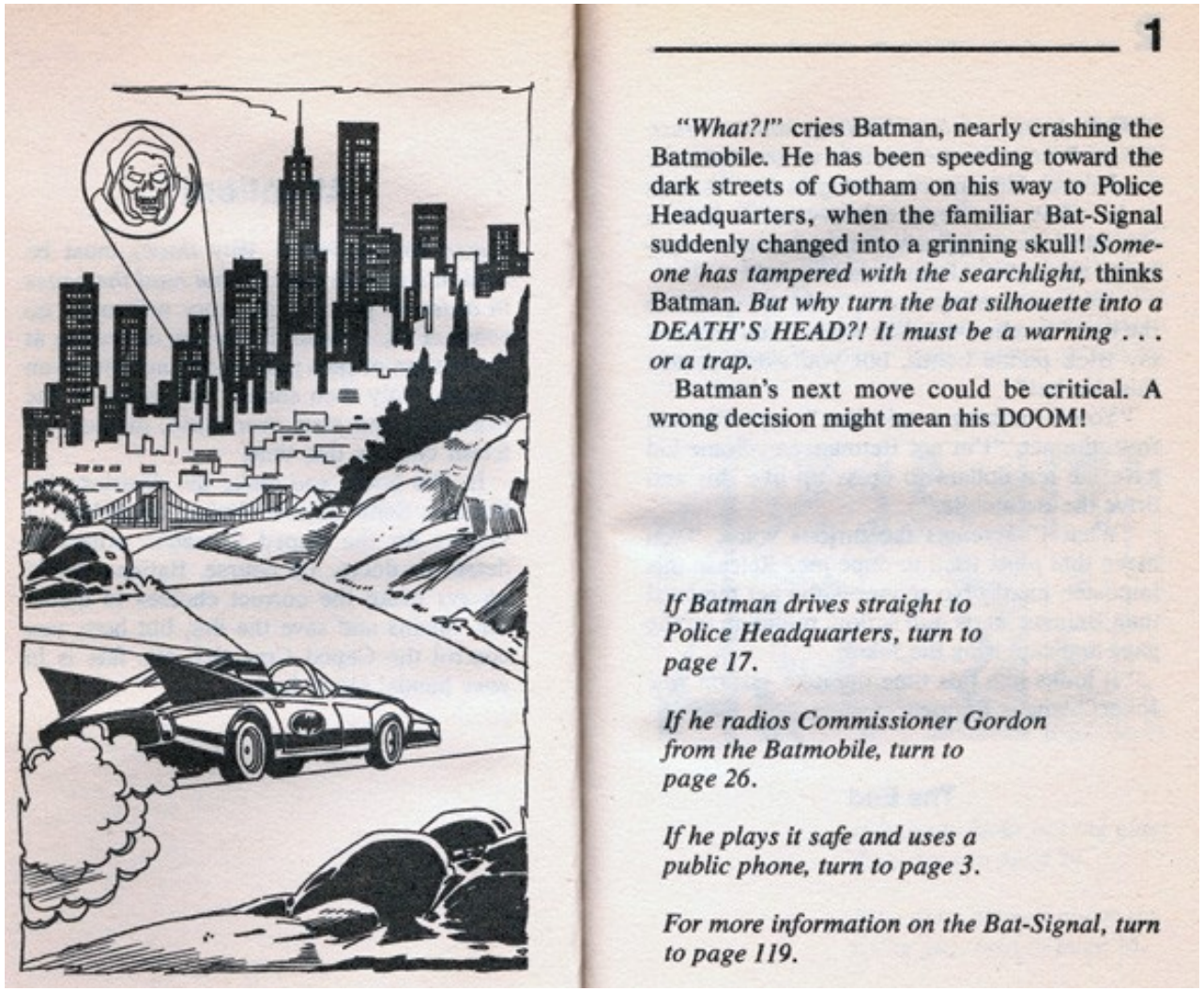
Twine Workshop

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.

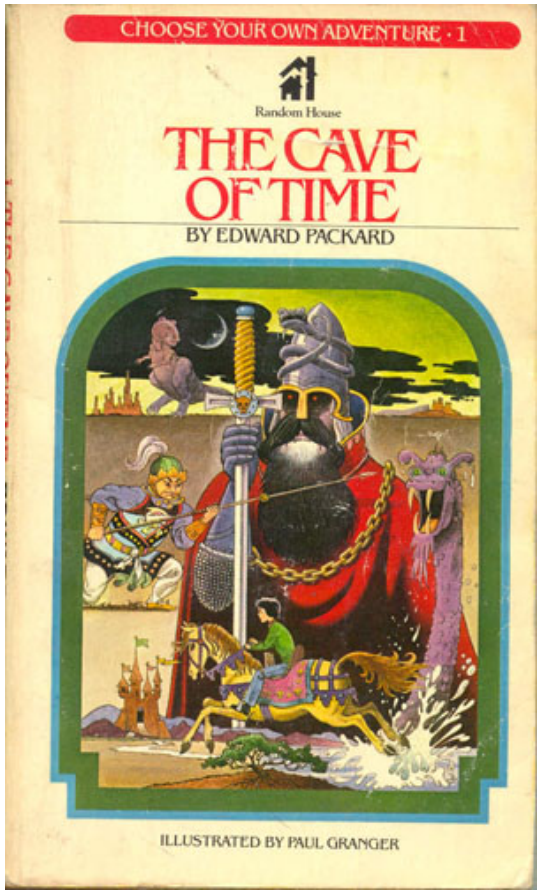
For more information on the Bat-Signal, turn to page 119.

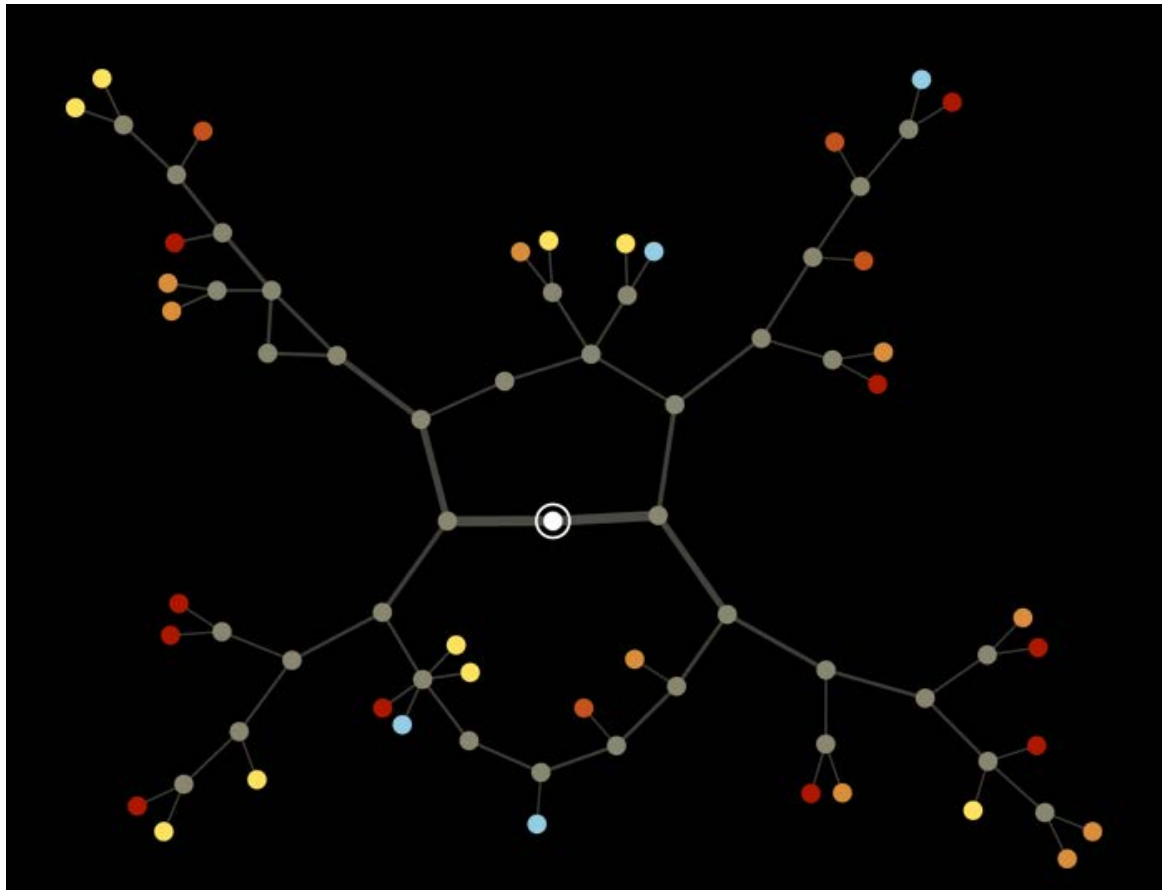
78

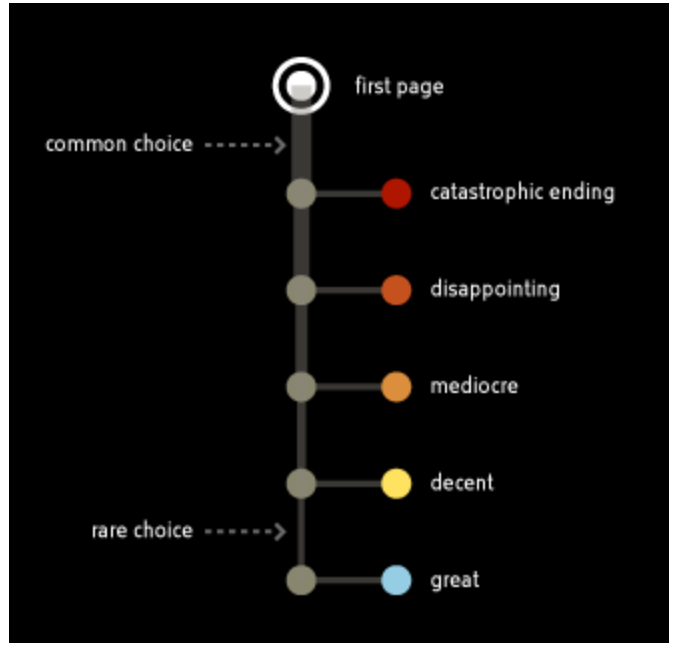
As you sit in the warm sunshine deciding about going to Morocco, you catch sight of a small girl—actually a midget—leading a dog. The midget walks up to you, hands you the leash, and before you realize that the dog is a mechanical dog, not a real one, it explodes into a thousand brilliant shards of metal.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

UGH! What a horrible way to go.

The End







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:

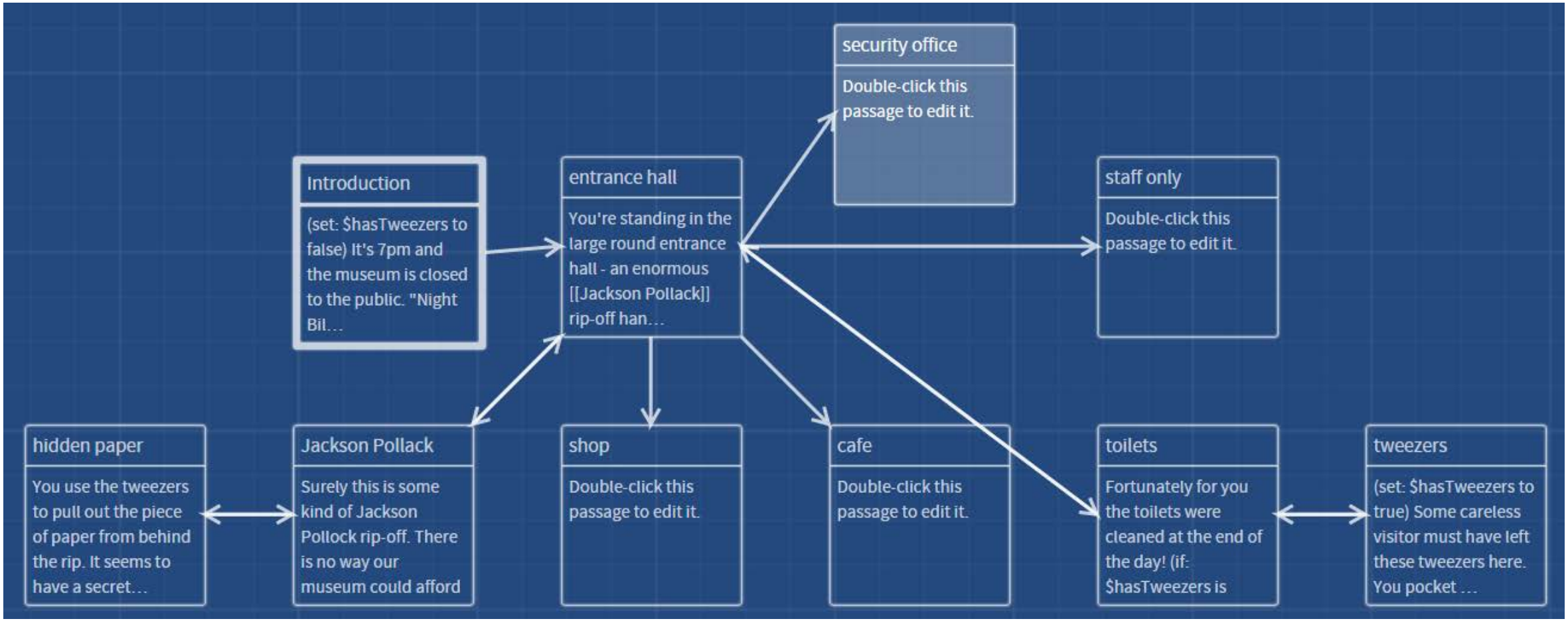
- Stories
- Poetry
- Text-based RPG
- Hypermedia art
- Prototyping

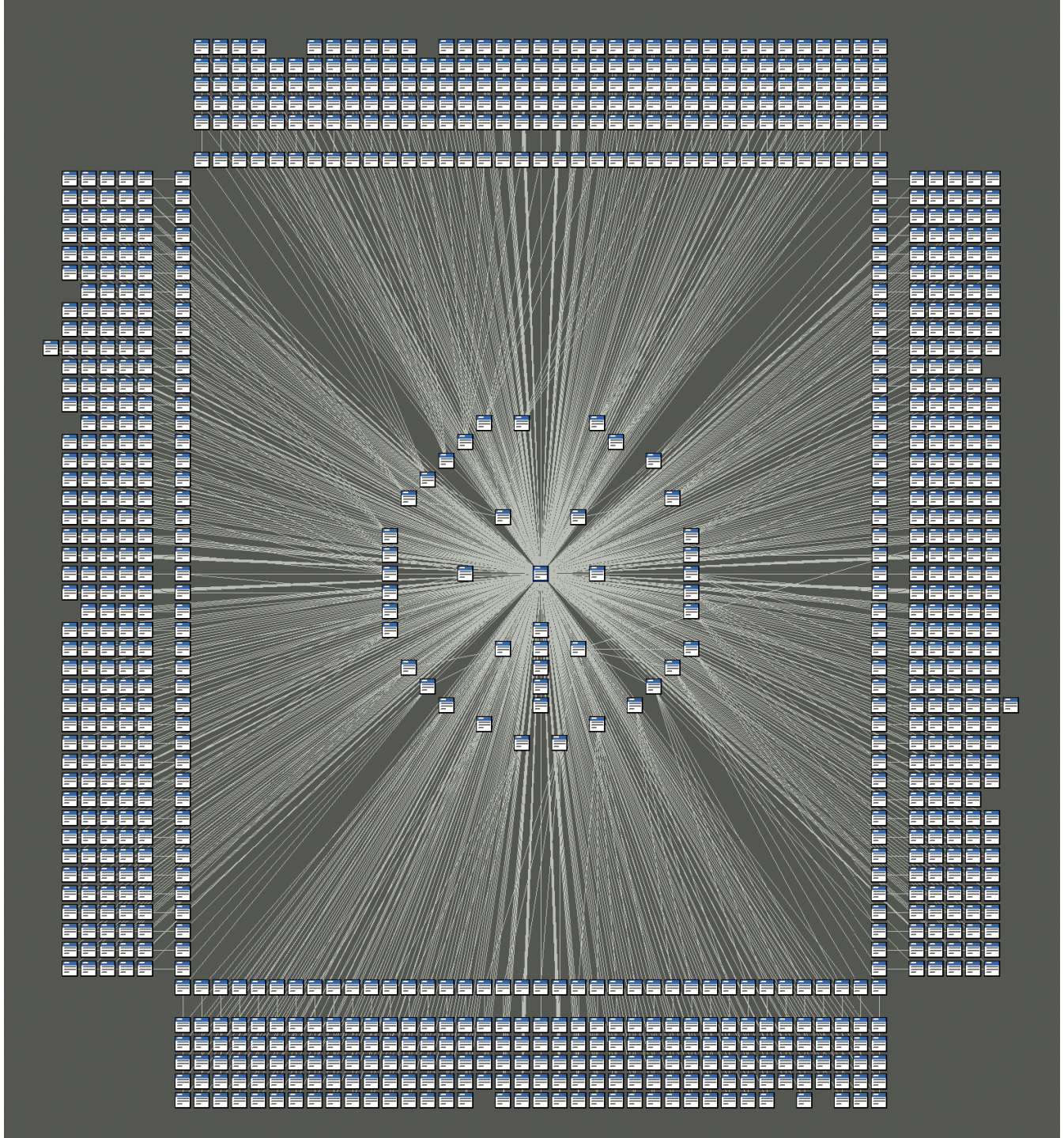
Twine Examples

- 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__`	Underline	`<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&mdash;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`