Twine Workshop

Interactive Fiction

1

"What?!" cries Batman, nearly crashing the Batmobile. He has been speeding toward the dark streets of Gotham on his way to Police Headquarters, when the familiar Bat-Signal suddenly changed into a grinning skull! *Someone has tampered with the searchlight, thinks Batman. But why turn the bat silhouette into a DEATH'S HEAD?! It must be a warning . . . or a trap.*

Batman's next move could be critical. A wrong decision might mean his DOOM!

If Batman drives straight to Police Headquarters, turn to page 17.

If he radios Commissioner Gordon from the Batmobile, turn to page 26.

If he plays it safe and uses a public phone, turn to page 3.

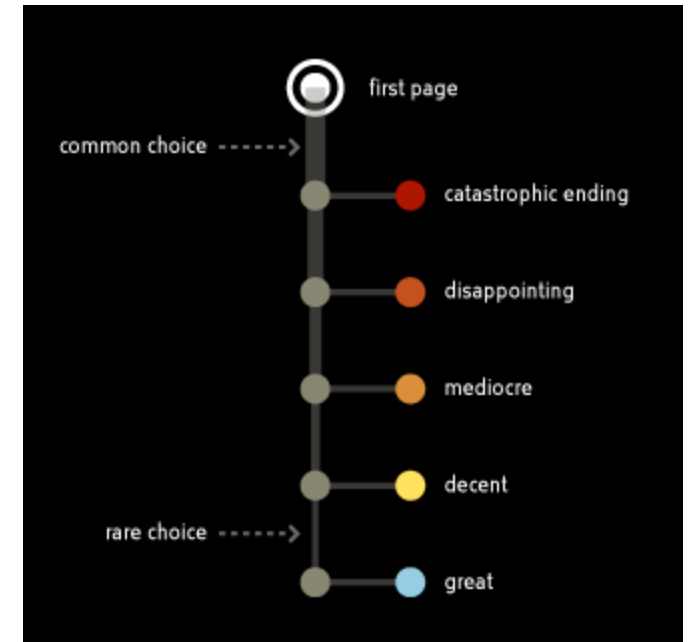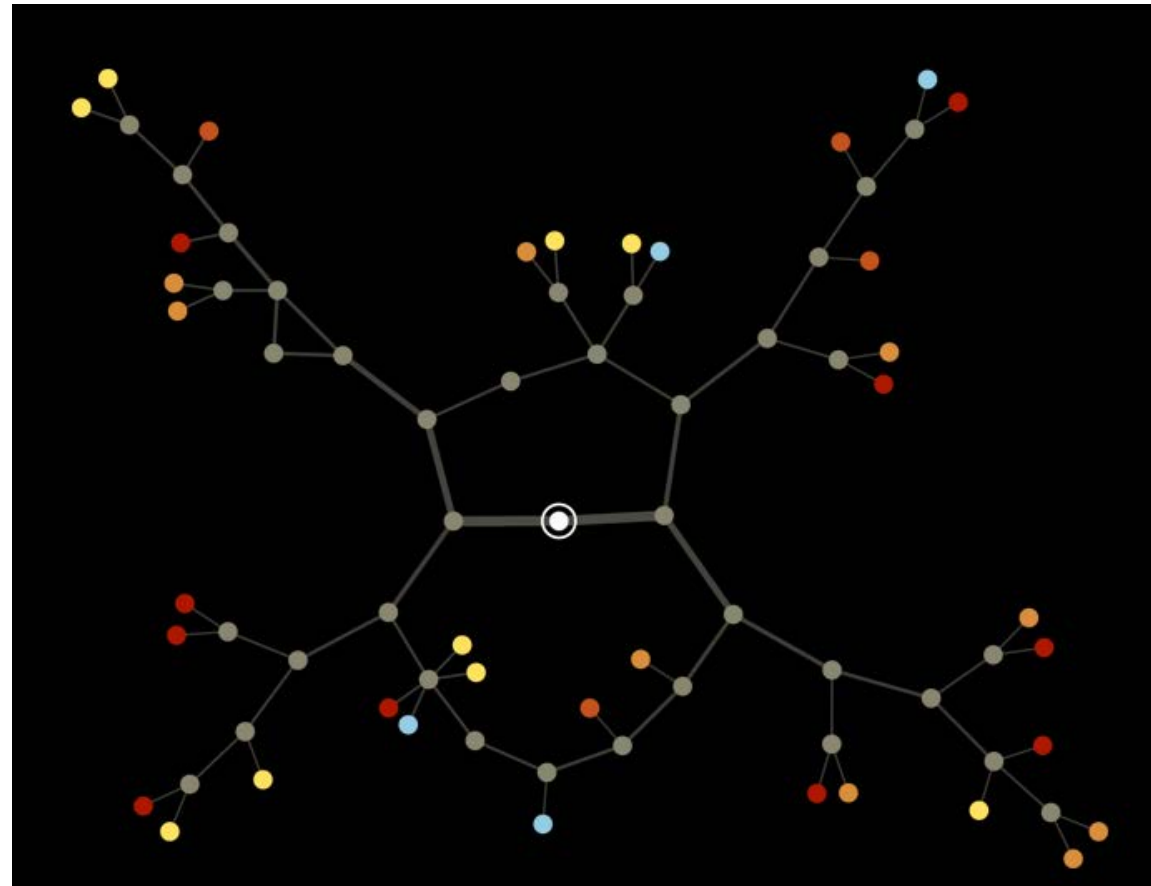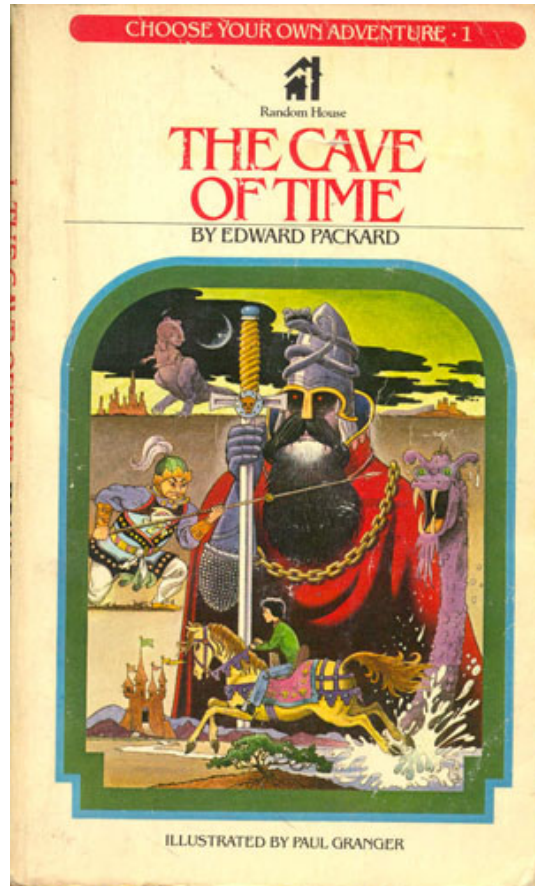*For more information on the Bat-Signal, turn to page 119.*

78

As you sit in the warm sunshine deciding about going to Morocco, you catch sight of a small girl—actually a midget—leading a dog. The midget walks up to you, hands you the leash, and before you realize that the dog is a mechanical dog, not a real one, it explodes into a thousand brilliant shards of metal. The explosion finishes you off.

UGH! What a horrible way to go.

The End

The Cave of Time, by Edward Packard. Choose Your Own Adventure · 1, Random House. Illustrated by Paul Granger.

common choice ·····>

first page

catastrophic ending

disappointing

mediocre

decent

rare choice ·····>

great

http://samizdat.cc/cyoa/gallery/cave-of-time.html#endings

Twine

- Open-source tool for building interactive stories
- Outputs to HTML
- Built upon HTML, CSS and JS

Twine is good at:

- Stories
- Poetry
- Text-based RPG
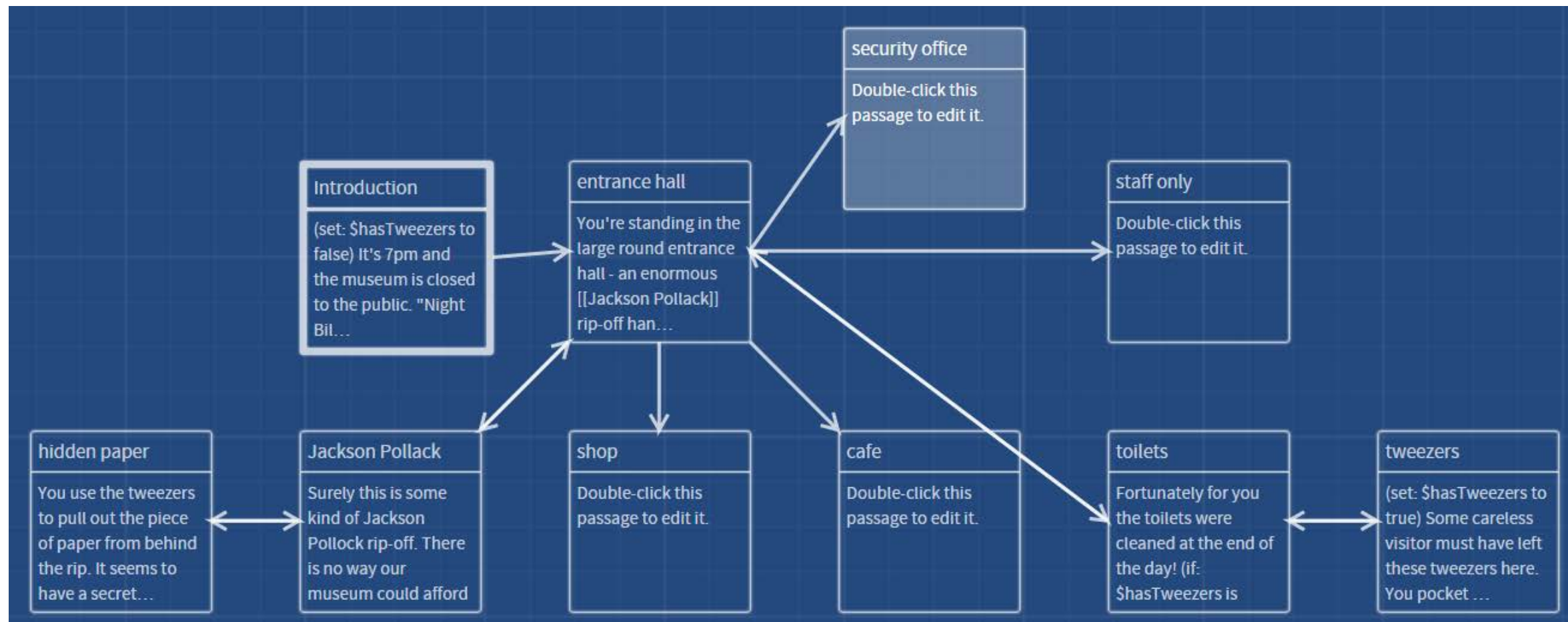- Hypermedia art
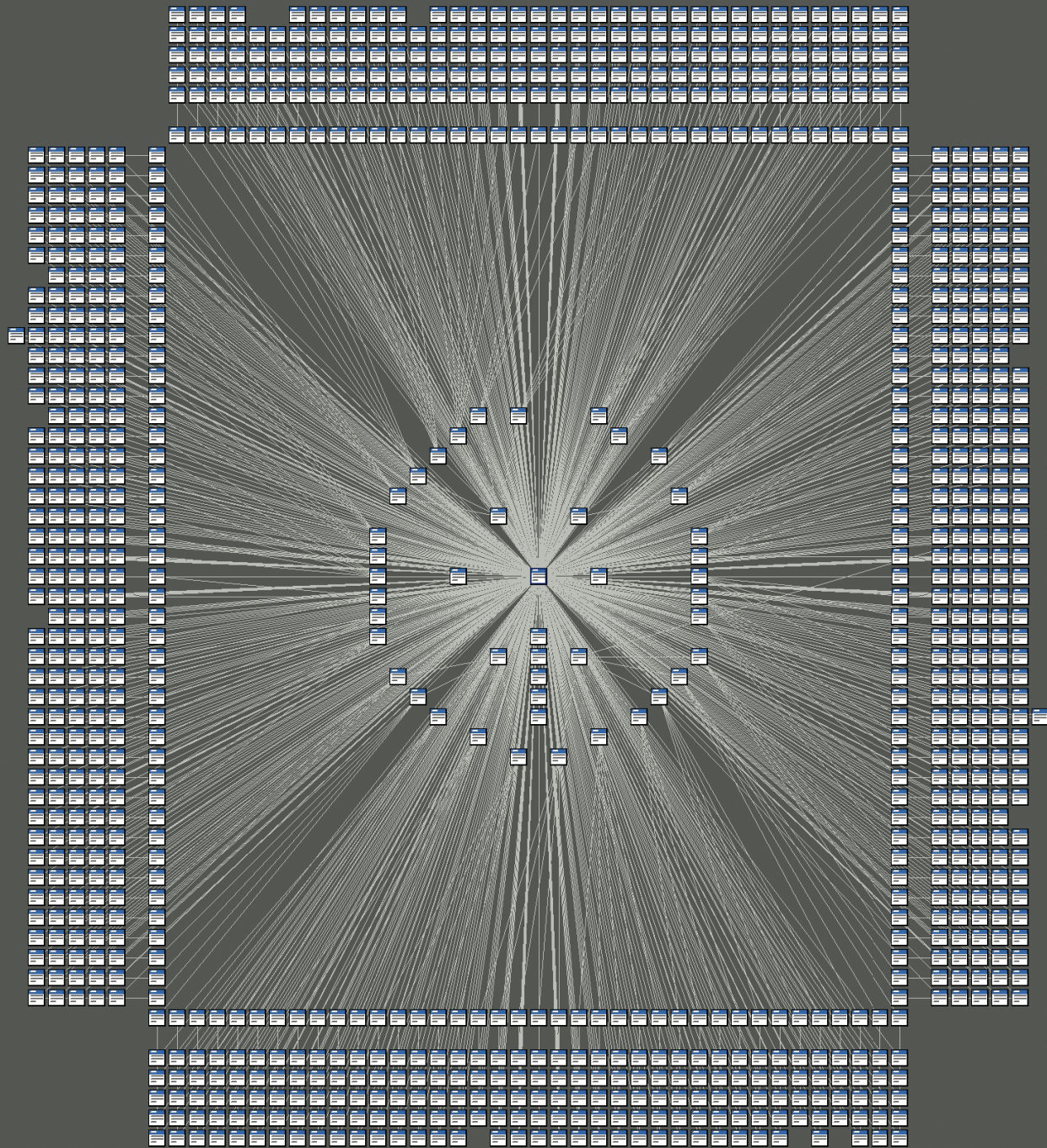- Prototyping

Twine Examples

- Story
  - [Lifeline](#) (mobile app based on twine)
  - [Player 2](#)
  - [Queers in Love at the End of the World](#)
  - [Even Cowgirls Bleed](#)
  - [The Uncle Who Works for Nintendo](#)
- RPG/Puzzle
  - [Candy Quest 3: Edge of Sweetness](#)
  - [Live, Run, Die Shop](#)
- Empathy Games
  - [Cis Gaze](#)
- Poetry
  - [A Kiss](#)
  - [Burnt Matches](#)
- Other
  - [Twineplat](#)
  - [HHH.exe](#)

Non-Twine Examples

- [My boyfriend came back from the war](#)
- [Sleep is Death](#)
- [Digital: A Love Story](#)
- [Hatoful Boyfriend](#)

security office

Double-click this passage to edit it.

Introduction

(set: $hasTweezers to false) It's 7pm and the museum is closed to the public. "Night Bil…

entrance hall

You're standing in the large round entrance hall - an enormous [[Jackson Pollack]] rip-off han…

staff only

Double-click this passage to edit it.

hidden paper

You use the tweezers to pull out the piece of paper from behind the rip. It seems to have a secret…

Jackson Pollack

Surely this is some kind of Jackson Pollock rip-off. There is no way our museum could afford

shop

Double-click this passage to edit it.

cafe

Double-click this passage to edit it.

toilets

Fortunately for you the toilets were cleaned at the end of the day! (if: $hasTweezers is

tweezers

(set: $hasTweezers to true) Some careless visitor must have left these tweezers here. You pocket …

Twine is not so good at:

- 2D/3D graphics-based games
- Platormers, first-person shooters, etc.
- For those, check out: [Phaser](), [Three.js](), [Unity]()

Installing Twine

- Options:
  - Twine 1.4.2 – the old standalone application
  - Browser-based – the latest version of twine (2.01), running in the browser
  - Twine 2.01 – the new standalone application
- Download 2.01 from here: twinery.org

Twine UI Demo

Story Formats

- SugarCube
  - Easy to pick up
  - Flexible, includes save system, widely used
- Harlowe
  - Default in Twine 2
  - A little more restrictive than SugarCube
- Snowman
  - Advanced
  - Allows you to write raw HTML/CSS/JS easily

SugarCube Installation

1. Download the current local version of [SugarCube 2.x for Twine 2](#).
2. Extract the archive to a safe location on your computer and take note of the path to it. I recommend some place like: Documents/Twine/Formats.
3. Click on the Formats link in the Twine 2 sidebar.
4. In the dialog that opens, click on the Add a New Format tab.
5. Finally, paste a [file URL](#) to the format.js file, based on the path from step #2, into the textbox and click the +Add button.
6. Set SugarCube as the default under "Story Formats"

SugarCube

SugarCube Documentation

- Documentation: motoslave.net/sugarcube/2/
- Important sections to start with:
  - Markup – info on formatting
  - TwineScript – info on variables
  - Macros – info on SugarCube's built-in functionality

Headings

An exclamation point which begins a line defines the heading markup. It consists of one to six exclamation points, each additional one beyond the first signifying a lesser heading.

Type	Syntax	Example	Rendered As
Level 1	`!Level 1 Heading`	# Level 1 Heading	`<h1>Level 1 Heading</h1>`
Level 2	`!!Level 2 Heading`	## Level 2 Heading	`<h2>Level 2 Heading</h2>`
Level 3	`!!!Level 3 Heading`	### Level 3 Heading	`<h3>Level 3 Heading</h3>`
Level 4	`!!!!Level 4 Heading`	#### Level 4 Heading	`<h4>Level 4 Heading</h4>`
Level 5	`!!!!!Level 5 Heading`	##### Level 5 Heading	`<h5>Level 5 Heading</h5>`
Level 6	`!!!!!!Level 6 Heading`	###### Level 6 Heading	`<h6>Level 6 Heading</h6>`

Basic Formatting

Type	Syntax	Example	Rendered As
Emphasis	`//Emphasis//`	*Emphasis*	`<em>Emphasis</em>`
Strong	`''Strong Emphasis''`	**Strong Emphasis**	`<strong>Strong Emphasis</strong>`
Underline	`__Underline__`	<u>Underline</u>	`<u>Underline</u>`
Strikethrough	`==Strikethrough==`	~~Strikethrough~~	`<s>Strikethrough</s>`
Superscript	`Super^^script^^`	Super^{script}	`Super<sup>script</sup>`
Subscript	`Sub~~script~~`	Sub$_{script}$	`Sub<sub>script</sub>`
Code, Inline	`{{{Code}}}`	Code	`<code>Code</code>`

Code, Block	`{{{`
	`Code`
	`}}}`
		Code	`<pre>Code</pre>`

Em-dash	`Em--Dash`	Em—Dash	Em—Dash
Avoiding formatting (all markup inside is not transformed and rendered as-is)			
	`"""Non-formatted"""`	No `''//formatting//''`	No `''//formatting//''`
	`<nowiki>Non-formatted</nowiki>`	No `''//formatting//''`	No `''//formatting//''`

Images

SugarCube's wiki image syntax consists of a required `Image` component and optional `Title`, `Link`, and `Setter` components. The `Image`, `Title`, and `Link` components may be either plain text or any valid TwineScript expression, which will be evaluated early (i.e. when the link is initially processed). The `Setter` component (which only works with passage links, not external links) must be a valid TwineScript expression, of the `<<set>>` macro variety, which will be evaluated late (i.e. when the link is clicked on).

The `Image` component value may be any valid URL to an image resource (local or remote) or the title of an embedded image passage (pre-Twine 2 only). The `Link` component value may be the title of a passage or any valid URL to a resource (local or remote).

Also, in addition to the standard pipe separator (`|`) used to separate the `Image` and `Title` components (as seen below), SugarCube also supports the arrow separators (`->` & `<-`). Particular to the arrow separators, the arrows' direction determines the order of the components, with the arrow always pointing at the `Image` component (i.e. the right arrow works like the pipe separator, `Title->Image`, while the left arrow is reversed, `Image<-Title`).

For the following examples assume: `$src` is `home.png`, `$go` is `Home`, and `$show` is `Go home`

Type	Syntax	Example	Result	
Image	`[img[Image]]`	`[img[home.png]]` `[img[$src]]`	**Image:**	home.png
Image w/ Link	`[img[Image][Link]]`	`[img[home.png][Home]]` `[img[$src][$go]]`	**Image:**	home.png
			Link:	Home
Image w/ Link & Setter	`[img[Image][Link][Setter]]`	`[img[home.png][Home][$done to true]]` `[img[$src][$go][$done to true]]`	**Image:**	home.png
			Link:	Home
			Setter:	$done to true
Image w/ Title	`[img[Title|Image]]`	`[img[Go home|home.png]]` `[img[$show|$src]]`	**Title:**	Go home
			Image:	home.png
Image w/ Title & Link	`[img[Title|Image][Link]]`	`[img[Go home|home.png][Home]]` `[img[$show|$src][$go]]`	**Title:**	Go home
			Image:	home.png
			Link:	Home
Image w/ Title, Link, & Setter	`[img[Title|Image][Link][Setter]]`	`[img[Go home|home.png][Home][$done to true]]` `[img[$show|$src][$go][$done to true]]`	**Title:**	Go home
			Image:	home.png
			Link:	Home
			Setter:	$done to true

Links

Single Link

+ Tag

```
[[Open Door]]
```


Single Link

[[Open Door]]

Open Door

The door has been opened.
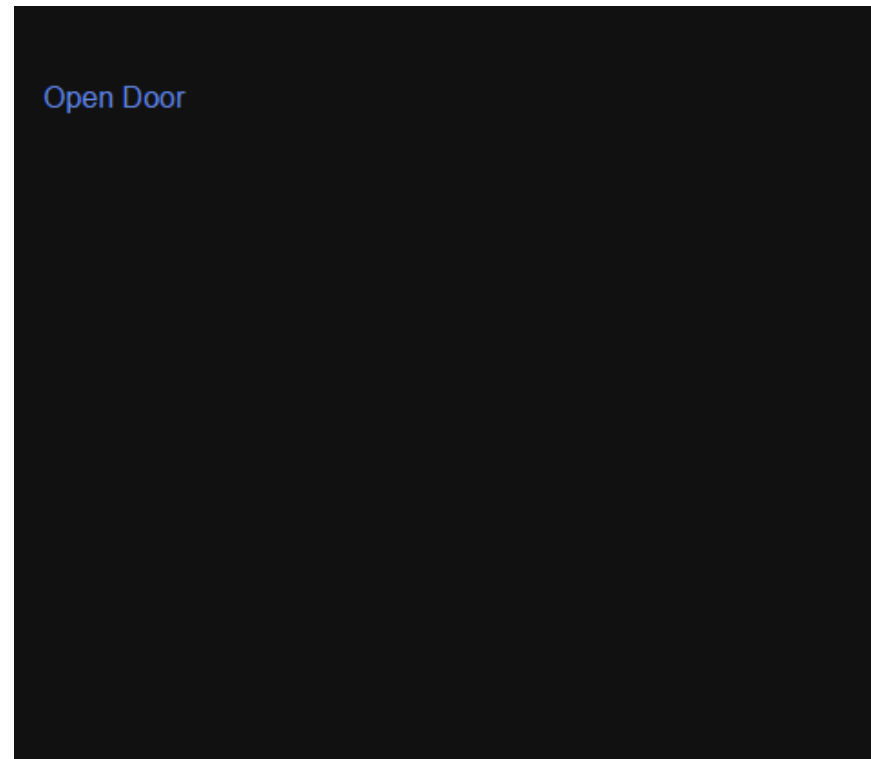

Open Door

Named Links

＋Tag

```
[[Open the spooky door->Spooky door]]
[[Eat a jellybean|Eat]]
[[Run<-Run away as fast as you can]]
```

Named Links

[[Open the spooky door->Spooky door]]
[[Eat a jellybean|Eat]]
[[Run<-Run away as fast as you ...

Spooky door

Double-click this passage to edit it.

Eat

Double-click this passage to edit it.

Run

Double-click this passage to edit it.

Open the spooky door

Eat a jellybean

Run away as fast as you can

Linking to Media

Strategies

- Hotlinking
  - Linking to a file someone else has hosted online
  - If that person takes the file down, you lose access to it
- Hosting Yourself
  - Create a GitHub repository called MyTwines
  - Upload images/sounds/video to folders there
  - Link to them using a URL: http://mikewesthad.github.io/MyTwine/images/cat.png
- Encoding & Embedding
  - More info later…

Variables

Variables Are Containers
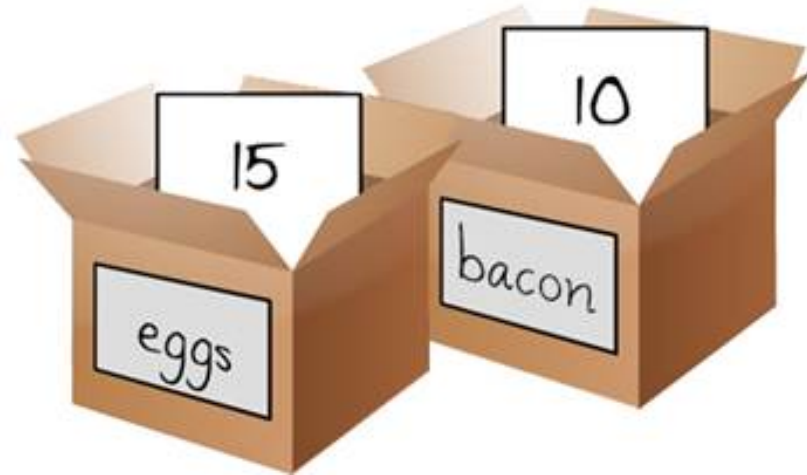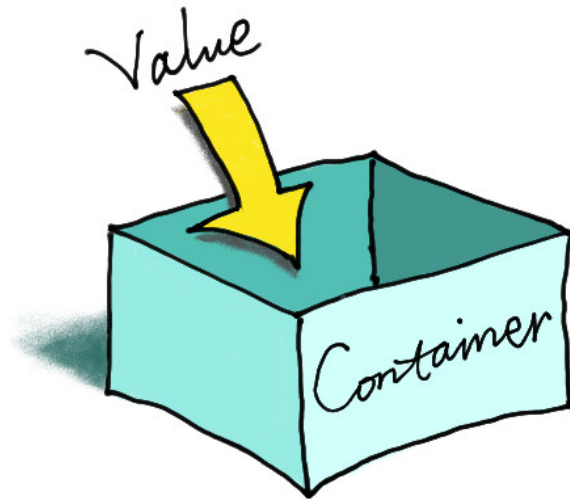

```
Variables                                    ✖

  ✚Tag

<<set $bacon = 10>>
<<set $eggs = 15>>
```

Twine Variable Types

- Strings
  - A series of characters inside of double quotation marks
- Numbers
  - Whole numbers or decimals
- Booleans
  - Either true or false

```
<<set $name = "Charizard">>
<<set $weight = 200>>
<<set $canFly = true>>
```

Twine Story Variables

- Must start with a dollar sign ($)
- Second character: A – Z, a – z, $ or _
- Any other characters: A – Z, a – z, 0 – 9, $ or _
- Pick meaningful, descriptive names!
- We will use lowerCamelCase

```
$cash
$age
$preferredNickname
$hasKeyCard2
$hasMetRobot
```

Macros

Macros

- IA macro is a piece of programming functionality
  - Under the hood, macros run a piece of JavaScript
- We've already used a macro:

```
<<set $canFly = true>>
```

- In SugarCube, there are three general types of macros
  - Documentation: motoslave.net/sugarcube/2/docs/macros.html

Set Macro
(Single Expression Macro)

`<<set expression>>`

Sets story $variables and temporary _variables based on the given expression.

Arguments:

- **expression:** A valid expression. See Variables and Expressions for more information.

Set Macro
(Single Expression Macro)

Start of Macro

TwineScript Expression

`<<set $canFly = true>>`

Macro Name

End of Macro

Textbox Macro
(Multiple Expression Macro)

Oh, hello there... *Er, I know this...* Okay. I knew this once. What's your name?

Continue

Textbox Macro
(Multiple Expression Macro)

```
<<textbox variable_name default_value [passage_name] [autofocus]>>
```

Creates a text input box, used to modify the value of the $variable with the given name, optionally forwarding the user to another passage.

SEE: Interactive macro warning.

Arguments:

- **variable_name:** The name of the $variable to modify, which *must* be quoted (e.g. `"$foo"`). Object and array property references are also supported (e.g. `"$foo.bar"`, `"$foo['bar']"`, & `"$foo[0]"`).
- **default_value:** The default value of the text box.
- **passage_name:** (optional) The name of the passage to go to if the return/enter key is pressed.
- **autofocus:** (optional) Keyword, used to signify that the text box should automatically receive focus. Only use the keyword *once* per page; attempting to focus more than one element is undefined behavior.

Usage:

```
→ Creates a text box which modifies $pie
What's your favorite pie? <<textbox "$pie" "Blueberry">>

→ Creates an automatically focused text box which modifies $pie
What's your favorite pie? <<textbox "$pie" "Blueberry" autofocus>>

→ Creates a text box which modifies $pie and forwards to the "Cakes" passage
What's your favorite pie? <<textbox "$pie" "Blueberry" "Cakes">>

→ Creates an automatically focused text box which modifies $pie and forwards to the "Cakes" passage
What's your favorite pie? <<textbox "$pie" "Blueberry" "Cakes" autofocus>>
```
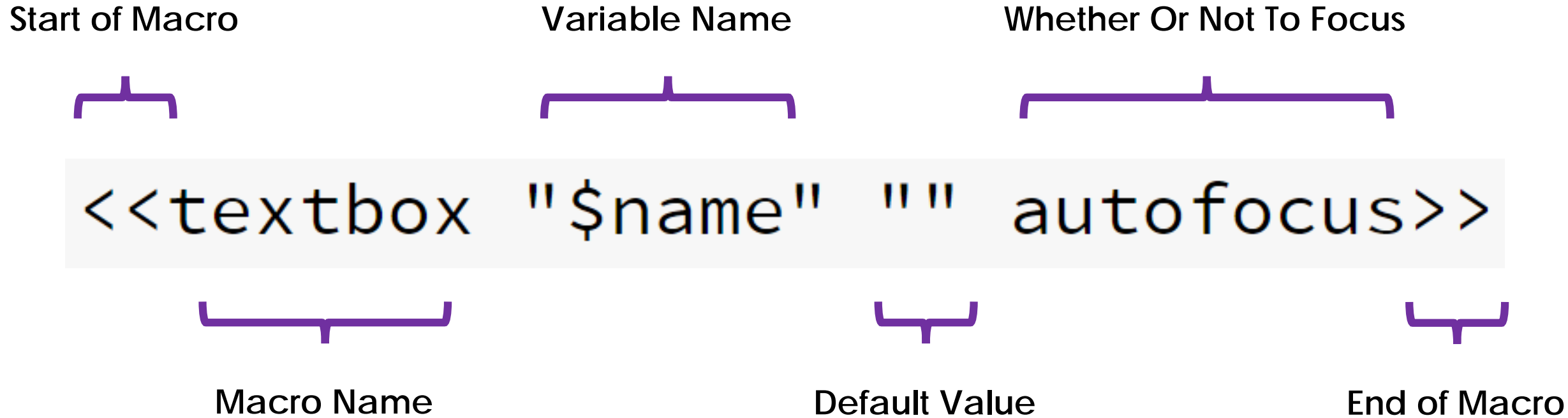
Textbox Macro
(Multiple Expression Macro)

Start of Macro

Variable Name

Whether Or Not To Focus

`<<textbox "$name" "" autofocus>>`

Macro Name

Default Value

End of Macro

Note: When a macro takes multiple parameters, you need to separate each parameter with whitespace!

Conditional Macros

Conditionals

- Oftentimes, you will need to do something only under certain conditions
  - E.g. if the player has the key, let them open the chest
  - E.g. if the player's health drops to zero, restart the game
- In order to express this in code, we need two things:
  - A way to compare variables (conditional operators)
  - A new set of macros (if, elseif, else)

JavaScript conditional operators: (not an exhaustive list)

`==`	Evaluates to `true` if both sides are equal (e.g. `$bullets == 6`).
`!=`	Evaluates to `true` if both sides are not equal (e.g. `$pie != "cherry"`).
`===`	Evaluates to `true` if both sides are *strictly* equal (e.g. `$bullets === 6`).
`!==`	Evaluates to `true` if both sides are *strictly* not equal (e.g. `$pie !== "cherry"`).
`>`	Evaluates to `true` if the left side is greater than the right side (e.g. `$cash > 5`).
`>=`	Evaluates to `true` if the left side is greater than or equal to the right side (e.g. `$foundStars >= $neededStars`).
`<`	Evaluates to `true` if the left side is less than the right side (e.g. `$shoeCount < ($peopleCount * 2)`).
`<=`	Evaluates to `true` if the left side is less than or equal to the right side (e.g. `$level <= 30`).
`!`	Flips a `true` evaluation to `false`, and vice versa (e.g. `!$hungry`).
`&&`	Evaluates to `true` if all subexpressions evaluate to `true` (e.g. `$age >= 20 && $age <= 30`).
`||`	Evaluates to `true` if any subexpressions evaluate to `true` (e.g. `$friend === "Sue" || $friend === "Dan"`).

Conditional Macros

(Opening/Closing Macro)

`<<if conditional_expression>>`, `<<elseif conditional_expression>>`, & `<<else>>`

Executes its contents if the given conditional expression evaluates to `true`. If the condition evaluates to `false` and an `<<elseif>>` or `<<else>>` exists, then other contents can be executed.

NOTE: SugarCube does not trim whitespace from the contents of `<<if>>` macros, so that authors don't have to resort to various kludges to get whitespace where they want it. However, this means that extra care must be taken when writing them to ensure that unwanted whitespace is not created within the final output.

Arguments:

- **conditional_expression:** A valid conditional expression, evaluating to either `true` or `false`. See Expressions for more information.

Conditional Macros
(Opening/Closing Macro)

Conditional Expression

Start of Macro {

```
<<if $score >= 50>>
        Hey, you passed.
<<​/if>>
```

End of Macro {

Note: assume we already have a variable called $score

Conditional Macros
(Opening/Closing Macro)

```
<<if $score >= 50>>
    Hey, you passed.
<<else>>
    Well, better luck next time.
<</if>>
```

Conditional Macros
(Opening/Closing Macro)

```
<<if $score >= 90>>
    Nice, an A.
<<elseif $score >= 50>>
    Hey, you passed.
<<else>>
    Well, better luck next time.
<</if>>
```

```
<!-- Checking a Boolean -->
<<if $isRaining>>
    Bring an umbrella.
<</if>>
```

```
<!-- Comparing a String -->
<<if $name == "Lassie">>
    What, Timmy's fallen in the well?
<</if>>
```

Special Passages

Passage Names

Passage	Description	
PassageDone	Used for post-passage-display tasks, like redoing dynamic changes (happens after the rendering and display of each passage). Equivalent to the postdisplay task object.	
PassageFooter	Appended to each rendered passage. Mostly equivalent to the postrender task object.	
PassageHeader	Prepended to each rendered passage. Mostly equivalent to the prerender task object.	
PassageReady	Used for pre-passage-display tasks, like redoing dynamic changes (happens before the rendering of each passage). Equivalent to the predisplay task object.	
Start	**Twine 1:**	*Required.* The starting passage, the first passage displayed. Configurable, see Config.passages.start for more information.
	Twine 2:	Not special. Choose the starting passage by marking it via the 🚀 (rocket ship icon) passage context-menu item.
StoryAuthor	Sets the authorial byline in the UI bar (element ID: story-author).	
StoryBanner	Set the story's banner in the UI bar (element ID: story-banner).	
StoryCaption	Sets the story's caption in the UI bar (element ID: story-caption).	
StoryInit	Used for pre-story-start initialization tasks, like variable initialization (happens at the beginning of story initialization).	
StoryMenu	Sets the story's menu items in the UI bar (element ID: menu-story).	
StorySettings	**Twine 1:**	Not special and not used. The Configuration Object serves the same basic purpose.
	Twine 2:	
StoryShare	Used to populate the contents of the Share dialog.	
StorySubtitle	Sets the story's subtitle in the UI bar (element ID: story-subtitle).	
StoryTitle	**Twine 1:**	*Required.* Sets the story's title in the UI bar and elsewhere (element ID: story-title).
	Twine 2:	Not special. The story's title/name is part of the story project.
Compiler Passages (not meaningful to story formats, listed only for completeness)		
StoryIncludes	**Twine 1:**	Used by the compiler to include other Twine (.tws) or Twee (.twee, .tw) source files during compilation.
	Twine 2:	Unsupported by the compiler (i.e. Twine 2).

StoryInit

+ Tag

```
<<set $hasKey = false>>
<<set $score = 0>>
```

Cycling Link Macro

Custom Macros

- If you know JavaScript, you can write your own macro
- The <<cyclinglink>> macro allows you to create a link that cycles through a series of options when you click on it
- Installing the macro
  - Download the macro here
  - Open up cyclinglink-macro.min.js in VS Code
  - Copy and paste it into you "Story JavaScript" in Twine

**Variable to Store
Current Link**

`<<cyclinglink "$pieType" "apple" "blueberry" "pumpkin">>`

Values to Cycle Through

Audio Macros

Supported Audio Formats

- MP3, MP4 – all browsers
- WAV, Ogg – decent support

Format	Container	Chrome	Internet Explorer	Edge	Firefox	Opera	Safari
PCM	WAV	Yes	No	Yes	Yes, in v3.5	Yes, in v11.00	Yes, in v3.1
MP3	-	Yes	Yes, in IE9	Yes	From OS[a]	From OS[a]	Yes, in v3.1
AAC	MP4	Yes	Yes, in IE9	Yes	From OS[a]	From OS[a]	Yes
AAC	ADTS[b]	Yes	No	Yes	From OS, in v45.0	No	Yes
Vorbis	Ogg	Yes, in v9	No	No	Yes, in v3.5	Yes, in v10.50	With Xiph QuickTime Components
Vorbis	WebM	Yes	No	No[15]	Yes, in v4.0	Yes, in v10.60	No
Opus	Ogg	Yes, in v25 (in v31 for Windows)	No	No	Yes, in v15.0	Yes, in v14	No
Opus	WebM	Yes	No	Yes[16][17]	Yes, in v28.0[18]	Yes	No
FLAC	FLAC	Yes, in v56[19]	No	?	Yes, in v51[20]	No	No
FLAC	Ogg	Yes, in v56[19]	No	?	Yes, in v51[20]	No	No

Supported Audio Formats

- MP3 – compressed files, small files
- WAV – most are uncompressed files, large files
- Online audio converter

Supported Audio Formats

- Twine desktop doesn't play MP3s
- When publishing as HTML, the browser *will* play MP3s
- WAV & Ogg will work in Twine

Sounds for Exercise

- github.com/mikewesthad/Class-TwineMedia

Sound Resources

- [Incompetech](#) – music
- [Freesound.org](#) – sound effects, ambient sounds
- [ChipTone](#) – online app for making game sounds
- [Newgrounds Audio](#) – music (not all is downloadable)
- [Audacity](#) – free audio editing software

Playing Sounds

1. Load all sounds using <<cacheaudio>>
   - Best to do this in a passage called "StoryInit"
2. Play/stop/loop/fade sounds with <<audio>>

`<<cacheaudio track_id source_list>>`

Caches an audio track for use by the other audio macros.

NOTE: The `StoryInit` passage is normally the best place to set up audio tracks.

Arguments:

- **track_id:** The ID of the audio track, which will be used by the other audio macros to reference it.
- **source_list:** A space separated list of sources for the audio track. Only one is required, though supplying additional sources in differing formats is recommended. A source must be either a URL (absolute or relative) to an audio resource or a data URI. In rare cases where the audio format cannot be automatically detected from the source (URLs are parsed for a file extension, data URIs are parsed for the media type), a format specifier may be prepended to the front of each source to manually specify the format (syntax: `format:{format};`, where `{format}` is the audio format; generally, use whatever the file extension would normally be, e.g. `mp3`, `mp4`, `ogg`, `weba`, `wav`).

Usage:

```
→ Cache a track with the ID "boom" and one source (mp3)
<<cacheaudio "boom" "media/audio/explosion.mp3">>

→ Cache a track with the ID "bgm_space" and two sources (mp3 & ogg)
<<cacheaudio "bgm_space" "media/audio/space_quest.mp3" "media/audio/space_quest.ogg">>

→ Cache a track with the ID "what" and one source (mp3) with a format specifier
<<cacheaudio "what" "format:mp3;http://an-audio-service.com/a-user/a-track-id">>
```

`<<audio track_id action_list>>`

Controls the playback of individual audio tracks, which must be set up via `<<cacheaudio>>`.

NOTE: Group IDs do not affect playlist tracks which have been copied into their respective playlist—meaning those set up via `<<createplaylist>>` with its `copy` action or all tracks set up via, the deprecated, `<<setplaylist>>`.

Arguments:

- **track_id:** The ID of the audio track. Special group IDs, which affect multiple tracks at once, may also be used. Available group IDs are: `:all`, `:looped`, `:muted`, `:paused`, and `:playing`.
- **action_list:** The list of actions to perform. Available actions are:
  - **play:** Start playback.
  - **pause:** Pause playback.
  - **stop:** Stop playback.
  - **fadein:** Start playback and fade the audio from its current volume level to `1` (loudest).
  - **fadeout:** Start playback and fade the audio from its current volume level to `0` (silent; will also pause playback).
  - **fadeto** *level*: Start playback and fade the audio from its current volume level to the specified level.
  - **fadeoverto** *seconds level*: Start playback and fade the audio from its current volume level to the specified level over the specified number of seconds.
  - **volume** *level*: Set the current volume level to the specified level. Valid values are floating-point numbers in the range `0` (silent) to `1` (loudest) (e.g. `0` is 0%, `0.5` is 50%, `1` is 100%).
  - **time** *seconds*: Set the current playback time to the specified number of seconds. Valid values are floating-point numbers in the range `0` (start) to the maximum duration (e.g. `60` is `60` is sixty seconds in, `90.5` is ninety-point-five seconds in).
  - **mute:** Mute the volume (effectively volume `0`, except without changing the volume level).
  - **unmute:** Unmute the volume.
  - **loop:** Set the audio to repeat playback upon ending.
  - **unloop:** Set the audio to not repeat playback (this is the default).
  - **goto** *passage*: Forwards the user to the passage with the given name when playback ends normally. May be called either with the passage name or with a wiki link.

Usage (normal track IDs):

```
→ Given the following (best done in the StoryInit special passage)
<<cacheaudio "bgm_space" "media/audio/space_quest.mp3" "media/audio/space_quest.ogg">>

→ Start playback
<<audio "bgm_space" play>>

→ Start playback at 50% volume
<<audio "bgm_space" volume 0.5 play>>

→ Start playback at 120 seconds in
<<audio "bgm_space" time 120 play>>

→ Start repeating playback
<<audio "bgm_space" loop play>>

→ Start playback and fade from 0% to 100% volume
<<audio "bgm_space" volume 0 fadein>>

→ Start playback and fade from 75% to 0% volume
<<audio "bgm_space" volume 0.75 fadeout>>

→ Start playback and fade from 25% to 75% volume
<<audio "bgm_space" volume 0.25 fadeto 0.75>>

→ Start playback and fade from 25% to 75% volume over 30 seconds
<<audio "bgm_space" volume 0.25 fadeoverto 30 0.75>>

→ Start playback and goto the "Peace Moon" passage upon ending normally
<<audio "bgm_space" play goto "Peace Moon">>

→ Pause playback
<<audio "bgm_space" pause>>

→ Stop playback
<<audio "bgm_space" stop>>

→ Mute playback, without changing the current playback state
<<audio "bgm_space" mute>>

→ Unmute playback, without changing the current playback state
<<audio "bgm_space" unmute>>

→ Change the volume to 40%, without changing the current playback state
<<audio "bgm_space" volume 0.40>>

→ Seek to 90 seconds in, without changing the current playback state
<<audio "bgm_space" time 90>>
```

Usage (group IDs):

```
→ Start playback of paused individual audio tracks
<<audio ":paused" start>>

→ Pause playback of playing individual audio tracks
<<audio ":playing" pause>>

→ Stop playback of playing individual audio tracks
<<audio ":playing" stop>>

→ Stop playback of all individual audio tracks (equivalent to <<stopallaudio>>)
<<audio ":all" stop>>
```