Of Caves, Programming, and Poetry

Domain-specific languages in interactive fiction

Daniel Janus

November 17, 2008

The Story Begins...

Mammoth Cave, Kentucky, c. 1840

The first adventurer:
Stephen Bishop
(1820?—1857)

First true explorer, guide and mapper of the Mammoth Cave complex

Names like *River Styx*, *Snowball Room*, *Giant Dome*, etc., used to this day, come from him

The Story Begins...

Mammoth Cave, Kentucky, c. 1840

The first adventurer:
Stephen Bishop
(1820?—1857)

First true explorer, guide and mapper of the Mammoth Cave complex

Names like *River Styx*, *Snowball Room*, *Giant Dome*, etc., used to this day, come from him

Fast forward 130 years...

The Story Continues

- **1972:** An expedition led by Patricia Crowther rediscovers the Tight Spot connecting the Flint Ridge cave system and the Mammoth Cave (the passage was later found on Stephen Bishop's 1850s map)
- **1975:** William Crowther, Patricia's husband, member of ARPANET development team and member of 1972 expedition, writes a PDP-10 program

  "...that was a re-creation in fantasy of my caving, and also would be a game for the kids...My idea was that it would be a computer game that would not be intimidating to non-computer people, and that was one of the reasons why I made it so that the player directs the game with natural language input, instead of more standardized commands."

The Story Continues

- **1972:** An expedition led by Patricia Crowther rediscovers the Tight Spot connecting the Flint Ridge cave system and the Mammoth Cave (the passage was later found on Stephen Bishop's 1850s map)
- **1975:** William Crowther, Patricia's husband, member of ARPANET development team and member of 1972 expedition, writes a PDP-10 program

  "...that was a re-creation in fantasy of my caving, and also would be a game for the kids...My idea was that it would be a computer game that would not be intimidating to non-computer people, and that was one of the reasons why I made it so that the player directs the game with natural language input, instead of more standardized commands."

The Birth of Interactive Fiction

- **1976:** Don Woods picks up Crowther's program from ARPANET and augments it with magic and puzzles

  Colossal Cave: The first piece of interactive fiction

```
> on
Your lamp is now on.

You are in a debris room filled with stuff washed
in from the surface.  A low wide passage with
cobbles becomes plugged with mud and debris here,
but an awkward canyon leads upward and west.  A
note on the wall says "Magic word XYZZY".
```

The Birth of Interactive Fiction

- **1976:** Don Woods picks up Crowther's program from ARPANET and augments it with magic and puzzles

Colossal Cave: The first piece of interactive fiction

```
> on
Your lamp is now on.

You are in a debris room filled with stuff washed
in from the surface.  A low wide passage with
cobbles becomes plugged with mud and debris here,
but an awkward canyon leads upward and west.  A
note on the wall says "Magic word XYZZY".
```

Programming the Original

- The original Adventure was written in about 700 lines of Fortran code, plus another 700 lines of data
- Ran only on PDP-10; high memory requirements (ca. 300 KB), hard to port
- No real world model; more like a finite-state automaton
- Example rule: "When in room 1, print out the strings in Table 1 marked with value 2 in response to the player typing keyword 2 or a keyword from group 44. Then, move the player to room 2."

Programming the Original

- The original Adventure was written in about 700 lines of Fortran code, plus another 700 lines of data
- Ran only on PDP-10; high memory requirements (ca. 300 KB), hard to port
- No real world model; more like a finite-state automaton
- Example rule: "When in room 1, print out the strings in Table 1 marked with value 2 in response to the player typing keyword 2 or a keyword from group 44. Then, move the player to room 2."

Programming the Original

- The original Adventure was written in about 700 lines of Fortran code, plus another 700 lines of data
- Ran only on PDP-10; high memory requirements (ca. 300 KB), hard to port
- No real world model; more like a finite-state automaton
- Example rule: "When in room 1, print out the strings in Table 1 marked with value 2 in response to the player typing keyword 2 or a keyword from group 44. Then, move the player to room 2."

Programming the Original

- The original Adventure was written in about 700 lines of Fortran code, plus another 700 lines of data
- Ran only on PDP-10; high memory requirements (ca. 300 KB), hard to port
- No real world model; more like a finite-state automaton
- Example rule: "When in room 1, print out the strings in Table 1 marked with value 2 in response to the player typing keyword 2 or a keyword from group 44. Then, move the player to room 2."

Consequences of Commercial Era

- The first and most successful commercial text adventure ever: “Zork” by Infocom (1981)
- It had to face great diversity on the home microcomputer market, which called for as high portability as possible
- Developing multiple games yielded the necessity of having a reusable code base
- Infocom developed a dedicated virtual machine, the **Z-machine**, its emulators for over 20 kinds of systems and a dedicated language for writing IF — **ZIL** (*Zork Implementation Language*), compiled to Z-machine code

Consequences of Commercial Era

- The first and most successful commercial text adventure ever: “Zork” by Infocom (1981)
- It had to face great diversity on the home microcomputer market, which called for as high portability as possible
- Developing multiple games yielded the necessity of having a reusable code base
- Infocom developed a dedicated virtual machine, the **Z-machine**, its emulators for over 20 kinds of systems and a dedicated language for writing IF — **ZIL** (*Zork Implementation Language*), compiled to Z-machine code

Consequences of Commercial Era

- The first and most successful commercial text adventure ever: “Zork” by Infocom (1981)
- It had to face great diversity on the home microcomputer market, which called for as high portability as possible
- Developing multiple games yielded the necessity of having a reusable code base
- Infocom developed a dedicated virtual machine, the **Z-machine**, its emulators for over 20 kinds of systems and a dedicated language for writing IF — **ZIL** (*Zork Implementation Language*), compiled to Z-machine code

Consequences of Commercial Era

- The first and most successful commercial text adventure ever: “Zork” by Infocom (1981)
- It had to face great diversity on the home microcomputer market, which called for as high portability as possible
- Developing multiple games yielded the necessity of having a reusable code base
- Infocom developed a dedicated virtual machine, the **Z-machine**, its emulators for over 20 kinds of systems and a dedicated language for writing IF — **ZIL** (*Zork Implementation Language*), compiled to Z-machine code

Authoring Systems

- ZIL/Z-machine (Infocom, 1981-1989)
- AGT (Adventure Game Toolkit, 1985-7)
- TADS (Text Adventure Development System; Mike Roberts, 1987-)
- Inform/Z-machine (Graham Nelson, 1993-)
- Alan (Thomas Nilsson, 1993-)
- Hugo (Kent Tessman, 1994-)
- ADRIFT (Campbell Wild, 1997-)

Inform at a Glance (1): The Basics

- A dynamically weakly typed, procedural and object-oriented language with multiple inheritance
- C-like syntax
- Objects are usually singleton instances of anonymous classes and correspond to objects of modelled game world
- Objects may have properties, attributes (boolean properties), and are linked together to form a set of trees of objects

Inform at a Glance (1): The Basics

- A dynamically weakly typed, procedural and object-oriented language with multiple inheritance
- C-like syntax
- Objects are usually singleton instances of anonymous classes and correspond to objects of modelled game world
- Objects may have properties, attributes (boolean properties), and are linked together to form a set of trees of objects

Inform at a Glance (1): The Basics

- A dynamically weakly typed, procedural and object-oriented language with multiple inheritance
- C-like syntax
- Objects are usually singleton instances of anonymous classes and correspond to objects of modelled game world
- Objects may have properties, attributes (boolean properties), and are linked together to form a set of trees of objects

Inform at a Glance (1): The Basics

- A dynamically weakly typed, procedural and object-oriented language with multiple inheritance
- C-like syntax
- Objects are usually singleton instances of anonymous classes and correspond to objects of modelled game world
- Objects may have properties, attributes (boolean properties), and are linked together to form a set of trees of objects

Inform at a Glance (2): Sample Game

```
Constant Story "Hello World";
Constant Headline "^An Interactive Example^";
Include "Parser";
Include "VerbLib";

[ Initialise;
   location = Living_Room;
   "Hello World";
];

Object Room "Living Room"
  with description "A comfortably furnished living room.",
  has  light;
Object -> Apple "juicy apple"
  with name 'juicy' 'apple' 'fruit',
       description "Looks yummy."
  has  edible;

Include "Grammar";
```

Inform at a Glance (3): Parsing and Actions

- The task of a parser is to break down a player's command into a sequence of *actions*
- Actions represent ways of changing state of the game world
- Actions may trigger other actions as they are carried out
- Actions may have zero to two arguments

Sample actions

```
Look
Take sword
Insert gold_coin cloth_bag
```

Inform at a Glance (3): Parsing and Actions

- The task of a parser is to break down a player's command into a sequence of *actions*
- Actions represent ways of changing state of the game world
- Actions may trigger other actions as they are carried out
- Actions may have zero to two arguments

Sample actions

```
Look
Take sword
Insert gold_coin cloth_bag
```

Inform at a Glance (3): Parsing and Actions

- The task of a parser is to break down a player's command into a sequence of *actions*
- Actions represent ways of changing state of the game world
- Actions may trigger other actions as they are carried out
- Actions may have zero to two arguments

Sample actions

```
Look
Take sword
Insert gold_coin cloth_bag
```

Inform at a Glance (3): Parsing and Actions

- The task of a parser is to break down a player's command into a sequence of *actions*
- Actions represent ways of changing state of the game world
- Actions may trigger other actions as they are carried out
- Actions may have zero to two arguments

Sample actions

```
Look
Take sword
Insert gold_coin cloth_bag
```

Inform at a Glance (3): Parsing and Actions

- The task of a parser is to break down a player's command into a sequence of *actions*
- Actions represent ways of changing state of the game world
- Actions may trigger other actions as they are carried out
- Actions may have zero to two arguments

Sample actions

```
Look
Take sword
Insert gold_coin cloth_bag
```

Inform at a Glance (4): Actions (continued)

- Some actions don't take up game time (Save, etc.), others do
- Actions are carried out in phases: "before", "during", and "after", with respect to interference with the standard game rules
- Each action also has an associated actor (typically the player) who performs it

Inform at a Glance (4): Actions (continued)

- Some actions don't take up game time (Save, etc.), others do
- Actions are carried out in phases: "before", "during", and "after", with respect to interference with the standard game rules
- Each action also has an associated actor (typically the player) who performs it

Inform at a Glance (4): Actions (continued)

- Some actions don't take up game time (Save, etc.), others do
- Actions are carried out in phases: "before", "during", and "after", with respect to interference with the standard game rules
- Each action also has an associated actor (typically the player) who performs it

Inform World Model (1)

Well thought-out, consistent, flexible and reasonably concise — probably the greatest strength of Inform.

Outline

Inform World Model (1)

Well thought-out, consistent, flexible and reasonably concise — probably the greatest strength of Inform.

Outline

1. Substance
2. Containment
3. Space
4. Sense
5. Time
6. Action

Inform World Model (2)

1.3 Objects are indivisible even if the player may see internal structure to them, such as the four legs which are part of a chair. Pieces of objects only appear in the model if additional objects are provided for them.

1.4 Objects have internal states and are therefore distinguishable from each other by more than their position in the containment tree [...]

2.3.3 The contents of the player fall into two categories: those which are "worn", and the rest.

2.3.3.1 Worn objects represent clothing or accessories held onto the body without the need for hands, such as a belt or a rucksack.

Inform World Model (3)

3.2.2.3 A "door" is an item representing something which comes between two locations, which must be passed through or by in order to go from one to the other, and which it requires some conscious decision to use.

4.2 Awareness = sight + touch, that is, the player is aware of something if it can be seen or touched.

4.4.2 In the dark, the player can touch (a) anything contained in the player and (b) the enterable object which the player is contained in (if any).

5.1 The passage of time is represented by describing and changing the model world at regular intervals, each cycle being called a "turn" [...]

Inform World Model (4)

Some of the ways the model can be altered and what one can do with it:

- Implement a character in thelepatical contact with the player, although not visible
- Implement a gas mask that prevents speech but also death from gas inhalation
- Reverse the directions of the world

The model is implemented within the Inform Standard Library, divided into several code modules.

Inform World Model (4)

Some of the ways the model can be altered and what one can do with it:

- Implement a character in thelepatical contact with the player, although not visible
- Implement a gas mask that prevents speech but also death from gas inhalation
- Reverse the directions of the world

The model is implemented within the Inform Standard Library, divided into several code modules.

Inform World Model (4)

Some of the ways the model can be altered and what one can do with it:

- Implement a character in thelepatical contact with the player, although not visible
- Implement a gas mask that prevents speech but also death from gas inhalation
- Reverse the directions of the world

The model is implemented within the Inform Standard Library, divided into several code modules.

Inform World Model (4)

Some of the ways the model can be altered and what one can do with it:

- Implement a character in thelepatical contact with the player, although not visible
- Implement a gas mask that prevents speech but also death from gas inhalation
- Reverse the directions of the world

The model is implemented within the Inform Standard Library, divided into several code modules.

Imperative IF Programming Considered Harmful?

Inform is good: it can be used to great effect by programmers to produce nice and well-behaved games.

…But storytellers are not necessarily programmers. Even if they are, it's more natural for them to think declaratively about the world they're creating.

Some of other authoring systems were devised to be beginner-friendly, even at the cost of sacrificing flexibility.

> *Alan is not a programming language. Instead Alan takes a descriptive view.*
>
> — Thomas Nilsson

Imperative IF Programming Considered Harmful?

Inform is good: it can be used to great effect by programmers to produce nice and well-behaved games.

…But storytellers are not necessarily programmers. Even if they are, it's more natural for them to think declaratively about the world they're creating.

Some of other authoring systems were devised to be beginner-friendly, even at the cost of sacrificing flexibility.

> *Alan is not a programming language. Instead Alan takes a descriptive view.*
>
> — Thomas Nilsson

Imperative IF Programming Considered Harmful?

Inform is good: it can be used to great effect by programmers to produce nice and well-behaved games.

…But storytellers are not necessarily programmers. Even if they are, it's more natural for them to think declaratively about the world they're creating.

Some of other authoring systems were devised to be beginner-friendly, even at the cost of sacrificing flexibility.

> *Alan is not a programming language. Instead Alan takes a descriptive view.*
>
> — Thomas Nilsson

Imperative IF Programming Considered Harmful?

Inform is good: it can be used to great effect by programmers to produce nice and well-behaved games.

…But storytellers are not necessarily programmers. Even if they are, it's more natural for them to think declaratively about the world they're creating.

Some of other authoring systems were devised to be beginner-friendly, even at the cost of sacrificing flexibility.

> *Alan is not a programming language. Instead Alan takes a descriptive view.*
>
> — Thomas Nilsson

Inform 7, a.k.a. Natural Inform

Sample Game Revisited

```
"Hello World" by Daniel Janus

The story headline is "An Interactive
Example".

The Living Room is a room.  "A comfortably
furnished living room." The apple is an edible
thing in the Living Room.  Description of the
apple is "Looks yummy." Understand "juicy",
"apple" and "fruit" as the apple.
```

Yes, this is actual code.

Another example

Section 1 - Subjects

A subject is a kind of thing. The current subject is a thing that varies. Blank is a subject. The printed name of blank is "whatever comes to mind".

Suggestion relates subjects to each other. The verb to suggest (it suggests, they suggest, it is suggested) implies the suggestion relation.

Declarative Approach: Pros and Cons

Pros

- Arguably easier to learn and more easily understood by non-programmers
- Self-documenting, readable code
- Seems to be just as expressive as the "traditional" approach

Cons

- Natural language makes it harder to spot bugs
- What is wrong with the following snippet of code?

```
Understand "where [a thing]" as locating something. Locating something is an action applying to one thing.
```

Answer: it should be "[any thing]" and "...to one visible thing."

Declarative Approach: Pros and Cons

Pros

- Arguably easier to learn and more easily understood by non-programmers
- Self-documenting, readable code
- Seems to be just as expressive as the "traditional" approach

Cons

- Natural language makes it harder to spot bugs
- What is wrong with the following snippet of code?

```
Understand "where [a thing]" as locating something. Locating something is an action applying to one thing.
```

Answer: it should be "[any thing]" and "...to one visible thing."

Declarative Approach: Pros and Cons

Pros

- Arguably easier to learn and more easily understood by non-programmers
- Self-documenting, readable code
- Seems to be just as expressive as the "traditional" approach

Cons

- Natural language makes it harder to spot bugs
- What is wrong with the following snippet of code?

```
Understand "where [a thing]" as locating something. Locating something is an action applying to one thing.
```

Answer: it should be "[any thing]" and "...to one visible thing."

Declarative Approach: Pros and Cons

Pros

- Arguably easier to learn and more easily understood by non-programmers
- Self-documenting, readable code
- Seems to be just as expressive as the "traditional" approach

Cons

- Natural language makes it harder to spot bugs
- What is wrong with the following snippet of code?

```
Understand "where [a thing]" as locating something. Locating something is an action applying to one thing.
```

Answer: it should be "[any thing]" and "...to one visible thing."

Declarative Approach: Pros and Cons

Pros

- Arguably easier to learn and more easily understood by non-programmers
- Self-documenting, readable code
- Seems to be just as expressive as the "traditional" approach

Cons

- Natural language makes it harder to spot bugs
- What is wrong with the following snippet of code?

```
Understand "where [a thing]" as locating something.  Locating something is an action applying to one thing.
```

Answer: it should be "[any thing]" and "...to one visible thing."

Declarative Approach: Pros and Cons

Pros

- Arguably easier to learn and more easily understood by non-programmers
- Self-documenting, readable code
- Seems to be just as expressive as the "traditional" approach

Cons

- Natural language makes it harder to spot bugs
- What is wrong with the following snippet of code?

```
Understand "where [a thing]" as locating something.  Locating something is an action applying to one thing.
```

Answer: it should be "[any thing]" and "...to one visible thing."

Writing Code in Iambic Pentameter

Will's Study is a room. The desk is here.
A hastily handwritten note is on it.
Description is "It's from your friend Shakespeare:
'I've gone to lunch. You'll have to write the sonnet.'"
Composing is an action applying to nothing.
The quill is a thing that is in the study.
Understand "write sonnet" as composing.
Description of the quill is "Old and cruddy".
Instead of composing when the player
has no quill, say "You have not got the quill."
Instead of composing, say "And... done. 'Heya',
says Will, returning. You say, 'Hello, Will!'
Says Shakespeare, 'Thank you for the time you've taken!
You really are a pal, Sir Francis Bacon.'"

— Robin Johnson

Polishing Foreign Versions

Dębowe biurko stoi w gabinecie Willa.
Na nim kartka z notatką skreśloną starannie.
Opis: "Od przyjaciela twojego, Szekspira:
'Poszedłem na kolację, napisz sonet za mnie.'"
Pisanie jest to akcja bezargumentowa.
Gęsie pióro to przedmiot, co jest w gabinecie.
Opis pióra: "Możesz nim ubrać myśli w słowa."
Powiedz: "Nie możesz pisać, pióra nie masz przecie!"
zamiast pisania, gdy gracz nie posiada pióra.
Zamiast pisania, powiedz: "I oto — skończone.
Will powraca z kolacji z uśmiechem ponurym,
lecz wnet się rozpromienia, widząc gotów sonet.
Uśmiechasz się, Will także, chyląc głowę w skłonie:
'Dobry z Waści przyjaciel, Franciszku Baconie.'"

— Translated by Daniel Janus

The *really* challenging part: port Inform 7 compiler to Polish so that this keeps compiling!

> **xyzzy**
You mutter the word "XYZZY." Suddenly a hollow voice says...

Thank you for your attention!

*** You have won ***

Would you like to RESTART, RESTORE a saved game, ask some QUESTIONS, or QUIT?