AW CONTENT WRITING GUIDE

In order ensure a consistent standard for the game, and to make writing as straight-forward as possible, I've put together this guide. This guide is only going to cover the basics of using SugarCube logic/macros but should be sufficient for most writing. Detailed information and examples are available at the MotoSlave website, should you need them.

CONTENTS

PERSPECTIVE & BASICS

All content in the game will be from the player's perspective and should be written in **second person perspective**. More information on second person form can be found online, but in general you could imagine it as the "narrator" explaining to the person what they are doing. The words "you, your, you're" are used to describe the player character. In first person, the narrator is describing what happened to the narrator. In second person the narrator is describing what happened to you. The exception to second person in this game is when handling internal thoughts, which use a special tag. Internal monologue is written in first person, as it could be considered a "quote" of the character's thoughts.

> You step into the filthy breakroom. *I can't believe I have to clean up this mess, can't these slobs clean up after themselves?*

TENSE

Generally, the writing should be in past or present tense, unless in the context of a hypothetical or internal thoughts. The game is happening as you read it, so usually it won't make sense to use future tense. Generally significant action should be in present tense, while past tense can be used to set the scene or explain previous events.

OTHER STANDARDS:

The player should not read text describing something that the player character would not know. This is a basic of second person, but unlike first person it's considered bad form to use the character's future version to relay information. You can have a player secretly observe

something, or have an NPC tell them, but avoid writing something like "little did you know, in the next room SadFlower was pouring aphrodisiac and Rohypnol into your drink".

In general, you should avoid putting character-defining thoughts into the player's head.

Good: That reminds me... I must pay the bills when I get home.
Bad: What I really need right now is a big stinky Cleveland Steamer, they're my favorite.

In addition, the writing style is to use primarily narrator description to describe things, rather than internal monologue. Try to avoid heavy use of internal monologue unless some aspect of the story necessitates it.

REQUIREMENTS

There are a few basic requirements that are really necessary to allow writing to be included in the game. The following items should be pretty self-explanatory and are fairly basic. If you have a question if something applies to what you're doing or would like to do something that seems like it may violate a rule, just ask about it.

WRITING SHOULD:

- Be proof read, without excessive typos or grammar and spelling errors.
- Appropriately accommodate different player characters and NPCs. For example: don't assume all PCs have wide hips or enjoy anal, or don't assume all NPCs have giant cocks.
    - Parser calls will help for descriptive purposes, and logic can be used to split up text by variable. *If you must use logic to separate different attributes, avoid going overboard and making a ton of custom variations!*
- Should provide alternatives for any gated or fetish content. For example, if you're writing a special NPC who suddenly unleashes a bunch of S&M or water-sports content, you should provide an obvious way for the player to opt out before the content happens. (I am open to including most fetishes/kinks, but do not want to force them on players!)
- Provide information on game mechanics for a scene as appropriate. Examples are the amount of time that passes, how pleasurable or arousing an action is, if it makes the player stressed or sad, etc.
- Be submitted with a brief summary of the content at the top, as well as a summary of any new, unusual, or complex game mechanics.
- Be organized so that it is obvious how the different parts of the text relate to each other.

BASIC FORMAT

1] The writing should be in either a Google Docs or Microsoft Word format. Google Docs makes sharing and collaboration easier, not to mention it's free.

2] The exact details of the font and design are up to you, as long as it is legible. If you use a special color scheme or fonts to signify special details, please provide a key for their meaning.

3] The beginning of the document should include a basic table of contents, and different sections should be well labeled so that it is clear how they fit together.

4] Write a brief summary describing the submission so that it's clear what it's about. Be sure to include basic information about location and involved NPCs as appropriate.

5] There should be a section at the beginning listing any new variables you would like to include in the game for your content, such as a story flag or some other value. Give a brief (one sentence or less) description of each variable's purpose.

6] Include a section if necessary describing any special mechanics the content requires to be implemented. In many cases this won't be necessary, but if you need something special built to accommodate the story, be sure to explain it.

7] Logic and parser calls should use twine script if at all possible. HTML tags should be used for any special formatting. If you are unable to determine the correct twine script, wrap a description of what you need in brackets that are bold and a noticeable color. If the description needs to be long, you can just write **[note 1]** and describe the note at the top of the section.

8] Break long segments of story into different sections. This can be done by using link/choice points, or by using time, encounters or other common-sense places to break up the content into sections.

9] Each section should have a basic set of information at the beginning covering changes like increased arousal or some other stat that occurs in the section, as well as basic information about location/scene/characters or other details relevant to implementing it. *It's also very helpful if you include justifications for why something happens, particularly if your content alters the player character or potentially forces them into a certain course of action.*

10] Try to keep similar sections in a group together and/or clearly organized. If you have three choices for a dialog option, for example, don't place the "result" sections all over the place if possible. You can organize the content in one of two ways, both have their advantages and disadvantages, and it's really a matter of preference.

Chronologically:

- Opening story, choice 1.
- Choice 1 option A, choice 2.
- Choice 1 option B, choice 3.
- Choice 1 option C, choice 4.
- Choice 2 option A, end
- Choice 2 option B, end
- Choice 3 option A, choice 5.
- Choice 3 option B, choice 6
- Choice 3 option C, choice 5
- Etc

Tree/Hierarchy:

1. Opening story, choice1
    a. Option A, choice 2
        i. Option A, end
        ii. Option B, end
    b. Option B, choice 3
        i. Option A, choice 5
            1. Option a
            2. Option b
        ii. Option B, choice 6
            1. Option a
            2. Option b
        iii. Option C, choice 5
            1. Option a
            2. Etc…

USEFUL SYNTAX

For the most part, writing will be displayed with html markup, really just the most basic tags are needed. There are also SugarCube Macros that can be used for logic and parser calls. In general html formatting is preferred to SugarCube markup because it is less prone to problems/conflicts. However it's fine to use SugarCube markup for simple style items.

Be sure not to use "smart quotes" like these! They cause a ton of problems. Use a standard quotation marks like this " instead.

Any special styling you make to the document itself won't be reflected in the game text. If you want bold or colored text, for example, you have to use some of the below items.

MARKUP & TAGS

<tag> HTML tag – used for HTML layout and style

<<macro>> SugarCube Macro – used for logic and parser calls.

/* This is a comment. */ Comment Block

Applying a class for use with dialog and internal monologue.

PC's spoken dialog | @@.pc;PC's spoken dialog@@

NPC's spoken dialog | @@.npc;NPC's spoken dialog@@

 internal monologue | @@.mono; internal monologue@@

Line Breaks

 Paragraph of writing
 single line break.

Printing a variable directly:

<<= $variable>>

Using a parser call: [see parser documentation for exact details!]

<<p "attribute" [npc]>>

Attribute = a specific keyword such as “breastSize” that is wrapped in quotes.

npc *optional* = if excluded returns text for PC. Otherwise returns text based on the npcid (ex n101 for Lily), active npc index number (0-9) for npcs 1 through up to 10 that are active in a scene. You can also use $T for targeted NPC such as during sex scenes.

Temporary Variable: _variable *erased when leaving a passage*

Story Variable: $variable *stored permanently. Generally you will only need to worry about $flag.variable avoid creating other variables.*

Type	Syntax & Example	Rendered As	Displays As (*roughly*)
Emphasis	`//Emphasis//`	`<em>Emphasis</em>`	*Emphasis*
Strong	`''Strong Emphasis''`	`<strong>Strong Emphasis</strong>`	**Strong Emphasis**
Underline	`__Underline__`	`<u>Underline</u>`	Underline
Strikethrough	`==Strikethrough==`	`<s>Strikethrough</s>`	~~Strikethrough~~
Superscript	`Super^^script^^`	`Super<sup>script</sup>`	Super^{script}
Subscript	`Sub~~script~~`	`Sub<sub>script</sub>`	Sub$_{script}$
Blockquote	`>Line 1` `>Line 2` `>>Nested 1` `>>Nested 2`	`<blockquote>Line 1 ` `Line 2 ` `<blockquote>Nested 1 ` `Nested 2 ` `</blockquote></blockquote>`	Line 1 Line 2 Nested 1 Nested 2
Code, Inline	`{{{Code}}}`	`<code>Code</code>`	`Code`

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

List, Unordered	`* A list item`
	`* Another list item`
		`<ul>`
		`<li>A list item</li>`
		`<li>Another list item</li>`
		`</ul>`
			• A list item
			• Another list item

List, Ordered	`# A list item`
	`# Another list item`
		`<ol>`
		`<li>A list item</li>`
		`<li>Another list item</li>`
		`</ol>`
			1. A list item
			2. Another list item

Horizontal rule	`----`	`<hr>`	---
Em-dash	`A--B`	`A&mdash;B`	A—B
No Format, triple quotes	`"""No //format//"""`	`No //format//`	No //format//
No Format, <nowiki>	`<nowiki>No //format//</nowiki>`	`No //format//`	No //format//

If you are familiar with CSS and styles for webpages, you can use the below markup to shorten applying them to text. Note that nesting them often doesn't work properly and should be avoided if possible. Additionally, you can't use colons or semicolons inside the @@ notation.

Type	Syntax	Example	Rendered As
Highlight, Inline	@@Text@@	@@Text@@	Text
Custom Style, Inline	@@*style-list*[1];Text@@	@@#foo;.bar;Text@@	Text
		@@color:red;Text@@	Text
Highlight, Block	@@ Text @@	@@ Text @@	Text
Custom Style, Block	@@*style-list*[1]; Text @@	@@#foo;.bar; Text @@	Text
		@@color:red; Text @@	Text

Links: to go to a new passage.

For the following examples assume: `$go` is `"Grocery"` and `$show` is `"Go buy milk"`

Type	Syntax	Example	Result	
Link	[[Link]]	[[Grocery]] [[$go]]	**Text:**	Grocery
			Link:	Grocery
Link w/ Text	[[Text|Link]]	[[Go buy milk|Grocery]] [[$show\|$go]]	**Text:**	Go buy milk
			Link:	Grocery
Link w/ Setter	[[Link][Setter]]	[[Grocery][$bought to "milk"]] [[$go][$bought to "milk"]]	**Text:**	Grocery
			Link:	Grocery
			Setter:	$bought to "milk"
Link w/ Text & Setter	[[Text|Link][Setter]]	[[Go buy milk|Grocery][$bought to "milk"]] [[$show|$go][$bought to "milk"]]	**Text:**	Go buy milk
			Link:	Grocery
			Setter:	$bought to "milk"

LOGIC

You don't need to be a programmer to contribute content! If you're writing an involved scene though, you do need to write it so that it works for the different character types. Once I get the descriptive parsers finished, I will publish a list of parser calls so that you can use them in your writing. An example of a call would be <<p "tits">>. You could write your sentence: "he reaches up and squeezes your <<p "tits">>". This automatically puts in some descriptive words for the player's breasts based on the variables, followed by a word for breasts. This one won't always describe size, so if you need to emphasize that you can always use <<p "breastSize">> <<p

“breasts”>> which will give a size adjective followed by an appropriate word for breasts based on the character. (If you think you have a good candidate for a new parser call, let me know and I can add it.)

More detailed differences based on different variables will require some programming logic, as will some other things to a certain extent. The most commonly used item is the IF ELSEIF ELSE tool, which can get you really far.

<<if $variable is true>>then print this<<else>>print this<</if>>

<<if $variable == 2>>then write this<<elseif $variable == 1>>then write this<<elseif $variable == 0>>then write this<<else>>or write this<</if>>

An example would be:

<<if $boobsize > 10 && not $wearBra>>they bounce ponderously with each step up the forest trail, making the hike more difficult.<<elseif $boobsize > 10>>they jiggle enticingly with each step up the forest trail, but the bra keeps them from slowing your progress.<<else>>they jiggle happily with the exercise as you make your way up the forest trail.<</if>>

There are more detailed instructions on using these in the MotoSlave documentation, and plenty of guides available online. I will eventually be creating a list of variables and functions that could be useful for creating content.