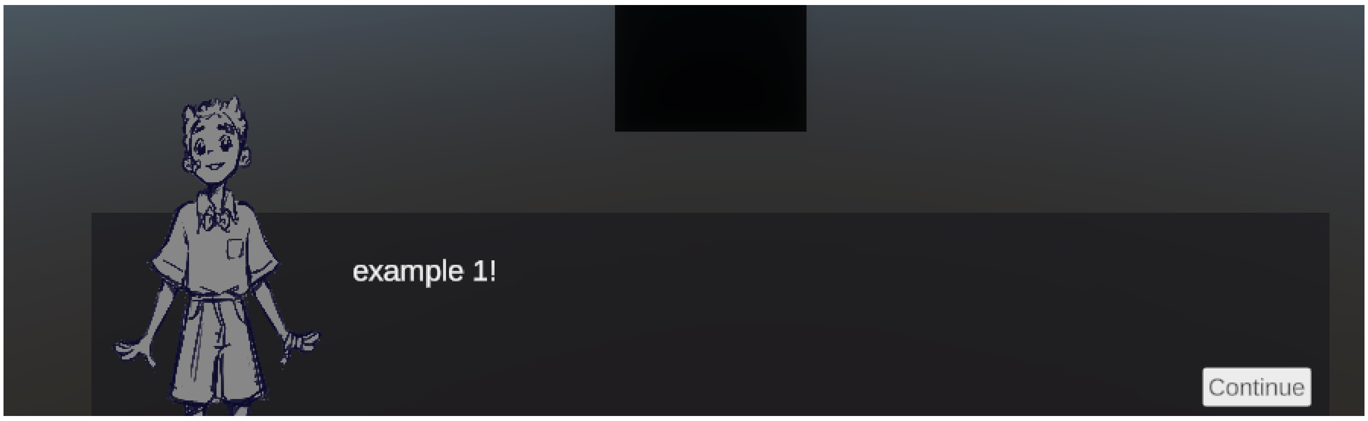
Yarn Spinner Cheat Sheet

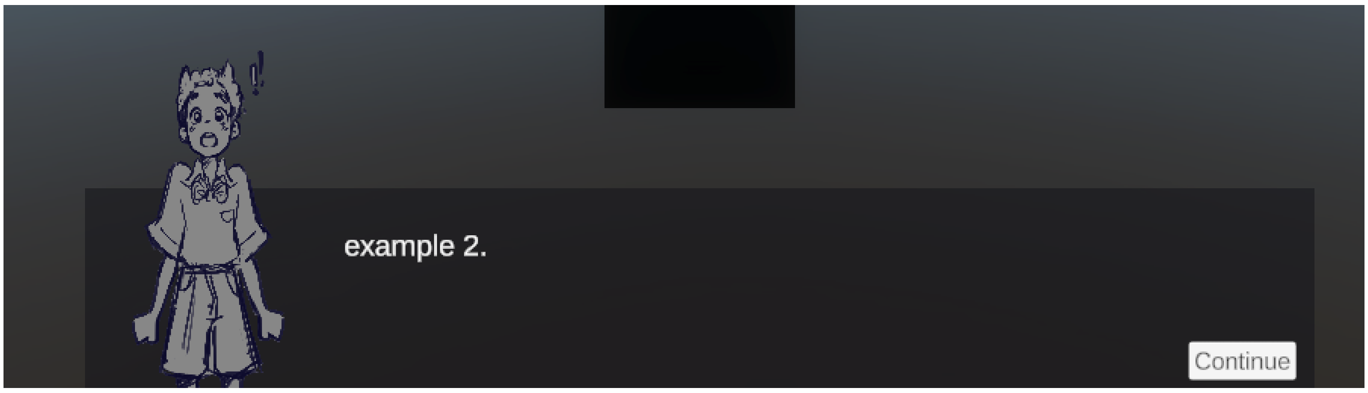
I apologize in advance, some of the internal links in this doc do not work. I have no idea why 😦.

Character Dialog

Dialog text will be shown with a typewriting effect that will also trigger appropriate sound effects. See [sprite=str/] for more info about sprite configuration.


example 1!

```
name: example 1!
```

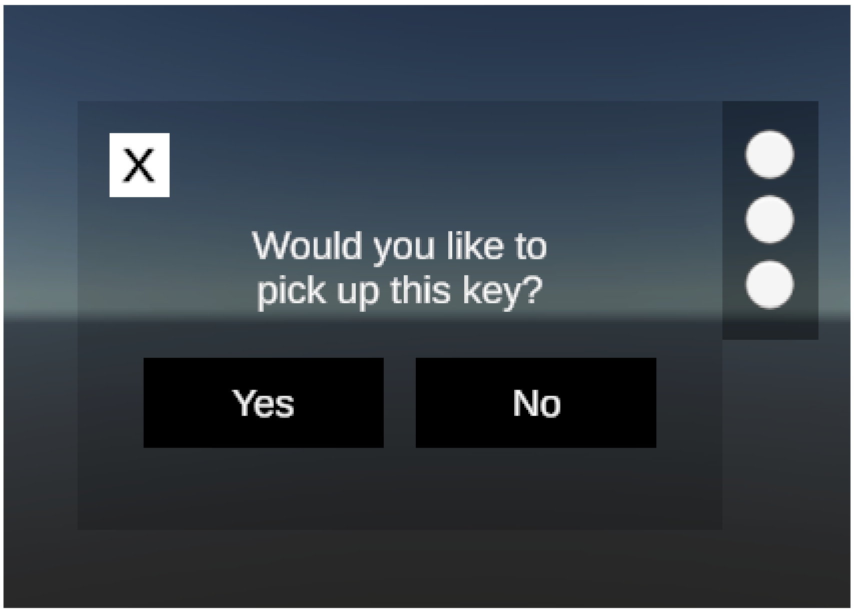

example 2.

```
name: [sprite=surprise/] example 2.
```

Interactions

Interactions should begin with the `[interaction/]` markup tag. If the cancel button should be displayed in the UI, include an option with the `[cancel/]`tag. `[spirit_cost/]` may also be used to configure the interaction UI. See below for example.

```
[interaction/][spirit_cost=2] A porcelain teacup...<br/>looks breakable
-> move
-> break
-> [cancel/]
```



```
title: key_from_book
---
[interaction/][spirit_points=3/]Would you like to pick up this key?
-> Yes
 <<locked_animate I_key pickup>>
 <<set $loc_key = "player">>
-> No
-> [cancel/]
```

^ Example Interaction

See [interaction/] and [cancel/] for more info.

Options

Options will use the InteractionView UI and provide the user the ability to chose of the listed options. Below is an example.

```
-> option text 1
    continued text
-> option text 2
    continued text
```

*note - empty lines may not be used within option blocks

```
-> option text 1
    continued text

-> option text 2
    continued text
```

In the above example, this will be considered as 2 different interactions.

Flow Control

Yarn variables can be used within if statements to decide the path of execution. See [set var](#set var val https docs yarnspinner dev getting-started writing-in-yarn logic-and-variables) and [declare var](#declare var val) for more info about variables in yarn.

```
<<if $var_bool == true>>
    var_bool is true.
<<elseif $var_int < 20>>
    var_int is less than 20.
<<endif>>
```

Markup

Markup tags may embeded in any line of dialog and will not be displayed in the shown lines of dialog. They provide information to the dialog system for further configuration.

A typical example may look like this: `hello [ex]Jaq[ex/]`. This would display `hello Jaq`, but the dialogue system may perform some kind operation over `Jaq`.

There are also self-closing markup tags that may look like this: `[ex/] hello`. They do not operate over a chunk of the text, but rather a single point in the line.

[interaction/]

This signals the game that the current line should not be shown in the CharacterView. The UI will not be displayed and it will automatically continue to the next line. The intent is for this line to be followed by a set of options to be displayed in the interaction UI. See Interactions for intended use.

[spirit_cost=int/]

Provides the spirit point cost to the Interaction View UI. If not provided to the Interaction View, the cost section of the UI will not be shown. See Interactions for intended use.

[cancel/]

Including this in one of the options of an interaction will ensure that the cancel button in the Interaction View will be shown. Warning there will be unintended consequences if multiple options in the same interaction have this markup tag. See Interactions for intended use.

[sprite=str/]

Specifies the character sprite that should be shown with a character dialog. Input strings are not case sensitive. If no sprite is provided for a Character Dialog, the neutral sprite will be used. See Character Dialog for example usage. Please contact Nick Maclean to change the below predefined strings.

Predefined strings

- `neutral`
- `surprise`/`surprised`

Yarn Commands

Yarn Commands are accessible within any part of a Yarn narrative script. They are called using `<<cmd param1 param2 ...>>`. Please look below for defined commands, their parameters, and descriptions.

Parameters wrapped in paranethesis are optional. If multiple optional parameters are defined all optional parameters to the left of any that you would like to provide must also be provided.

```
<<cmd param1 optional1 optional2
```

For example (with the above command), `<<cmd p1>>`, `<<cmd p1 p2>>`, and `<<cmd p1 p2 p3>>` would be legal calls, but `<<cmd p1 p3>>` would either be an illegal call or p3 would be mistaken as param2.

<<locked_animate obj timeline>>

Attempts to play the timeline `timeline` on the YarnObject `obj`. This will halt narrative script execution untill the timeline is done playing.

Will return an error to the console if unable to find the object or timeline.

<<animate obj timeline>>

Attempts to play the timeline `timeline` on the YarnObject `obj`. Unlike `locked_animate`, this will not wait for the timeline to complete before continuing narrative script execution.

Will return an error to the console if unable to find the object or timeline.

<<activate obj instanceName>>

Attempts to set the YarnObject's, `obj`, active instance to `instanceName`.

<<fade_in (duration)>>

Fades the screen to black over `duration`. While the animation is being performed, execution of the narrative script will be halted. If no duration is provided a default value in game will be defined.

<<fade_in_unlocked (duration)>>

Fades the screen to black over `duration`. The narrative script will continue execution after beginning the animation. If no duration is provided a default value in game will be defined.

<<fade_out (duration)>>

Fades the screen from black to display the scene over `duration`. While the animation is being performed, execution of the narrative script will be halted. If no duration is provided a default value in game will be defined.

<<fade_out_unlocked (duration)>>

Fades the screen from black to display the scene over `duration`. The narrative script will continue execution after beginning the animation. If no duration is provided a default value in game will be defined.

<<fade_in_out (inDuration) (outDuration)>>

Fades the screen to black over `inDuration` and back to display the scene to display the scene over `outDuration`. While the animation is being performed, execution of the narrative script will be halted. If `inDuration` and/or `outDuration` is provided a default value in game will be defined.

<<fade_out_unlocked (inDuration) (outDuration)>>

Fades the screen to black over `inDuration` and back to display the scene to display the scene over `outDuration`. The narrative script will continue execution after beginning the animation. If `inDuration` and/or `outDuration` is provided a default value in game will be defined.

Built-in commands

<<jump nodeName>>

This will find the node `nodeName` in the YarnProject and immediately begin its execution.

```
<<if false>>
    <<jump node>>
<<endif>>
```

Example hack to render a connection in VSCode between the current node and `node` without affecting the execution of the script.

<<wait duration>>

This will pause narrative script execution for `duration` seconds. The below would wait for 2 seconds before beginning to execute `a_node`.

```
<<wait 2>>
<<jump a_node>>
```

<<stop>>

The `stop` command will immediately end narrative script execution. The below would stop narrative script execution if `$we_should_stop` is true.

```
<<if $we_should_stop == true>>
    <<stop>>
<<endif>>
```

<<set $var = val>>

Can be used to set the value of variable. Available variable types include numbers, strings, and booleans.

<<declare $var = val>>

Is used within a YarnProject file to declare a default value for a variable. A description can be provided with `/// {description` on the line above or to the right of the declaration.

Example Node

```
title: key_from_book
---
[interaction/][spirit_points=3/]Would you like to pick up this key?
-> Yes
 <<locked_animate I_key pickup>>
 <<set $loc_key = "player">>
-> No
-> [cancel/]

// requisite to
<<if false>>
 <<jump door_sisters>>
<<endif>>
===
```

*Note

We are using Text Mesh Pro to display text, so all Rich Text Tags should be available to use in the script. Here is some documentation: Text Mesh Pro