Getting Started with Inform 7

PART FOUR: Making Your World Behave

Actions, rules, the say command, if statements, conditions, and ending the game.

ACTIONS

An action is what happens behind the scenes when a player types in a command.

You can see these actions by typing in the command 'actions' in any game you are working on. The actions will then be displayed within square brackets after you type in game play commands. See the following example:

```
Foyer
There is an exit to the north.

>actions
Actions listing on.

>n
[going north]

Kitchen
There is an exit to the south.

You can see a hot potato here.

[going north - succeeded]
>x potato
[examining the hot potato]
This potato looks steaming hot.

[examining the hot potato - succeeded]
>take potato
[taking the hot potato]
Taken.
[taking the hot potato - succeeded]

>
```

Typing 'actions off' will turn off this feature.

It will be important to understand actions (and how to find them) before you learn how to manipulate the rules affecting these actions.

RULES

Inform has various rules that provide the ability to manipulate, change, and customize actions to serve our needs. The report rule has been discussed already in regards to traveling messages. We will discuss four of these rules here: the after rule, the before rule, the carry out rule, and the instead of rule.

The After Rule

This is the simplest rule to understand and implement. It simply means 'do something after an action has been carried out.' The after rule is most commonly utilized to customize the message printed out to the player after they perform an action.

The after rule takes this form:

```
After the action:
    do something.
```

For example, whenever the player 'takes' some object they are given the default message of 'taken':

```
>take shard of glass
Taken.

>
```

This may be satisfactory for most situations but what if that 'shard of glass' is sharp and we want to customize the message to imply that the player is taking care not to cut themselves when they 'take' it.

```
>take shard of glass
Being careful not to cut yourself, you gingerly take the shard
of glass.

>
```

First, we need to know what that action is. We do that with the 'actions' command as explained earlier:

```
>actions
Actions listing on.

>take shard of glass
[taking the shard of glass]
Taken.
[taking the shard of glass - succeeded]

>
```

Now we see the action that we wish to manipulate is [taking the shard of glass].

Here is how we use the after rule to manipulate this action and create a custom message:

```
After taking the shard of glass:
   say "Being careful not to cut yourself, you gingerly take the
piece of glass.
```

Here is another example when the player eats an edible hot potato:

```
>eat hot potato
You eat the hot potato. Not bad.

>
```

Add this code to change the above default response after eating the potato:

```
After eating the potato:
  say "Being careful not to burn yourself, you nibble at the
potato until it is all gone."
```

Here is how it will now look in play:

```
>eat potato
Being careful not to burn yourself, you nibble at the potato
until it is all gone.

>
```

The Before Rule
The before rule is used when we need to check some condition or do something before an action will be allowed to continue or be stopped.

For example, suppose we want to ensure that the player wears some heavy gloves before they are allowed to pick up that hot potato.

Here is the code for the gloves and the hot potato:

```
A pair of heavy gloves are in the Foyer. They are wearable.

The hot potato is in the Foyer. It is edible.
```

Let's break the before rule down line by line:

The first line invokes the before rule and says that when the player types 'take the hot potato' we need to check a few things before doing anything.

```
Before taking the hot potato:
```

The next line checks if the player is wearing the heavy gloves and, if they are, the action continues normally. (Note the use of the comma after the if statement and the semi-colon at the end of the line.)

```
if the heavy gloves are worn, continue the action;
```

The last line, beginning with 'otherwise', is invoked if the player is not wearing the heavy gloves (the previous if statement is false.) and prints the text within double quotations instead. (Note: The statement ends with the word 'instead' to stop the action.)

```
otherwise say "The potato is so hot that you immediately drop
it!" instead.
```

These format for the 'if' and 'otherwise' statements are used when we have a situation that we simply want an "either/or" outcome.

Here is the code in its entirety:

```
A pair of heavy gloves are in the Foyer. They are wearable.

The hot potato is in the Foyer. It is edible.

Before taking the hot potato:
  if the heavy gloves are worn, continue the action;
  otherwise say "The potato is so hot that you immediately
drop it!" instead.
```

Here is how this will look (and behave) in play:

```
Foyer
You can see a pair of heavy gloves and a hot potato here.

>take potato
The potato is so hot that you immediately drop it!

>wear gloves
(first taking the pair of heavy gloves)
You put on the pair of heavy gloves.

>take potato
Taken.

>
```

The Carry Out Rule

We utilize the carry out rule when we want some action to behave as it normally would but wish to include additional commands when the action is carried out.

In this example, we want the player to be able to turn a flashlight on and off.

```
The Foyer is a room.

The Closet is north of the Foyer. It is dark.

The flashlight is a device in the Foyer.

Carry out switching on the flashlight:
    now the flashlight is lit.

Carry out switching off the flashlight:
    now the flashlight is dark.
```

This is how it will look in play:

```
Foyer
You can see a flashlight here.

>go north

Darkness
It is pitch dark, and you can't see a thing!

>turn on flashlight
You switch the flashlight on.

Closet

>
```

The Instead Rule

The instead rule allows you to override or bypass an action in order to force something else to happen. A good time to use the instead rule is when we want the player to be able to drink something.

Here we have encountered a mountain stream and the player decides to take a drink.

```
>drink the water
[drinking the water]
There's nothing suitable to drink here.
[drinking water - succeeded]

>
```

The action we want to create an instead rule for is [drinking the water]. Look carefully at the following two lines:

```
Instead of drinking the water:
  say "With great trepidation, you take some water in the palm
of your hand, bring it to your lips, and drink..."
```

The first line says stop the action of drinking the water (followed by a colon.) The second line is what to do instead: print ('say') the text within double quotations.

Now, when the player types 'drink water', the second line will be printed and the illusion is created that the player drinks the water.

```
>drink water
With great trepidation, you take some water in the palm of
your hand, bring it to your lips, and drink...

>
```

Now that you know how to find the many actions going on, you can use the instead rule to alter any of them.

Ending the Game
The instead rule works well when we want some particular action to end the game successfully, in failure, or with a custom message.

Here are the four commands that will end a game:

```
end the game in victory

end the game in death

end the game saying "The potato burns your hand so badly that
you can no longer type."

end the story
```

Using the instead rule, here is how each of them is coded and will look:

Ending the Game in Victory

```
Instead of taking the hot potato:
  end the game in victory.
```

Here is how it will look in play:

```
You can see a hot potato here.

>take potato


    *** You have won ***


In that game you scored 0 out of a possible 0, in 2 turns.

Would you like to RESTART, RESTORE a saved game or QUIT?
>
```

Ending the Game in Death

```
Instead of taking the hot potato:
  end the game in death.
```

Here is how it will look in play:

```
You can see a hot potato here.

>take potato


    *** You have died ***


In that game you scored 0 out of a possible 0, in 2 turns.

Would you like to RESTART, RESTORE a saved game or QUIT?
>
```

If these examples seem a little too abrupt an ending, you can include the 'say' command to print a little more information. Note that after 'say' the message to be printed is enclosed in double quotations and the sentence ends with a semi-colon.

```
Instead of taking the hot potato:
  say "The potato is so hot that you burst into flames!";
  end the game in death.
```

or

```
Instead of taking the hot potato:
  say "The potato does not burn you. You are invincible!";
  end the game in victory.
```

Ending the Game with a Custom Message

```
Instead of taking the hot potato:
  end the game saying "The potato burns your hand so badly
that you can no longer type."
```

Here is how it will look in play:

```
You can see a hot potato here.

>take potato


    *** The potato burns your hand so badly that you can no
longer type. ***


In that game you scored 0 out of a possible 0, in 2 turns.

Would you like to RESTART, RESTORE a saved game or QUIT?
>
```

Ending the Story

```
Instead of taking the hot potato:
 end the story.
```

```
You can see a hot potato here.

>take potato


    *** The End ***


In that game you scored 0 out of a possible 0, in 2 turns.

Would you like to RESTART, RESTORE a saved game or QUIT?
>
```