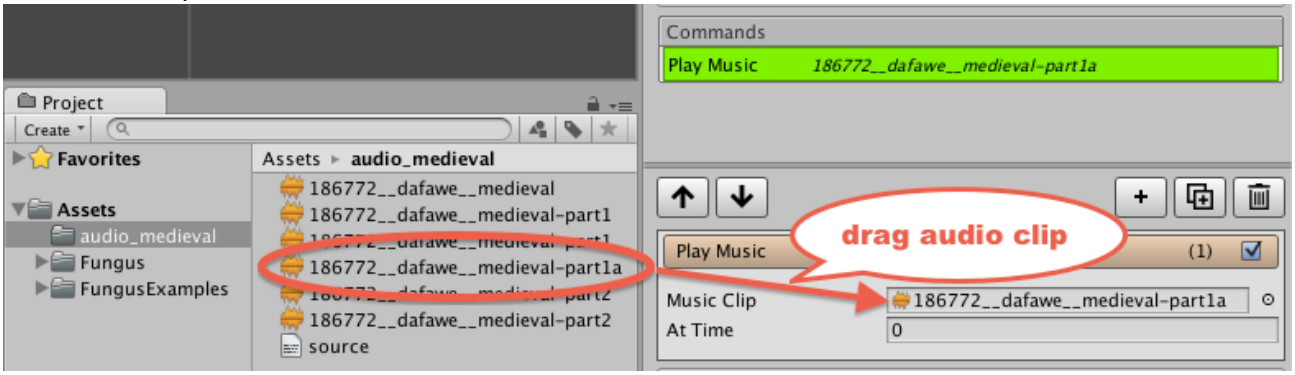


Play Music command

Music sound clips loop, so they are restarted once they have finished playing. Often the first Command in a Block is a **Play Music** Command. Add music to a Block as follows:

1. (if you have not already done so: Create a new scene, add a Fungus Flowchart to the scene, and select the Block in the Flowchart).
2. Add a Play Music Command to the current Block by clicking the Add Command (plus-sign "+" button) in the Inspector, and then choosing menu: Audio | Play Music.
3. Ensure the Play Music command is selected (green highlight) in the top of the Inspector, and then drag the desired music clip file into the "Music Clip" property in the bottom

half of the Inspector:



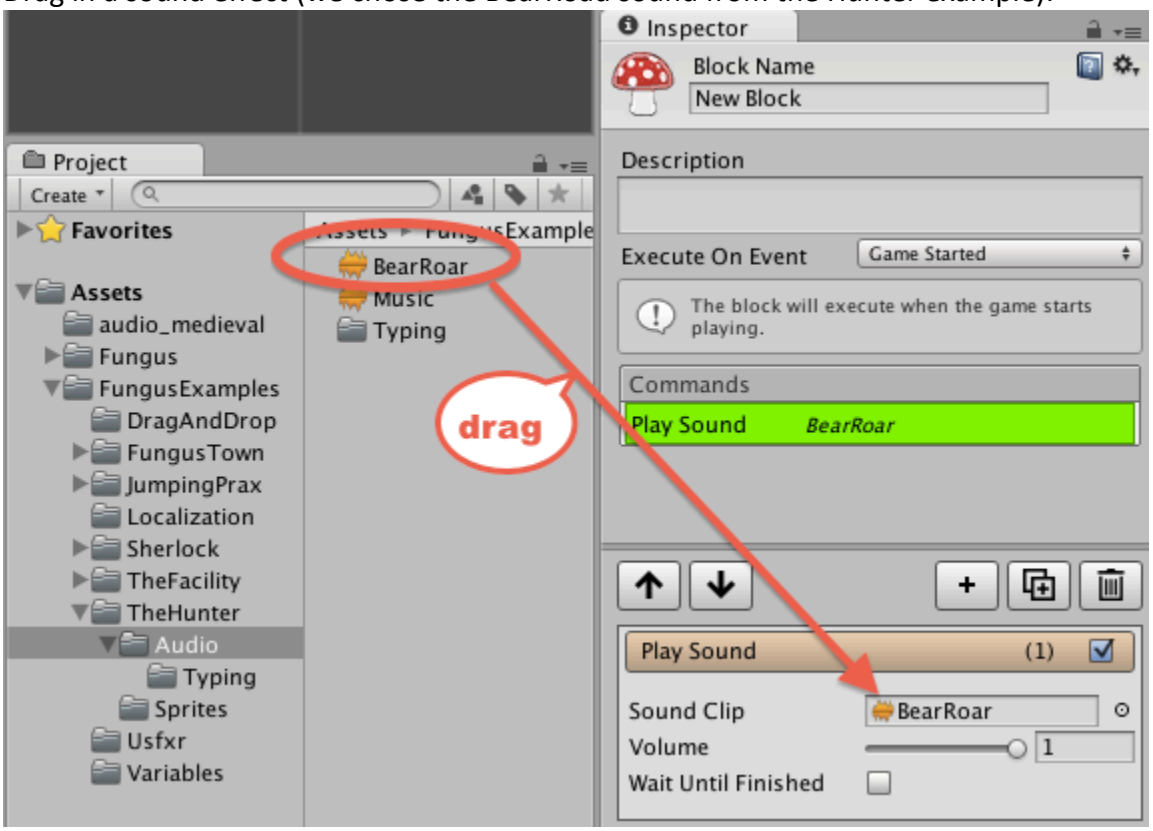
4. Change the volume as desired
   (the default is 1, values are between 0.0 and 1.0, representing percentages of volume from 0% - 100%).
5. Play your scene - the music clip should play, and keep looping.

NOTE: If you wish to start playing the music clip from a known time-point (rather than from the beginning), then enter the desired timepoint in the Inspector property "At Time" for your Play Music command.

Play Sound command

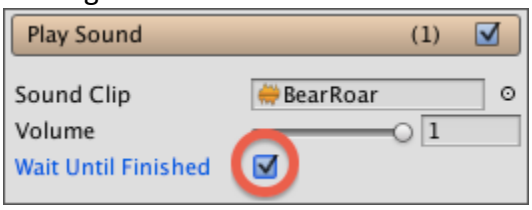
The Fungus Play Sound Command will play a stated audio clip once. With your Flowchart Block selected, click the Add Command button in the Inspector and choose menu: Audio | Play Sound.

Drag in a sound effect (we chose the BearRoad sound from the Hunter example):



Play the scene, you should hear your sound effect play once.

Note. The default Fungus setting is for the sound effect to start playing, and while it is playing the next Command in the Block will start executing. However, you if you check the "Wait Until Finished" checkbox, then Fungus will wait until the sound effect has finished playing, before moving on to execute the next Command in the block:

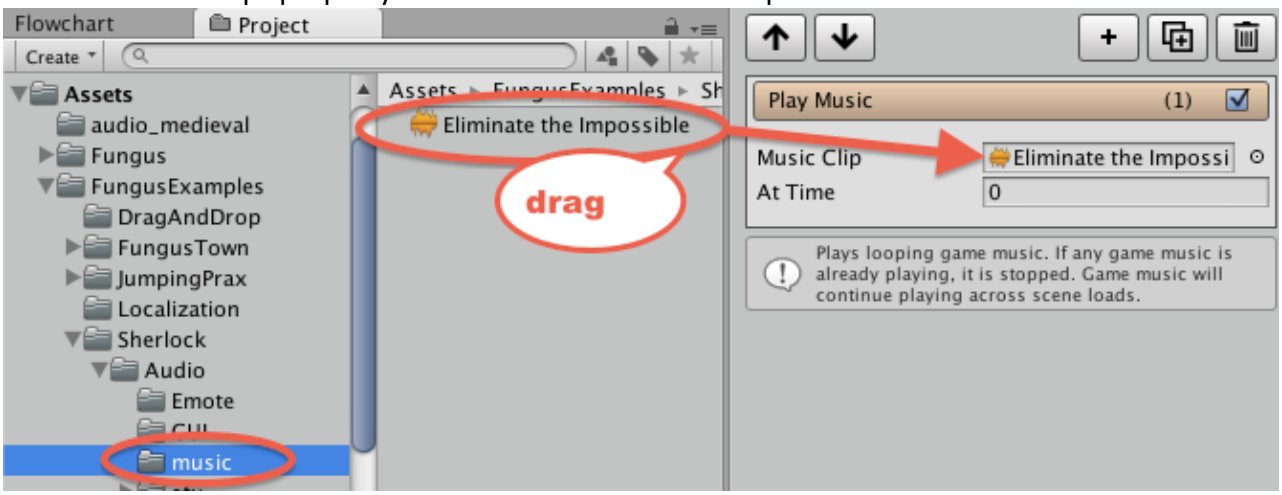


Set Audio Volume command

The default volume for music being played is 1 (100%). However, we can change this easily with the Set Audio Volume command. Do the following:

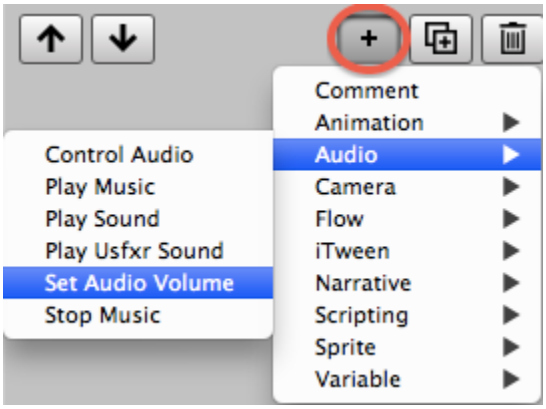
1. (if you have not already done so: Create a new scene, add a Fungus Flowchart to the scene, and select the Block in the Flowchart).
2. Rename this Block "Play Music".

3. Add a Play Music Command to the current Block, then drag the desired music clip file into the "Music Clip" property in the bottom half of the Inspector:



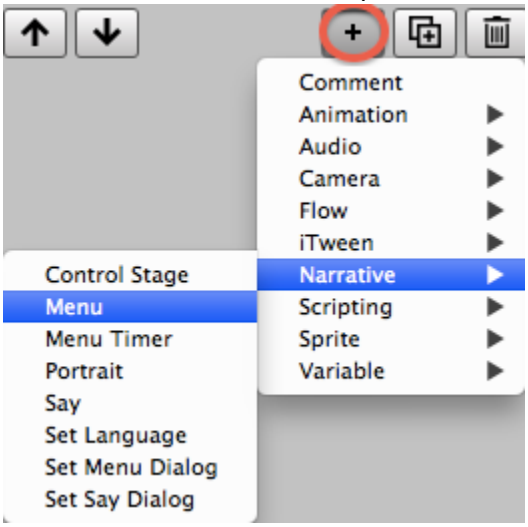
4. If you play the scene now, the music will play at full volume (1 = 100%).
5. Now create a second Block in the Flowchart window named "quieter".

6. Add to this new Block a Set Audio Volume Command.

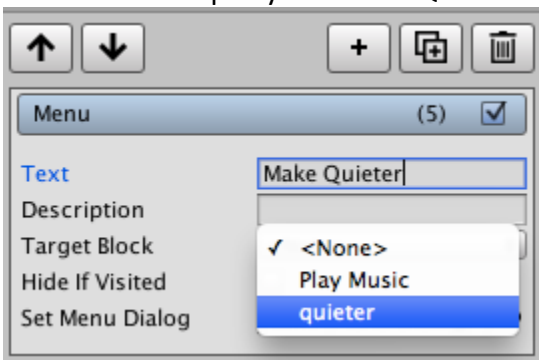


7. In the properties of the Set Audio Volume Command set the volume to 0.25 (25%).

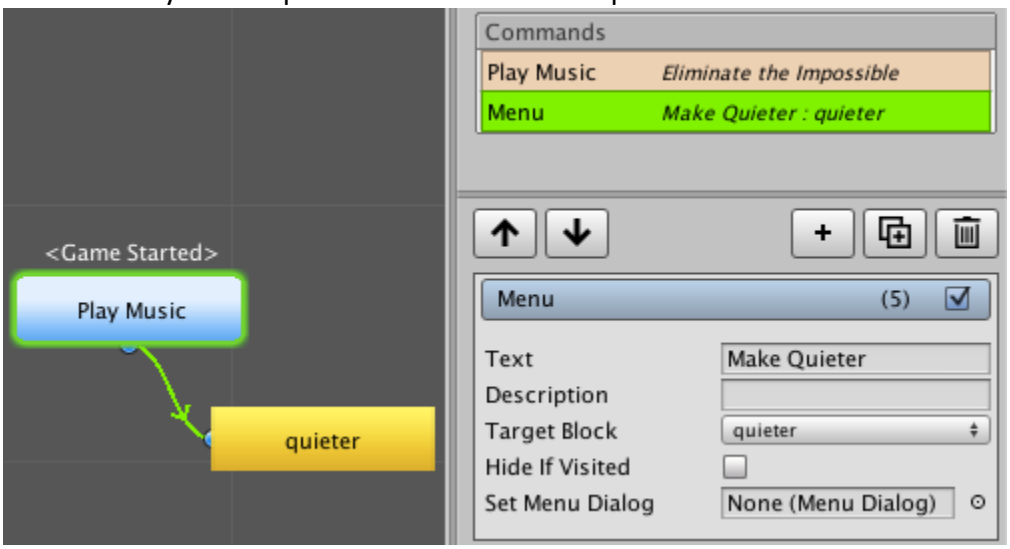
8. Select the "Play Music" block, and add a Menu command by clicking the plus-sign add Command button in the Inspector and then choosing menu: Narrative | Menu.



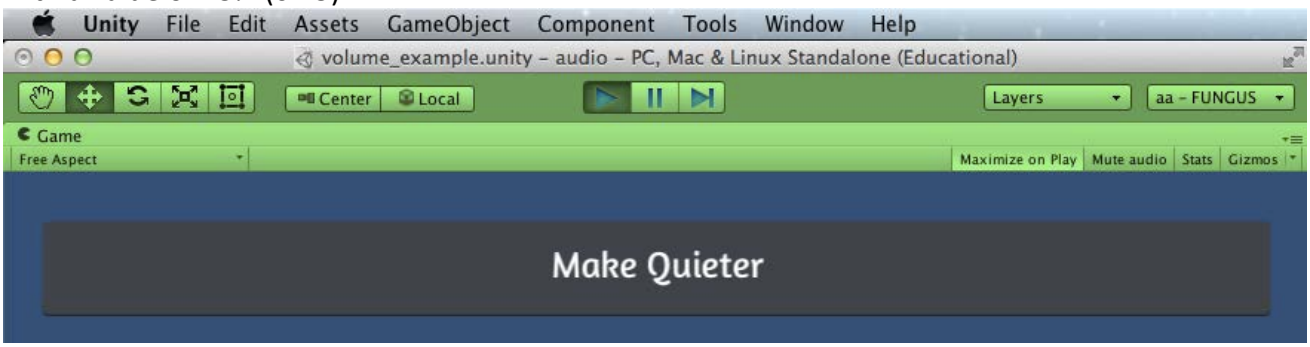
9. Set the Text Property to "Make Quieter" and the Target Block to Block "Quieter":



10. In the Flowchart window you should now see a green arrow, showing that a Command in Block "Play Music" passes control to Block "quieter":



11. When you run the scene, the music will start playing at full volume, then when y ou click the "Make Quieter" button, control will pass to the "quieter" block, and you'll hear the music become much quieter after the Set Audio Volume Command has been executed with a value of 25% (0.25):



The 3 Unity audio concepts

Unity has 3 different kinds of Audio 'object', that it is worth understanding when working with audio in Fungus (or any other) Unity project:

1. Audio Clip
2. Audio Listener
3. Audio Source

Unity Audio Clip

Unity uses the term Audio "Clip" to refer to the physical sound files (.mp3, .wav, .ogg etc.) that are stored in your Project folder. It is these Audio Clip files that you drag and drop into the

"Music Clip" and "Sound Clip" properties in the Inspector Window, when creating Play Music and Play Sound Commands in a Fungus Block.

Unity Audio "Listener"

Basically, if you want sound to be played there must be an Audio Listener component inside one of the gameObjects in your scene. The Main Camera of a scene has one by default, so in most cases you just leave this alone and can rest assured that you have an Audio Listener.

If a scene has no Audio Listener in any gameObject, then no audio will be heard by the user of the game, regardless of how many music and sound clips might be playing.

Sometimes you may add gameObjects to your scene that contain another Audio Listener component. In this case, Unity will present a warning message stating that more than 1 Audio Listener is present in the scene. If you see such a message, then its best to resolve this problem by disabling all but one Audio Listener...

If you are working with a 3D game, and/or you wish to present a sophisticated stereo sound experience for your user, then you may need to learn about 3D audio. In such games the 3D "position" of the gameObject containing the Audio Listener becomes important - but don't worry about this if you are just getting started with audio in Fungus. For 3D effects the Audio Listener is like an "electronic ear", so its location determines things like how loud a sound is played (distance from "ear") and left-right stereo balance (which "side" audio is to the "ear") etc.

Unity Audio Source

In Unity the link between an Audio Clip (music/sound) file that we wish to be played, and the Audio Listener in the scene is a Unity Audio Source component of a gameObject. However, in most cases Fungus creates one of these if needed, so we don't need to worry about them!

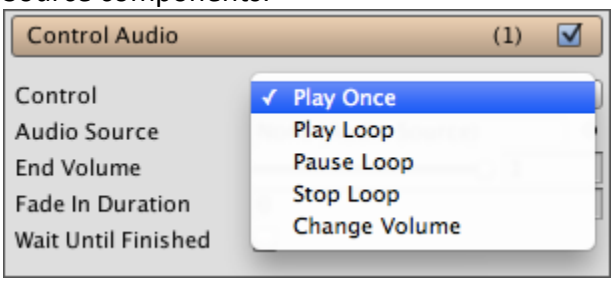
However, for sophisticated control of music and sound and speech in your game there is the facility to make Fungus have detailed control of Unity Audio Sources. It is an Audio Source component that controls how and when and which part of an audio clip is playing (and whether it should loop or not), and whether it is playing or paused, and when resuming should continue from where paused or restart. The volume of a playing clip can also be controlled by properties of an Audio Source.

Learn more about audio in Unity at the Unity Manual Audio Page.

Control Audio command

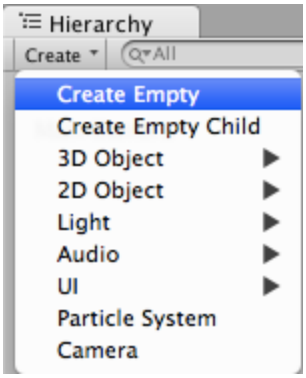
The Fungus Audio Commands cover all common music and sound effect actions, for specialist audio work you may need to access the raw (and complicated) power of Unity audio. The Fungus Command "Control Audio" lets Fungus Blocks communicate directly with Unity Audio Source components in any gameObject in the current scene, so your wizard audio team member can do what they need to do with the Unity toolkit, and you can still control playing / looping / volume etc. of the audio in those complex gameObjects.

The Control Audio Command offers five actions that can be communicated to Unity Audio Source components:



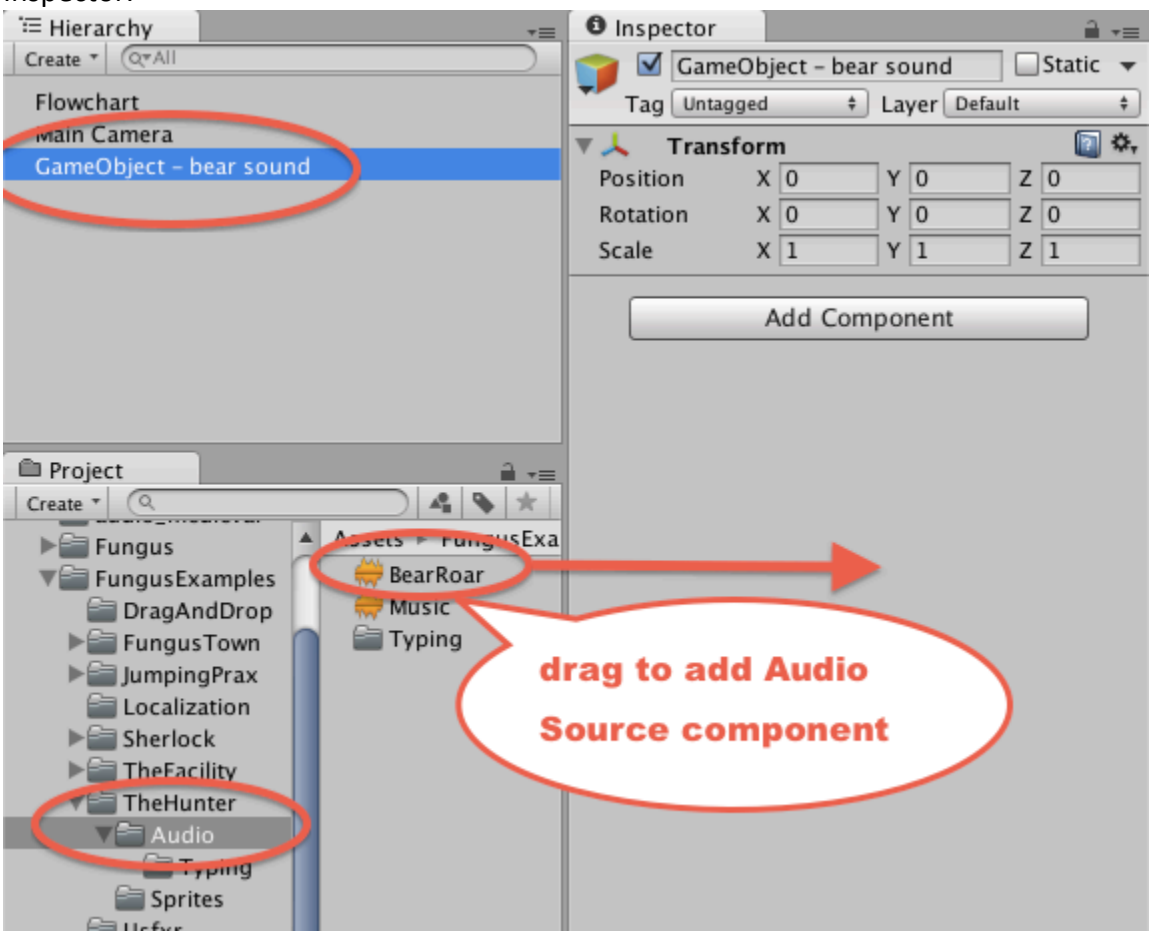
We'll learn about this with a simple Unity gameObject that plays a bear sound two times, first at full volume and then reduces the volume before playing a second time, using three "Control Audio" Fungus Commands. Do the following:

1. (setup) If you have not already done so: Create a new scene, add a Fungus Flowchart to the scene, and select the Block in the Flowchart.
2. Rename this Block "Control Audio".
3. In the Hierarchy Window create a new Empty gameObject, choose menu: Create | Create Empty:



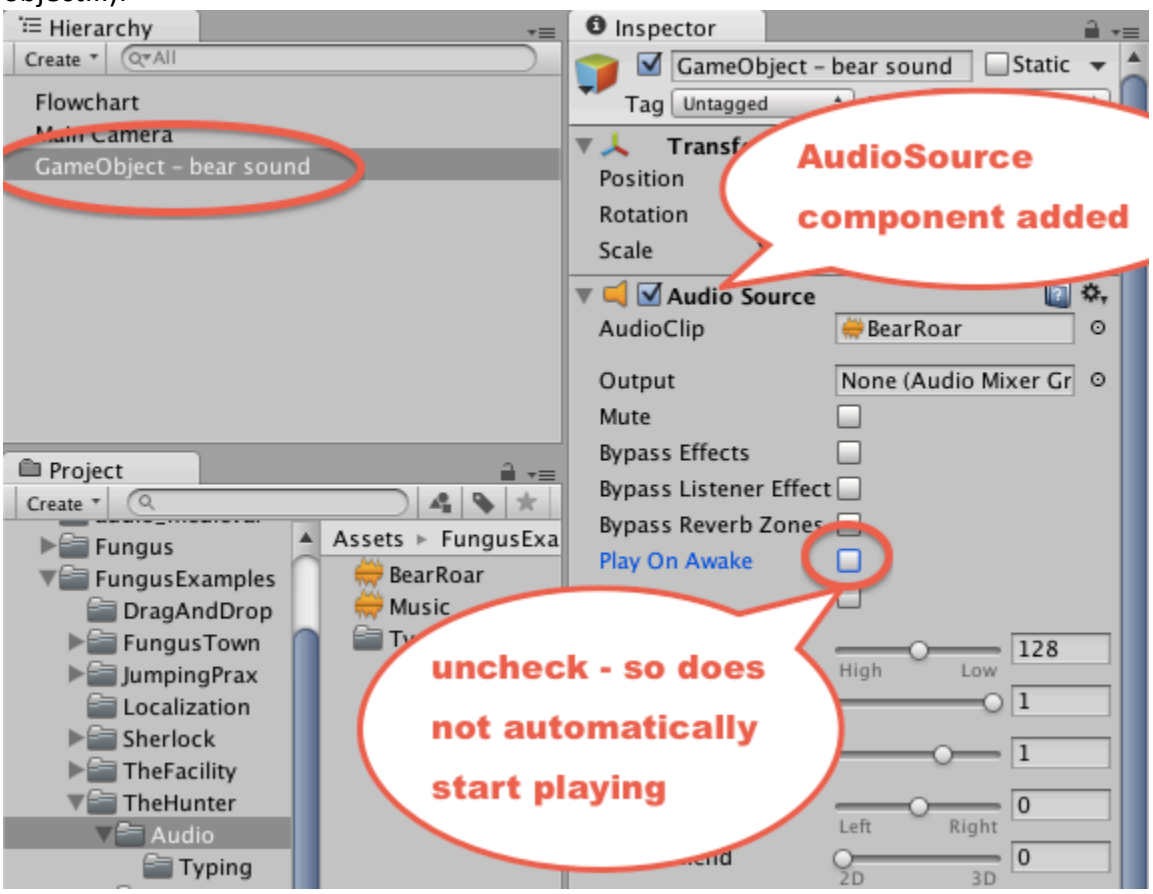
4. Rename this new empty gameObject "GameObject - bear sound" (or whatever sound name makes sense in your project).
5. Locate your desired audio clip file in the Project window, and (with "GameObject - bear sound" selected), drag a reference to the audio clip from the Project window into the

Inspector:

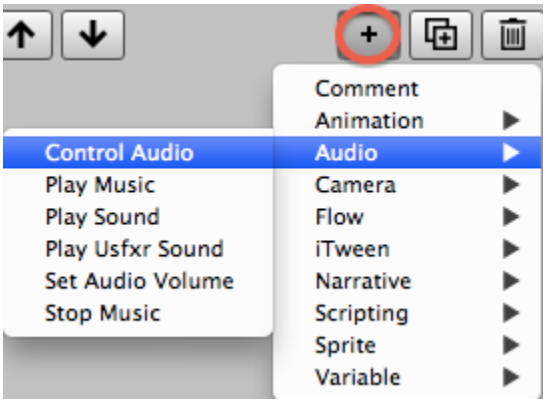


6. An Audio Source component should now be added in the Inspector to "GameObject - bear sound". Un-check the Play On Awake property in the Inspector (to stop this sound from playing as soon as the scene begins - **we** want to be in control of this Audio Source

object...):



7. With your Flowchart Block selected, click the Add Command button in the Inspector and choose menu: Audio | Control Audio.



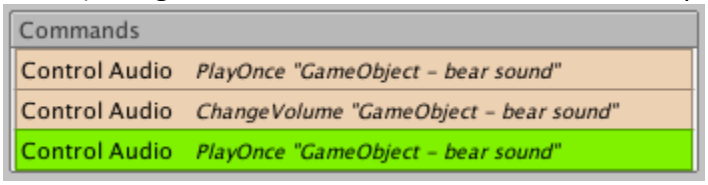
8. Drag the "GameObject - bear sound" gameObject from the Hierarchy into the Audio Source property in the Inspector, and select the Wait Until Finished checkbox (so Fungus

will wait for the sound to finish playing before moving on):

9. Note that the default Control action for a Control Audio Command is "Play Once" - we'll leave this property unchanged, since we want our bear sound inside our gameObject to be played once.
10. If you run the scene now, you'll hear the bear sound play once at full volume.
11. Since all three of the Control Audio commands we need use a link to "GameObject - bear sound" the fastest workflow is to **duplicate** each Command, and just change the bits we need. So duplicate your Control Audio command by clicking the Duplicate Command button:

12. In the newly copied command, change the Command action to Change Volume, and choose a volume of 0.25 (25%):

13. Once again, make a duplicate of the first Control Audio Command (that Plays the sound). Drag this new Command to be last in the sequence of commands.



14. Now when you play the scene, first the bear sound linked to in the Audio Source component of gameObject "GameObject - bear sound" should play at full volume, then (after having its volume reduced to 25%) it should play a second time at a much reduced volume.

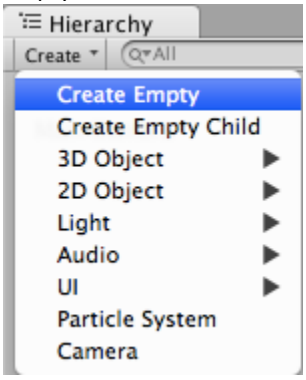
Audio Tags (in Say Commands)

Since often dialog authors will know just what sound effects or music they wish to associate with their characters utterances, Fungus allows audio-related 'tags' to be embedded in the text of Say Commands. There are four audio related tags:

{audio=AudioObjectName} Play Audio Once
{audioloop=AudioObjectName} Play Audio Loop
{audiopause=AudioObjectName} Pause Audio
{audiostop=AudioObjectName} Stop Audio

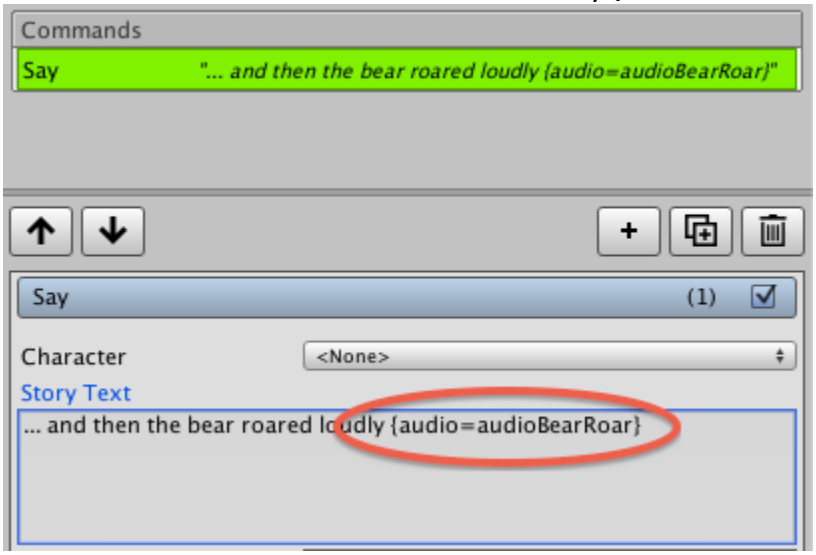
Using tags like this make it important to carefully **name** the gameObjects in the Hierarchy window. To explore how to control sounds in gameObjects do the following:

1. (setup) If you have not already done so: Create a new scene, add a Fungus Flowchart to the scene, and select the Block in the Flowchart.
2. Rename this Block "Say audio tags".
3. In the Hierarchy Window create a new Empty gameObject, choose menu: Create | Create Empty:

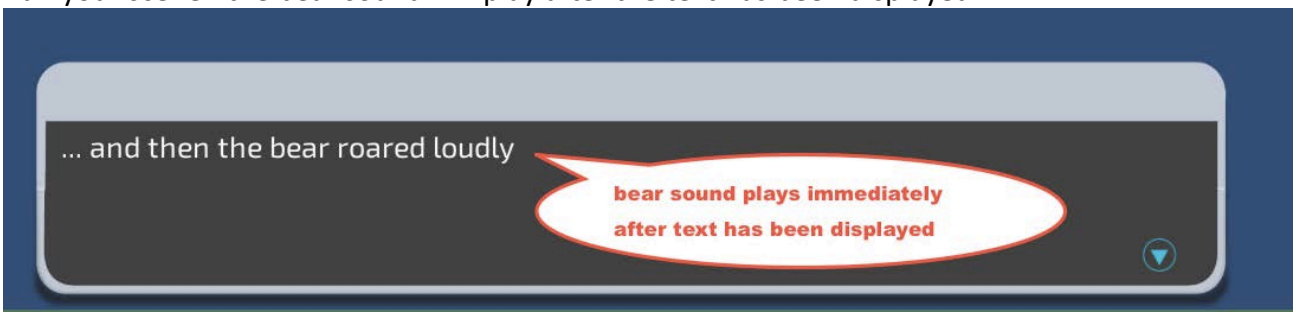


4. Rename this new empty gameObject "audioBearRoar", or whatever sound name makes sense in your project, but avoid spaces in the name of this game object.

5. Locate your desired audio clip file in the Project window, and (with "audioBearRoar" selected), drag a reference to the audio clip from the Project window into the Inspector.
6. Un-check the Play On Awake property in the Inspector for the Audio Source component (to stop this sound from playing as soon as the scene begins - **we** want to be in control of this Audio Source object...):
7. With your Flowchart Block selected, click the Add Command button in the Inspector and choose menu: Narrative | Say. Enter the following for the Story Text property of this Say Command "... and then the bear roared loudly {audio=audioBearRoar}":



8. Run your scene - the bear sound will play after the text has been displayed:



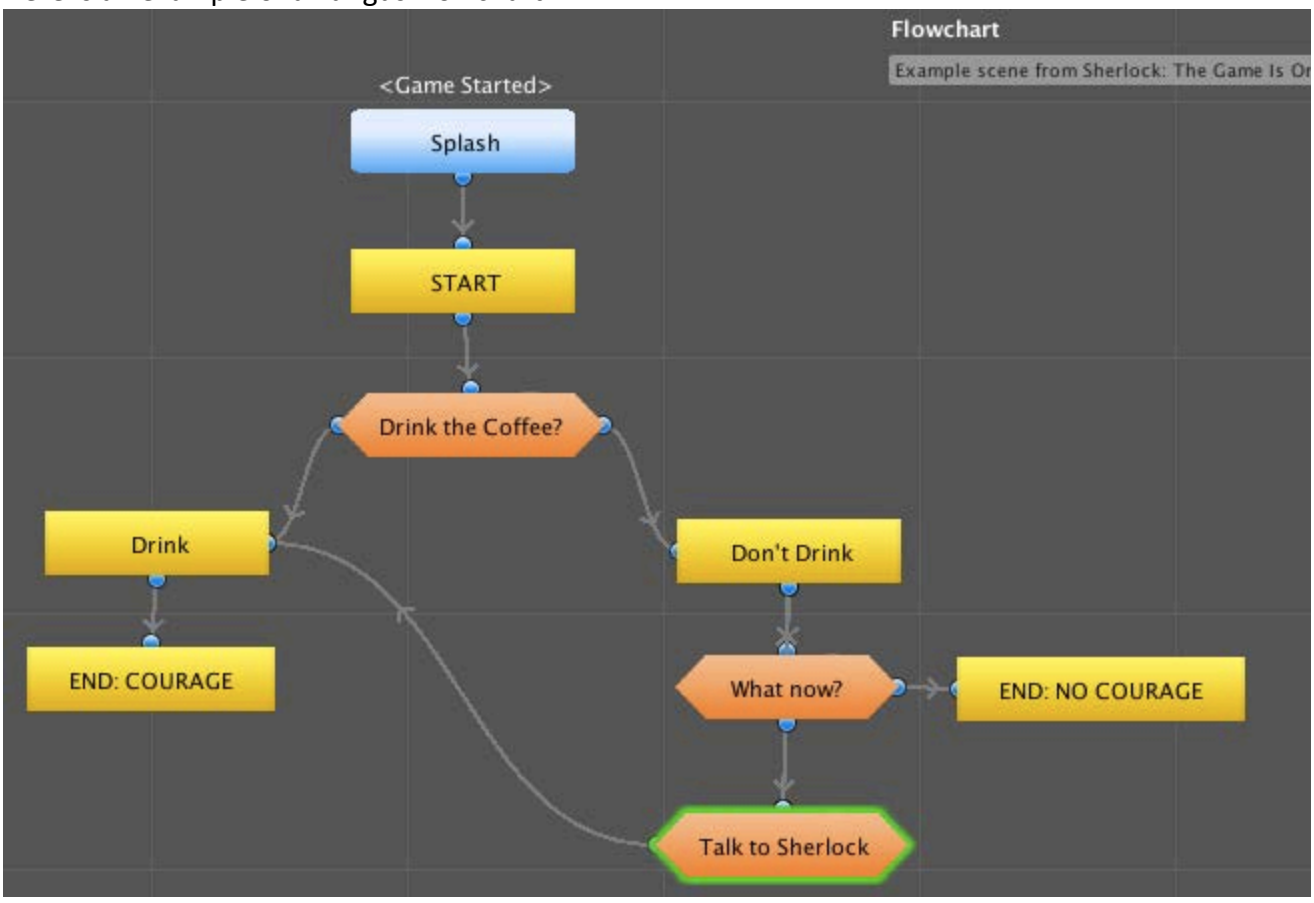
Fungus Fundamentals - Flowcharts

A fundamental concept of Fungus is the **Flowchart**. Scenes can contain a single Flowchart, or multiple Flowcharts.

What is a Flowchart?

A Fungus **Flowchart** contains the Blocks in which all your Fungus Commands are located. A Unity scene can contain multiple Flowcharts, and commands can be executing simultaneously in different Flowcharts. However, for many games it is sufficient for one Block in one Flowcart to be executing at any one time.

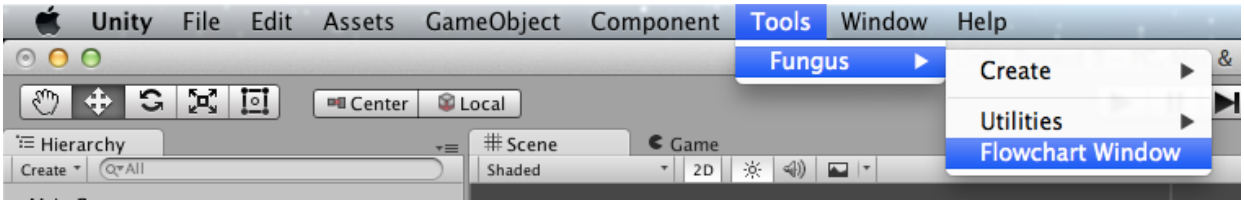
Here is an example of a Fungus Flowchart:



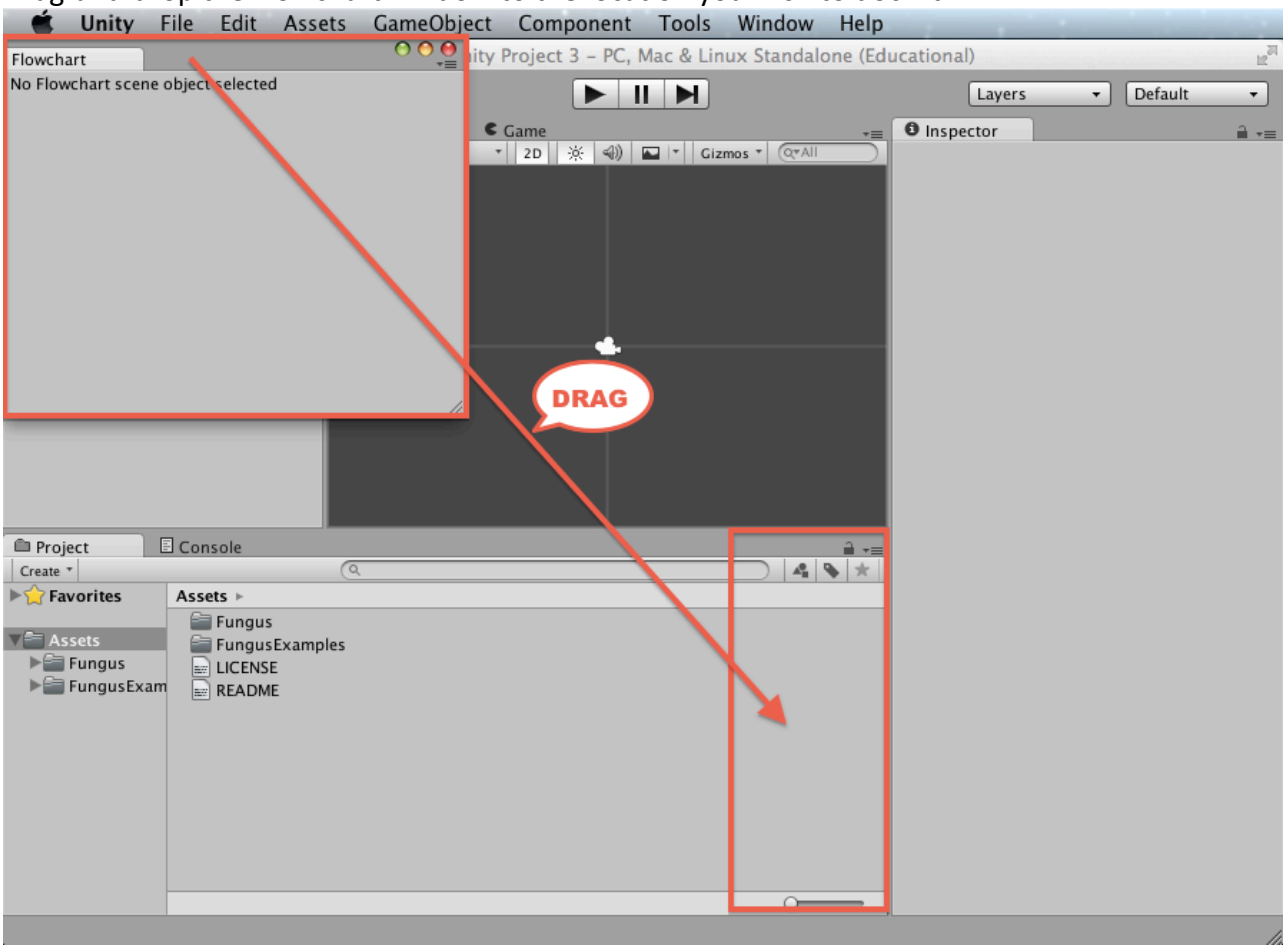
Opening and docking the Flowchart window

You'll need the Fungus Flowchart window when working with Fungus. Open and dock this window somewhere handy by following these steps:

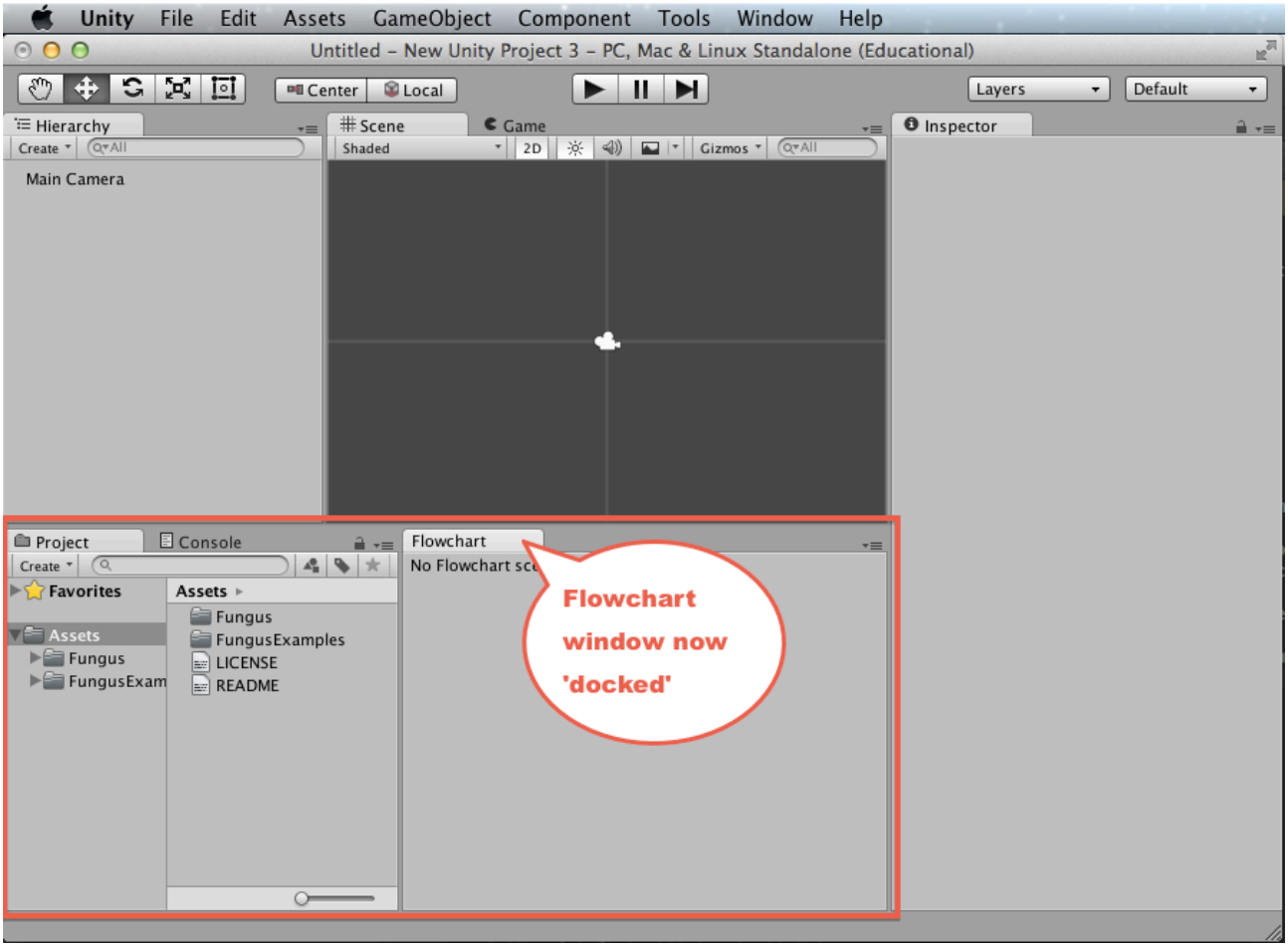
1. Choose menu: Tools | Fungus | Flowchart Window



2. Drag-and-drop the Flowchart window to the location you wish to dock it:



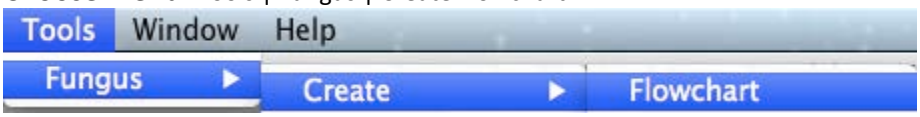
3. The Flowchart window is now docked and part of your Unity window layout:



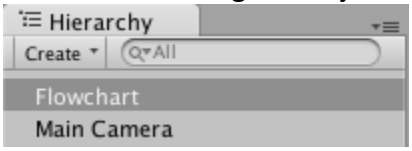
Creating a Flowchart

To create a Fungus Flowchart do the following:

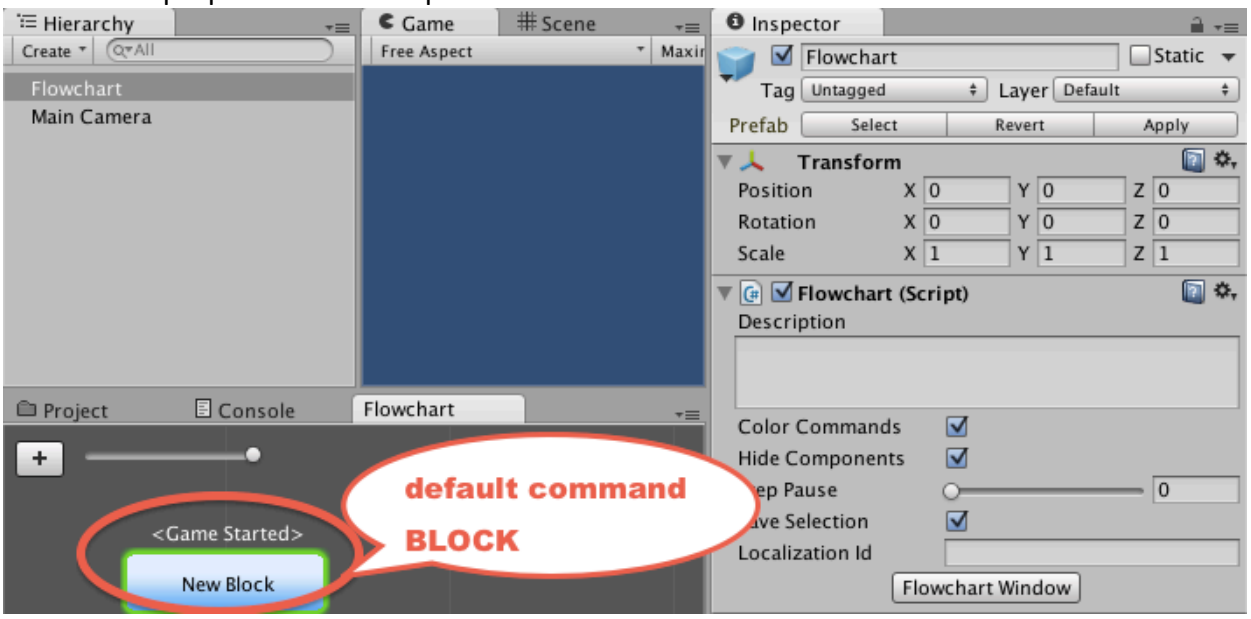
1. Choose menu: Tools | Fungus | Create Flowchart



2. A new **Flowchart** gameObject should appear in the Hierarchy window.



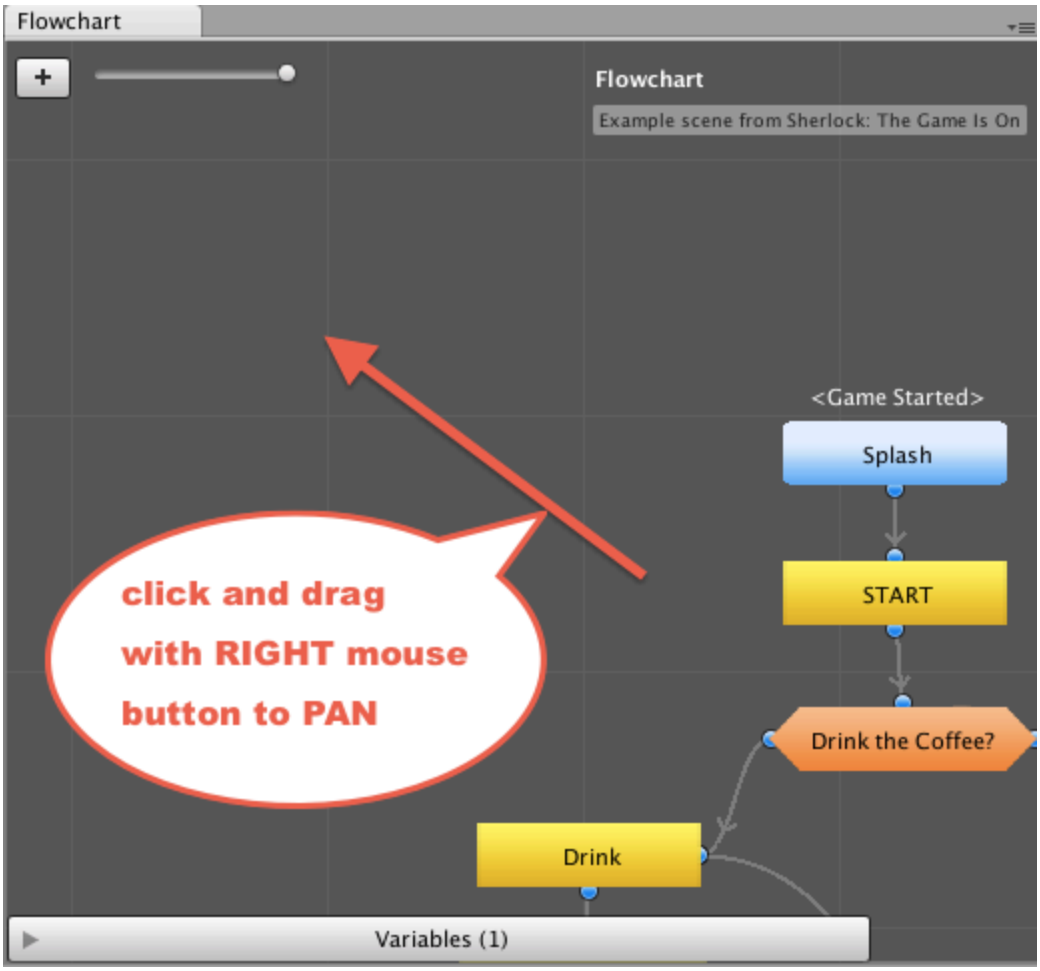
3. Select the **Flowchart** gameObject in the Hierarchy window, and you'll see the **Flowchart's** properties in the Inspector Window:

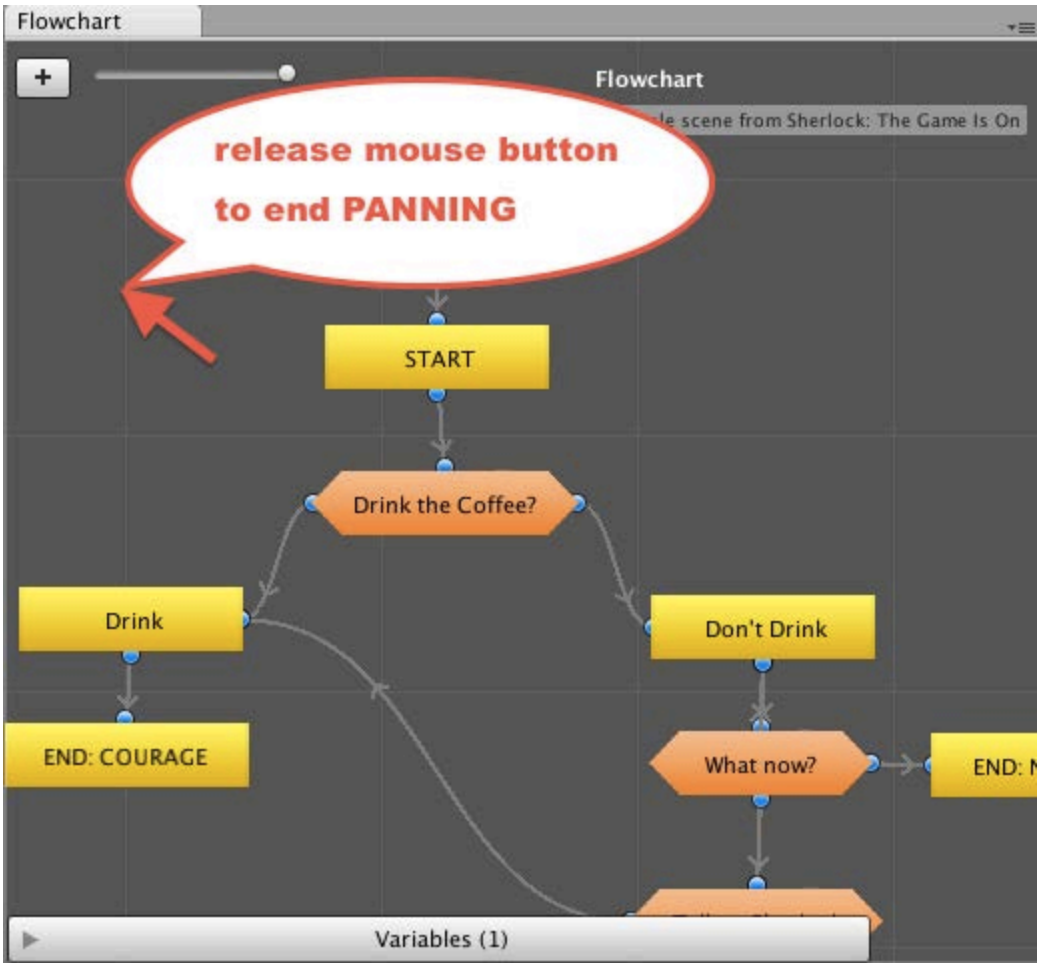


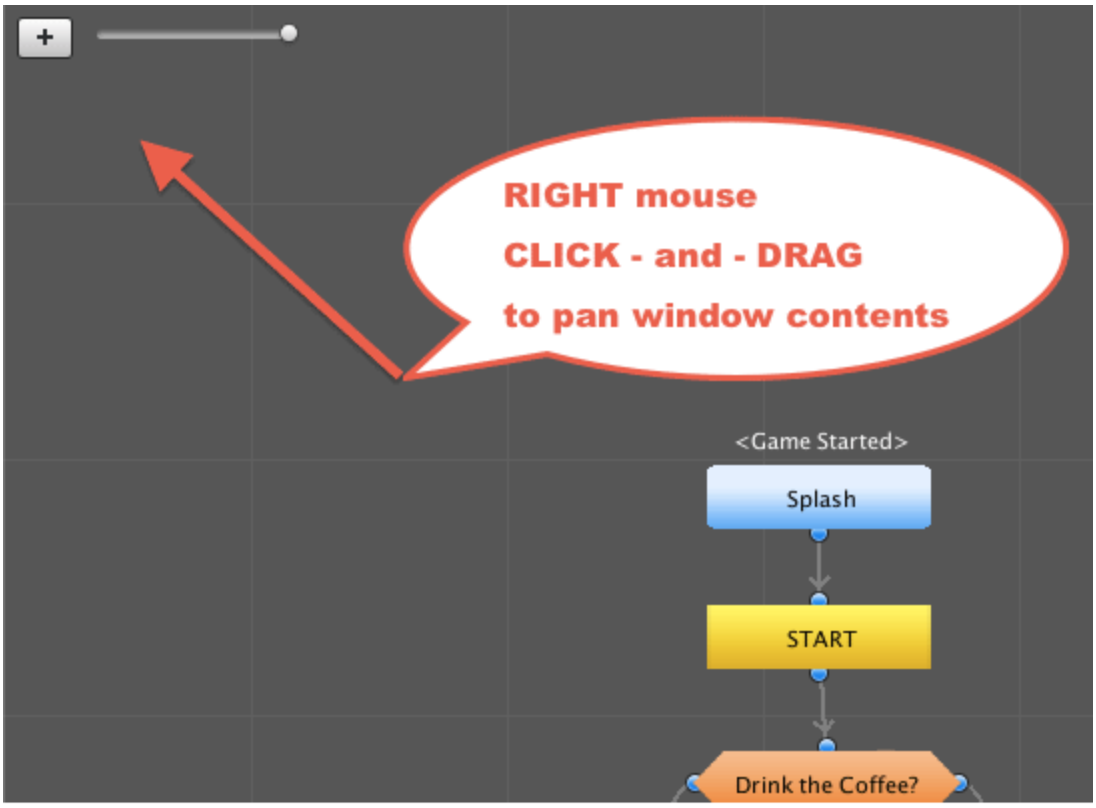
4. If you have not already displayed the Flowchart Window, you can do so by clicking the Flowchart Window button in the Inspector.
5. As you can see, when a new Flowchat is created a single command Block named "New Block" is automatically created, with the Event handler "Game Started" (so it will start executing Fungus commands as soon as the scene goes into **Play Mode**).

Panning the Flowchart window

Panning means moving the contents of the Flowchart window as if they are on a piece of paper. Click and drag with the RIGHT mouse button to pan the contents of the Flowchart window.

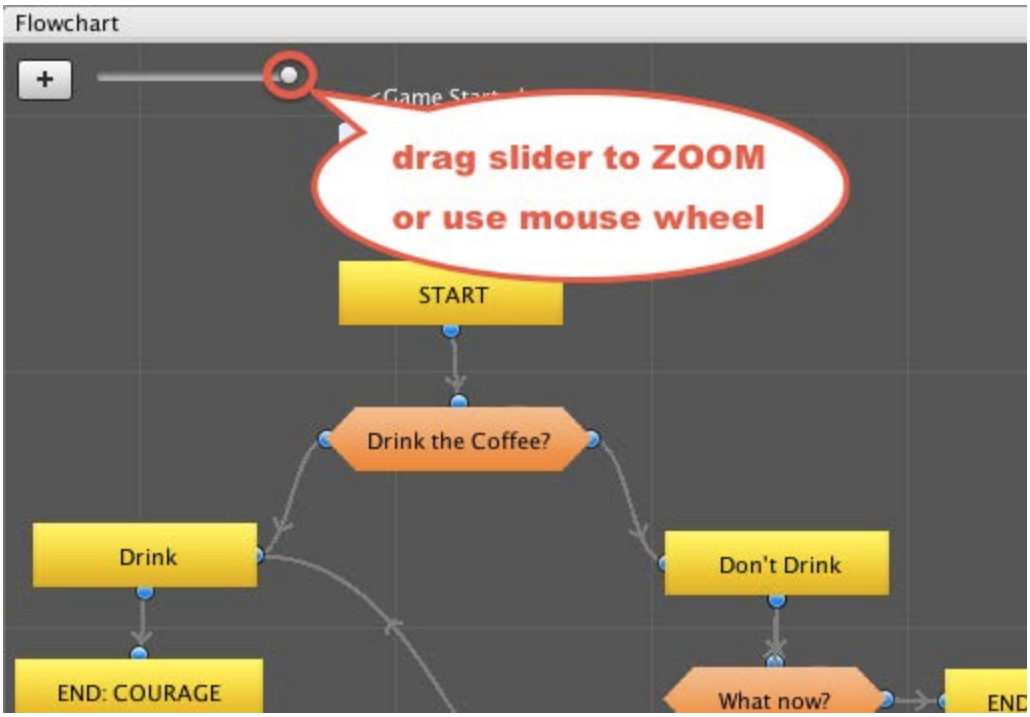

Flowchart
Flowchart
Example scene from Sherlock: The Game Is On
<Game Started>
Splash
START
Drink the Coffee?
Drink
click and drag
with RIGHT mouse
button to PAN
Variables (1)

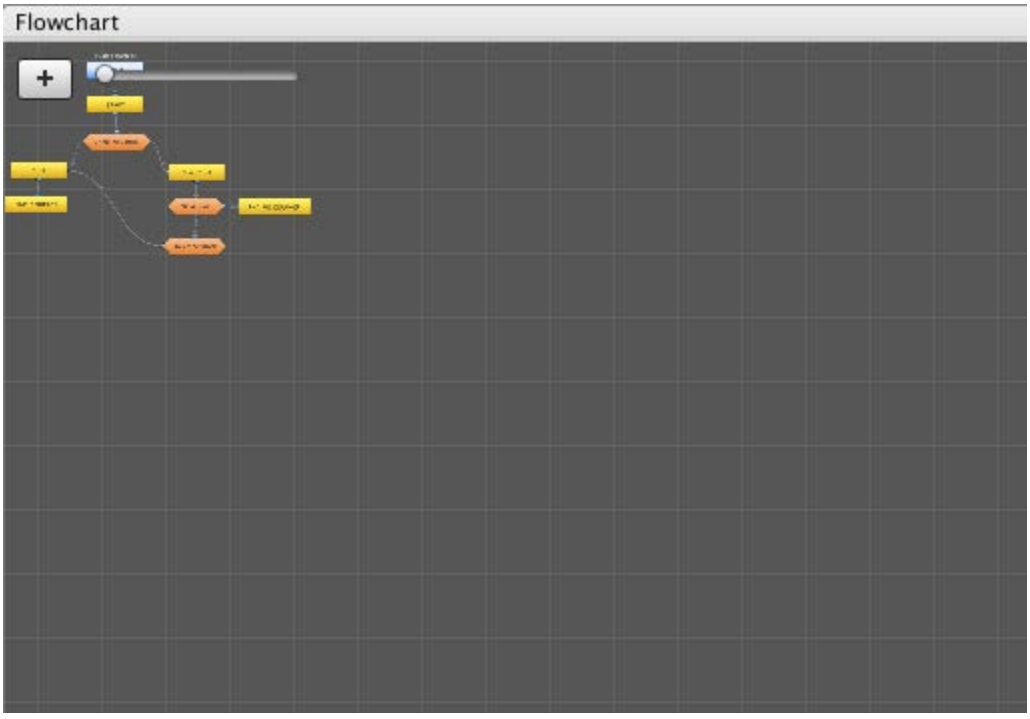

Flowchart
Flowchart
release mouse button
to end PANNING
START
Drink the Coffee?
Drink
Don't Drink
END: COURAGE
What now?
Variables (1)



Zooming the Flowchart window

Zooming refers to making the contents larger or smaller. To zoom the Flowchart window contents either click and drag the UI slider, or use the mouse wheel (or trackpad).


Flowchart
drag slider to ZOOM
or use mouse wheel
START
Drink the Coffee?
Drink
Don't Drink
END: COURAGE
What now?


Flowchart

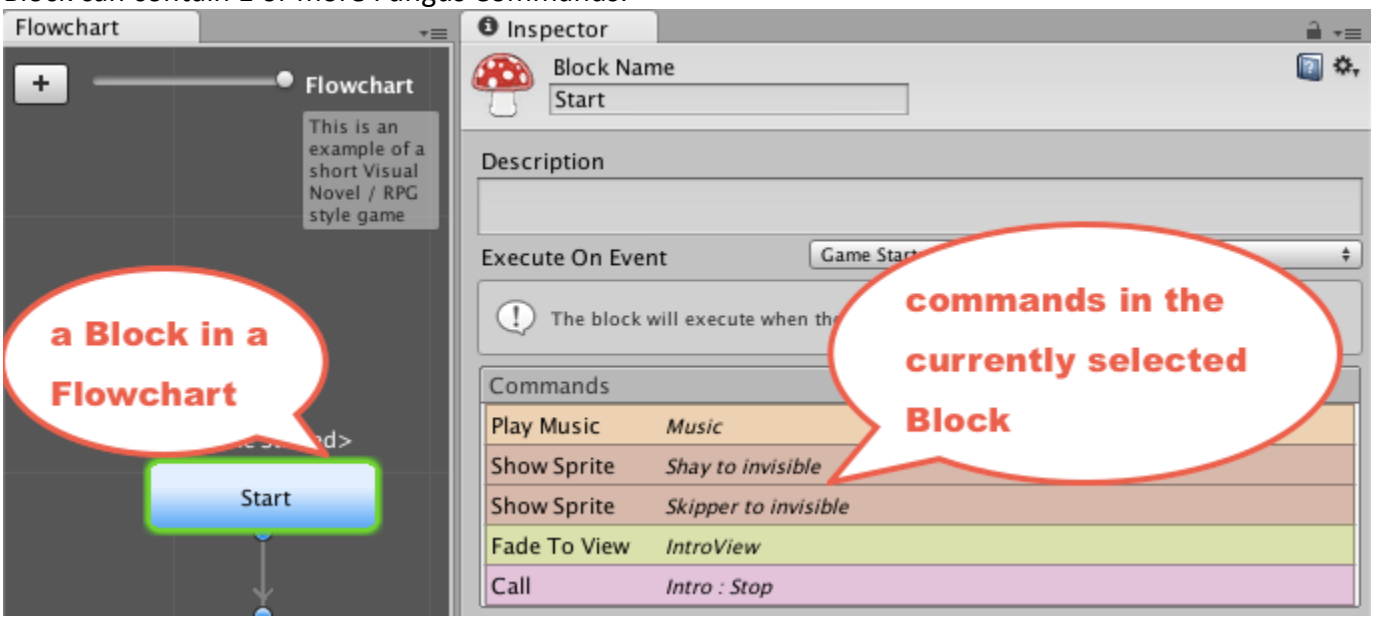

Flowchart
drag slider to ZOOM
or use mouse wheel
START
Drink the Coffee?
Drink
Don't Drink
END: COURAGE
What now?

Fungus Fundamentals - Blocks

A fundamental concept of Fungus is the **Block**. Blocks contain your Fungus Commands, and reside inside Flowcharts.

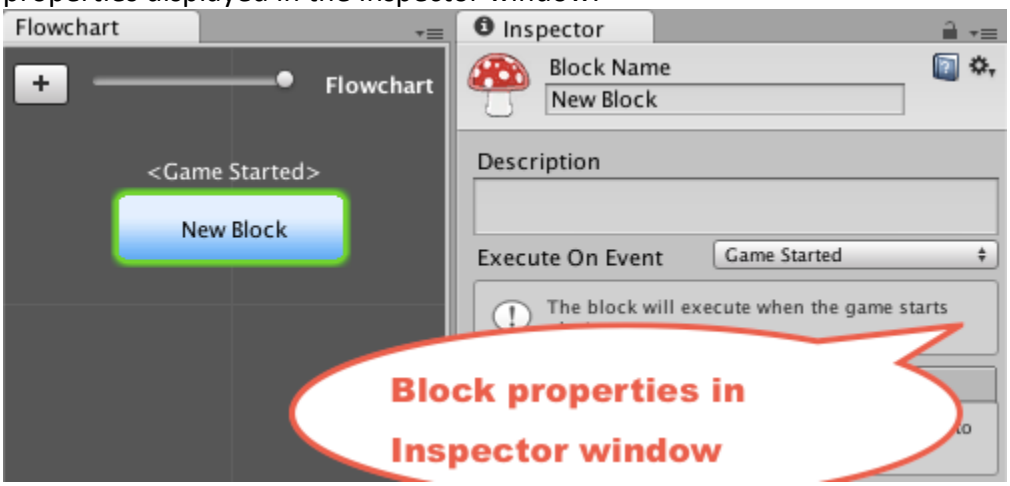
Blocks (and how to inspect Block properties)

Blocks are found inside Flowcharts. Blocks are where your Fungus Commands are stored. Each Block can contain 1 or more Fungus Commands:



To inspect the properties of a Block do the following:

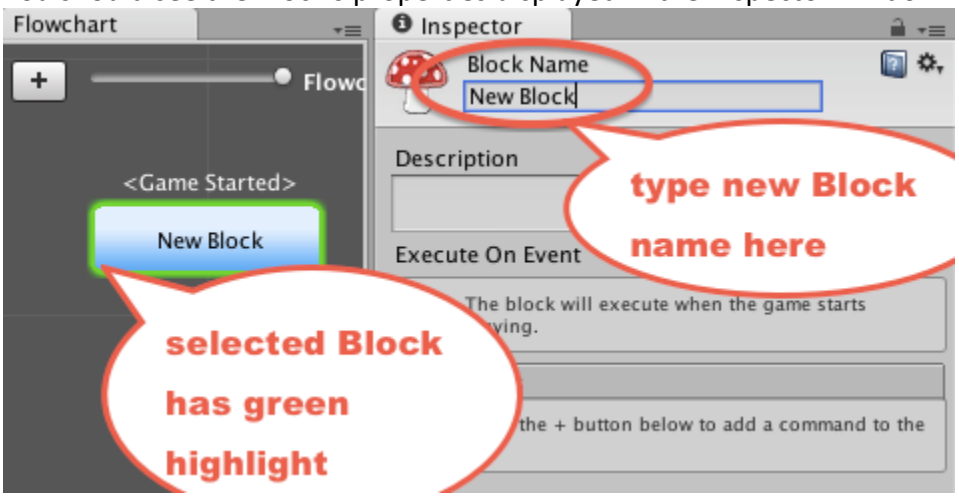
1. (setup) Create a Fungus Flowchart.
2. Click to select the default Block in the new Flowchart. You should see the Block's properties displayed in the Inspector window:



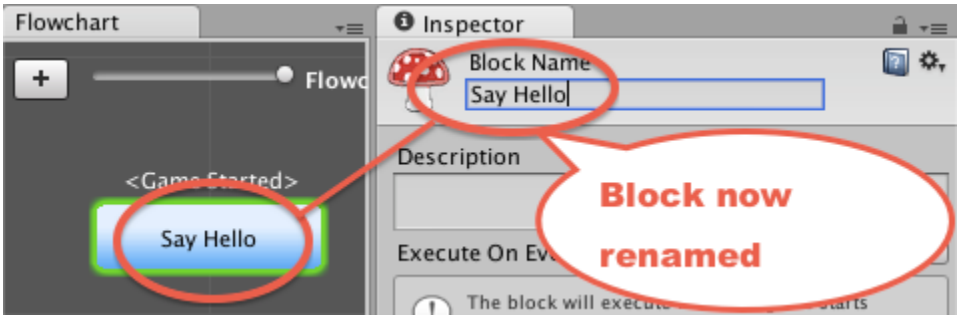
Setting Block name and description

When working with more than one Block, its important to name each Block in a meaningful way. To rename a Block do the following:

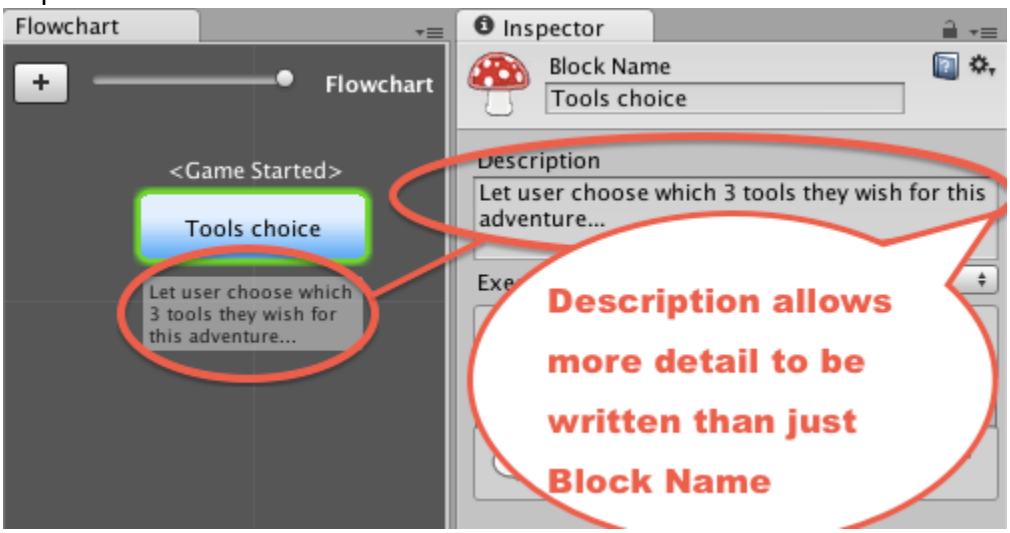
1. (setup) Create a Fungus Flowchart.
2. Click to select the default Block in the new Flowchart. The selected block has a green outline.
3. You should see the Block's properties displayed in the Inspector window:



4. In the Inspector change the text for the Block Name property to "Say Hello".
5. You should now see the Block has been renamed in the Flowchart window:



6. Now add a detailed description about the Block in the Description property in the Inspector window:

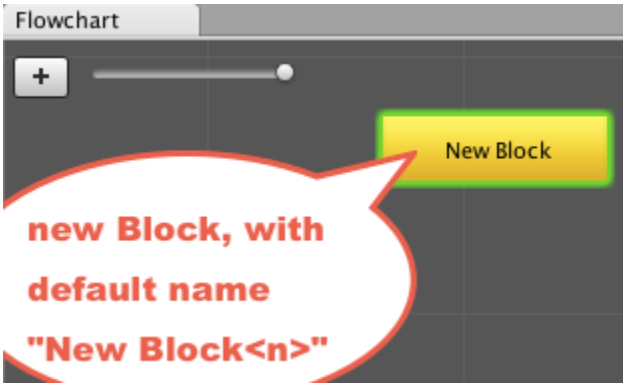


Creating a block

To create a new Block do the following:

1. (setup) Create a Fungus Flowchart (or be viewing the Flowchart for your current project).
2. Click the Add New Block button (the plus-sign "+") in the top-left of the Fungus Flowchart window:

3. A new Block should have been added to your Flowchart (with the default name "New Block", or "New Block1/2/3 etc." so each name is unique)

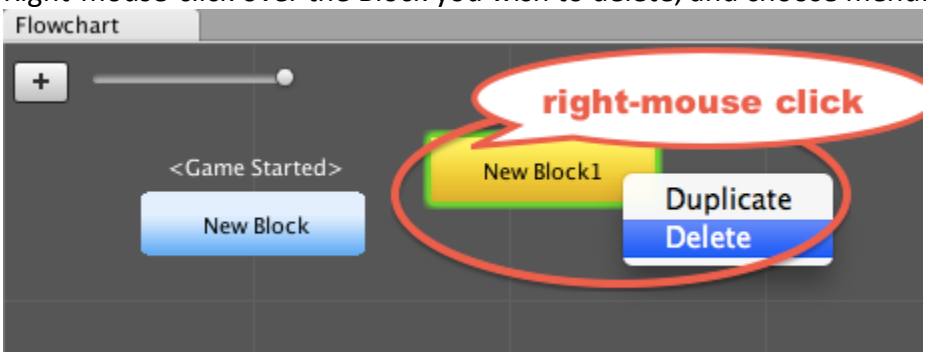


Note - a good time to choose a meaningful name a Block is immediately after creating a new Block ...

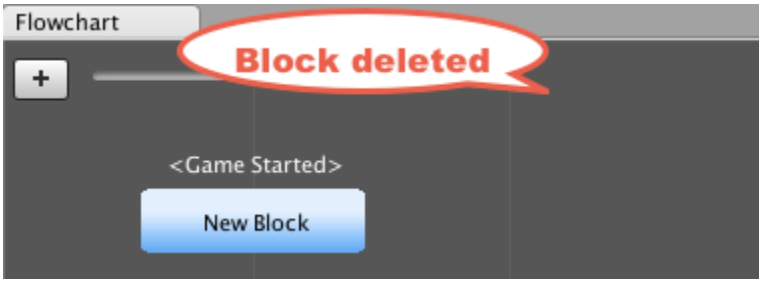
Delete a Block

To delete a Block from the current Flowchart, do the following:

1. (setup) Create a Fungus Flowchart (or be viewing the Flowchart for your current project).
2. Right-mouse-click over the Block you wish to delete, and choose menu: Delete:



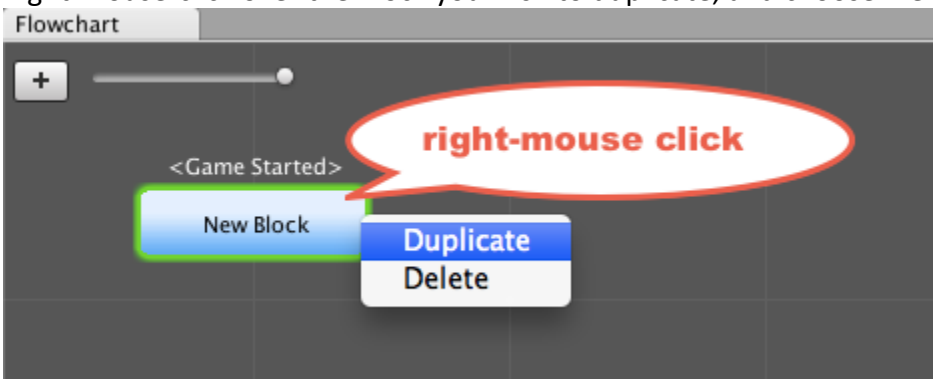
3. The Block should now have been removed from the Flowchart:



Duplicate a Block

To duplicate (clone / make an exact copy of) a Block from the current Flowchart, do the following:

1. (setup) Create a Fungus Flowchart (or be viewing the Flowchart for your current project).
2. Right-mouse-click over the Block you wish to duplicate, and choose menu: Duplicate:



3. A copy of the Block should now have been added to the Flowchart (with "(copy)" appended the name of the duplicate):

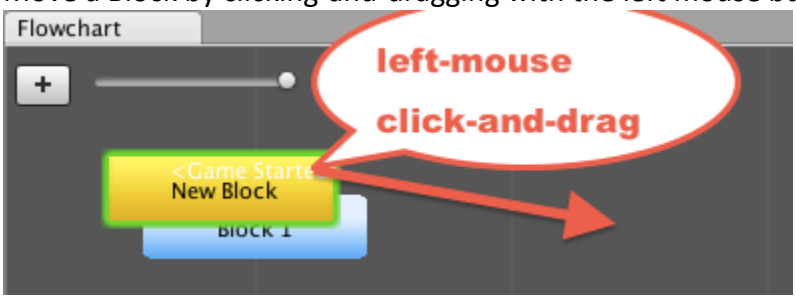
Note - a good time to choose a meaningful name a Block is immediately after duplicating one ...

Moving blocks

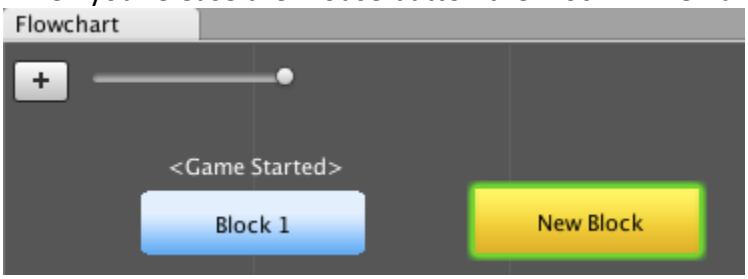
To move / rearrange Blocks in the Flowchart window do the following:

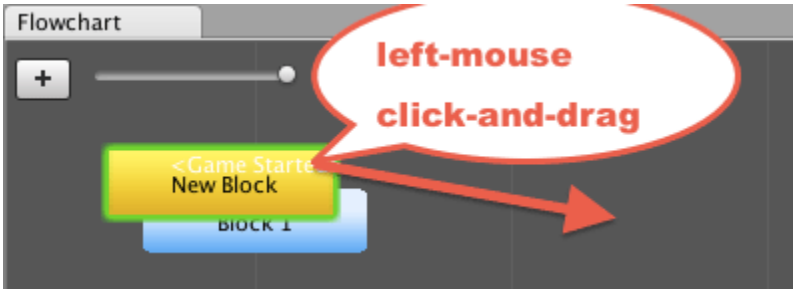
1. (setup) Create a Fungus Flowchart (or be viewing the Flowchart for your current project).

2. Move a Block by clicking-and-dragging with the left mouse button:



3. When you release the mouse button the Block will remain where it was dragged:





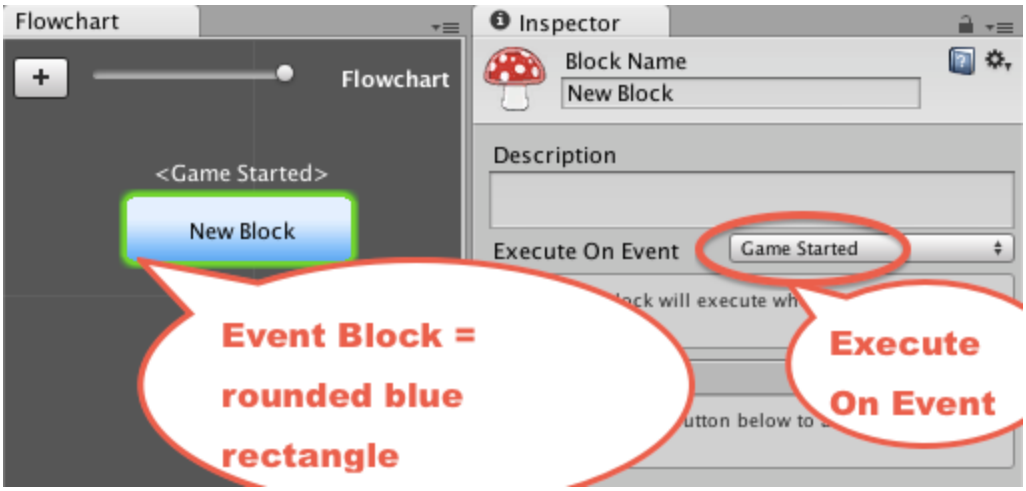
The 3 types of Block (Event Block, Branching Block, Standard Block)

Fungus Flowcharts visually differentiate three differents kinds of Block behavior:

1. Event Block - blue rounded rectangle (Block execution triggered by an event)
2. Branching Block - orange polygon (passes control to 2 or more other Blocks, and not and Event Block)
3. Standard Block - yellow rectangle (no event, passes control to 0 or 1 other blocks)

1: Event Block - blue rounded rectangle

One way Fungus decides when to start executing the Commands inside a Block is if a Block has defined an Execute On Event. If a Block has any of the possible Execute On Events chosen (such as Game Started, Message Received, Sprite Drag Completed, Key Pressed etc.) then this Block will be displayed as a **blue rounded rectangle**:

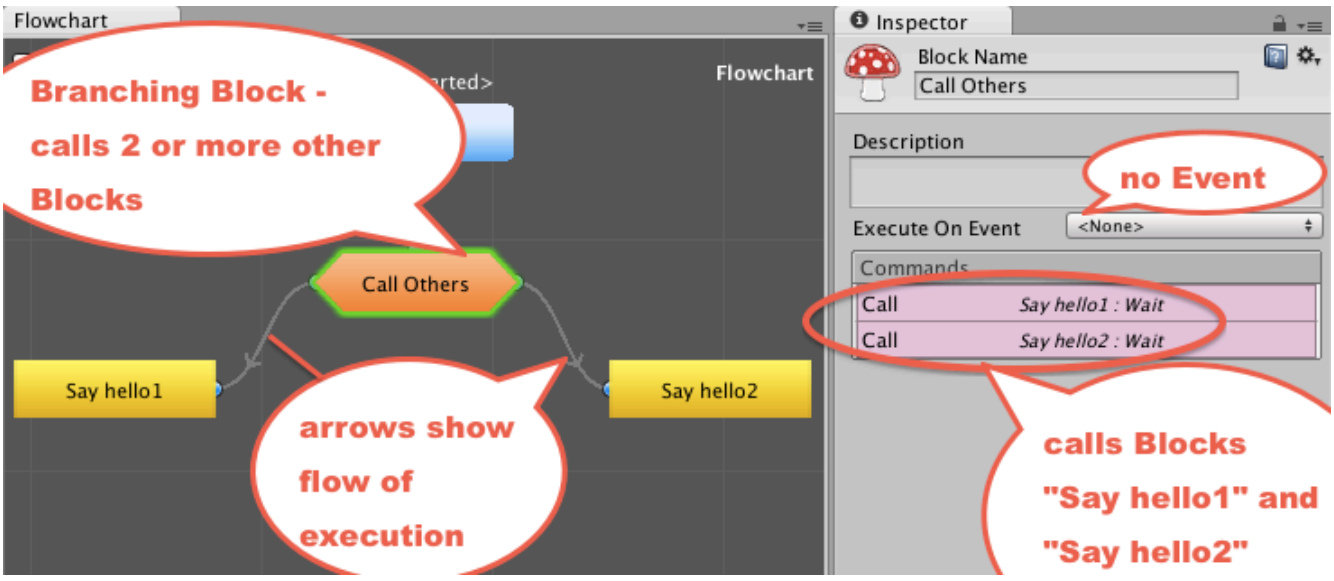


As we can see, the default Block created with every new Flowchart automatically defines the Game Started Execute On Event, so this default Block always is displayed as an Event Block.

NOTE: A Block with an Execute On Event will always appear as an Event Block in the Flowchart, regardless of whether its Commands contain menus or calls to 2 or more Blocks). In other words, in terms of Flowchart appearance, display of an Event Block overrides display of a Branching Block.

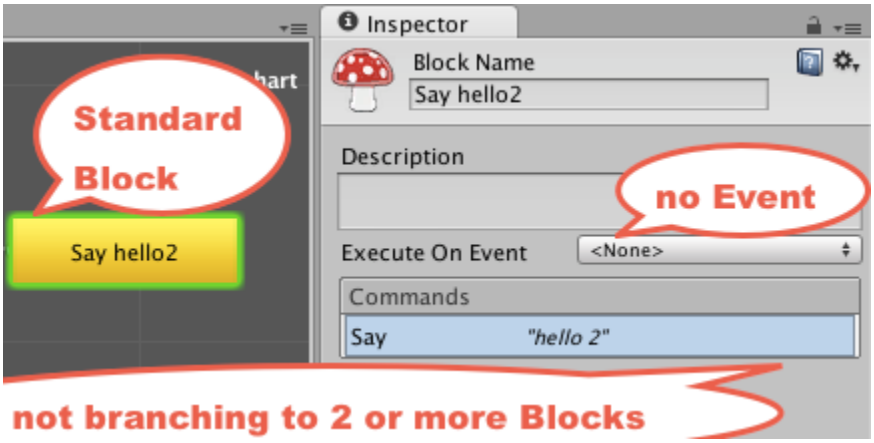
2: Branching Block - orange polygon

If a Block is *not* an Event Block, and its Commands include Calls and/or Menu commands to execute Commands in 2 or more other Blocks, then in the Flowchart window this Block will be displayed as a Branching Block, visually appearing as an **orange polygon**.



3: Standard Block - yellow rectangle

If a Block is *neither* an Event Block *nor* a Command Block, then it is a Standard Block, and will be displayed as an **yellow rectangle**.



Connections between Blocks: Flow of Execution

A whole scene's logic can rarely fit all into a single Block, therefore we need ways for one Block to pass execution control to other Blocks. There are several ways to do this, the most common being:

1. the Call Command
2. the Menu Command

Executing Commands in another Block with the Call Command

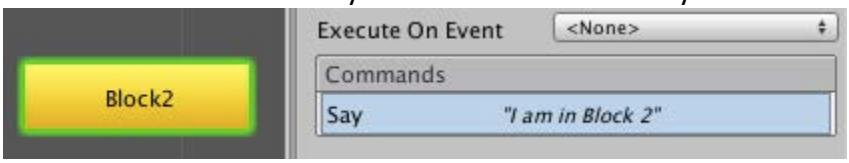
The Call Command tells Fungus to go and start executing the Commands in named Block. There are several ways to do this, we can tell Fungus to Stop execution completely in the current Block, and just pass control to named Block. We can also tell Fungus to go and completed all Commands in the named Block, and when they are finished, to then continue executing any remaining commands in the current Block. Finally, and perhaps the most complicated/sophisticated technique, we can tell Fungus to both started executing Commands in a named Block WHILE simultaneously continuing to execute remaining Commands in the current Block.

To pass control to another Block, and stop executing Commands in the current Block, do the following:

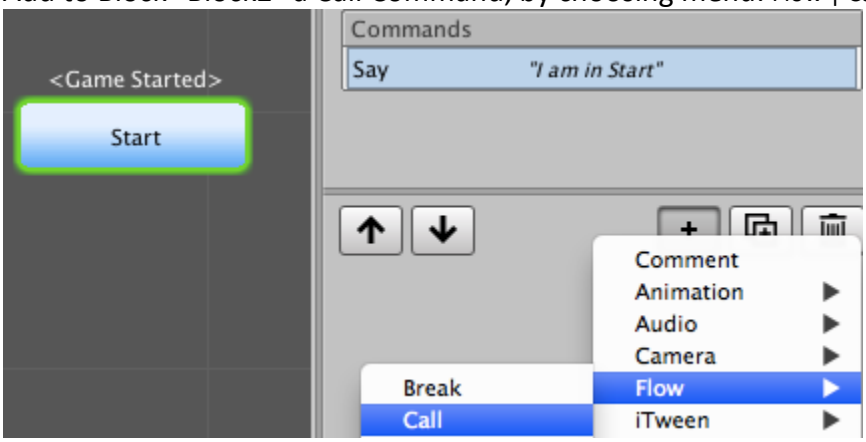
1. (setup) If you have not already done so: Create a new scene, add a Fungus Flowchart to the scene, and select the Block in the Flowchart.
2. Rename this Block "Start".

3. Add to Block "Start" a Say Command with the Story Text "I am in Start".

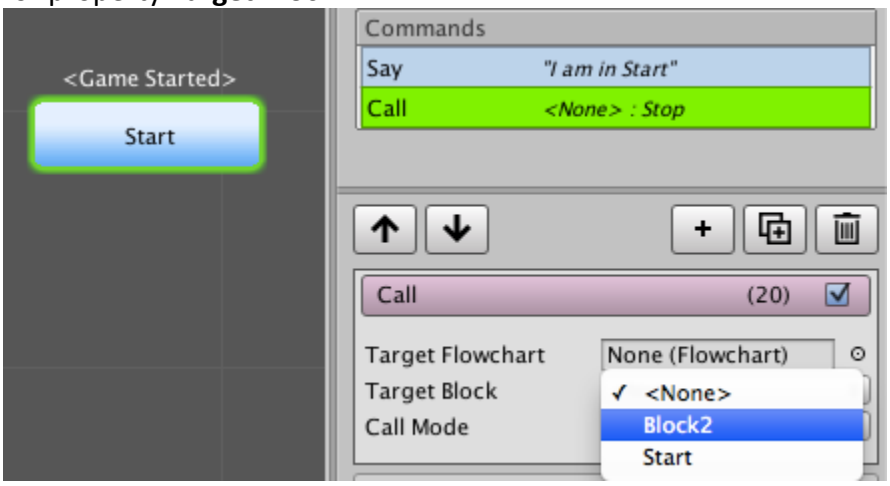
4. Add a new Block to your Flowchart named "Block2".
5. Add to Block "Block2" a Say Command with the Story Text "I am in Block2".



6. Add to Block "Block2" a Call Command, by choosing menu: Flow | Call:

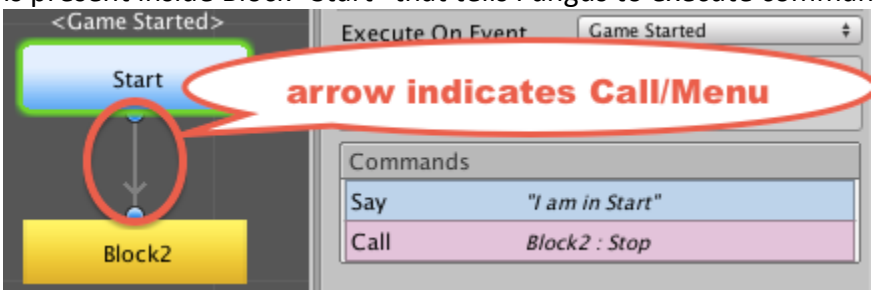


7. With this Call Command Selected, in the Inspector choose Block2 from the list of Blocks for property **Target Block**:



8. Note: We will keep the default of **Target Flowchart** (None), which means the current Flowchart.

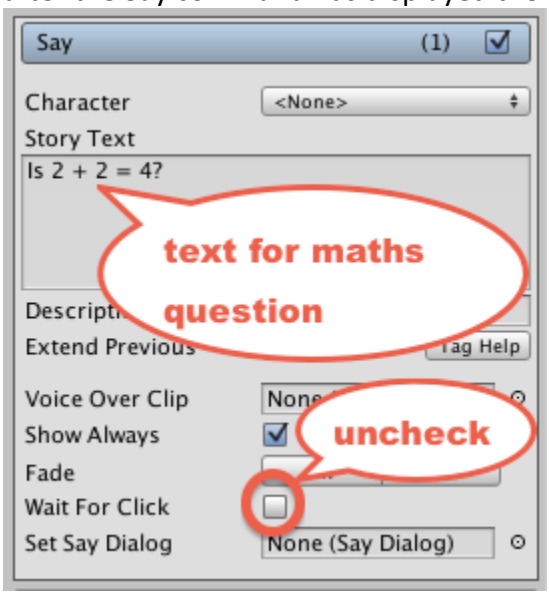
9. Note: We will keep the default of **Call Mode** Stop, which means that execution in the current Block (Start) will stop once execution of the called Block has begun.
10. You should now see an arrow in the Flowchart window, connecting Block "Start" with Block "Block2". This visually tells us (the game developer) that a Call or Menu Command is present inside Block "Start" that tells Fungus to execute commands in Block "Block2":



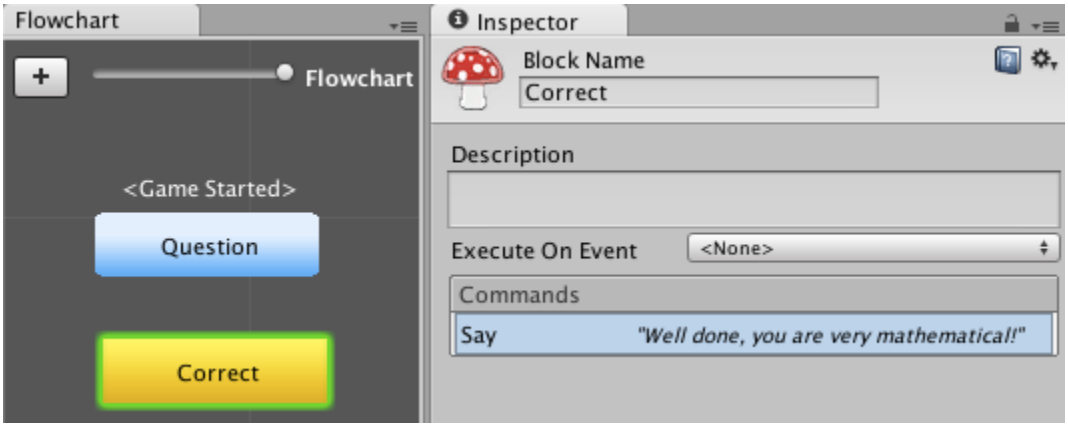
Executing Commands in another Block with Menu Commands

Let's use a Say command above to ask a tricky mathematical question, and demonstrate the Menu command by offering the user a choice been "correct' and "incorrect" answers. Menu commands transfer control to another block - so we'll need to add 2 new blocks to correspond to the 2 answers. Do the following:

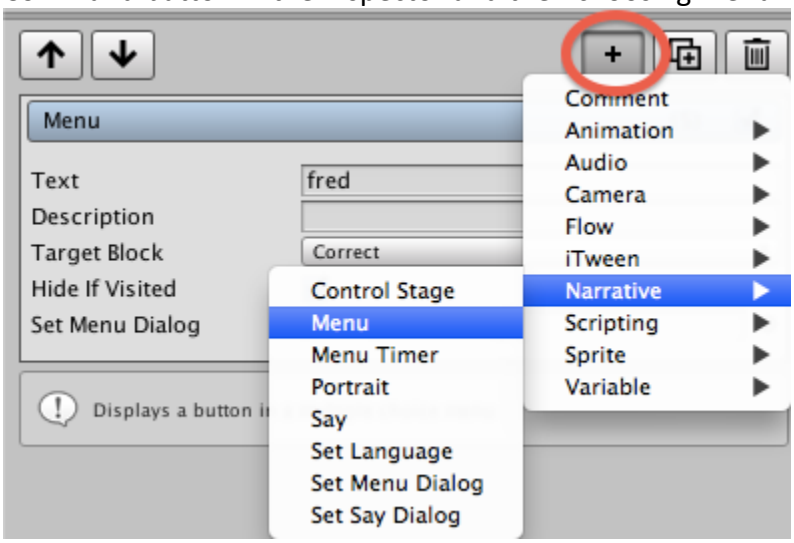
1. (setup) Create a new scene, add a Fungus Flowchart to the scene, and select the Block in the Flowchart.
2. Rename the Block in the Flowchart to "Question".
3. Create a Say command, with **Story Text** to ask the question: "Is 2 + 2?".
4. Uncheck the "Wait For Click" checkbox (this is so we see the menu options immediately after the Say command has displayed the question):



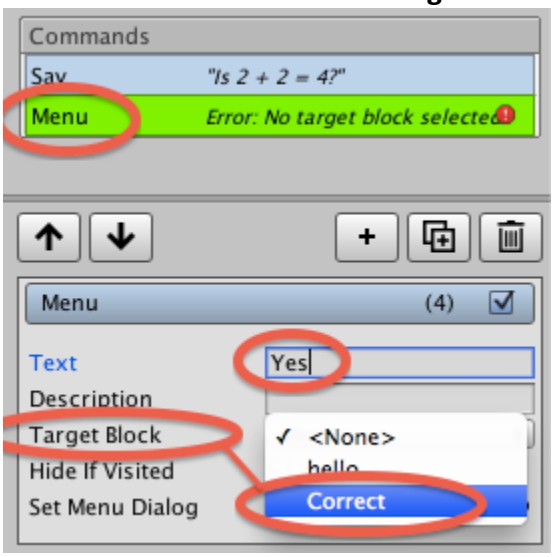
5. Create a new Block, named "Correct" which contains a **Say** command with the text "Well done, you are very mathematical!". Click the plus-sign button in the Flowchart window to add a new Block to the Flowchart, rename it "Correct" and then add that Say command:



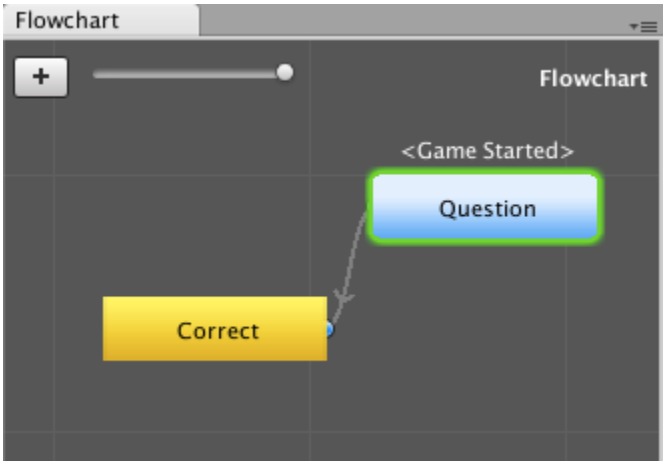
6. Select the "Question" block, and add a Menu command by clicking the plus-sign add Command button in the Inspector and then choosing menu: Narrative | Menu.



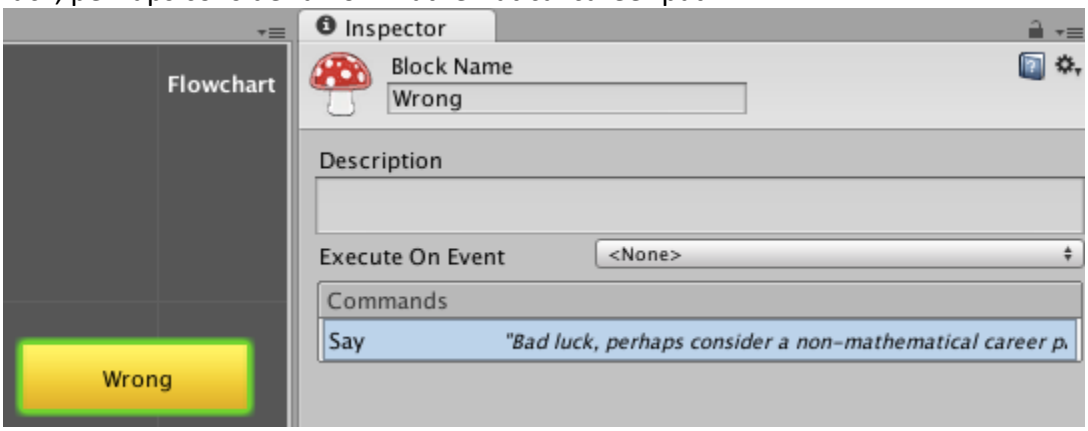
7. With this new Menu command selected (green) in the top half of the Inspector window, set the **Text** to "Yes" and the **Target Block** to your new "Correct" block:



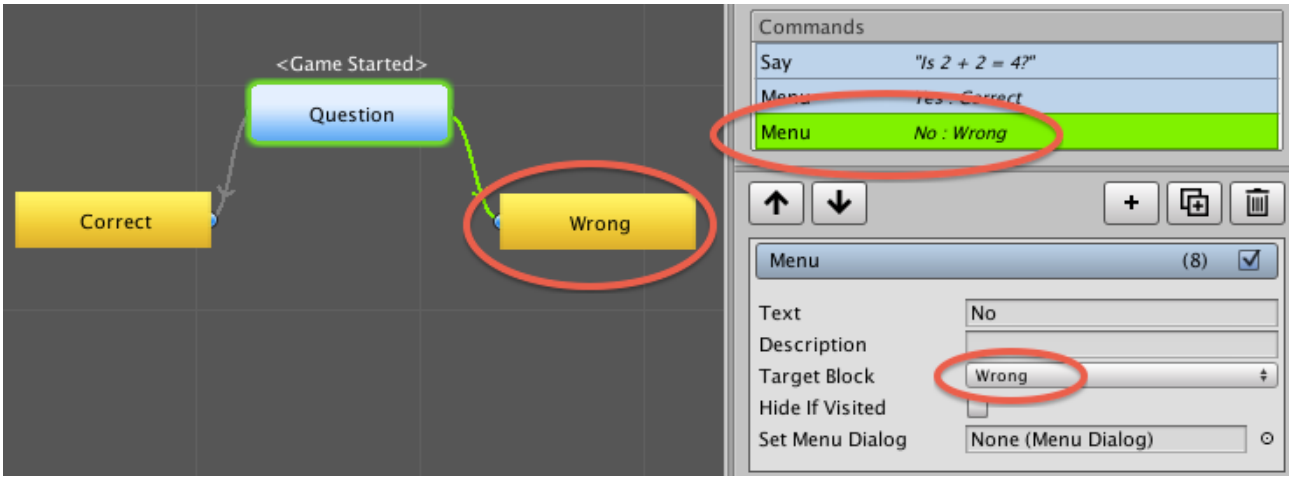
8. You should now see how the 'flow' of commands can change from Block "hello" to Block "Correct" in the Flowchart window:



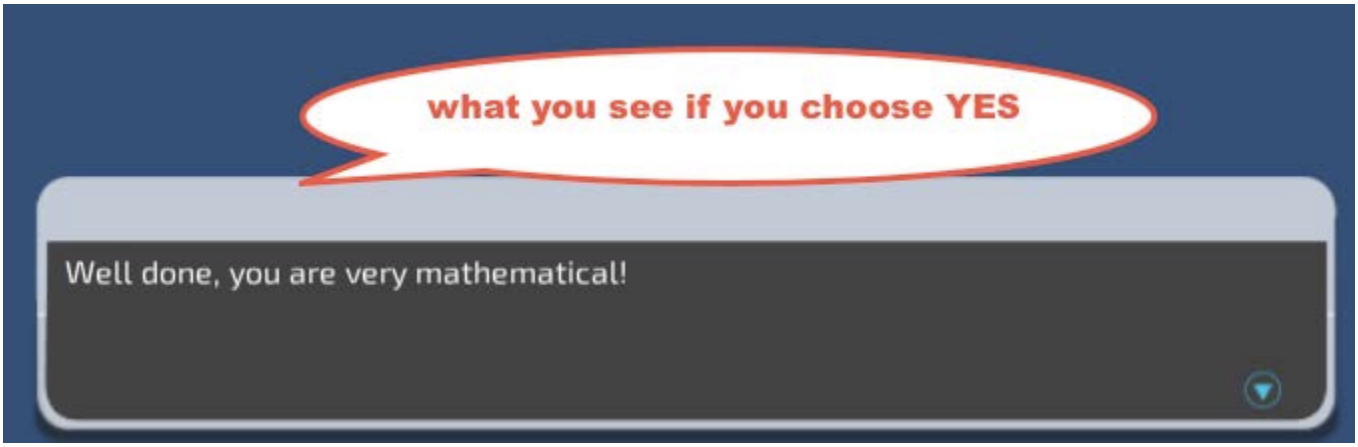
9. Add a second new Block named "Wrong", containing a Say command with text "Bad luck, perhaps consider a non-mathematical career path..."

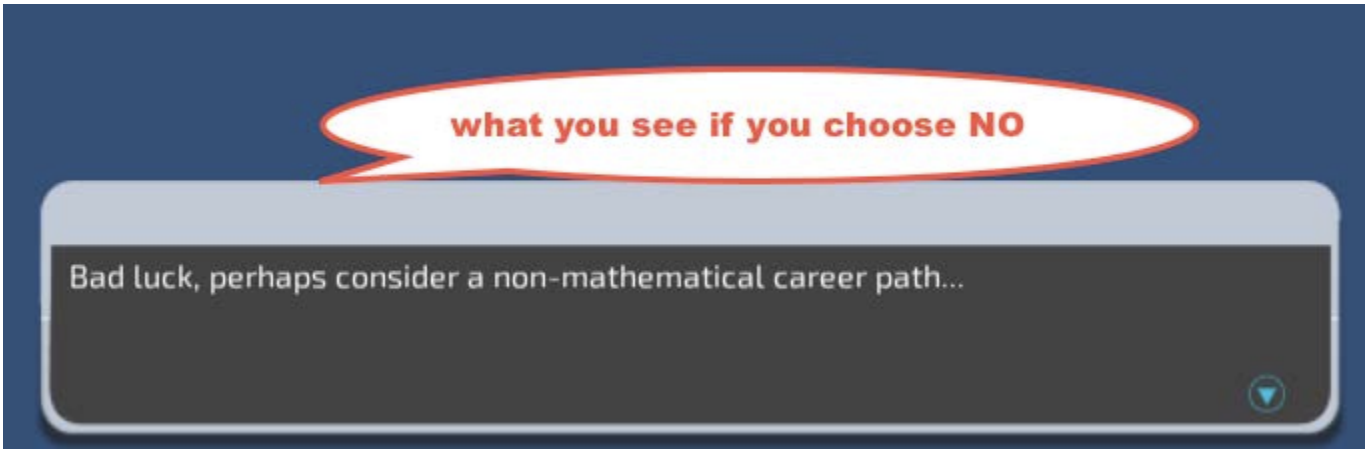


10. Now we need to add another Menu command to our "hello" block, offering the user the "No" answer to our maths question, and passing control to Block "Wrong" if they disagree that 2 + 2 = 4. Select the "hello" block, and add a Menu command. With this new Menu command selected (green) in the top half of the Inspector window, set the **Text** to "No" and the **Target Block** to your new "Wrong" block.
11. You should now see in the Flowchart window how block "hello" can pass control to either block "Correct" or Block "Wrong" - depending on which menu answer the user selects.



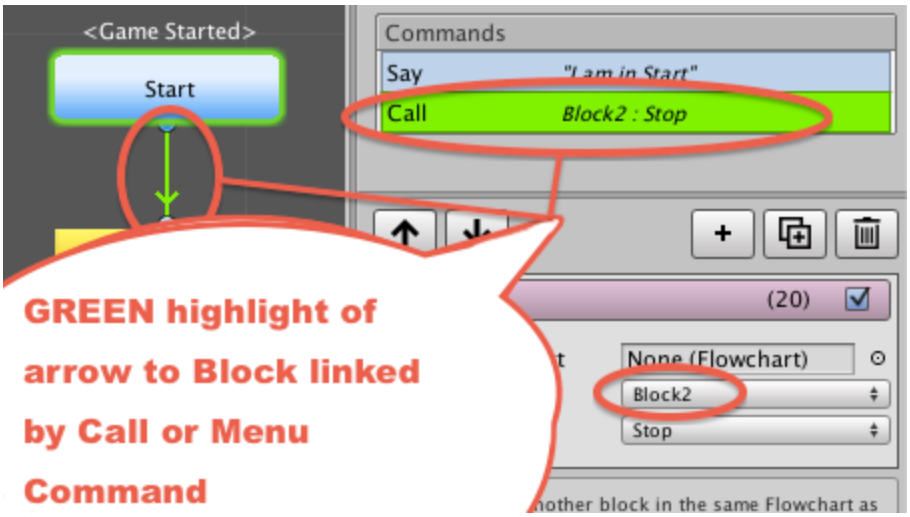
12. Run the scene, and you should see the Say question appear at the bottom of the screen, and also the two Menu buttons "Yes" and "No" in the middle of the screen. Clicking "Yes" then runs the "Correct" Block's commands, and clicking "No" runs the "Wrong" block's commands:





Highlighting connection between blocks by selecting command

In the Inspector window, if you select a Call or Menu Command (executing Commands in another Block in **the same Flowchart**), then you'll see the arrow between the 2 Blocks highlighted in GREEN:

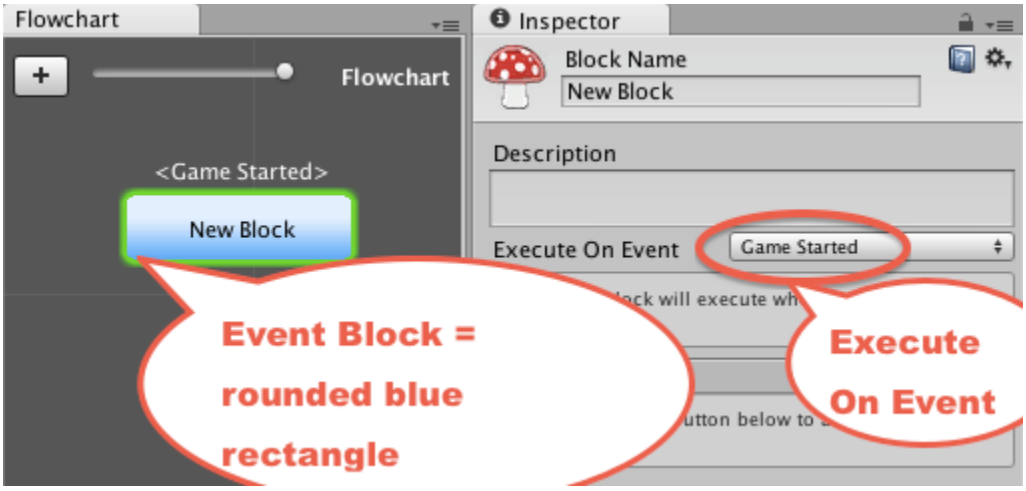


Setting a Block event handler

Events are one way to declare when you want execution of the Commands in a Block to begin. Typical events include:

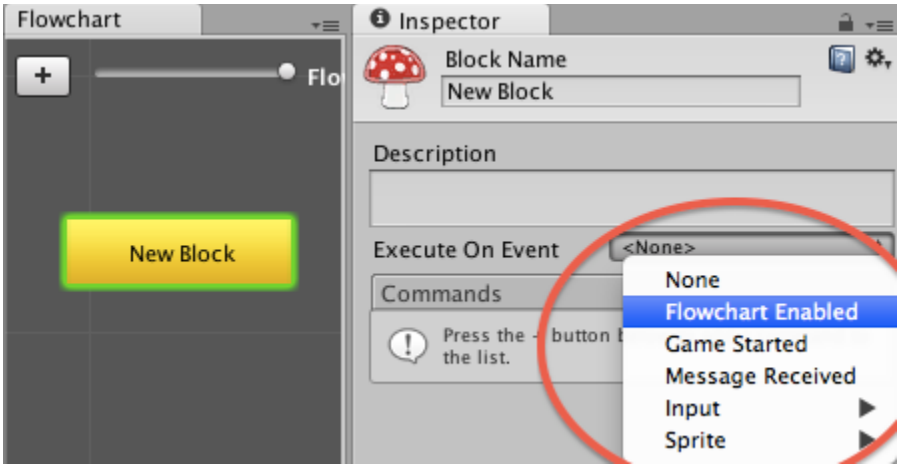
- Game Started
- Flowchart enabled
- Message Received
- Key Pressed (Up / Down / Repeat)
- Sprite clicking / drag-dropping interactions

The most common Event used to define when a Block should start execution is, of course, the **Game Started** event. Each new Flowchart automatically starts with a single empty Block that is defined to begin execution when the game starts:



To change the Event defined for a Block, or to assign an event for a Block that had None, do the following:

1. Select the Block in the Flowchart window.
2. In the Inspector window, for property **Execute On Event** choose from the popup menu the desired Event:



Glossary

A

Animator parameter

Special variables that can be used to change what happens in an animation chart, such as when to trigger a chance from one animation state to another.

Assets

The files on the computer/device, that are used in a game. Asset files include audio clips, video clips, 2D images, 3D models, and text files containing computer program code.

Audio tag

One of the tags specially allowing the control of Audio from within the Story Text of a Say Command.

There are four audio related tags:

{audio=AudioObjectName} Play Audio Once
{audioloop=AudioObjectName} Play Audio Loop
{audiopause=AudioObjectName} Pause Audio
{audiostop=AudioObjectName} Stop Audio

For more information see: Audio Tags recipe

Audiosource

A special component of Unity objects that can refer to a particular audio clip file, and control its playing / pausing / 3d effects etc.

B

Block

A Fungus Block is a sequence of Commands that have been created inside a Fungus Flowchart.

Block connection

When Commands in one Block cause the execution of Commands in another Block.

Boolean variable

A true/false variable.

Branching block

A Block that is **not** an Event Block, and causes execution of 2 or more other Blocks.

Note: Branching Blocks are displayed as an orange polygon in the Flowchart window.

Branching narrative

When choices by the user (via the Menu Command for example) will change what Narrative Commands are executed and displayed to the user.

Build platform

The target device for which a Unity game application will be built.

Build settings

Build settings include the target platform, screen resolution, input devices, scenes to be included etc.

C

Csharp (C♯)

A computer programming language developed by Microsoft, similar to Java. C♯ is one of the programming languages that can be used to write scripts for controlling Unity games.

Character

A Fungus gameObject allowing the personalisation of narrative, allowing characters to have names, special colours for their text on screen, and a range of different named images so character images can be displayed corresponding to the content of their dialog.

Clickable Sprite

A 2D image on screen that has been code to do something when clicked with the mouse or selected by the trackpad.

Coding / Scripting

Writing lines of computer code to control execution of a computer program, such as a Unity game (as opposed to using a visual language such as Fungus Flowcharts).

Command

An instruction for the Unity game engine to do something, examples are to play or stop a sound, or to display some text to the user in a dialog.

Command category

THe Fungus commands are grouped according to what kind of action they will lead to.

Command property panel

When a Block is selected in a Fungus Flowchart, the top-half of the Inspector window will show the properties and list of Commands for that Block. When one of the Commands is selected, that its properties are displayed in the bottom-half of the Inspector window.

Condition

Conditions are tests that are either "TRUE" or "FALSE" at a particular point in time. Different Commands can be defined to be executed depending on the value of a condition test in a Fungus Block.

CSV

Comma Separated Variable - a computer file format where the values of rows and columns are recorded, in a form that can be loaded into a spreadsheet..

Custom strings

As well as the automatically generated rows for each Say, Menu, Command etc., you can also add your own rows to the localisation file using any string id you want. To use these custom strings, you use variable substitution.

For example, say you have a custom string called "PlayerName", you can put {$PlayerName} in any Say text to substitute the correct localised value from the localisation file.

D

Draggable Sprite

A 2D image on screen that has been code to do something when clicked-and-dragged with the mouse or trackpad.

DragTarget Sprite

A 2D image that has been defined to execute some action(s) if a Draggable Sprite is dragged and released over it.

E

Editor

An application allowing the editing of game resources. The Unity Editor has multiple windows for viewing / editing properties of files, scene contents, UI dialogs, camera settings etc.

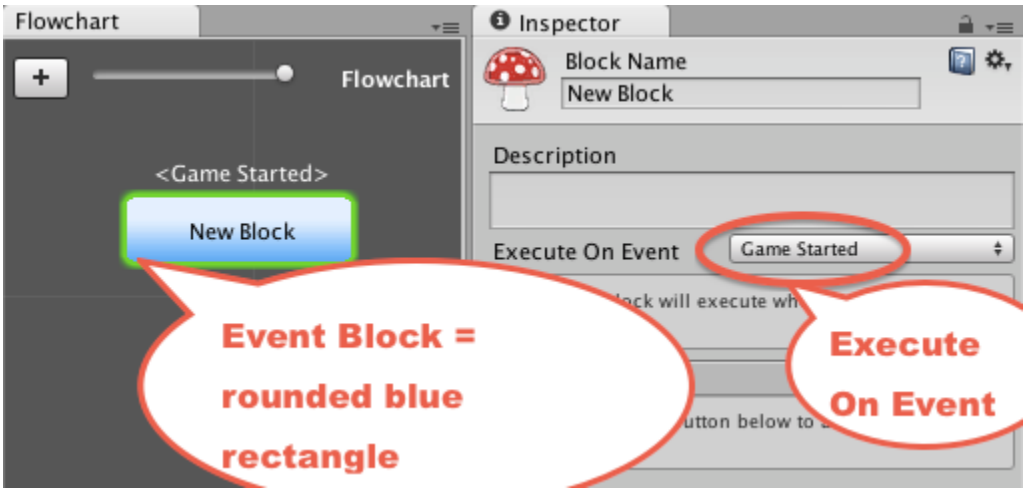
Event Handler

A Fungus Block or a Unity coded "method" that will be executed when a particular "Event" occurs. For example when the "SPACE" key is pressed the game should respond to the user having executed a spell or fired a gun or whatever.

Event block

Any Block whose execution is triggered by an Event, such as Game Started or Key Preses.

Note: Event Blocks are displayed as rounded blue rectangles in the Flowchart window.



Excel

A computer spreadsheet application program from Microsoft, part of the MS Office suite.

see the Microsoft website: Microsoft Excel application

Execution

The running of Commands or statements in a computer script / program. Making the computer do something.

F

Float variable

A variable that stores decimal numbers, such as 5.5, -0.11 etc.

Flowchart

A Fungus Flowchart contains a set of Blocks, each containing Commands to be executed.

Flow control

A general term referring to how a computer decides what to do next. So it is deciding which Fungus Command or Unity code statement to execute next. Typical Flow commands include decision choices such as loops or IF-conditions etc.

Flowchart item id

The unique number that is created for each Block of a Flowchart. This ID is used fpr serialisation (save and load) purposes, and not usually seen by the developer.

Flowchart Localization id

A special, unique name, used for associating language localization data for each Fungus game project.

Flowchart Message

A Message is basically a piece of text, but one that plays the role of being able to act as a 'trigger event' to cause Blocks to start executing.

Messages can be sent (via the Flow | Send Message Command), either to the current Flowchart, or to **ALL** Flowcharts. One of the Events that can be defined to start the execution of a Block is if a particular Message has been received by the Flowchart.

Flowchart Window

A special window available in the Unity Editor for editing and working with Fungus Flowcharts and their Blocks.

I

Integer variable

A variable that stores whole numbers, such as 100, 2, 0, -2 etc.

L

Landscape orientation

An orientation of an image or computer screen, where the width is greater than the height.

Language Column

When you first export the localization .csv (Comma-Separated-Variable) file it has 3 columns (Key, Description, Standard). When you want to add a new language to the file you add a new column for that language. You can use whatever column name you like, though it's typical to use two letter language codes (FR, ES, DE, etc.).

The Set Language command is provided with the name of the column for the language you wish to use in Fungus.

Libre Office

A free and Open Source alternative to Microsoft Office.

Lean more and download from: LibreOffice.org

Localization

Localization and Internationalization refer to coding a game in such a way that it can be deployed in a manner supporting playing and interactions in multiple human languages.

Logging

The Scripting | Debug Log Command allows the writing ('logging') of messages to the Console window while a scene is running. Logs provide a record of what has happened when the game is running, useful for checking what happened was what was intended, or for debugging (hunting down where errors are located).

M

Mecanim Animation

A Unity file representing an animation clip or pose. Animators manage the changes or mixing between one Animation clip and another.

Mecanim Animator

A Unity state-chart controller, which manages animated gameObjects - deciding when and how quickly objects should 'blend' into a different state or animation.

Menu Dialog

The UI elements (background / lines / text) that define the visual style of how the Text of Menu Commands is presented to the user.

Fungus provides a default Menu Dialog, but custom variations can be defined, for different games / scenes / characters etc.

N

Integer variable

A variable that stores whole numbers, such as 100, 2, 0, -2 etc.

O

Order in layer

If multiple objects have been assigned to the same Unity Sorting Layer, then the numeric value of the Order in Layer property determines their visual sorting order (what appears in front of what).

P

Parallax Sprite

The use of 2D sprites, moving at different speeds, to give the impression of 3D movement (where sprites for objects meant to be further away from the user move more slowly than objects closer to the game object.

Platform build

Refers to the device and type of application that the Unity editor will create. Examples are stand alone applications for Windows, Mac OS, iOS phone, Android phone, X-BOX etc.

Portrait

The Fungus Portrait Command is used to make the game display one of the Character Portrait images on the Stage. A particular image can be selected, and it can be made to move on/off stage.

Portrait orientation

An orientation of an image or computer screen, where the height is greater than the width.

Procedural Sound

A method of creating sounds using computer algorithms.

Private / Public variables

A Private variable as one that can only be accessed by Cmmands in Blocks in that Flowchart. A Public variable is one that can be accessed by Commands in Blocks in other Flowcharts as well as its own one.

S

Save profile

The Save Profile is basically a string that gets prepended to the key used to save & load variables in PlayerPrefs. This allows you to create separate save profiles using different names. For example, player 1's save data might use the Save Profile 'Chris', while player 2's data would use the Save Profile 'Matt', and their data can be saved / loaded independently.

The typical workflow would be:

1. Choose a suitable value fo the Save Profile (e.g. "player1")
2. Set Save Profile.
3. Save Variable(s).

Say Dialog

The UI elements (background / lines / text) that define the visual style of how the Story Text of Say Commands is presented to the user.

Fungus provides a default Say Dialog, but custom variations can be defined, for different games / scenes / characters etc.

Scenes

Unity Scenes are like chapters in a book, or "screens" in a game, or "levels" in a game. They allow the logic of a computer game to be broken into components. The "gameObjects" in a scene determine what software components will be created when a Scene starts running (others may be created or deleted once the Scene has started).

Screen aspect ratio

The relationship of the width of an image or screen to its height. Typical rations include 4:3 ad 16:10.

Screen resolution

The number of pixels wide and high a window or device supports.

Sorting layer

Unity 2D objects are assigned to a "Sorting Layer". This allows images/text to be assigned to layers such as Background, Foreground, Middleground, UI (on top of everything) etc.

Spine

Spine is a third-party 2D animation system from EsotericSoftware.com.

Spine Animation

Animation clip based on the Spine 2D system, that can be controlled from Fungus Commands (once the Unity Spine and Fungus Spine add-on packages have been installed).

Sprites

2D Images, that may be hidden / revealed. Sprite may be moved through Fungus Commands, or Unity code. Sprites can also be defined to be "draggable" by the users computer mouse pointer or mobile device touch gestures.

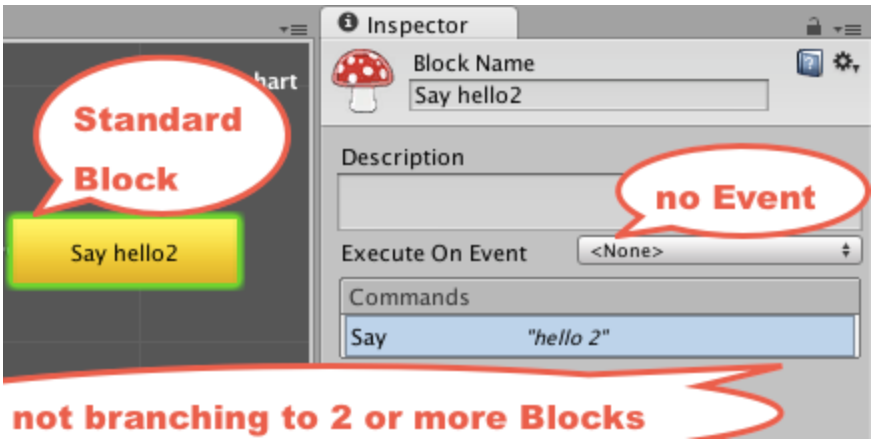
Stage

A Fungus gameObject in the scene, used to define the position and start-end movements of Character Portrait images controlled by Fungus Portrait Commands.

Standard block

A Fungus Block whose execution is **not** triggered by an event (so it is not an Event Block), and which does **not** cause the execution of 2 or more other Blocks (so is **not** a Branching Block).

Note: Standard Blocks are displayed as yellow rectangles in the Flowchart window.



Standard Text

When you export the localization file, there is a Standard Text field. This is populated with the text that has been entered into the Say, Menu, etc. text fields in the Unity editor. In normal usage, this field is in the localisation file for reference so localizers know what text needs to be translated. For example, if the project is drafted using English, then the text in Say, Menu Commands etc. would be in English, and in the exported localization file this text would appear in the Standard Text fields.

Note, if you use the Set Language command with an empty language string then the values in the Standard Text field will be used when Fungus runs.

Story Text Tags

Special instructions that can be embedded inside the Story Text of Say commands. Examples are tags that start/stop playing a sound, or send a message.

String ID

This is an automatically generated ID code, that is stored in the first column of the localisation file. These IDs are generated automatically when you export the strings using the Localization object.

The ID is needed since every localised text string needs a unique identifying name so that it can be mapped it back to the Say, Menu, etc. command that references it.

String variable

A variable that stores text characters, such as the player's name, or colour of the potion they are carrying.

T

Tag

Special characters that can be inserted into Fungus Say Commands, to control other aspects of the scene, such as the playing of a sound, or the shaking of the camera, or the sending of a message to cause other Blocks to start executing.

U

Unity

A game engine and IDE (Interactive Development Environment) - with program code editor, and audio/graphics editing facilities. Able to build applications for deployment to Windows, Mac, Linus, iOS, Android, XBox etc.

Unity UI

The building of visual interfaces for computer programs deployed by Unity.

Usfxr

A system for generating sound effects from mathematical parameters.

Built into Fungus and based on the open source project at: Usfxr at GitHub

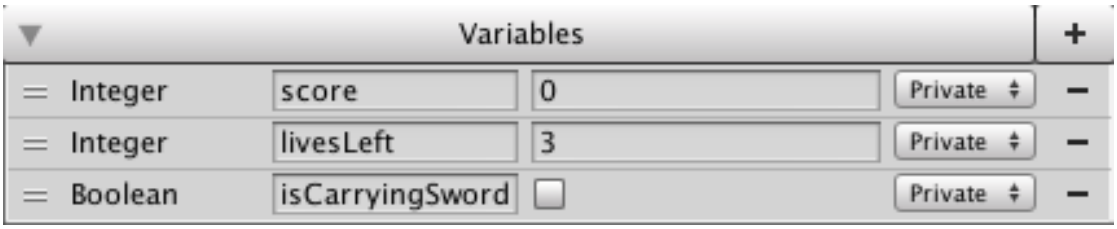
V

Variable

A named memory location, from which values can be retrieved at a later date.

Variable default value

A "default" value refers to the value given to a variable automatically, if no particular value is specified. In the Variables section at the bottom of the Flowchart window, as well as defining the names and data types of variables, you can also set default values here.



Variables Panel

The very bottom section of the Flowchart window is where variables (such as score, playerName, carryingGreenKey etc.) can be defined for use in Fungus Commands.

Variable saving

The saving of the value of a variable to the devices 'disk'. So when another Unity Scene is loaded, the saved value can be loaded from saved memory. The saved value can also be loaded the next time the game application is executed on a device.

Variable substitution

This relates to the use of a tag in the Story Text of a Say Command, whereby the contents of a Variable will be inserted into that part of the text for the Say Command

{$VarName} Substitute variable

View, Move, Rotate, Scale & Rectangle tools

The core tools offered in the Unity editor for changing basic properties of 2D and 3D objects

View

A Fungus gameObject that helps ensure good visual experience regardless of device aspect ration, and used for camera movements and control by Fungus Commands.

Voice over

A sound clip file containing recorded audio that corresponds to text being displayed on screen.

W

Webplayer

A Build option in Unity that creates an application that can play inside web pages that have the Unity Web Player plug-in installed. The need fo the plug-in and web-player option are being replaced by the WebGL Unity build facility.

WebGL

WebGL is a technology available in modern browsers that allows for interactive, high quality computer graphics within web pages, without the need for any special plug-ins (such as Flash or the Unity Player). It promised to be the future for multimedia web page content and interaction.

Animation Commands

Reset Anim Trigger

Resets a trigger parameter on an Animator component.

Property	Type	Description
Animator	UnityEngine.Animator	Reference to an Animator component in a game object
Parameter Name	System.String	Name of the trigger Animator parameter that will be reset

Set Anim Bool

Sets a boolean parameter on an Animator component to control a Unity animation

Property	Type	Description
Animator	UnityEngine.Animator	Reference to an Animator component in a game object
Parameter Name	System.String	Name of the boolean Animator parameter that will have its value changed
Value	Fungus.BooleanData	The boolean value to set the parameter to

Set Anim Float

Sets a float parameter on an Animator component to control a Unity animation

Property	Type	Description
Animator	UnityEngine.Animator	Reference to an Animator component in a game object
Parameter Name	System.String	Name of the float Animator parameter that will have its value changed
Value	Fungus.FloatData	The float value to set the parameter to

Set Anim Integer

Sets an integer parameter on an Animator component to control a Unity animation

Property	Type	Description
Animator	UnityEngine.Animator	Reference to an Animator component in a game object
Parameter Name	System.String	Name of the integer Animator parameter that will have its value changed
Value	Fungus.IntegerData	The integer value to set the parameter to

Set Anim Trigger

Sets a trigger parameter on an Animator component to control a Unity animation

Property	Type	Description
Animator	UnityEngine.Animator	Reference to an Animator component in a game object
Parameter Name	System.String	Name of the trigger Animator parameter that will have its value changed

Audio Commands

Control Audio

Plays, loops, or stops an audiosource.

Property	Type	Description
Control	Fungus.ControlAudio+controlType	What to do to audio
Audio Source	UnityEngine.AudioSource	Audio clip to play
Start Volume	System.Single	Start audio at this volume
End Volume	System.Single	End audio at this volume
Fade Duration	System.Single	Time to fade between current volume level and target volume level.
Wait Until Finished	System.Boolean	Wait until this command has finished before executing the next command.

Play Music

Plays looping game music. If any game music is already playing, it is stopped. Game music will continue playing across scene loads.

Property	Type	Description
Music Clip	UnityEngine.AudioClip	Music sound clip to play
At Time	System.Single	Time to begin playing in seconds. If the audio file is compressed, the time index may be inaccurate.

Play Sound

Plays a once-off sound effect. Multiple sound effects can be played at the same time.

Property	Type	Description
Sound Clip	UnityEngine.AudioClip	Sound effect clip to play
Volume	System.Single	Volume level of the sound effect
Wait Until Finished	System.Boolean	Wait until the sound has finished playing before continuing execution.

Play Usfxr Sound

Plays a usfxr synth sound. Use the usfxr editor [Tools > Fungus > Utilities > Generate usfxr Sound Effects] to create the SettingsString. Set a ParentTransform if using positional sound. See https://github.com/zeh/usfxr for more information about usfxr.

Property	Type	Description
Parent Transform	UnityEngine.Transform	Transform to use for positional audio
Settings String	System.String	Settings string which describes the audio
Wait Duration	System.Single	Time to wait before executing the next command

Set Audio Volume

Sets the global volume level for audio played with Play Music and Play Sound commands.

Property	Type	Description
Volume	System.Single	Global volume level for audio played using Play Music and Play Sound
Fade Duration	System.Single	Time to fade between current volume level and target volume level.

Stop Music

Stops the currently playing game music.

Camera Commands

Fade Screen

Draws a fullscreen texture over the scene to give a fade effect. Setting Target Alpha to 1 will obscure the screen, alpha 0 will reveal the screen. If no Fade Texture is provided then a default flat color texture is used.

Property	Type	Description
Duration	System.Single	Time for fade effect to complete
Target Alpha	System.Single	Current target alpha transparency value. The fade gradually adjusts the alpha to approach this target value.
Wait Until Finished	System.Boolean	Wait until the fade has finished before executing next command
Fade Color	UnityEngine.Color	Color to render fullscreen fade texture with when screen is obscured.
Fade Texture	UnityEngine.Texture2D	Optional texture to use when rendering the fullscreen fade effect.

Fade To View

Fades the camera out and in again at a position specified by a View object.

Property	Type	Description
Duration	System.Single	Time for fade effect to complete
Fade Out	System.Boolean	Fade from fully visible to opaque at start of fade
Target View	Fungus.View	View to transition to when Fade is complete
Wait Until Finished	System.Boolean	Wait until the fade has finished before executing next command
Fade Color	UnityEngine.Color	Color to render fullscreen fade texture with when screen is obscured.
Fade Texture	UnityEngine.Texture2D	Optional texture to use when rendering the fullscreen fade effect.

Move To View

Moves the camera to a location specified by a View object.

Property	Type	Description
Duration	System.Single	Time for move effect to complete
Target View	Fungus.View	View to transition to when move is complete
Wait Until Finished	System.Boolean	Wait until the fade has finished before executing next command

Shake Camera

Applies a camera shake effect to the main camera.

Property	Type	Description
Duration	System.Single	Time for camera shake effect to complete
Amount	UnityEngine.Vector2	Magnitude of shake effect in x & y axes
Wait Until Finished	System.Boolean	Wait until the shake effect has finished before executing next command

Start Swipe

Activates swipe panning mode where the player can pan the camera within the area between viewA & viewB.

Property	Type	Description
View A	Fungus.View	Defines one extreme of the scrollable area that the player can pan around
View B	Fungus.View	Defines one extreme of the scrollable area that the player can pan around
Duration	System.Single	Time to move the camera to a valid starting position between the two views
Speed Multiplier	System.Single	Multiplier factor for speed of swipe pan

Stop Swipe

Deactivates swipe panning mode.

Flow Commands

Break

Force a loop to terminate immediately.

Call

Execute another block in the same Flowchart as the command, or in a different Flowchart.

Property	Type	Description
Target Flowchart	Fungus.Flowchart	Flowchart which contains the block to execute. If none is specified then the current Flowchart is used.
Target Block	Fungus.Block	Block to start executing
Call Mode	Fungus.Call+CallMode	Select if the calling block should stop or continue executing commands, or wait until the called block finishes.

Else

Marks the start of a command block to be executed when the preceding If statement is False.

Else If

Marks the start of a command block to be executed when the preceding If statement is False and the test expression is true.

Property	Type	Description
Variable	Fungus.Variable	Variable to use in expression
Boolean Data	Fungus.BooleanData	Boolean value to compare against
Integer Data	Fungus.IntegerData	Integer value to compare against
Float Data	Fungus.FloatData	Float value to compare against
String Data	Fungus.StringData	String value to compare against
Compare Operator	Fungus.CompareOperator	The type of comparison to be performed

End

Marks the end of a conditional block.

If

If the test expression is true, execute the following command block.

Property	Type	Description
Variable	Fungus.Variable	Variable to use in expression
Boolean Data	Fungus.BooleanData	Boolean value to compare against
Integer Data	Fungus.IntegerData	Integer value to compare against
Float Data	Fungus.FloatData	Float value to compare against
String Data	Fungus.StringData	String value to compare against
Compare Operator	Fungus.CompareOperator	The type of comparison to be performed

Jump

Move execution to a specific Label command

Property	Type	Description
Target Label	Fungus.Label	Label to jump to

Label

Marks a position in the command list for execution to jump to.

Property	Type	Description
Key	System.String	Display name for the label

Load Scene

Loads a new Unity scene and displays an optional loading image. This is useful for splitting a large game across multiple scene files to reduce peak memory usage. Previously loaded assets will be released before loading the scene to free up memory.The scene to be loaded must be added to the scene list in Build Settings.

Property	Type	Description
Scene Name	System.String	Name of the scene to load. The scene must also be added to the build settings.
Loading Image	UnityEngine.Texture2D	Image to display while loading the scene

Send Message

Sends a message to either the owner Flowchart or all Flowcharts in the scene. Blocks can listen for this message using a Message Received event handler.

Property	Type	Description
Message Target	Fungus.SendMessage+MessageTarget	Target flowchart(s) to send the message to
Message	System.String	Name of the message to send

Stop

Stop executing the current Flowchart.

Wait

Waits for period of time before executing the next command in the block.

Property	Type	Description
Duration	System.Single	Duration to wait for

While

Continuously loop through a block of commands while the condition is true. Use the Break command to force the loop to terminate immediately.

Property	Type	Description
Variable	Fungus.Variable	Variable to use in expression
Boolean Data	Fungus.BooleanData	Boolean value to compare against
Integer Data	Fungus.IntegerData	Integer value to compare against
Float Data	Fungus.FloatData	Float value to compare against
String Data	Fungus.StringData	String value to compare against
Compare Operator	Fungus.CompareOperator	The type of comparison to be performed

iTween Commands

Look From

Instantly rotates a GameObject to look at the supplied Vector3 then returns it to it's starting rotation over time.

Property	Type	Description
From Transform	UnityEngine.Transform	Target transform that the GameObject will look at
From Position	UnityEngine.Vector3	Target world position that the GameObject will look at, if no From Transform is set
Axis	Fungus.iTweenAxis	Restricts rotation to the supplied axis only
Target Object	UnityEngine.GameObject	Target game object to apply the Tween to
Tween Name	System.String	An individual name useful for stopping iTweens by name
Duration	System.Single	The time in seconds the animation will take to complete
Ease Type	Fungus.iTween+EaseType	The shape of the easing curve applied to the animation
Loop Type	Fungus.iTween+LoopType	The type of loop to apply once the animation has completed
Wait Until Finished	System.Boolean	Wait until the tween has finished before executing the next command

Look To

Rotates a GameObject to look at a supplied Transform or Vector3 over time.

Property	Type	Description
To Transform	UnityEngine.Transform	Target transform that the GameObject will look at
To Position	UnityEngine.Vector3	Target world position that the GameObject will look at, if no From Transform is set
Axis	Fungus.iTweenAxis	Restricts rotation to the supplied axis only
Target Object	UnityEngine.GameObject	Target game object to apply the Tween to
Tween Name	System.String	An individual name useful for stopping iTweens by name
Duration	System.Single	The time in seconds the animation will take to complete
Ease Type	Fungus.iTween+EaseType	The shape of the easing curve applied to the animation
Loop Type	Fungus.iTween+LoopType	The type of loop to apply once the animation has completed
Wait Until Finished	System.Boolean	Wait until the tween has finished before executing the next command

Move Add

Moves a game object by a specified offset over time.

Property	Type	Description
Offset	UnityEngine.Vector3	A translation offset in space the GameObject will animate to
Space	UnityEngine.Space	Apply the transformation in either the world coordinate or local cordinate system
Target Object	UnityEngine.GameObject	Target game object to apply the Tween to
Tween Name	System.String	An individual name useful for stopping iTweens by name
Duration	System.Single	The time in seconds the animation will take to complete
Ease Type	Fungus.iTween+EaseType	The shape of the easing curve applied to the animation
Loop Type	Fungus.iTween+LoopType	The type of loop to apply once the animation has completed
Wait Until Finished	System.Boolean	Wait until the tween has finished before executing the next command

Move From

Moves a game object from a specified position back to its starting position over time. The position can be defined by a transform in another object (using To Transform) or by setting an absolute position (using To Position, if To Transform is set to None).

Property	Type	Description
From Transform	UnityEngine.Transform	Target transform that the GameObject will move from
From Position	UnityEngine.Vector3	Target world position that the GameObject will move from, if no From Transform is set
Is Local	System.Boolean	Whether to animate in world space or relative to the parent. False by default.
Target Object	UnityEngine.GameObject	Target game object to apply the Tween to
Tween Name	System.String	An individual name useful for stopping iTweens by name
Duration	System.Single	The time in seconds the animation will take to complete
Ease Type	Fungus.iTween+EaseType	The shape of the easing curve applied to the animation
Loop Type	Fungus.iTween+LoopType	The type of loop to apply once the animation has completed
Wait Until Finished	System.Boolean	Wait until the tween has finished before executing the next command

Move To

Moves a game object to a specified position over time. The position can be defined by a transform in another object (using To Transform) or by setting an absolute position (using To Position, if To Transform is set to None).

Property	Type	Description
To Transform	UnityEngine.Transform	Target transform that the GameObject will move to
To Position	UnityEngine.Vector3	Target world position that the GameObject will move to, if no From Transform is set
Is Local	System.Boolean	Whether to animate in world space or relative to the parent. False by default.
Target Object	UnityEngine.GameObject	Target game object to apply the Tween to
Tween Name	System.String	An individual name useful for stopping iTweens by name
Duration	System.Single	The time in seconds the animation will take to complete
Ease Type	Fungus.iTween+EaseType	The shape of the easing curve applied to the animation
Loop Type	Fungus.iTween+LoopType	The type of loop to apply once the animation has completed
Wait Until Finished	System.Boolean	Wait until the tween has finished before executing the next command

Punch Position

Applies a jolt of force to a GameObject's position and wobbles it back to its initial position.

Property	Type	Description
Amount	UnityEngine.Vector3	A translation offset in space the GameObject will animate to
Space	UnityEngine.Space	Apply the transformation in either the world coordinate or local cordinate system
Target Object	UnityEngine.GameObject	Target game object to apply the Tween to
Tween Name	System.String	An individual name useful for stopping iTweens by name
Duration	System.Single	The time in seconds the animation will take to complete
Ease Type	Fungus.iTween+EaseType	The shape of the easing curve applied to the animation
Loop Type	Fungus.iTween+LoopType	The type of loop to apply once the animation has completed
Wait Until Finished	System.Boolean	Wait until the tween has finished before executing the next command

Punch Rotation

Applies a jolt of force to a GameObject's rotation and wobbles it back to its initial rotation.

Property	Type	Description
Amount	UnityEngine.Vector3	A rotation offset in space the GameObject will animate to
Space	UnityEngine.Space	Apply the transformation in either the world coordinate or local cordinate system
Target Object	UnityEngine.GameObject	Target game object to apply the Tween to
Tween Name	System.String	An individual name useful for stopping iTweens by name
Duration	System.Single	The time in seconds the animation will take to complete
Ease Type	Fungus.iTween+EaseType	The shape of the easing curve applied to the animation
Loop Type	Fungus.iTween+LoopType	The type of loop to apply once the animation has completed
Wait Until Finished	System.Boolean	Wait until the tween has finished before executing the next command

Punch Scale

Applies a jolt of force to a GameObject's scale and wobbles it back to its initial scale.

Property	Type	Description
Amount	UnityEngine.Vector3	A scale offset in space the GameObject will animate to
Target Object	UnityEngine.GameObject	Target game object to apply the Tween to
Tween Name	System.String	An individual name useful for stopping iTweens by name
Duration	System.Single	The time in seconds the animation will take to complete
Ease Type	Fungus.iTween+EaseType	The shape of the easing curve applied to the animation
Loop Type	Fungus.iTween+LoopType	The type of loop to apply once the animation has completed
Wait Until Finished	System.Boolean	Wait until the tween has finished before executing the next command

Rotate Add

Rotates a game object by the specified angles over time.

Property	Type	Description
Offset	UnityEngine.Vector3	A rotation offset in space the GameObject will animate to
Space	UnityEngine.Space	Apply the transformation in either the world coordinate or local cordinate system
Target Object	UnityEngine.GameObject	Target game object to apply the Tween to
Tween Name	System.String	An individual name useful for stopping iTweens by name
Duration	System.Single	The time in seconds the animation will take to complete
Ease Type	Fungus.iTween+EaseType	The shape of the easing curve applied to the animation
Loop Type	Fungus.iTween+LoopType	The type of loop to apply once the animation has completed
Wait Until Finished	System.Boolean	Wait until the tween has finished before executing the next command

Rotate From

Rotates a game object from the specified angles back to its starting orientation over time.

Property	Type	Description
From Transform	UnityEngine.Transform	Target transform that the GameObject will rotate from
From Rotation	UnityEngine.Vector3	Target rotation that the GameObject will rotate from, if no From Transform is set
Is Local	System.Boolean	Whether to animate in world space or relative to the parent. False by default.
Target Object	UnityEngine.GameObject	Target game object to apply the Tween to
Tween Name	System.String	An individual name useful for stopping iTweens by name
Duration	System.Single	The time in seconds the animation will take to complete
Ease Type	Fungus.iTween+EaseType	The shape of the easing curve applied to the animation
Loop Type	Fungus.iTween+LoopType	The type of loop to apply once the animation has completed
Wait Until Finished	System.Boolean	Wait until the tween has finished before executing the next command

Rotate To

Rotates a game object to the specified angles over time.

Property	Type	Description
To Transform	UnityEngine.Transform	Target transform that the GameObject will rotate to
To Rotation	UnityEngine.Vector3	Target rotation that the GameObject will rotate to, if no To Transform is set
Is Local	System.Boolean	Whether to animate in world space or relative to the parent. False by default.
Target Object	UnityEngine.GameObject	Target game object to apply the Tween to
Tween Name	System.String	An individual name useful for stopping iTweens by name
Duration	System.Single	The time in seconds the animation will take to complete
Ease Type	Fungus.iTween+EaseType	The shape of the easing curve applied to the animation
Loop Type	Fungus.iTween+LoopType	The type of loop to apply once the animation has completed
Wait Until Finished	System.Boolean	Wait until the tween has finished before executing the next command

Scale Add

Changes a game object's scale by a specified offset over time.

Property	Type	Description
Offset	UnityEngine.Vector3	A scale offset in space the GameObject will animate to
Target Object	UnityEngine.GameObject	Target game object to apply the Tween to
Tween Name	System.String	An individual name useful for stopping iTweens by name
Duration	System.Single	The time in seconds the animation will take to complete
Ease Type	Fungus.iTween+EaseType	The shape of the easing curve applied to the animation
Loop Type	Fungus.iTween+LoopType	The type of loop to apply once the animation has completed
Wait Until Finished	System.Boolean	Wait until the tween has finished before executing the next command

Scale From

Changes a game object's scale to the specified value and back to its original scale over time.

Property	Type	Description
From Transform	UnityEngine.Transform	Target transform that the GameObject will scale from
From Scale	UnityEngine.Vector3	Target scale that the GameObject will scale from, if no From Transform is set
Target Object	UnityEngine.GameObject	Target game object to apply the Tween to
Tween Name	System.String	An individual name useful for stopping iTweens by name
Duration	System.Single	The time in seconds the animation will take to complete
Ease Type	Fungus.iTween+EaseType	The shape of the easing curve applied to the animation
Loop Type	Fungus.iTween+LoopType	The type of loop to apply once the animation has completed
Wait Until Finished	System.Boolean	Wait until the tween has finished before executing the next command

Scale To

Changes a game object's scale to a specified value over time.

Property	Type	Description
To Transform	UnityEngine.Transform	Target transform that the GameObject will scale to
To Scale	UnityEngine.Vector3	Target scale that the GameObject will scale to, if no To Transform is set
Target Object	UnityEngine.GameObject	Target game object to apply the Tween to
Tween Name	System.String	An individual name useful for stopping iTweens by name
Duration	System.Single	The time in seconds the animation will take to complete
Ease Type	Fungus.iTween+EaseType	The shape of the easing curve applied to the animation
Loop Type	Fungus.iTween+LoopType	The type of loop to apply once the animation has completed
Wait Until Finished	System.Boolean	Wait until the tween has finished before executing the next command

Shake Position

Randomly shakes a GameObject's position by a diminishing amount over time.

Property	Type	Description
Amount	UnityEngine.Vector3	A translation offset in space the GameObject will animate to
Is Local	System.Boolean	Whether to animate in world space or relative to the parent. False by default.
Axis	Fungus.iTweenAxis	Restricts rotation to the supplied axis only
Target Object	UnityEngine.GameObject	Target game object to apply the Tween to
Tween Name	System.String	An individual name useful for stopping iTweens by name
Duration	System.Single	The time in seconds the animation will take to complete
Ease Type	Fungus.iTween+EaseType	The shape of the easing curve applied to the animation
Loop Type	Fungus.iTween+LoopType	The type of loop to apply once the animation has completed
Wait Until Finished	System.Boolean	Wait until the tween has finished before executing the next command

Shake Rotation

Randomly shakes a GameObject's rotation by a diminishing amount over time.

Property	Type	Description
Amount	UnityEngine.Vector3	A rotation offset in space the GameObject will animate to
Space	UnityEngine.Space	Apply the transformation in either the world coordinate or local cordinate system
Target Object	UnityEngine.GameObject	Target game object to apply the Tween to
Tween Name	System.String	An individual name useful for stopping iTweens by name
Duration	System.Single	The time in seconds the animation will take to complete
Ease Type	Fungus.iTween+EaseType	The shape of the easing curve applied to the animation
Loop Type	Fungus.iTween+LoopType	The type of loop to apply once the animation has completed
Wait Until Finished	System.Boolean	Wait until the tween has finished before executing the next command

Shake Scale

Randomly shakes a GameObject's rotation by a diminishing amount over time.

Property	Type	Description
Amount	UnityEngine.Vector3	A scale offset in space the GameObject will animate to
Target Object	UnityEngine.GameObject	Target game object to apply the Tween to
Tween Name	System.String	An individual name useful for stopping iTweens by name
Duration	System.Single	The time in seconds the animation will take to complete
Ease Type	Fungus.iTween+EaseType	The shape of the easing curve applied to the animation
Loop Type	Fungus.iTween+LoopType	The type of loop to apply once the animation has completed
Wait Until Finished	System.Boolean	Wait until the tween has finished before executing the next command

Stop Tween

Stops an active iTween by name.

Property	Type	Description
Tween Name	System.String	Stop and destroy any Tweens in current scene with the supplied name

Stop Tweens

Stop all active iTweens in the current scene.

Narrative Commands

Control Stage

Controls the stage on which character portraits are displayed.

Property	Type	Description
Display	Fungus.StageDisplayType	Display type
Stage	Fungus.Stage	Stage to display characters on
Replaced Stage	Fungus.Stage	Stage to swap with
Use Default Settings	System.Boolean	Use Default Settings
Fade Duration	System.Single	Fade Duration
Wait Until Finished	System.Boolean	Wait until the tween has finished before executing the next command

Menu

Displays a button in a multiple choice menu

Property	Type	Description
Text	System.String	Text to display on the menu button
Description	System.String	Notes about the option text for other authors, localization, etc.
Target Block	Fungus.Block	Block to execute when this option is selected
Hide If Visited	System.Boolean	Hide this option if the target block has been executed previously
Set Menu Dialog	Fungus.MenuDialog	A custom Menu Dialog to use to display this menu. All subsequent Menu commands will use this dialog.

Menu Timer

Displays a timer bar and executes a target block if the player fails to select a menu option in time.

Property	Type	Description
Duration	System.Single	Length of time to display the timer for
Target Block	Fungus.Block	Block to execute when the timer expires

Portrait

Controls a character portrait.

Property	Type	Description
Stage	Fungus.Stage	Stage to display portrait on
Display	Fungus.DisplayType	Display type
Character	Fungus.Character	Character to display
Replaced Character	Fungus.Character	Character to swap with
Portrait	UnityEngine.Sprite	Portrait to display
Offset	Fungus.PositionOffset	Move the portrait from/to this offset position
From Position	UnityEngine.RectTransform	Move the portrait from this position
To Position	UnityEngine.RectTransform	Move the portrait to this positoin
Facing	Fungus.FacingDirection	Direction character is facing
Use Default Settings	System.Boolean	Use Default Settings
Fade Duration	System.Single	Fade Duration
Move Speed	System.Single	Movement Speed
Shift Offset	UnityEngine.Vector2	Shift Offset
Move	System.Boolean	Move
Shift Into Place	System.Boolean	Start from offset
Wait Until Finished	System.Boolean	Wait until the tween has finished before executing the next command

Say

Writes text in a dialog box.

Property	Type	Description
Description	System.String	Notes about this story text for other authors, localization, etc.
Character	Fungus.Character	Character that is speaking
Portrait	UnityEngine.Sprite	Portrait that represents speaking character
Voice Over Clip	UnityEngine.AudioClip	Voiceover audio to play when writing the text
Show Always	System.Boolean	Always show this Say text when the command is executed multiple times
Show Count	System.Int32	Number of times to show this Say text when the command is executed multiple times
Extend Previous	System.Boolean	Type this text in the previous dialog box.
Fade In	System.Boolean	Fade in this dialog box.
Fade Out	System.Boolean	Fade out this dialog box.
Wait For Click	System.Boolean	Wait for player to click before hiding the dialog and continuing. If false then the dialog will display and execution will continue immediately.
Set Say Dialog	Fungus.SayDialog	Sets the active Say dialog with a reference to a Say Dialog object in the scene. All story text will now display using this Say Dialog.

Set Language

Set the active language for the scene. A Localization object with a localization file must be present in the scene.

Property	Type	Description
Language Code	System.String	Code of the language to set. e.g. ES, DE, JA

Set Menu Dialog

Sets a custom menu dialog to use when displaying multiple choice menus

Property	Type	Description
Menu Dialog	Fungus.MenuDialog	The Menu Dialog to use for displaying menu buttons

Set Say Dialog

Sets a custom say dialog to use when displaying story text

Property	Type	Description
Say Dialog	Fungus.SayDialog	The Say Dialog to use for displaying Say story text

Scripting Commands

Comment

Use comments to record design notes and reminders about your game.

Property	Type	Description
Commenter Name	System.String	Name of Commenter
Comment Text	System.String	Text to display for this comment

Call Method

Calls a named method on a GameObject using the GameObject.SendMessage() system.

Property	Type	Description
Target Object	UnityEngine.GameObject	Target monobehavior which contains the method we want to call
Method Name	System.String	Name of the method to call
Delay	System.Single	Delay (in seconds) before the method will be called

Debug Log

Writes a log message to the debug console.

Property	Type	Description
Log Type	Fungus.DebugLog+DebugLogType	Display type of debug log info
Log Message	Fungus.StringData	Text to write to the debug log. Supports variable substitution, e.g. {$Myvar}

Destroy

Destroys a specified game object in the scene.

Property	Type	Description
Target Game Object	UnityEngine.GameObject	Reference to game object to destroy

Get Text

Gets the text property from a UI Text object and stores it in a string variable.

Property	Type	Description
Text Object	UnityEngine.UI.Text	Text object to get text value from
Variable	Fungus.Variable	String variable to store the text value in

Set Active

Sets a game object in the scene to be active / inactive.

Property	Type	Description
Target Game Object	UnityEngine.GameObject	Reference to game object to enable / disable
Active State	Fungus.BooleanData	Set to true to enable the game object

Set Text

Sets the text property on a UI Text object.

Property	Type	Description
Text Object	UnityEngine.UI.Text	Text object to set text on
String Data	Fungus.StringData	String value to assign to the text object

Spawn Object

Spawns a new object based on a reference to a scene or prefab game object.

Property	Type	Description
Source Object	UnityEngine.GameObject	Game object to copy when spawning. Can be a scene object or a prefab.
Parent Transform	UnityEngine.Transform	Transform to use for position of newly spawned object.
Spawn Position	UnityEngine.Vector3	Local position of newly spawned object.
Spawn Rotation	UnityEngine.Vector3	Local rotation of newly spawned object.

Sprite Commands

Fade Sprite

Fades a sprite to a target color over a period of time.

Property	Type	Description
Sprite Renderer	UnityEngine.SpriteRenderer	Sprite object to be faded
Duration	System.Single	Length of time to perform the fade
Target Color	UnityEngine.Color	Target color to fade to. To only fade transparency level, set the color to white and set the alpha to required transparency.
Wait Until Finished	System.Boolean	Wait until the fade has finished before executing the next command

Set Clickable 2D

Sets a Clickable2D component to be clickable / non-clickable.

Property	Type	Description
Target Clickable2 D	Fungus.Clickable2D	Reference to Clickable2D component on a gameobject
Active State	Fungus.BooleanData	Set to true to enable the component

Set Collider

Sets all collider (2d or 3d) components on the target objects to be active / inactive

Property	Type	Description
Target Objects	System.Collections.Generic.List`1[UnityEngine.GameObject]	A list of gameobjects containing collider components to be set active / inactive
Target Tag	System.String	All objects with this tag will have their collider set active / inactive
Active State	Fungus.BooleanData	Set to true to enable the collider components

Show Sprite

Makes a sprite visible / invisible by setting the color alpha.

Property	Type	Description
Sprite Renderer	UnityEngine.SpriteRenderer	Sprite object to be made visible / invisible
Visible	System.Boolean	Make the sprite visible or invisible

Variable Commands

Delete Save Key

Deletes a saved value from permanent storage.

Property	Type	Description
Key	System.String	Name of the saved value. Supports variable substition e.g. "player_{$PlayerNumber}

Load Variable

Loads a saved value and stores it in a Boolean, Integer, Float or String variable. If the key is not found then the variable is not modified.

Property	Type	Description
Key	System.String	Name of the saved value. Supports variable substition e.g. "player_{$PlayerNumber}"
Variable	Fungus.Variable	Variable to store the value in.

Random Float

Sets an float variable to a random value in the defined range.

Property	Type	Description
Variable	Fungus.FloatVariable	The variable whos value will be set
Min Value	Fungus.FloatData	Minimum value for random range
Max Value	Fungus.FloatData	Maximum value for random range

Random Integer

Sets an integer variable to a random value in the defined range.

Property	Type	Description
Variable	Fungus.IntegerVariable	The variable whos value will be set
Min Value	Fungus.IntegerData	Minimum value for random range
Max Value	Fungus.IntegerData	Maximum value for random range

Reset

Resets the state of all commands and variables in the Flowchart.

Property	Type	Description
Reset Commands	System.Boolean	Reset state of all commands in the script
Reset Variables	System.Boolean	Reset variables back to their default values

Save Variable

Save an Boolean, Integer, Float or String variable to persistent storage using a string key. The value can be loaded again later using the Load Variable command. You can also use the Set Save Profile command to manage separate save profiles for multiple players.

Property	Type	Description
Key	System.String	Name of the saved value. Supports variable substition e.g. "player_{$PlayerNumber}
Variable	Fungus.Variable	Variable to read the value from. Only Boolean, Integer, Float and String are supported.

Set Save Profile

Sets the active profile that the Save Variable and Load Variable commands will use. This is useful to crete multiple player save games. Once set, the profile applies across all Flowcharts and will also persist across scene loads.

Property	Type	Description
Save Profile Name	System.String	Name of save profile to make active.

Set Variable

Sets a Boolean, Integer, Float or String variable to a new value using a simple arithmetic operation. The value can be a constant or reference another variable of the same type.

Property	Type	Description
Variable	Fungus.Variable	The variable whos value will be set
Set Operator	Fungus.SetVariable+SetOperator	The type of math operation to be performed
Boolean Data	Fungus.BooleanData	Boolean value to set with
Integer Data	Fungus.IntegerData	Integer value to set with
Float Data	Fungus.FloatData	Float value to set with
String Data	Fungus.StringData	String value to set with

Events (Core) Commands

Flowchart Enabled

The block will execute when the Flowchart game object is enabled.

Game Started

The block will execute when the game starts playing.

Message Received

The block will execute when the specified message is received from a Send Message command.

Property	Type	Description
Message	System.String	Fungus message to listen for

Events (Input) Commands

Key Pressed

The block will execute when a key press event occurs.

Property	Type	Description
Key Press Type	Fungus.KeyPressed+KeyPressType	The type of keypress to activate on
Key Code	UnityEngine.KeyCode	Keycode of the key to activate on

Events (Sprite) Commands

Drag Cancelled

The block will execute when the player drags an object and releases it without dropping it on a target object.

Property	Type	Description
Draggable Object	Fungus.Draggable2D	Draggable object to listen for drag events on

Drag Completed

The block will execute when the player drags an object and successfully drops it on a target object.

Property	Type	Description
Draggable Object	Fungus.Draggable2D	Draggable object to listen for drag events on
Target Object	UnityEngine.Collider2D	Drag target object to listen for drag events on

Drag Entered

The block will execute when the player is dragging an object which starts touching the target object.

Property	Type	Description
Draggable Object	Fungus.Draggable2D	Draggable object to listen for drag events on
Target Object	UnityEngine.Collider2D	Drag target object to listen for drag events on

Drag Exited

The block will execute when the player is dragging an object which stops touching the target object.

Property	Type	Description
Draggable Object	Fungus.Draggable2D	Draggable object to listen for drag events on
Target Object	UnityEngine.Collider2D	Drag target object to listen for drag events on

Drag Started

The block will execute when the player starts dragging an object.

Object Clicked

The block will execute when the user clicks or taps on the clickable object.

Property	Type	Description
Clickable Object	Fungus.Clickable2D	Object that the user can click or tap on