Exploring Custom Languages for Software Development

Abe Pralle

August 2, 2017

About Me

Abe Pralle

- Indie game developer (Runegate, Plasmaworks)
- Rogue language designer
- github.com/AbePralle

Programming Interests

- Games, languages, API's

Contact

- abe.pralle@gmail.com

Overview

Topics

- Manifesto
- Custom Language Concepts
- HTML & JSON
- PostScript
- ASCIImage
- GLSL
- FGB
- Custom Language Benefits
- Overcoming the Mystique
- Z-Machine
- SCUMM
- Conclusion

Manifesto

I Believe That...

- Custom languages make software systems easier to build, debug, maintain, and port
- Custom languages are much simpler to create and maintain than people believe
- Developers should consider writing software frameworks using general-purpose languages and using custom languages to manage data and provide logic

Custom Language Concepts

- General-Purpose Language
  More direct & versatile but more overhead required per action
- Domain-Specific Language (DSL)
  More abstract with simple commands implying complex behavior, geared to a specific purpose
- Custom Language (CL)
  A language developed for a particular software system or data format, often in-house. Often a mix of GP and DSL, usually hosted by a framework written in a GP language.
- Host Language /Layer/Framework
  The software layer responsible for launching and updating Custom Language runtimes
- Scripting Language
  Used synonymously with CL and HLL. A proper scripting language controls the high-level state and behavior of framework objects without being responsible for low-level maintenance.

HTML & JSON

HTML

- Ubiquitous Domain-Specific Language
- 1990
- 'Nuff said

JSON

- Ubiquitous data format - and DSL
- 1999
- 'Nuff said

index.html

```
<html>
  <body>
    <b>Hello World!</b>
  </body>
</html>
```

Name.json

```
{
  "name":
  {
    "first":"Abe",
    "last":"Pralle"
  }
}
```

PostScript

PostScript

- Domain-Specific Language
- Developed in early 80's
- Still in heavy use today

HelloWorld.ps

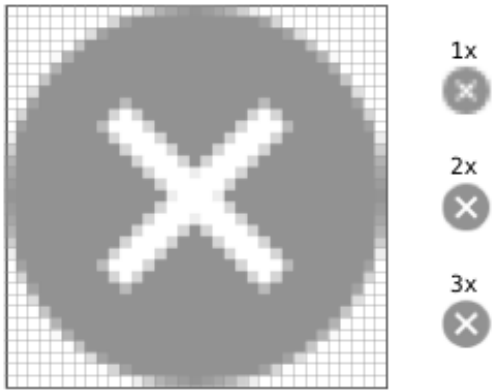
```
%!PS
 /Courier                 % name of the desired font
 20 selectfont            % choose the size in points and establish
                          % the font as the current one
 72 500 moveto            % position the current point at
                          % coordinates 72, 500 (the origin is at
                          % the lower-left corner of the page)
 (Hello world!) show      % stroke the text in parentheses
 showpage                 % print all on the page
```

ASCIImage

ASCIImage

- ASCIImage is an image generating DSL implemented in ObjectiveC
- Source code is "ASCII art"
- Output is PNG image
- Sample:

```
@"· · · 1 1 1 1 1 · · ·",
@"· · 1 · · · · · 1 · ·",
@"· 1 · · · · · · · 1 ·",
@"1 · · 2 · · · 3 · · 1",
@"1 · · · # · # · · · 1",
@"1 · · · · # · · · · 1",
@"1 · · · # · # · · · 1",
@"1 · · 3 · · · 2 · · 1",
@"· 1 · · · · · · · 1 ·",
@"· · 1 · · · · · 1 · ·",
@"· · · 1 1 1 1 1 · · ·",
```



- Website: http://cocoamine.net/blog/2015/03/20/replacing-photoshop-with-nsstring/

GLSL

GLSL

- OpenGL Shading Language
- Domain-Specific Language
- Used by all modern OpenGL apps
- Used to write programs that handle vertex and transformations and coloring
- Runs on video card GPU

Vertex Shader

```
attribute vec4 position;
attribute      vec2 uv;
varying        vec2 vertex_uv;

void main()
{
  gl_Position = position;
  vertex_uv = uv;
}
```

Pixel Shader (Fragment Shader)

```
uniform sampler2D texture_0;
varying vec2      vertex_uv;

void main()
{
  gl_FragColor = texture2D(texture_0,vertex_uv);
}
```

FGB

FGB

- FGB is an unreleased Game Boy Color game I programmed circa 2000 (can find on the internet)

- Video example:

  https://abepralle.github.io/Presentations/FGB-Flight.mov

FGB

FGB

- FGB's music was implemented using a Custom DSL
- Commands included instrument definition, JMP to label, call subroutine, and subroutine definition consisting of musical notes and holds/skips
- Source code converted to byte code at compile-time, only byte code present in game ROM
- Small VM written in assembly to play music

FGB

lady_flower.gbm (Source)

```
//lady flower
//Jacob Stevens
notesPerSecond 4    //quarter = 120, one
   note equals one eighth

track 1   //countermelody
instrument1 0044A23286

counterA

.repeat
counterB,counterC,counterB,counterC
counterD,counterE,counterD,counterF
counterB,counterC,counterB,counterC
counterD,counterE,counterD,counterF
counterH,counterJ,counterK,counterB
counterH,counterJ,counterK,counterL
counterH,counterJ,counterB,counterC
jmp .repeat
```

lady_flower.gbm (Cont'd)

```
track 2   //melody
instrument2 80F70887

melodyA

.repeat
melodyB,melodyB,melodyC,melodyD
melodyC,melodyE,melodyB,melodyB
melodyC,melodyD,melodyC,melodyF
melodyG,melodyG,melodyH,melodyJ
melodyG,melodyG,melodyH,melodyK
melodyL,melodyM,melodyM,melodyN
jmp .repeat
```

Case Study: FGB

lady_flower.gbm (Cont'd)

```
counterA:
-2

counterB:
C4 E4 G4 B4 E5 G5

counterC:
B5 G5 E5 B4 G4 E4

counterD:
D4 F4 A4 D5 F5 A5

counterE:
D6 B5 G5 F5 D5 B4

counterF:
G3 B3 D4 F4 G4 B4

...
```

lady_flower.gbm (Cont'd)

```
...

melodyH:
F6 D6# C6# D6# F6 D6#

melodyJ:
E6,h5

melodyK:
G6 D6# F6 G6 D6# G6

melodyL:
A6,h5

melodyM:
-6

melodyN:
-4 E6,h
```

Custom Language Benefits

Versatile

- Express computation and control logic
- Dynamically generate data (images, music)

Compact

- DSL programs generally take much less space than their GP counterparts
- Often much easier (and more compact) to describe data programmatically rather rather than pre-generating data. Ex: tremendous savings describing a circle rather than including a circle bitmap

Clarity & Collaboration

- Commands are simple, focused, & relevant to app
- Non-programmers can easily understand & modify "scripts"
- Don't need full compiler setup in order to modify scripts

Easier to Debug and Port

- Using a CL automatically decouples framework from logic, forming separate abstraction layers; a bug in either layer is fail-fast
- Apps expressed in CL can easily be ported by writing new framework or VM

Overcoming the Mystique

Common Perceptions

- Languages are often placed on a pedestal
- "Languages are hard to program and difficult to debug and maintain"
- "Better to use a well-established language that has been proven"

Setting the Record Straight

- People don't have similar concerns about custom data structures or file formats
- CL's are a way to implement a more sophisticated and flexible data-driven app
- A custom language is just a code module like any other
- Compilers can be buggy, but language runtimes tend to be fail-fast - if it's not crashing then it's probably bug-free
- An in-house CL can be easily customized and optimized
- Best use cases aren't usually on critical path

Z-Machine

Z-Machine

- Byte code-based virtual machine used to create all Infocom text adventures
- 1979
- Ported to virtually all platforms, including Game Boy and JS
- Porting VM makes all games available on new platform
- General Purpose + Domain Specific VM specification
- Built-in data movement, arithmetic, control flow, game object manipulation, text parsing
- Z-Machine Specification: http://inform-fiction.org/zmachine/standards/z1point0

ZIL

ZIL

- Zork Implementation Language, 1979
- Variant of MDL, itself a variant of Lisp
- Compiles to Z-Machine format
- General-Purpose + Domain-Specific Language hybrid
- Language Guide: http://xlisp.org/zil.pdf

Room Definition

```
<ROOM LIVING-ROOM
  (LOC ROOMS)
  (DESC "Living Room")
  (EAST TO KITCHEN)
  (WEST TO STRANGE-PASSAGE IF
   CYCLOPS- FLED ELSE
    "The wooden door is nailed shut.")
  (DOWN PER TRAP-DOOR-EXIT)
  (ACTION LIVING ROOM-F)
  (FLAGS RLANDBIT ONBIT SACREDBIT)
  (GLOBAL STAIRS)
  (THINGS <> NAILS NAILS-PSEUDO)>
```

Routines

```
<ROUTINE TURN-OFF-HOUSE-LIGHTS ()
  <FCLEAR ,LIVING-ROOM ,ONBIT>
  <FCLEAR ,DINING-ROOM ,ONBIT>
  <FCLEAR ,KITCHEN ,ONBIT>>

<ROUTINE INCREMENT-SCORE (NUM)
  <SETG SCORE <+ ,SCORE .NUM>>
  <COND (,SCORE-NOTIFICATION-ON
    <TELL "[Your score has just gone up by "
      N .NUM ".]" CR>)>>
```

Inform 7

Inform 7

- 2006
- Declarative programming language
- Compiles to Z-Machine format
- Domain-Specific Language
- http://inform7.com

Room Definition

```
The Living Room is a room. "A comfortably
furnished living room."
The Kitchen is north of the Living Room.
The Front Door is south of the Living
Room.
The Front Door is a door. The Front Door
is closed and locked.

The insurance salesman is a man in the
Living Room. The description is "An
insurance salesman in a tacky polyester
suit. He seems eager to speak to you."
Understand "man" as the insurance
salesman.
```

Interaction Defintion

```
Instead of listening to the
insurance salesman:
   say "The salesman bores you
with a discussion of life
insurance policies.  From his
briefcase he pulls some
paperwork which he hands to
you.";
   move the insurance paperwork
to the player.
```

SCUMM

Inform 7

- Script Creation Utility for Maniac Mansion, 1987
- Point & Click Graphic Adventure DSL
- Inspired by Lisp but became its own thing
- Concurrent script execution fundamental to system
- Used for all LucasFilm/LucasArts graphic adventures
- Runs on ScummVM, freeware adaptation of original VM

Example Script 1

```
script clock-tick
  do {
    clock-state = not clock-state
    object living-room-clock state clock-state
    play-sound clock-tick
    break-here 60
  }
}
```

SCUMM

Example Script 2

```
cut-scene {
  ...
  actor nurse-edna in-room edna-bedroom at 60,20
  camera-follow nurse-edna
  actor nurse-edna walk-to 30,20
  wait-for-actor nurse-edna
  say-line nurse-edna "WHAT'S YOUR POINT ED!!!"
  wait-for-talking nurse-edna
}
```

Conclusion

- Separating app logic into a framework + one or more custom languages has numerous benefits
  - Separate abstraction layers are each easier to debug and reason about
  - App logic is more compact, with more abstract commands that are directly relevant to app purpose
  - Porting host framework allows all apps written in custom language to run on new platforms
  - Easier to have non-programmers contributing to project
- It's really not that hard to write or maintain a custom language
- Consider adding a custom language to your next project!

The End