Best Practices for Modular Development in Inform 7

For the purposes of this project, we ask that you pursue an isolationist policy with your code, preventing anything written for your room from affecting gameplay in other rooms. If you're experienced with Inform 7, you're probably already thinking of some possible vectors for conflict and strategies for avoiding them. Please let us know what we haven't thought of.

We also need to be aware that this project is going to be huge and slow, and every millisecond of lag counts when it comes time for somebody to play this monstrosity. **I refuse to urge restraint** but I must ask you to keep efficiency in mind and be flexible—Your code that runs unproblematically on its own may turn out to be an albatross around the neck of a 75-room game, and if it does, we'll want to work with you to correct it.

Naming Conventions

Room Names: In your prompt you'll get a default name for your room (e.g. "the laboratory") and an internal name (e.g. "R11"). Please use the internal name internally, and for the printed name of the room, assign the name in your prompt—or any variation you care to use. **Then, in parentheses, credit yourself as the author of your room!**

```
R11 is a room. Printed name of R11 is “Professor Lake’s Laboratory (Ryan
Veeder)”.
```

Object Names: For each object you include, assume someone else is including a similar object. Give your object an internal name that Inform 7 won't confuse with anyone else's, and give the player a way to distinguish it as well. A "beaker" in your room is going to cause a problem if there's another generic "beaker" in another room, so be specific:

```
The veeder-beaker is in R11. Printed name of the veeder-beaker is “stained
beaker”. Understand “beaker” and “stained” and “stained beaker” as the
veeder-beaker. Initial appearance of the veeder-beaker is “There’s a beaker
here.” Description of the veeder-beaker is “This beaker is kind of gross.”
```

The "author-hyphen-object" style seen above is what Zarf says he's gonna use. There are other ways to go about it—but make sure you're consistent. In every rule and every declaration, **use the unambiguous internal names for all your objects**. Something like

```
The chipped flask is in R11. Description of the flask is “Why can’t people be
careful?”
```

can lead to problems, because Inform 7 won't necessarily know what flask is being referenced in the phrase "Description of the flask..."

Variable Names: Sorry to state the obvious, but you should use variable names that won't conflict with other people's variable names:

```
Veederprofsize is a number that varies. Veederprofsize is 10.

Every turn when the location is R11 and Professor Lake is visible (this is
the Veeder Shrinking Professor rule):
      decrement Veederprofsize;
      if Veederprofsize is greater than 0:
            say "Professor Lake gets even smaller!";
      otherwise:
            say "Professor Lake shrinks into nothing and disappears!";
            remove Professor Lake from play.
```

The same goes for rule names, table names, adjective names, verb names, relation names, etc.

Portable Objects

Portable objects removed from your room can continue to operate as you intended, as long as they don't affect the contents of other rooms. By this I mainly mean that the descriptions of your portable objects will display no matter where they're examined, and that's fine.

If you feel like getting a little fancier, something that "interacts" with other rooms in an unintrusive way is okay:

```
The food detector is in R11.

Instead of examining the food detector:
      let x be a random visible edible thing;
      if x is a thing:
            say "The food detector points at [the x].";
      otherwise:
            say "The food detector shrugs unhelpfully."
```

But something that unlocks arbitrary doors or removes objects from play would be bad. Obviously.

Default Actions and Default Text

We won't be adding global text replacements for stuff like the description of the player character or commands like >JUMP, >WAVE, etc. You're free to modify these things as you wish, but make sure your changes only apply to your room:

```
Instead of jumping when the location is R11 (this is the Veeder jumping
rule):
      say “You jump up and down! Whee!”
```

Often I will phrase such rules in the form “while the player is in R11”—but apparently “when the location is R11” is a little bit faster (and we're gonna need all the tiny optimizations we can get). Also, “the player is in” doesn't trigger if the player is supported or contained by something. **So please favor “when the location is”** unless otherwise necessary.

Critically, **make sure you add your own description of the player character:**

```
Instead of examining the player when the location is R11 (this is the Veeder
PC description rule):
      say “Even better-looking than ever!”
```

New Actions

You can include custom actions in your room, if you give them unique names and make them impossible to perform in other rooms:

```
Veeder finger snapping is an action applying to nothing. Understand “snap”
and “snap fingers” as Veeder finger snapping when the location is R11.

Instead of Veeder finger snapping (this is the Veeder finger snapping
response rule):
      say “You snap your fingers! Ouch!”
```

Including your name in the name of the action makes it impossible to confuse with a similar action added by somebody else. Adding the “when the location is [your assigned room]” clause to the “Understand…” line means that in all other rooms, your action functionally doesn't exist—so no other authors have to implement it or worry about it breaking their puzzles.

This is a little bit dangerous: A project with too many “Understand *this* as *that* while *some condition*” statements can end up running really slow. **Please try to accomplish fancy activities with Inform 7's default actions if you can.** (If the organizers find sufficient cause, we might add certain especially useful or popular actions to the whole project, but we really don't want any one person's ideas to create a burden on everybody else's.)

NPCs

Include whatever NPCs you wish, and bend their interactivity to your whim. If you choose to use Inform 7's built-in conversational actions (answering it that, telling it about, asking it about, asking it for), you should have nothing to worry about. If you want to subvert their "default" usage, you can do that without introducing new actions:

```
Professor Lake is a man in R11.

Instead of asking Professor Lake about something (this is the Veeder Slap
Professor Lake rule):
      say "'You're the worst instructor I've ever seen,' you say, and then
you slap the professor across the face."

Instead of telling Professor Lake about something:
      try asking Professor Lake about the topic understood.
```

We will probably include an extension or pseudo-extension in the project to make more complex forms of conversation possible, but we haven't decided how yet.

Doors

If your room has any adjoining doors, their basic information will be included in your prompt. Don't change any of it! Obviously! That door is there for a reason!

Make sure that everything about your side of the door refers *only* to your side of the door. This includes the description: Instead of

```
Description of 10door11 is "You've always hated this door."
```

use

```
Instead of examining 10door11 when the location is R11, say "You've always
hated this door."
```

If you follow these guidelines, we should be able to drop in the code for both sides of the door and make it work without much fuss:

```
Description of R11 is "Here you are in the laboratory.[paragraph break]In the
north wall is that contemptible steel door."
```

```
10door11 is a closed locked scenery container in R11.

Rule for printing the name of 10door11 when the location is R11: say “steel
door”.

Instead of examining 10door11 when the location is R11, say “You’ve always
hated this door.”

Check opening 10door11 when the location is R11:
      if 10door11 is open:
            say “That’s already open.” instead;
      otherwise if 10door11 is locked:
            say “The door won’t budge. What a stupid door!” instead.
```

Kinds and Adjectives

Declaring a kind creates a change in I7’s value taxonomy that propagates, if not necessarily meaningfully, through the entire project. If you name them uniquely, you can introduce new kinds that won’t interfere with other people’s work:

```
A veederbug is a kind of animal. The iridescent beetle is a veederbug in R11.
The merciless ant is a veederbug in R11.

Instead of attacking a veederbug (this is the Veeder Bug Smashing rule):
      remove the noun from play;
      say “You smash [the noun] into nothing.”
```

Using uniquely-named adjectives is in many ways equivalent to using kinds...

```
A thing can be veederbug. The iridescent beetle is a veederbug animal in R11.
The hungry ant is a veederbug animal in R11.
```

But! Zarf has informed me that kinds are more efficient than adjectives. If you can, please try to use kinds when implementing your bug-smashing minigame.

Extensions

You cannot decide unilaterally to include an extension. Extensions apply to the whole project, and we’re not going to let you force any new rules on everybody’s rooms.

If you want to use an extension, you can do one of two things. First, you can port the source of the extension into the code for your room, *and then you can edit every single line of that extension so that* *its effects only apply in your room*. In this manner you can add "whatever you want," but if the extension you add is a memory hog, you might be asked to remove or simpliy it.

The other thing you can do is *suggest* adding it to the entire project, and then the organizers will determine whether or not it'll get added. The criteria we'll use should be pretty easy to guess: We don't want any global changes to interfere with anybody else's work—and that means *we don't want anyone to have to rewrite their code to work around an extension they didn't know they'd be put in contact with*. And we don't want any global changes to adversely affect performance. (The organizers will ask Zarf and Emily whether a proposed extension has the potential to adversely affect performance, and defer to their expertise.)

Zarf has already suggested including Modified Exit by Emily Short, and the organizers are inclined to acquiesce (especially since one of the organizers more or less recreates Modified Exit in all his own projects). Zarf has also assured the organizers that Ryan's favorite extension, Basic Screen Effects by Emily Short, won't cause any problems, so we'll probably be including that too.

We'll also include a small amount of universal code about books (see below). The sum of these global changes will be included in a project-specific extension that you can include when you start working and hopefully won't have to update very much as the project moves along.

Books

There are lots of books in Anchorhead, and there will be *an unconscionable number of books* in Cragne Manor. In Anchorhead, examining a book tells you what it looks like, but not what it says inside. Instead, you have to use the verb READ (which by default is synonymous with EXAMINE, but in Anchorhead is attached to a new action) to read books. We'll copy that behavior for Cragne Manor.

Some of the books in Cragne Manor will be library books, which must be collected to complete one of the PC's quests. We'll get to how you should implement these books in just a moment.

Part of our project-specific extension will include this (or something very similar):

```
A book is a kind of thing. Understand “book” as a book. A library book is a
kind of book. Understand “library book” as a library book.

A book has some text called passage.

Understand the command “read” as something new.

Reading is an action applying to one carried thing. Understand “read
[something]” as reading.
```

```
Instead of reading a book:
      say “[passage of the noun][paragraph break]”.
```

```
Instead of reading something:
      try examining the noun.
```

For that last rule, Anchorhead would instead say

```
Instead of reading something:
      say “There is nothing written on [the noun].”
```

but I like my way better. It lets the player >READ SIGN and get the same description as >LOOK AT SIGN without costing you guys any extra work.

If you’re assigned a Mercury Track puzzle, your room will need to include a library book. Implement that book like this:

```
The Dungeon Master’s Guide is a library book.

Printed name of the Dungeon Master’s Guide is “[italic type]Dungeon Master’s
Guide[roman type]”.

Description of the Dungeon Master’s Guide is “On the cover of the [italic
type]Guide[roman type] is a picture of a dragon. It also bears some sort of
insignia that makes it clear the book belongs to the public library.”

Passage of the Dungeon Master’s Guide is “You paw through the book and read
some descriptions of magical diseases.”
```

In addition to being marked internally as a library book, every library book should bear some sort of insignia that makes it clear the book belongs to the public library. It is the job of the person assigned to the public library room to decide what this insignia is, and I’m sure this person will get around to doing that soon.

If you’re not assigned a Mercury Track puzzle, you can still put a book in your room. Call it a “book,” give it a “passage” so players can read it, but don’t add the public library’s insignia! That would be confusing and cruel.

Fancy Stuff

Don’t shy away from doing fancy stuff. There are limits to how wild we can get with this project, but we’d rather try to do something amazing and fail than start out with a big list of stuff you can’t do.

Show us what you want to do and we'll help you make it work. Or just make it work and then show us!