Phil Mercurio
Thyrd.org

Poet: Prototype Object Extension for Tcl

poet.sourceforge.net

Tcl'2007
New Orleans

Poet

- Poet: Prototype Object Extension for Tcl
  - Dynamic, prototype-based inheritance
  - One-way constraints
  - Persistence
  - Assimilation of Tk widgets via introspection
  - Current status: stable
- Poetics: Poet Integrated Construction Set
  - End-user modification of a running Poet program
  - Type annotations
  - Object and code editors
  - Goal: provide some of the functionality of an IDE to the user
  - Current status: experimental, handy for Poet developer

Inspiration and History

- Self (Ungar and Smith)
  - Live, directly-manipulated objects
  - Prototypes
- Garnet and Amulet (Myers *et al.*)
  - One-way constraints
  - Desktop application platform
- History:
  - 1994: theObjects (Juergen Wagner)
  - 1996: ported to Tcl7.5/Tk4.1
  - 1997: redesign Poet 1
  - 1999: Poet 2 begun
  - 2007: Poet 2.0.0 released

C vs. Tcl

- Poet started out as a C extension
- Primordial Poet object **`Object`** implemented in C
  - Low overhead in choosing C vs. Tcl for a method
  - 1/3 of **`Object`**'s methods are C
  - Constraint network is C

C code	~5000 lines
Tcl code, non-GUI	~4300 lines
Tcl code, GUI handwritten	~13000 lines
Tcl code, GUI autogenerated	~25000 lines

Object Creation and Destruction

- Objects are constructed by their parent

```
Object construct NewObject
```

- Objects destroy themselves

```
$self destruct
```

  - No garbage collection, override `destruct` to clean up
  - An object may have *goodbye* scripts which are automatically invoked upon destruction

```
modelObj addGoodbye {uiObj unrender modelObj}
```

- Tcl's autoloading used to load object source when first referenced
  - First line in source refers to parent, so parent autoloaded
  - Multiple inheritance via method `mixin`, which also autoloads

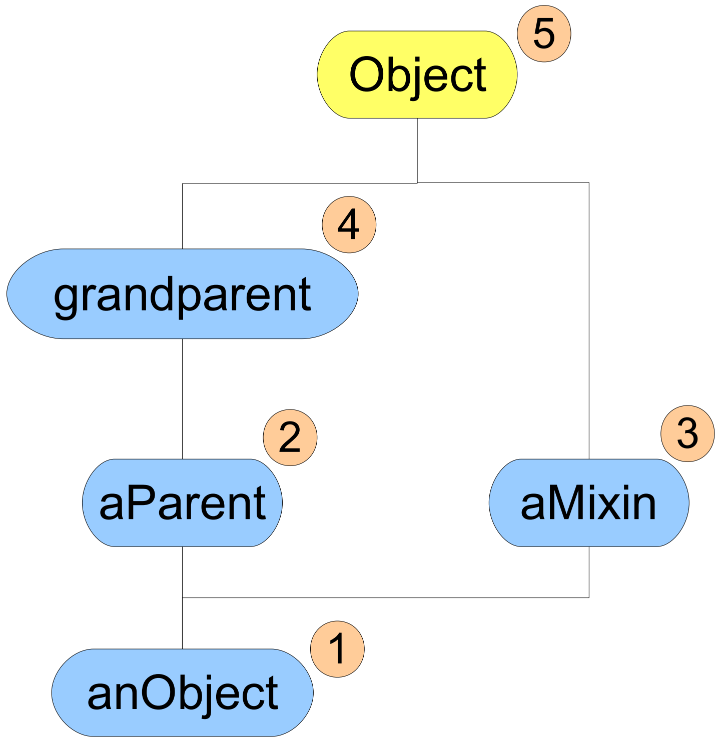
Anonymous Names

- If the argument to `construct` ends in `*`, an anonymous name is returned
  - Guaranteed to be unique in this interpreter
- If it ends in `@`, a persistent version is returned
  - Guaranteed to be unique in this persistent storage
  - Object not persistent yet, need to mixin `Thing`

```
% Object construct *
*a
% Object construct @
@a
% @a mixin Thing
```

Prototype Inheritance

- No classes, any object may be a parent for any other object
- Objects have multiple *dimensions* (methods, slots, formulas, etc.) subject to inheritance
  - Search order: object, parent, mixins, ancestors



Methods

- A method is a Tcl procedure with target object as `$self`
- No method chaining, but any method can be called on any object using `as`
- Complete definition of an object that announces its demise:

```
Object construct VerboseObject

VerboseObject method destruct {} {
   puts stderr “$self destructing”
   $self as [VerboseObject parent] destruct
}
```

Slots

- Object attributes are set or retrieved via the method `slot`
  - Slot names beginning with _ are *private* and are not inherited
  - Only *public* slots are persistent
- Slots may be designated *active* for reading and/or writing
  - A corresponding method is invoked when slot read or written
    - Write method may reject proposed value
  - The method may be on a different object than the slot value

```
% alpha slot test1 42
42
% alpha slot test1
42
% alpha method test1> {x} {puts "Set test1 to $x"}
% alpha slotOn test1 >
Set test1 to 42
% alpha slot test1 24
Set test1 to 24
24
```

Persistence

- Objects are made persistent by mixing in `Thing`
- `ThingPool` is used to specify the storage
  - Either a directory or a single file using tcllib's VFS
  - Each `Thing` written as a Tcl script
  - `Thing`s are autoloaded when referenced
- Setup for persistence:

```
ThingPool setFile [lindex $::argv 0]
ThingPool slot writable 1
ThingPool open

rename exit crash
proc exit {{returnCode 0}} {
   ThingPool close
   crash $returnCode
}
```

Constraints

- A slot's value may be constrained via `slotConstrain`
- A *formula* matching the slot is sought via inheritance
  - A formula is arbitrary Tcl code
  - Like a method, `$self` is available and a value is returned
  - References to other slots are recorded as dependencies

```
btn formula state {
   expr {[scl slot value] == 0 ? "disabled" : "normal"}
}

btn slotConstrain state
```

Controlling Constraints

- Automatic dependencies can lead to irrelevancies

  `Poet limitConstraints <object>`

  - Only descendants of `<object>` participate in network

  `Poet sideEffect <script>`

  - Ignores slot accesses inside `<script>`
- Formulas that take too long negatively impact liveness
- A formula may indicate it's not done yet with a special error

  `error "suspend <token>"`

  - `<token>` is any unique string, e.g. an object name
  - No value set on dependent slot

  `Object resumeFormula <token>`

  - Continuation of work on dependent slot

  `Object completeFormula <token> <value>`

  - Computation of `<value>` complete, set slot

Type Annotations

- Poet slots are Tcl variables and can hold any value
- A slot may have a *type annotation* indicating the sorts of values it may contain

```
alpha slot test1 42
alpha type test1 <integer>
```

- Types are subject to inheritance, a slot's value and type may reside on different objects
- Not a traditional type system
  - Slot values are not made to conform to their types
  - No type inferencing to validate expressions
- Poetics uses type annotations when introspecting Poet objects

Assimilation

- Megawidgets supported by *assimilating* Tk widgets into Poet objects
- Assimilation performed by preprocessor using Tk introspection
  - Only needs to be rerun if widget API changes

```
Tk_Button slot background #d9d9d9
Tk_Button method background> {value} {
    set p [$self primary]
    if {$p ne ""} {
        $p configure -background $value
    }
}

Tk_Button type background <color>
Tk_Button slotOn background >
```

ProtoWidget

- Poet assimilates Tk, BWidget, TkTable, and BLT
- Widget slots may participate in constraint network
- All widgets descendant from `ProtoWidget`
- `ProtoWidget construct` takes additional arguments of the form `-slotname value`
  - Result is cosmetically similar to Tk
- Additional slot `layout` contains geometry manager and options
  - If layout begins with `-`, assumed to be `pack` options
  - Otherwise, first word must be `grid`, `place`, etc.
- Assimilated widgets may be augmented with additional handwritten methods
- A few custom widgets included

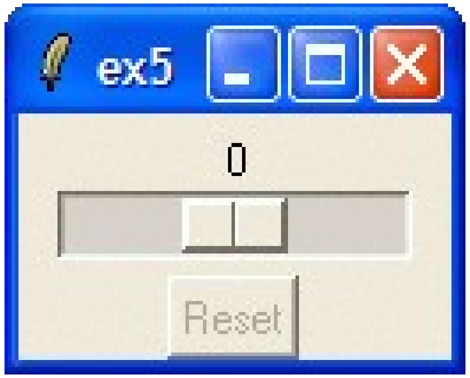
Example

```
package require Poet

Tk_Scale construct scl . \
    -from -7 -to 7 \
    -orient horizontal \
    -layout {-side top}

Tk_Button construct btn . \
    -text "Reset" \
    -layout {-side top} \
    -command "scl slot value 0"

btn formula state {
    expr {[scl slot value] == 0 ?
        "disabled" : "normal"}
}
btn slotConstrain state
```

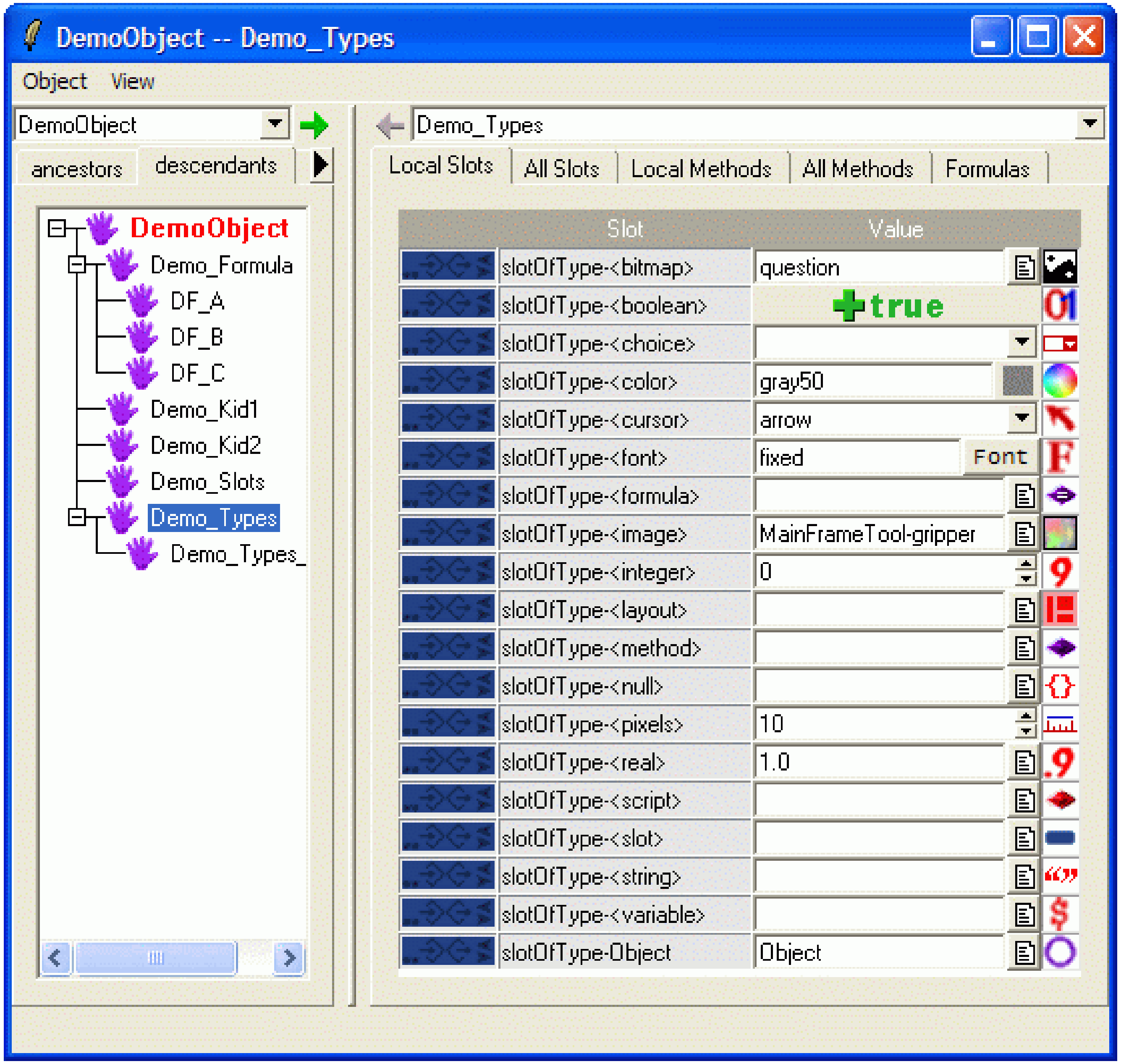


Poetics Types

- Poetics consists of tools to directly manipulate Poet objects
  - Not enabled by default, meant for use by “gardeners”
- We begin by defining types for editing Tk widgets, used to present correct editing tool in object editor

```
Object
<color>
<cursor>
<font>
<boolean>
<real>
<integer>
<real> -1.0 1.0 0.1
<integer> 0
<choice> alpha beta gamma
```

Object Editor



DemoObject -- Demo_Types

Object View

DemoObject

Demo_Types

ancestors descendants

- DemoObject
  - Demo_Formula
    - DF_A
    - DF_B
    - DF_C
  - Demo_Kid1
  - Demo_Kid2
  - Demo_Slots
  - Demo_Types
    - Demo_Types_

Local Slots | All Slots | Local Methods | All Methods | Formulas

Slot	Value
slotOfType-<bitmap>	question
slotOfType-<boolean>	true
slotOfType-<choice>	
slotOfType-<color>	gray50
slotOfType-<cursor>	arrow
slotOfType-<font>	fixed Font
slotOfType-<formula>	
slotOfType-<image>	MainFrameTool-gripper
slotOfType-<integer>	0
slotOfType-<layout>	
slotOfType-<method>	
slotOfType-<null>	
slotOfType-<pixels>	10
slotOfType-<real>	1.0
slotOfType-<script>	
slotOfType-<slot>	
slotOfType-<string>	
slotOfType-<variable>	
slotOfType-Object	Object

Problems and Future Work

- Poet is very tolerant of errors, perhaps too tolerant
  - Accessing undefined slots returns `{}`, not an error
  - All slots and methods (even private ones) accessible from any object
  - Most errors trapped by dialog that allows user to ignore error
- Browsing and editing of existing objects supported, not creation of new objects
  - Only autoloaded code editable in code browser
- Slot editors for more types need to be implemented
  - Layout editor particularly tricky
- Code editor could be enhanced with programming-by-demonstration features

Demo

- At Tcl'2007, these slides were shown via a slideshow program written in Poet. The program displayed a widget on this slide that could be inspected via Poetics. That demo program is available in the `sample/` folder of the Poet release at

`poet.sourceforge.net`