

Strand IF Authoring

Introduction

Terms

Flow is a fundamental concept. Flow is the way in which the event sequence of a story unfolds.

Here is a story;

```
STORY
Once there were three bears, then they all died!
The End.
```

OK, not a very exciting story (or even very original), but it illustrates the building block called the *term*. ;

Anything in capital letters is a *term* and the rest is *flow*.

STORY is a *term* and the text, "Once there were three bears, then they all died! The End." is its *flow*.

Let's run this story and see the output;

```
Once there were three bears, then they all died! The End.
```

You may have noticed the line break in the **STORY**, before "The End", does not appear in the output. Line breaks in text flows are ignored.

You can however have blank lines in text flow, for example;

```
STORY
Once there were three bears, then they _all_ **died!**

**The End.**
```

Produces the output;

```
Once there were three bears, then they all died!

The End.
```

Notice you can use markdown in text to indicate **bold** and *italic*.

Let's put in some more terms;

```
STORY
ONCE there were THREE bears, then they all died!
The End.

ONCE
Once

THREE
three
```

This produces the exact same output as before, except now the terms `ONCE` and `THREE` emit their own flow.

Selectors

Now, we can give these terms branching flow using *selectors*. For example;

```
STORY
ONCE there were THREE bears, then they all died!
The End.

ONCE
* Once
* Once upon a time
* A long time ago

THREE
* two
* three
* four
* five
```

The term `ONCE` has three *selectors* which give alternative flows. `ONCE` will choose one of the three selectors *randomly*. The same for `THREE`.

Now we get random stories, each time we run!

Once upon a time there were five bears, then they all died! The End.

A long time ago there were four bears, then they all died! The End.

So, just with these bear-bones (sic) basics, we can already make stuff;

```
STORY
ONCE there were THREE bears, PART1. DIED. THEEND

ONCE
* Once
* Once upon a time
* A long time ago

THREE
* two
* three
* four
* five

PART1
* they were forced into lockdown in their HOUSE
* they all lived in a HOUSE

HOUSE
* small cottage
* tiny house
* woodland shack
* garden shed
```

DIED
Unfortunately they
* contracted VIRUS
* were poisoned by POISON

and died

POISON
* tainted porridge
* a little blond girl
* dodgy mushrooms

VIRUS
* Covid19
* a nasty flu
* tuberculosis
* the plague

THEEND
* The End.
* It's all over.
* That's the end!

Some sample outputs;

A long time ago there were five bears, they were forced into lockdown in their garden shed.
Unfortunately they were poisoned by a little blond girl and died. It's all over.

Once upon a time there were two bears, they all lived in a woodland shack. Unfortunately they were
poisoned by tainted porridge and died. It's all over.

Once upon a time there were four bears, they were forced into lockdown in their woodland shack.
Unfortunately they contracted tuberculosis and died. That's the end!

You can have text *before* selectors and/or more text *after* selectors (eg **DIED**) but you **cannot have more than one set of selectors in a term**. You'll see why in the next section.

Sticky Selectors

A problem with random selectors is that sometimes you need to pick something at random and then stick with it. For example, a person's name.

Here's an example where we want to be random, but maintain consistency;

STORY
MARY had a little LAMB its FLEECE was WHITEASSNOW.
And everywhere that MARY went, the LAMB was sure to go.

MARY!
* Mary
* Larry
* Barry
* Harry
* Gary

```
* Sally

LAMB!
* lamb
* chicken
* puppy
* dog
* cat

FLEECE
* fleece
* coat
* fur
* tail

WHITEASSNOW
* white as snow
* red as blood
* green as grass
* pink as a fairy
* blue as sky
```

In this example, the terms **MARY** and **LAMB** have the *indicator* **!** after their definition, but not their usage in the **STORY** .

The **!** indicates the term is *sticky*.

A *sticky* term is one whose flow occurs just *once* and thereafter the flow output remains the same.

Here's some outputs;

Larry had a little puppy its fur was green as grass. And every where that Larry went the puppy was sure to go.

Harry had a little chicken its tail was red as blood. And every where that Harry went the chicken was sure to go.

Unfortunately, it doesn't always rhyme anymore!

There are other indicators you can apply to a term such as ones to control the ordering, the randomness and so on.

Term Indicators

These affect the manner in which selectors are used.

Here are the possible term indicators, many of these are explained later.

Indicator	Name	Description
!	sticky	Runs flows <i>once</i> thereafter emits the same output.
&	shuffle	Selectors are "shuffled" and executed in a random order.
#	nonrandom	Selectors chosen with biased random preventing starvation.
<	sequence	Selectors run in order.
=	first	First valid selector is run.
?	choice	Term is a choice.
@	object	Term is an object.

Choices

So far we've only covered terms used as generators. But;

The difference between a player choice and a generator is whether the choice is made by the *player* or by the *machine*.

When you want to make a player choice, you use the *exact same syntax* but mark the term as a choice using **?**.

Here's a new story to illustrate;

STORY

Once upon a time there were three bears.

What's your name? NAME

Ok NAME, here's the deal, the three bears have just gone out for a walk.

ROBTHEM That's the end of the game.

NAME?!

- * Goldilocks
- * Mousilocks
- * Dreadlocks

ROBTHEM?

Do you want to rob their place?

- * yes

BREAKIN

- * no

Well then!

BREAKIN

You horrible person! Well ok then since that's the plot.

You break into their house like a pro.

Whoa! Lookee here; on the table are three bowls of porridge. LIKEPORRIDGE

VANDAL

You

- * set the place on fire and leave.
- * smash the place up. That will show 'em.
- * throw the porridge all over the walls, about all it's good for really.

LIKEPORRIDGE?

Do you like porridge?

* Hell yes!

Right on! WHICHBOWL

* A bit

WHICHBOWL

* No, it's awful!

Oh well, not much else here then. VANDAL Bears suck anyway!

WHICHBOWL

There are three bowls; a large one, a medium one and a small one. BOWL

BOWL?

Which do you want to try NAME?

* the biggest of course!

Eww! It's cold and lumpy! BOWL

* the medium sized one

Eww! It's way too sweet! BOWL

* the smallest one

This one's not too bad. A bit small though, so you eat the lot!

Suddenly the door slams open and three angry bears are glaring right at you! WHATNOW

WHATNOW?

What now?

* Run like hell!

Nice try, but they swiftly block your exit. MAULED

* Climb a tree.

Be serious, there are no trees in here! WHATNOW

* Put your hands up and plead innocent.

Sorry, the cute routine doesn't work with bears! MAULED

* Say; NAME went that-a-way, and point out the window!

It worked! The bears all rush out to find the culprit.

Quickly, you exit the shack and run away as fast as you can. That was close! MORAL

* Fight!

You rush them, kicking and screaming like a banshee, perhaps they'll run off? MAULED

YOU DIE

They

* leave you to die slowly and painfully,

* watch you die slowly, your blood squirting out in slow motion,

* laugh as you die screaming in agony,

while they eat their porridge!

VICIOUSLY

* viciously

* ferociously

* mercilessly

* easily

MAULED

The bears grab you and VICIOUSLY tear all your arms and legs off! Bears are _that_ strong you know. YOU DIE MORAL

MORAL

Perhaps you won't mess with bears next time!

Let's see how this plays and explain the details;

Once upon a time there were three bears.

What's your name?

- (1) Goldilocks
- (2) Mousilocks
- (3) Dreadlocks

2

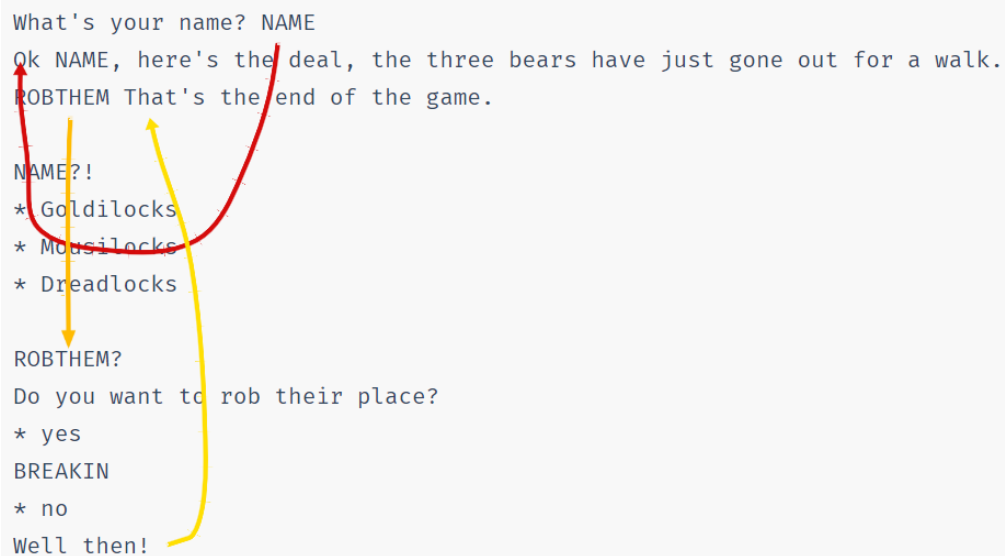
Ok Mousilocks, here's the deal, the three bears have just gone out for a walk. Do you want to rob their place?

- (1) yes
- (2) no

2

Well then! That's the end of the game.

The flow taken by this short playthrough is illustrated here;



What's your name? NAME
Ok NAME, here's the deal, the three bears have just gone out for a walk.
ROBTHEM That's the end of the game.
NAME?!
* Goldilocks
* Mousilocks
* Dreadlocks
ROBTHEM?
Do you want to rob their place?
* yes
BREAKIN
* no
Well then!

Here's another, slightly longer play;

Once upon a time there were three bears.

What's your name?

- (1) Goldilocks
- (2) Mousilocks
- (3) Dreadlocks

1

Ok Goldilocks, here's the deal, the three bears have just gone out for a walk. Do you want to rob their place?

- (1) yes
- (2) no

1

You horrible person! Well ok then since that's the plot. You break into their house like a pro.

Whoa! Lookee here; on the table are three bowls of porridge. Do you like porridge?

- (1) Hell yes!
 - (2) A bit
 - (3) No, it's awful!
- 3

Oh well, not much else here then. You throw the porridge all over the walls, about all it's good for really. Bears suck anyway! That's the end of the game.

Here's the flow this time;

ROBTHEM?

Do you want to rob their place?

* yes

BREAKIN

* no

Well then!

BREAKIN

You horrible person! Well ok then since that's the plot.
You break into their house like a pro.

Whoa! Lookee here; on the table are three bowls of porridge. LIKEPORRIDGE

VANDAL

You

* set the place on fire and leave.

* smash the place up. That will show 'em.

* throw the porridge all over the walls, about all it's good for really.

LIKEPORRIDGE?

Do you like porridge?

* Hell yes!

Right on! WHICHBOWL

* A bit

WHICHBOWL

* No, it's awful!

Oh well, not much else here then. VANDAL Bears suck anyway!

You can see from these flow diagrams that flow enters terms and continues to flow into subterms until there is nowhere for it to go, at which point, it returns to the original flow and continues. When the flow finally stops, the game is over.

Finally, here is a longer play;

Once upon a time there were three bears.

What's your name?

- (1) Goldilocks
- (2) Mousilocks
- (3) Dreadlocks

1

Ok Goldilocks, here's the deal, the three bears have just gone out for a walk. Do you want to rob their place?

(1) yes

(2) no

1

You horrible person! Well ok then since that's the plot. You break into their house like a pro.

Whoa! Lookie here; on the table are three bowls of porridge. Do you like porridge?

(1) Hell yes!

(2) A bit

(3) No, it's awful!

1

Right on! There are three bowls; a large one, a medium one and a small one. Which do you want to try Goldilocks?

(1) the biggest of course!

(2) the medium sized one

(3) the smallest one

1

Eww! It's cold and lumpy!

Which do you want to try Goldilocks?

(1) the medium sized one

(2) the smallest one

2

This one's not too bad. A bit small though, so you eat the lot! Suddenly the door slams open and three angry bears are glaring right at you! What now?

(1) Run like hell!

(2) Climb a tree.

(3) Put your hands up and plead innocent.

(4) Say; Goldilocks went that-a-way, and point out the window!

(5) Fight!

2

Be serious, there are no trees in here!

What now?

(1) Run like hell!

(2) Put your hands up and plead innocent.

(3) Say; Goldilocks went that-a-way, and point out the window!

(4) Fight!

1

Nice try, but they swiftly block your exit. The bears grab you and viciously tear all your arms and legs off! Bears are *that* strong you know. They leave you to die slowly and painfully, while they eat their porridge! Perhaps you won't mess with bears next time! That's the end of the game.

It should now be clear how the flow works in this playthrough, but let's look at choice terms in a detail;

```
LIKEPORRIDGE?
Do you like porridge?
* Hell yes!
Right on! WHICHBOWL
* A bit
WHICHBOWL
* No, it's awful!
Oh well, not much else here then. VANDAL Bears suck anyway!
```

`LIKEPORRIDGE` is a choice term as indicated by `?`. It has an (optional) initial text flow `Do you like porridge?`, then it has three selectors each introduced by `*` at the start of a line.

You'll notice each selector has two parts:

- The *choice* flow on the same line as `*`.
- The *action* flow on the subsequent lines.

eg

```
* Hell yes!    // choice flow
Right on! WHICHBOWL // action flow
```

If the choice is selected, the *action* flow is taken. Terms can also occur in the choice flow as `NAME` in the following;

```
BOWL?
Which do you want to try NAME?
* the biggest of course!
Eww! It's cold and lumpy! BOWL
* the medium sized one
Eww! It's way too sweet! BOWL
* the smallest one
This one's not too bad. A bit small though, so you eat the lot!
Suddenly the door slams open and three angry bears are glaring right at you! WHATNOW
```

You'll also notice some of the actions in the `BOWL` term reference `BOWL` itself. This causes a loop back to the `BOWL` term.

By default, choice terms remember which choices have previously been used and these do not appear again. For example the option `Climb a tree` did not appear again when the options were presented again. The same happened as the different porridge bowls were chosen.

It turns out that selectors can also have indicators, and this can be used to tweak behaviour. eg. Indicating `*+ Climb a tree.` allows the option to be used more than once.

Conditionals

We can give selectors conditionals to allow them to be affected by previous choice decisions.

Selectors can be conditional in *both* generator *and* choice terms.

The good news for non-programmers is that there are no, programming style, *variables* in this language. Instead conditionals are built from whether terms have been visited or not.

Let's look at an example;

