

# Re: Design problem

---

*Source:* <http://coding.derkeiler.com/Archive/General/comp.object/2006-02/msg00093.html>

---

- *From:* "Mark Nicholls" <Nicholls.Mark@xxxxxxxxxx>
  - *Date:* 6 Feb 2006 02:40:31 -0800
- 

Robert C. Martin wrote:

On 1 Feb 2006 02:29:31 -0800, "Mark Nicholls"  
<Nicholls.Mark@xxxxxxxxxx> wrote:

Robert C. Martin wrote:

On 6 Jan 2006 13:38:11 -0800, roberts.noah@xxxxxxxxxx  
wrote:

I'm having trouble with what seems like it  
should be a relatively  
simple problem and am wondering what  
ideas people here might have.

I have a shape class and inside this shape are  
"elements" (vector drawn  
based system). Certain things can be done to  
shapes, namely you can  
rotate, skew, flip, zoom..ummm...that's about  
it but you never know  
what could be added. Elements at this point  
can comprise of one of the  
following three constructs: A line; a  
polygon/polyline; or a circle.  
Lines are two point elements that contain a  
line from one to the other.  
Polygons are point coordinate arrays with a  
start and an end; empty  
polygons are actually lines from point to  
point otherwise it is a  
filled element. A circle is a center and a  
radius.

Now the problem. Sometimes these

Re: Design problem

operations are done one after the other and I can increase speed (and yes we want it to be fast) if I can do them all at once. I want a raw "perform" function on either the element or the operation so that I can pass in individual operations or constructs of operations. The easy way to do this is to build matrices and pass them in, but how to apply these to a circle?? I can of course have enums representing types of operations and have some ifs in the circle construct but this fails the LSP (and I happen to agree with that principle). I do have a couple of ideas but there are failings in all of them (including rethinking how a circle is represented).

What ideas do you guys have?

Think of a circle as being defined by the radius line. It's not a point and a length, it's two points.

Thats fine and dandy, but the space of circles is not closed under the operations described...

i.e. compressing the Y axis makes a circle into a ellipse, not a little circle.

Compressing the Y axis wasn't on the list above.

True, I assumed the etc...meant all the linear transformations on a vector space, but it doesn't matter,skew was mentioned....a skewed circle is not a circle, so the same logic applies.

I presume, then, that you can compress or expand either axis. Can you define the behavior of the system when you expand or compress an axis?

yes, 'linear vector spaces'.....though you need to throw in something about ellipses or more generally conic sections.

Re: Design problem

## Re: Design problem

For example, does the aspect ratio of a square change if you expand or compress the Y axis?

yes....so it's not a closed space.

If not, then how does the square behave?

it would have to change into a rectangle else exception.

What about a rectangle, does the aspect ratio change?

yes....but it's still a rectangle...it is closed...thus we do not need to know it's concrete shape in order to define what it becomes....it's always a rectangle.

but it doesn't work for skew....so you'd need quadrilateral....though the OP has solved this by using line segments.

Why would it change for a rectangle and not for a square?

because squares cannot change aspect ratio, so you would have to map a square to a rectangle....in which case you need to know both operation and concrete class...i.e. double dispatch.

What about equilateral triangles, or any regular polyhedron?

same as above, not closed.

Do they become irregular?

yes

Or do they maintain their regularity?

no

What should the behavior of a circle be if the Y axis is compressed?

it should 'become' an ellipse.

Some spaces are closed and some not....so if we want a single model (and not have to dispatch on shape and operation), we need a representation that is closed.

using points as the general representation is fantastic in theory because points are closed under all transformation, and points on a locus=>points on a locus.

but you have to pick a finite set of points that uniquely identify the 'shape' of a given shape....or end up with unbounded sets of points.....or a representation that does not identify the resultant shape.

.