Inform 7

- Inform is a language for authoring interactive fiction
- It targets the Z-machine (generates Z-machine bytecode)
  - The Z-machine was the virtual machine originally developed by Infocom for its text adventures
- Earlier versions of Inform looked like fairly standard procedural object-oriented programming
  - What makes Inform powerful is the large amount of content that already exists as standard libraries
- Inform 7 is a fairly radical departure
  - English-like syntax – uses the same parser for compiling the program as it does for handling player interaction
  - Declarative (rule-based) programming
  - Nice IDE with sophisticated project debugging and browsing support

Let's take a look at the IDE
Things and kinds

- Things (world objects)
- Kinds (types of world objects)
  - Creating a thing: A wicker cage is here.
  - Creating a specific kind of thing: A container called wicker cage is here.
  - Defining a new kind: An on/off button is a kind of device.

- Objects have properties. One of the properties is a text property, called description. This is the text presented when someone looks at an object.
  - There is a small wicker cage here. “There is a small wicker cage discarded nearby.”

- Rooms, containers and people are kinds that can contain other objects. You put objects inside an other object by typing: The <object name> is in <object name>. A shorthand for referring to the current room is to type: The <object name> is here.

Properties

- Properties are like fields on an object – they hold values
  - Either/or properties are binary – we can give whatever names we can to the two distinctions
    - The built-in kinds all define appropriate binary properties
      - Creating a closed cardboard box: The cardboard box is a container. The cardboard box is closed. (Note that we’re creating an instance of container called cardboard box, not a new kind)
  - We can create our own binary properties
    - An on/off button is a kind of device. An on/off button is either popped or depressed.

- Value properties hold values (e.g. text, numbers, objects)
  - All things have a text property called the description. The description of the wicker cage is “A small wicker cage lies discarded nearby”
  - Creating a new value property: The lantern has a number called a brightness level. (Adding a property to a thing (instance) rather than a kind)
  - We can define enum values: Brightness is a kind of value. The brightnesses are guttering, weak, radiant and blazing. (Note: we can not assign a brightness to a number property – it must be a brightness property)
  - Look at section 4.9 and 4.10 for some short cuts.

Text descriptions

- Text descriptions aren’t just static
  - We need dynamic descriptions in order to be able to write template text
  - Anything in square brackets within a piece of text is a description that will be evaluated to yield text
    - “You admire [the lantern].” becomes “You admire the candle lantern.” if lantern is the name of the candle lantern.
  - Lists
    - “Mr Darcy glares proudly at you. He is wearing [list of things worn by Darcy] and carrying [list of things carried by Darcy].”
    - This is also an example of how descriptions can get fancy.
Descriptions of objects

- Descriptions of objects consist of adjectives and nouns
  - The cargo trunk is an openable container. (adjective and noun combined in description)

- Two sources of adjectives
  - Property values (we’ve seen this)
  - Derived adjectives (computed by rules specifying how to combine simple values)

- Defining derived adjectives
  - Definition: A supporter is occupied if something is on it.
  - Definition: A room is occupied if a person is in it.
  - Note – we can’t use derived properties to create new things.

- Chapter 6 has more info on built-in derived adjectives and fancier descriptions

Actions

- Actions are performed by entities (primarily the player) within the world
  - Not to be confused with activities where are performed by the computer to simulate the world
- Inform is an event-based architecture – actions are the events triggered by actions within the simulated world
  - E.g. If the player types “take napkin” or “get the napkin”, the resulting action is taking the napkin

- Our first kind of rule: instead rules.
  - Instead rules can intercept the action (event) and perform special handling
  - Instead of eating the napkin: say “Why not wait for the actual dinner to arrive?”

- The types of action intercept rules
  - Instead
  - Before
  - After

An aside about rules

- Rules are divided up into rulebooks (the boxes in the diagram)
- Actions (events) are generated through typed commands or try phrases

- The before, instead and after rulebooks are common places to insert special handling without changing the fundamental semantics of supported actions
  - The built-in actions are those for which the blue rulebooks are provided by the standard library
  - If you invent your own actions, you have to write the blue rulebooks
Basic action handling

- Instead rules, by default, stop action processing
  - Instead of eating the napkin, say "Why not wait for the actual dinner to arrive?"
  - In this example, we won’t even get to the check rules

- Before rules, by default, continue action processing
  - Before taking the napkin, say "(first unfolding its delicate origami swan)."

The phrases "stop the action" and "continue the action" can be appended to any rule

- Before taking the napkin, say "Why not wait for the actual dinner to arrive?", stop the action
  - This acts the same as an instead rule now (though processed earlier in the tiers of rule processing)

In your rules you can initiate actions via the try statement

- Example: Try taking the napkin.
  - Try silently prints nothing if the action succeeds, normal output if it fails.

Scenes

- Scenes temporally segment the world, just like rooms (and regions) spatially segment the world

- Defining a scene: Train Stop is a scene.
  - Specifying a beginning: Train Stop begins when the player is in the station for the third turn (the italics can be any inform condition)
  - Specifying a terminating condition: Train Stop ends when the time since Train Stop began is 3 minutes.

- Initiating a scene:
  - When Train Stop begins:
    - move the Flying Scotsman to the Station;
    - say "The Flying Scotsman pulls up at the platform, to a billow of steam and hammering."

- Terminating a scene:
  - When Train Stop ends:
    - remove the Flying Scotsman from play;
    - if the player is in the Station, say "The Flying Scotsman inches away, with a squeal of released brakes, gathering speed invincibly until it disappears around the hill. All is abruptly still once more."

- Modifying action during a scene
  - Before going north during the Train Stop, say "The train blocks your way." instead.
  - Every turn during the Train Stop, say "Water is sluiced out of the tank and into the engine."

- Linking scenes
  - Brief Encounter is a scene. Brief Encounter begins when Train Stop ends.

- Scenes can have multiple named endings (allows one to differentiate termination actions in the scene termination rulebook)

Phrases

- Phrases are Inform’s statements (just like descriptions are one type of Inform expression)
  - There are a bunch of built-in phrases

- You can define your own phrases
  - Example: "To award (some – a number) points: ..."
  - Now you can use phrases like "award 2 points," "award 30 points," but not "award bogus points"

- Conditions will likely be useful in adding to the phrase book
  - Conditions are descriptions that are true or false

- Conditions can appear in text descriptions
  - "If (condition) then (otherwise) bar [end if]"
  - The Customs Wharf is a room. "Amid the bustle of the quayside, [if the cask is open] many eyes stray to your broached cask. [otherwise] nobody takes much notice of a man heaving a cask about. [end if] Sleek gondolas jostle at the plank pier."

- In defining phrases all the usual control constructs are available (see chapter 11).
Activities

- An activity is a real task for the computer program performing the simulation.
  - Vs. an action, which is a simulated task for a fictional entity within the world.

- An activity is like an internal method or subroutine – they provide functional abstraction for computational activities that might be performed during the handling of many actions.
  - An example is the built-in activity deciding the concealed possessions of a person.
  - It will be used by multiple action rules and activities when actions or activities are performed pertaining to the objects carried by a person.

- You can write your own specialized rules for handling the standard activities.
  - Example: Rule for deciding the concealed possessions of the Cloaked Villain: if the particular possession is the sable cloak, no; otherwise yes.

Understanding

- One can define objects, properties, actions (which modify properties) all you want, but the player can't act on them unless it's connected to the grammar.
  - Defining a new action: Photographing is an action applying to one visible thing and requiring light.
  - Attaching it to the grammar: Understand “photograph [someone]” as photographing.

- Even if you're not defining new actions, you may want to define new ways to invoke (from the parser) existing actions:
  - Understand “deposit [something] in [an open container]” as inserting it into
  - Understand “fill [an open container] with [something]” as inserting it into (with nouns reversed).

- Simplified example of creating an action from scratch:
  - Going by name is an action applying to one thing.
  - Carry out going by name: say “You walk to [the noun].”; move the player to the noun.
  - Understand “go to [any adjacent visited room]” as going by name.