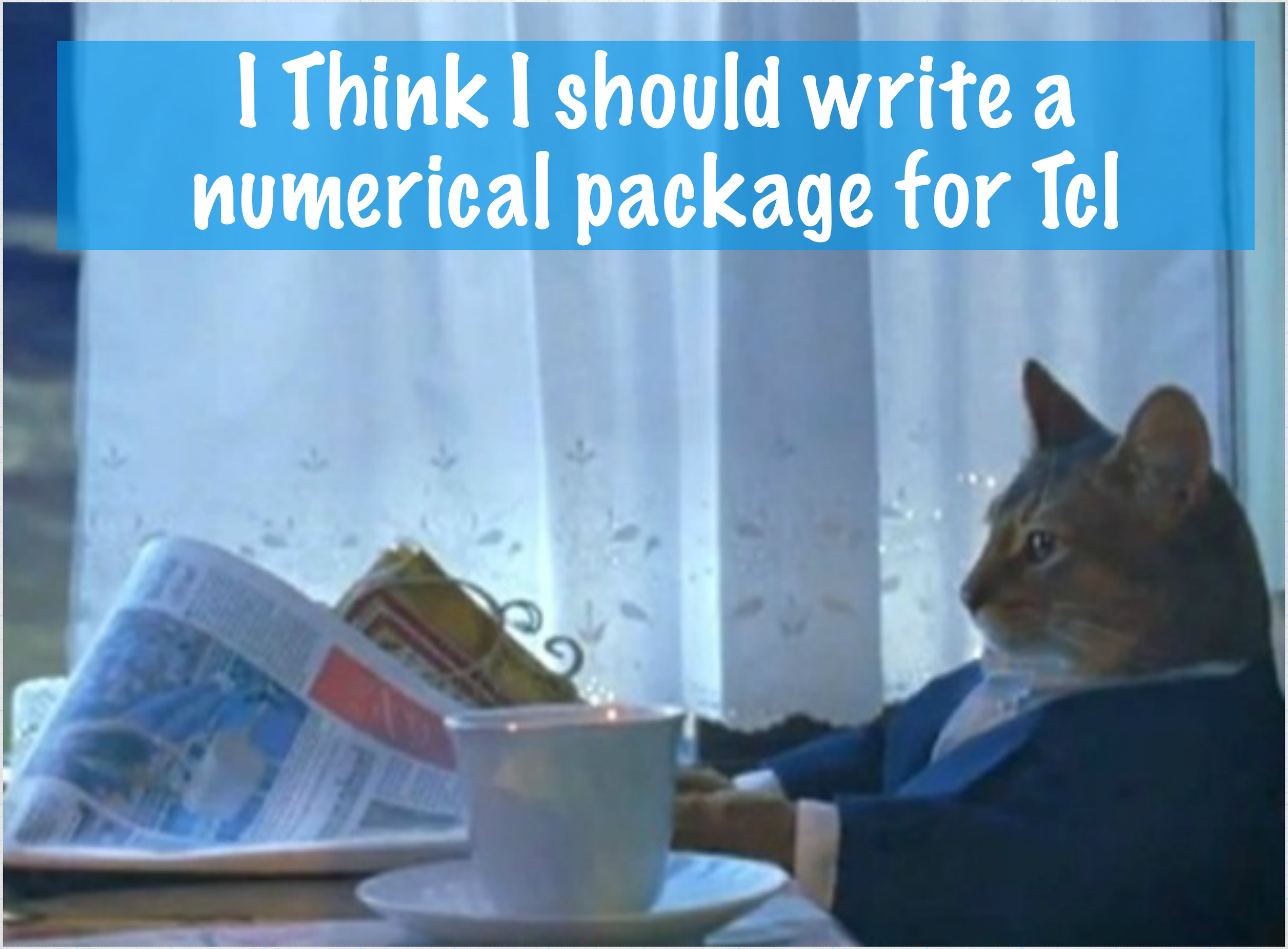
Odielib

Making Boring Math Exciting in C

I Think I should write a numerical package for Tcl



Odielib

* Collection of Tools, Math Functions, and other Assorted Routines
* Modular Build system, implemented in Tcl
* Uses the Practcl Build System from Last Year's Paper
* Packaged as a Loadable Extension

Trying Odielib

```
package require odielibc
set a {0 1}
set b {1 2 3}
set c [::vector::add $a $b]
{1 3 3}
::vectorxyz::midpoint {} $c
{0.5 1.5 1.5}
```

Trying Odielib

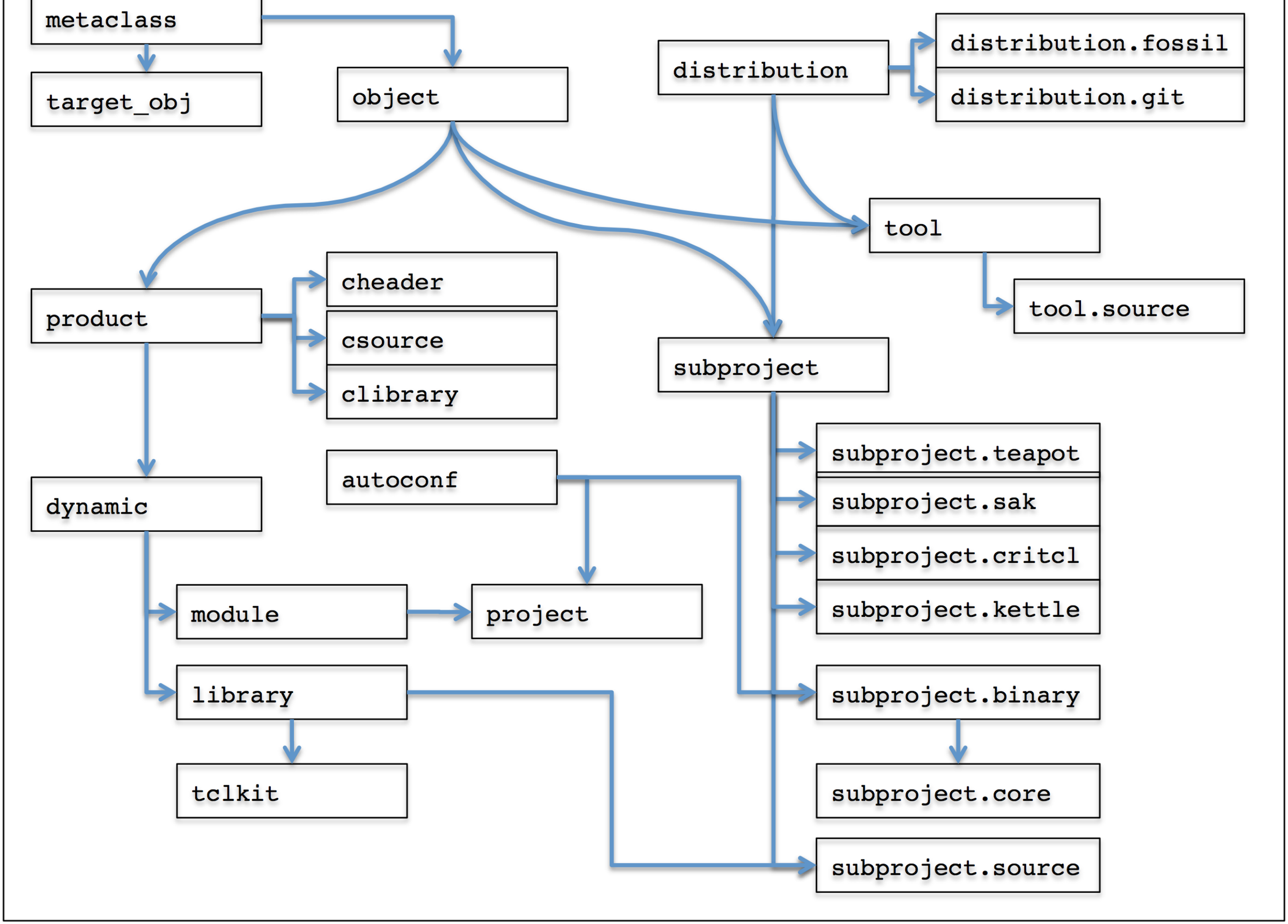
```
package require odielibc
set a {0 1}
set b {1 2 3}
set c [::vector::add $a $b]
{1 3 3}
::vectorxyz::midpoint {} $c
{0.5 1.5 1.5}
```

::vector::add sizes the output to match the size of the largest input

Arguments to ::vectorxyz are massaged to be a 3x1 matrix internally. Missing columns are assumed to be zero

The Practcl Build System

* Utilizes A Tcl based build automation
* Has it's own notation, but...
* Next version will be able to read Critcl
* Extends TEA beyond building standalone shared libraries



Practcl Class Hierarchy

We're All Mad Here

```
###
# A Tcl command in Practcl markup
###
my c_tclcmd  ::thestate::two_plus_two {
/*
The state says it's 5, Winston
*/
Tcl_SetObjResult(interp,Tcl_NewIntObj(5));
return TCL_OK;
}
###
# And back to Tcl
###
```

```
# A C function in Practcl markup
my c_function {
static inline double Vector_GridScaler(
  double x,
  double grid,
  double grain
)
} {
  double q;
  q=grid*round(x/grid);
  if((x-q)>grain) {
    q+=grain;
  }
  return q;
}
```

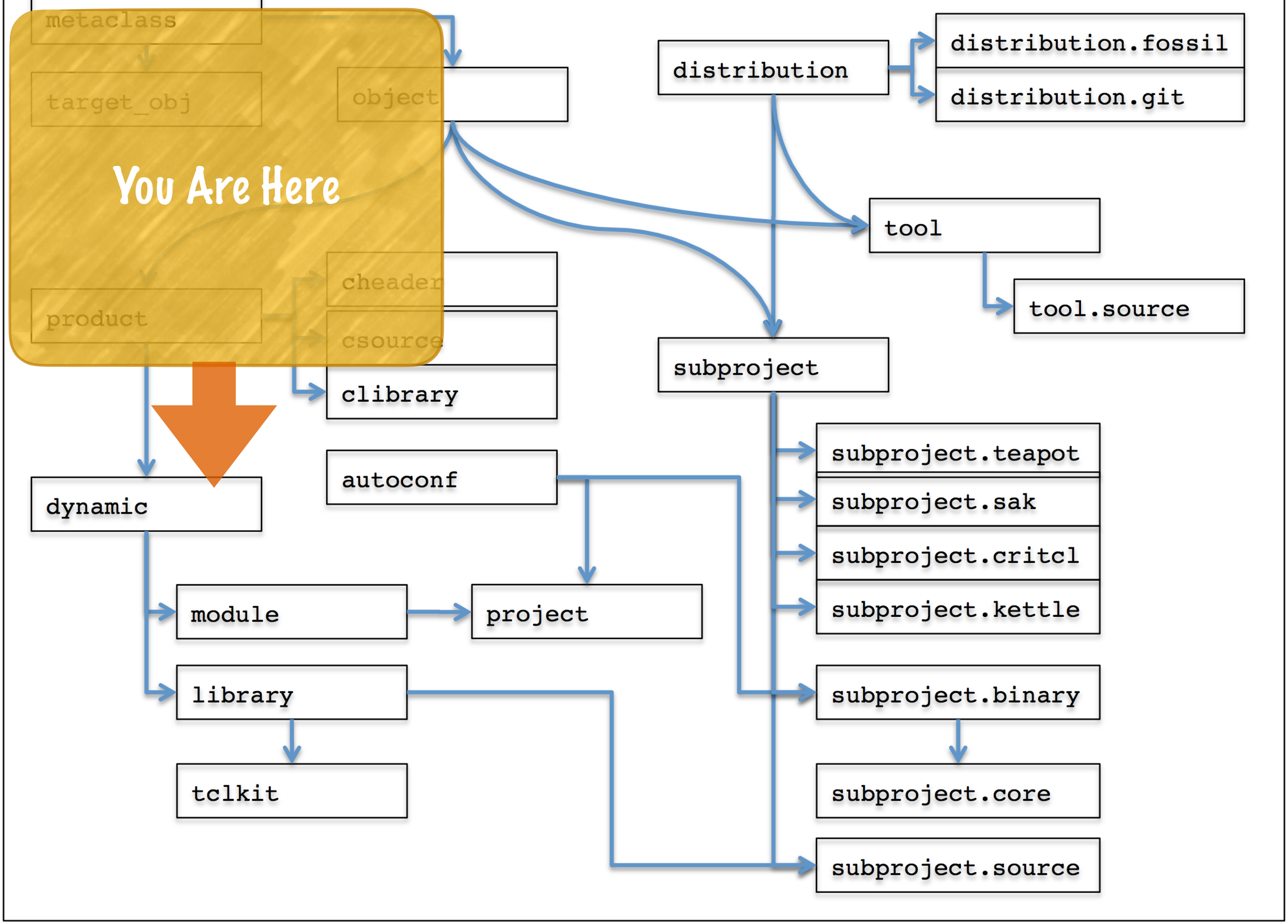
```
# A C function in Practcl markup
my c_function {
static inline double Vector_GridScaler(
   double x,
   double grid,
   double grain
)
```

Raw C Function Prototype

```
} {
   double q;
   q=grid*round(x/grid);
   if((x-q)>grain) {
      q+=grain;
   }
   return q;
}
```

Raw C Function Body

I Should Implement Critcl, But With None of the Nice Bits



Practcl Class Hierarchy

Numerical Values

* Odielib implements a Obj_Type to represent matrices and vectors

* This Obj_Type is optimized to perform transformations on 3x1 vectors via 4x4 affine matrices

* Within the C-API, All Functions Accept a pointer to an array of Doubles

Numerical Values

* Supports Pass by Reference and Pass by Value

```
##
# Pass by Value
##
set A {1 2 3}
set B {4 5 6}
set C [::vectorxyz::add $A $B]
```

```
##
# Pass by Value
##
my c_tclcmd  ::vectorxyz::add {
 VectorXYZ A,B,C;
 if(objc < 3) {
   Tcl_WrongNumArgs( interp, 1, objv, "A B" );
   return TCL_ERROR;
 }
 if(
Odie_GetVectorXYZFromTclObj(interp,objv[1],A)
) return TCL_ERROR;
 if(
Odie_GetVectorXYZFromTclObj(interp,objv[2],B)
) return TCL_ERROR;
 VectorXYZ_Add(C,A,B);
 Tcl_SetObjResult(interp,VectorXYZ_To_TclObj(C));
 return TCL_OK;
}
```

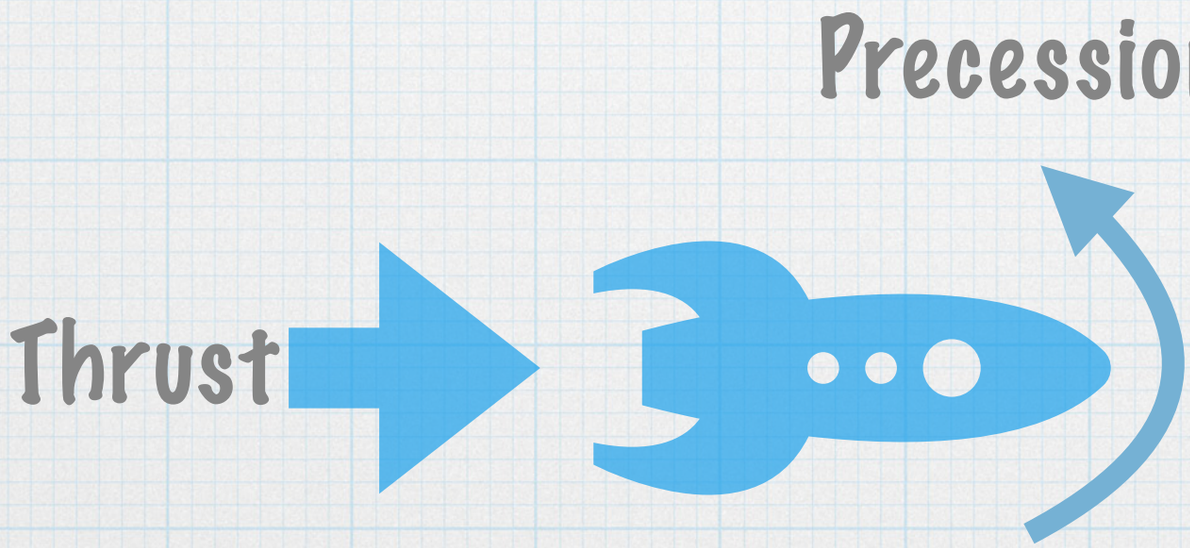
```
##
# Pass by Reference
##
set A {1 2 3}
set B {4 5 6}
::vectorxyz::incr A $B
```

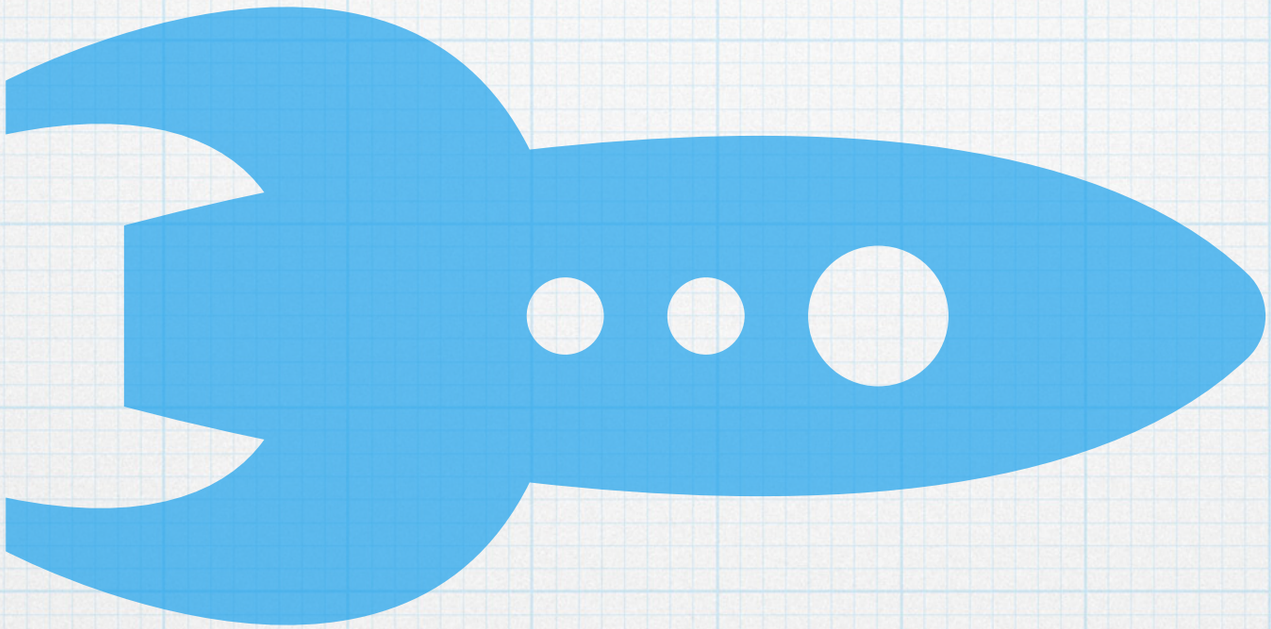
```
##
# Pass by Reference
##
my c_tclcmd  ::vectorxyz::incr {
  Odie_MatrixObj *A;
  VectorXYZ B;
  Tcl_Obj *varname;
  if(objc < 3) {
    Tcl_WrongNumArgs( interp, 1, objv, "A B" );
    return TCL_ERROR;
  }
  varname=Tcl_ObjGetVar2(interp,objv[1],NULL,0);
  if(
Odie_GetMatrixFromTclObj(interp,varname,MATFORM_vectorxyz,&A)
) return TCL_ERROR;
  Tcl_ResetResult(interp);
  if(
Odie_GetVectorXYZFromTclObj(interp,objv[2],B)
) return TCL_ERROR;
  VectorXYZ_Add(A->matrix,A->matrix,B);
  Tcl_InvalidateStringRep(varname);
  return TCL_OK;
}
```

```
# Calculate the location, orientation, thrust, and spin
# of an object in 3d space
set location       [vectorxyz::create {0 0 0}]
set attitude       [affine4x4::identity]
set attitude_delta [affine4x4::rotate_precession 0.9]
set deltap         [vectorxyz::create {1 0 0}]
set velocity       [vectorxyz::create {0 0 0}]

for {set x 0} {$x < 10} {incr x} {
   # Compute one step
   affine4x4::*= attitude $attitude_delta
   set F [vectorxyz::transform $attitude $deltap]
   vectorxyz::+= velocity $F
   vectorxyz::+= location $velocity
   puts [list f: $F x: $location v: $velocity]
}
```

```
f: {0.62161 0.783327 0} x: {0.62161 0.783327 0} v: {0.62161 0.783327 0}
f: {-0.227202 0.973848 0} x: {1.01602 2.5405 0} v: {0.394408 1.75717 0}
f: {-0.904072 0.42738 0} x: {0.506354 4.72506 0} v: {-0.509664 2.18455 0}
f: {-0.896758 -0.44252 0} x: {-0.900069 6.46709 0} v: {-1.40642 1.74203 0}
f: {-0.210796 -0.97753 0} x: {-2.51729 7.23159 0} v: {-1.61722 0.764504 0}
f: {0.634693 -0.772765 0} x: {-3.49981 7.22333 0} v: {-0.982526 -0.00826063 0}
f: {0.999859 0.0168139 0} x: {-3.48248 7.23189 0} v: {0.017333 0.00855327 0}
f: {0.608351 0.793668 0} x: {-2.8568 8.03411 0} v: {0.625684 0.802221 0}
f: {-0.243544 0.96989 0} x: {-2.47466 9.80622 0} v: {0.38214 1.77211 0}
f: {-0.91113 0.412118 0} x: {-3.00365 11.9904 0} v: {-0.52899 2.18423 0}
```





Geometry Operations

* Convex Decomposition of Polygons
* Synthesizing Polygons from Vectors
* Decomposing Polygons into Line Segments
* Set Operations on Polygons

Raw Shapes

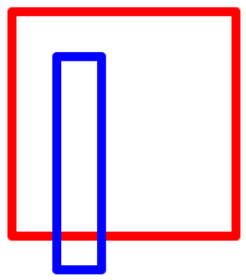
Union

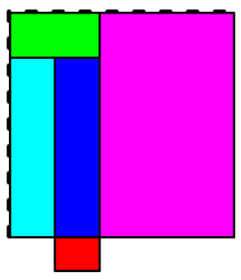
Intersection

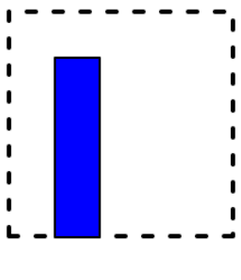
A - B

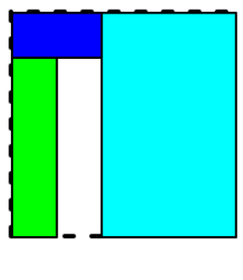
B - A

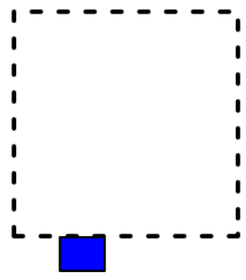
A - O



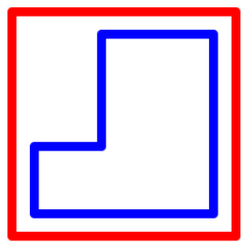


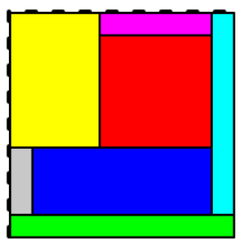


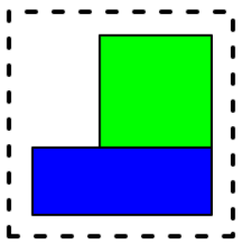


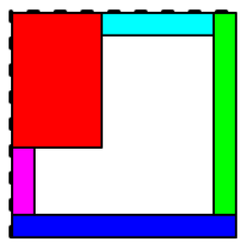


A - P



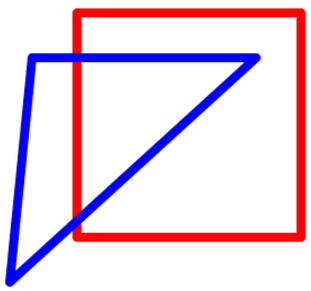


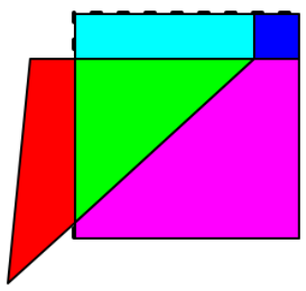


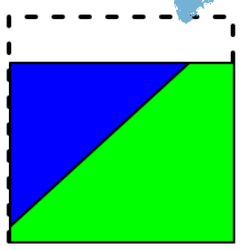


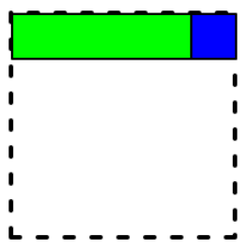
Bad Case:
Bug in polygon::within

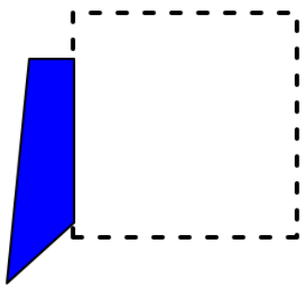
A - Q











I Should Put an Obvious Flaw In My Demonstration...

Future Direction

* Develop a more coherent standard for namespaces
  * Ex: affine::4x4::multiply
* Allow modules to be written in Critcl notation
* Adapt visual tests to standard regression tests

Comparison to Vectcl

* Vectcl is an expression engine.
* Odielib is designed to perform batch operations
* Odielib Doesn't Need Patches to the Core
* Vectcl Allows arbitrary dimensions
* Odielib is currently limited to a 4x4 matrix.

Where to Get it

Fossil Repo:

http://fossil.etoyoc.com/fossil/odielib

Contact:
Sean Woods
yoda@etoyoc.com