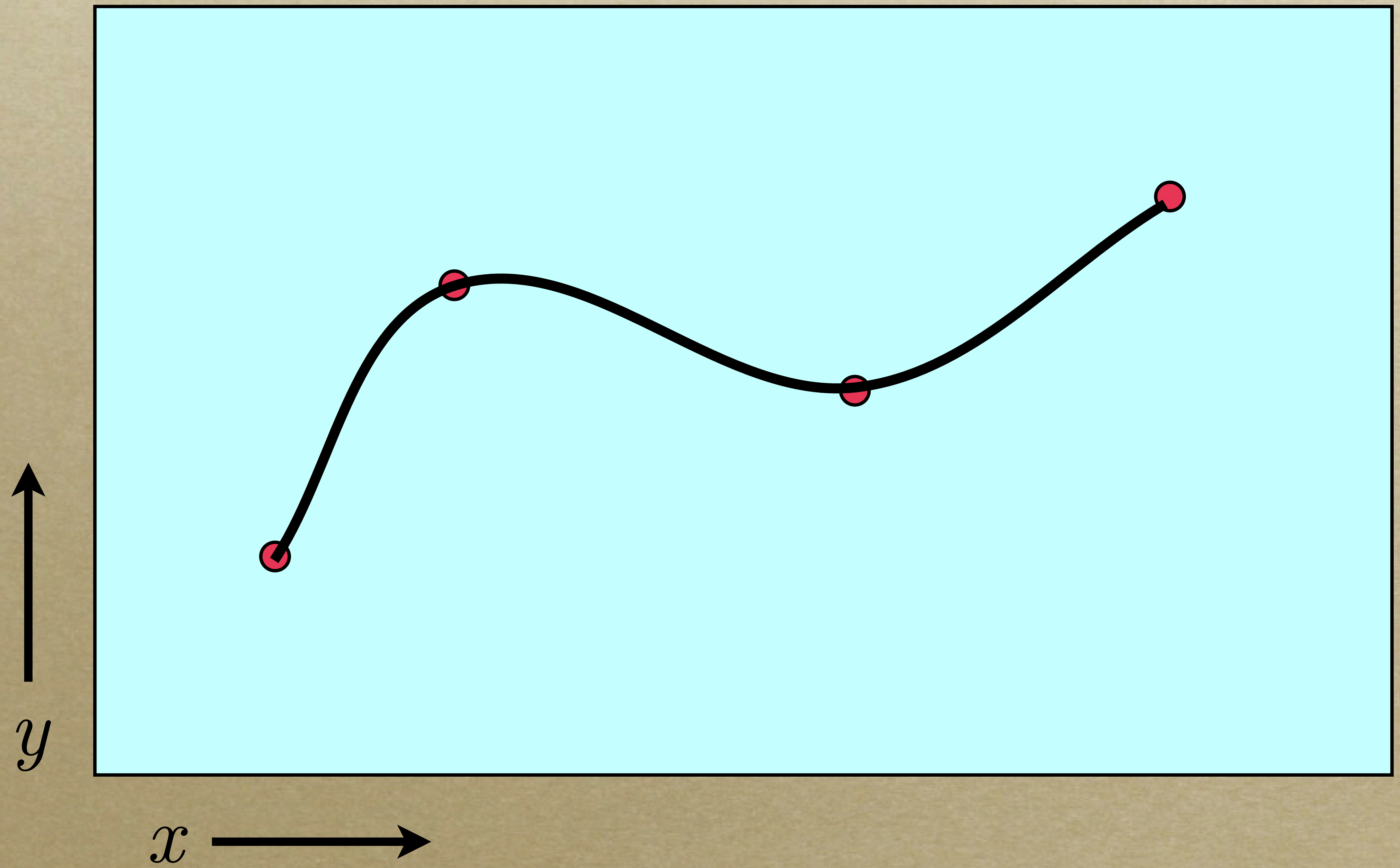
Modelling Techniques

- *Parametric Patches*
- *Constructive Solid Geometry (CSG)*
- *Subdivision Surfaces*
- *Implicit Surfaces*

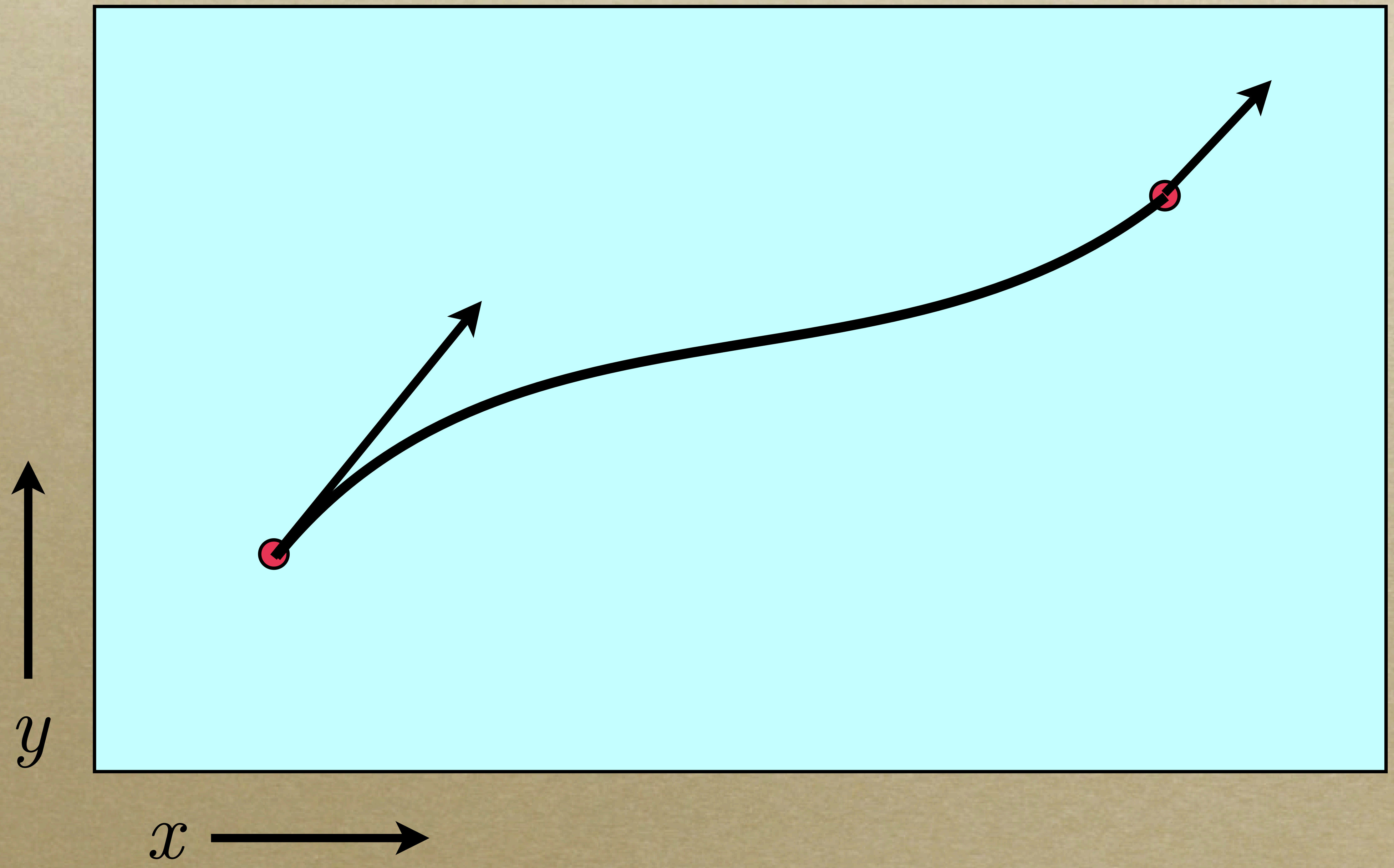
Parametric patches

- *Polygons are flat*
- *Objects have curved faces*
- *Can we fit a curve to points in space?*
- *Can we fit a surface to points in space?*

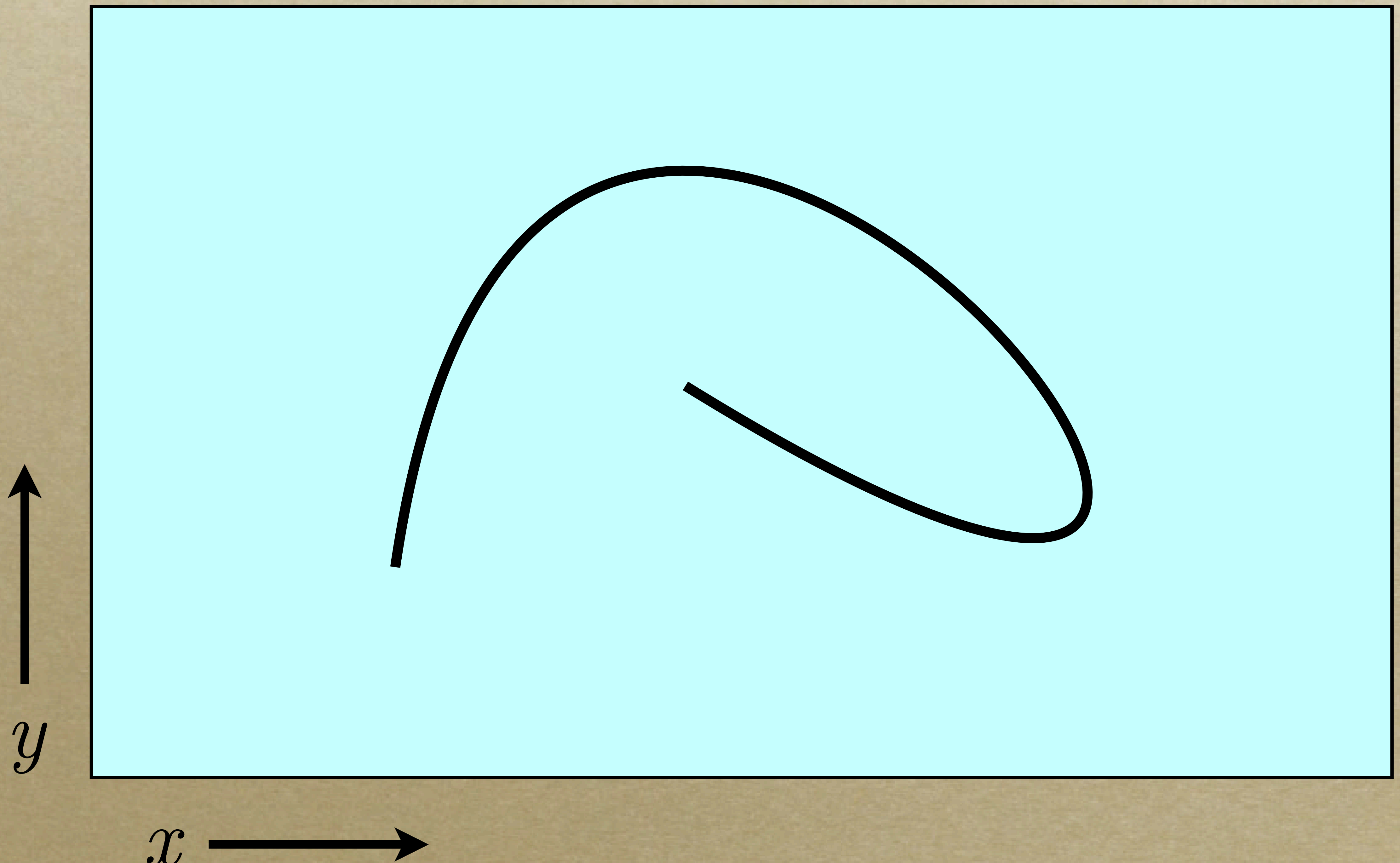
Curve fitting



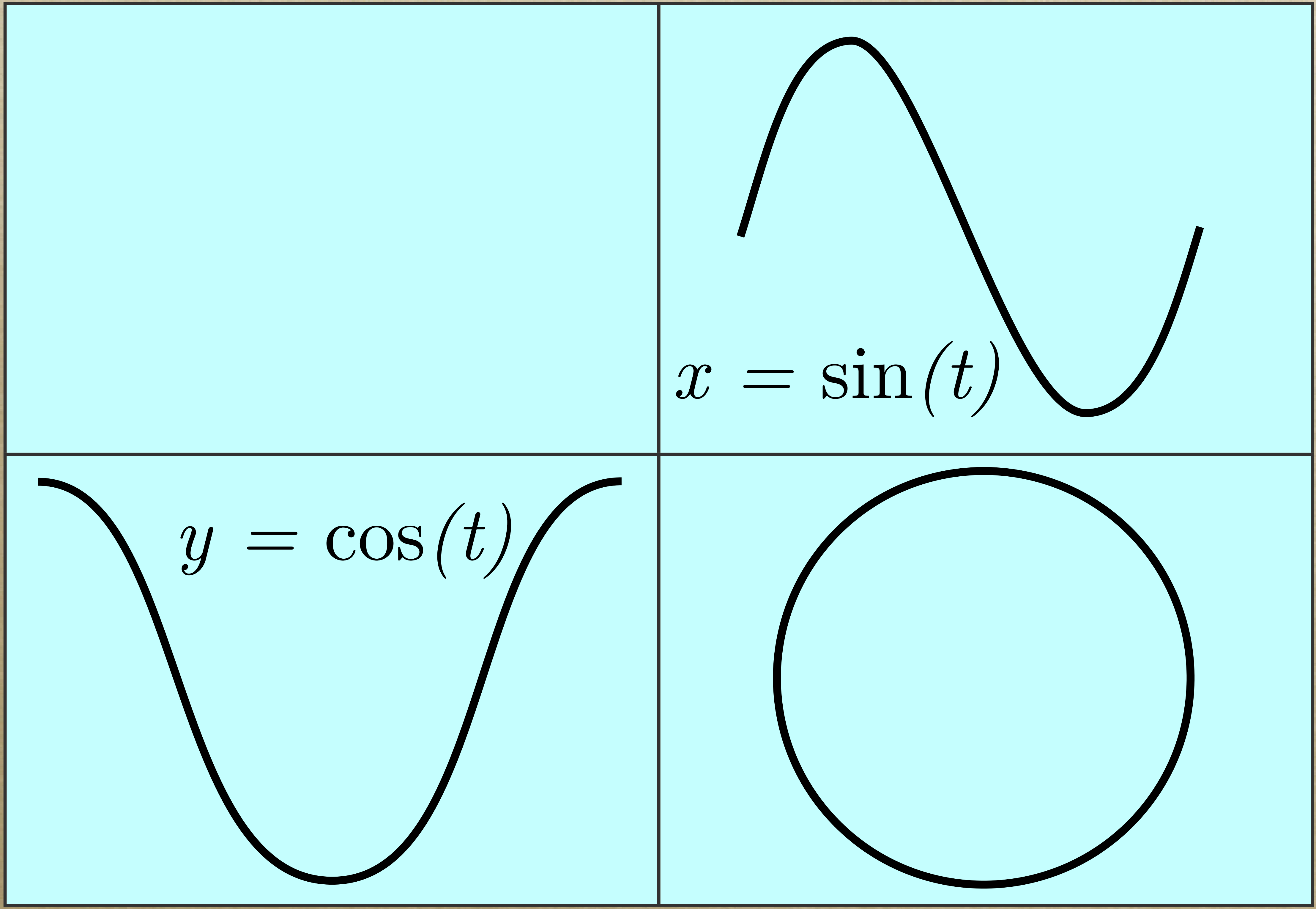
Curve fitting



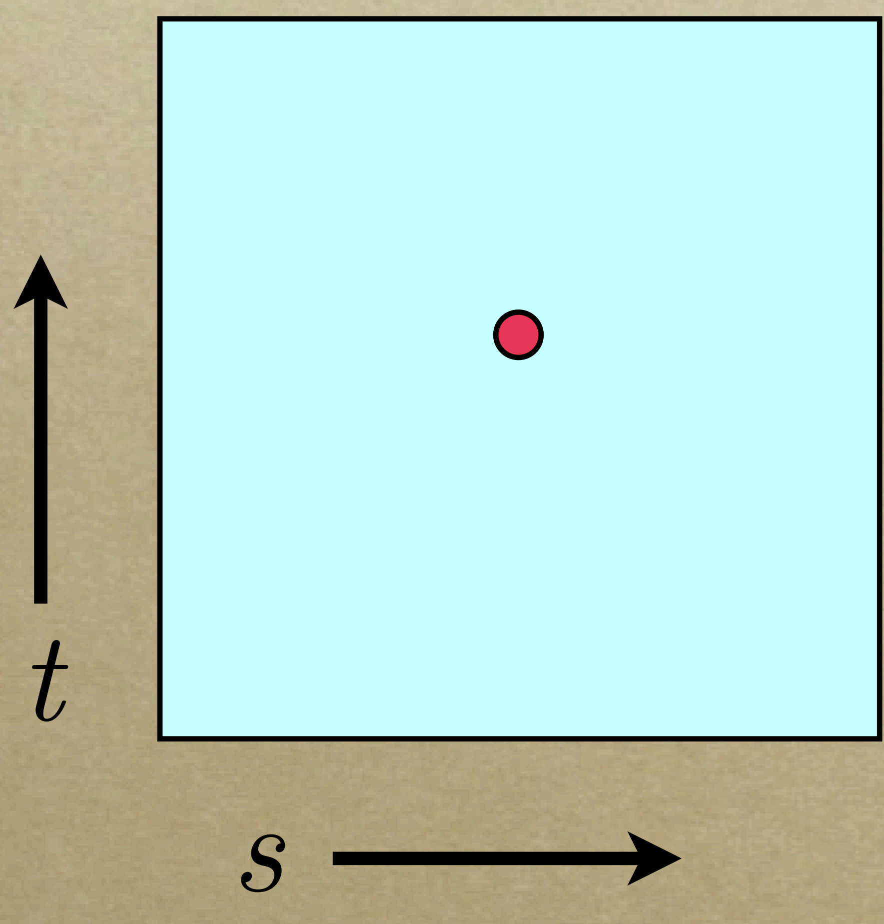
How do we do this one?

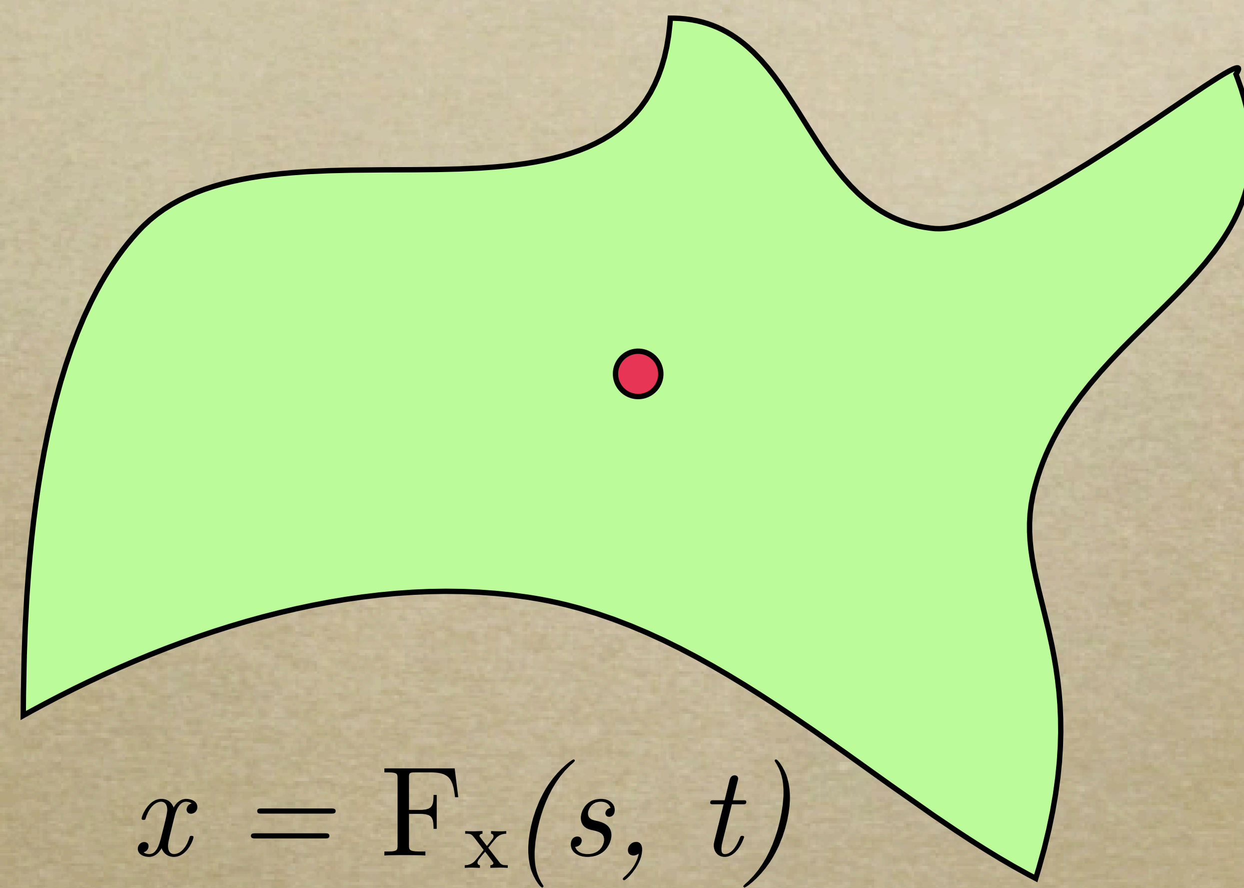


Plot $x(t)$, $y(t)$



Patches





$$x = \mathrm{F_x}(s,\ t)$$

$$y = \mathrm{F_y}(s,\ t)$$

$$z = \mathrm{F_z}(s,\ t)$$

What function?

$F_x(s,\ t) =$

$X_{3,3}s^3t^3\ +$	$X_{3,2}s^3t^2\ +$	$X_{3,1}s^3t\ +$	$X_{3,0}s^3\ +$
$X_{2,3}s^2t^3\ +$	$X_{2,2}s^2t^2\ +$	$X_{2,1}s^2t\ +$	$X_{2,0}s^2\ +$
$X_{1,3}st^3\ +$	$X_{1,2}st^2\ +$	$X_{1,1}st\ +$	$X_{1,0}s\ +$
$X_{0,3}t^3\ +$	$X_{0,2}t^2\ +$	$X_{0,1}t\ +$	$X_{0,0}$

16 term cubic for x, y, z

That means there are 48 coefficients:

$X_{i,j}$, $Y_{i,j}$, $Z_{i,j}$,

Often we derive the values from 16 points: $\mathbf{u}_{i,j}$

Pros and cons

✓ *Can make a huge variety of shapes*

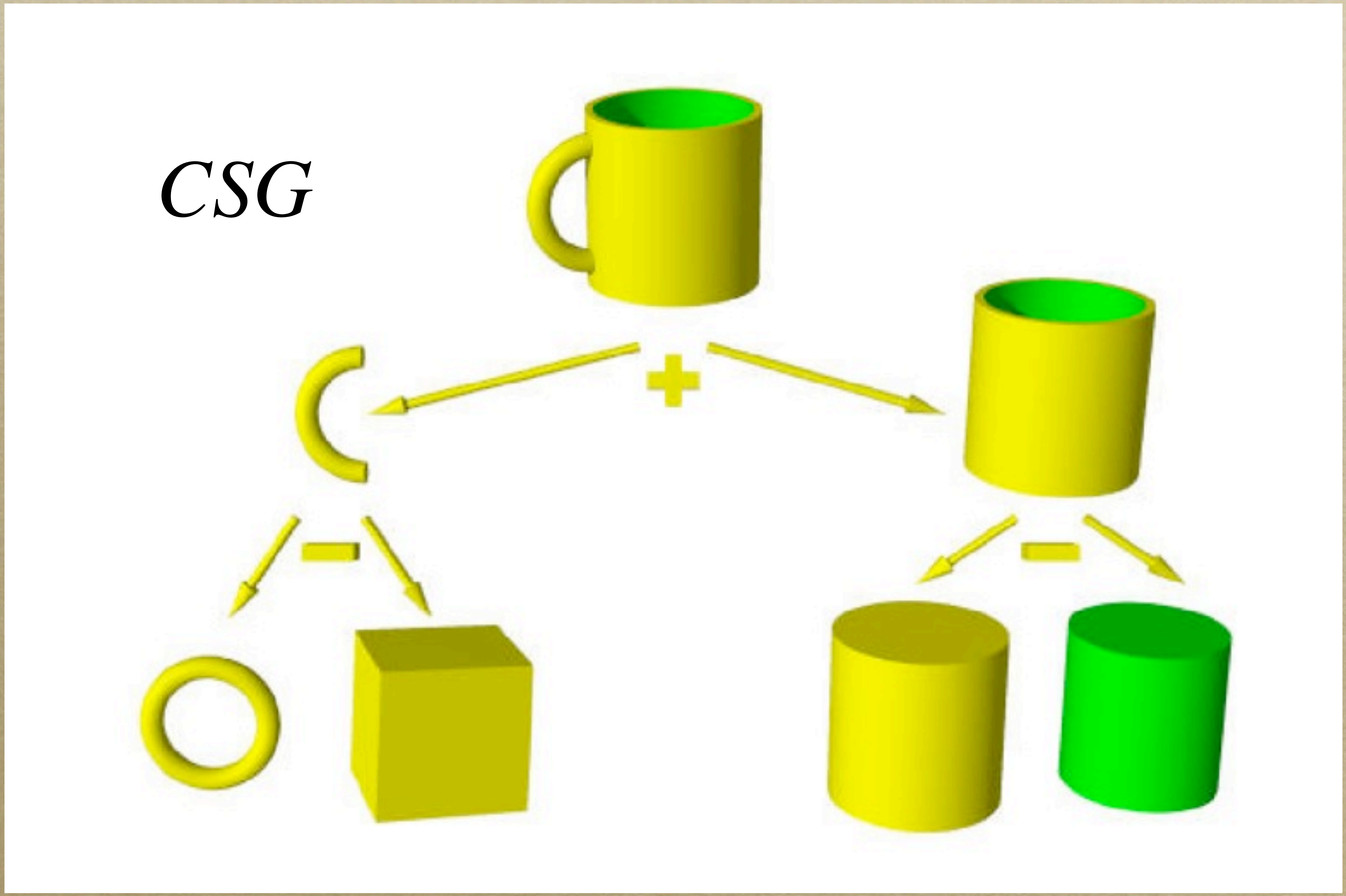
✓ *Smooth surfaces*

✓ *Can represent spheres and cylinders*

✗ *Difficult boundary conditions*

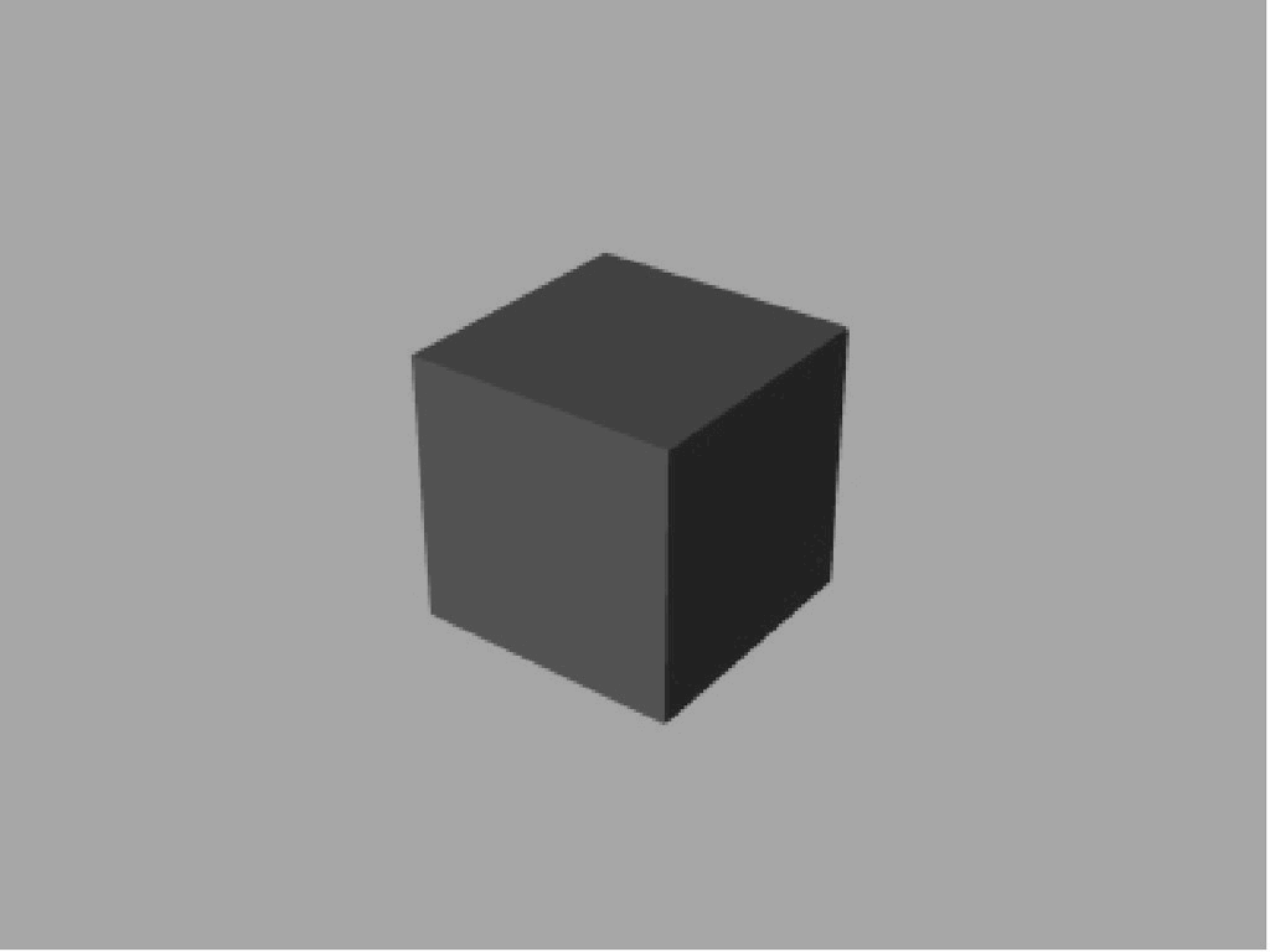
✗ *Only defines the surface—no inside*

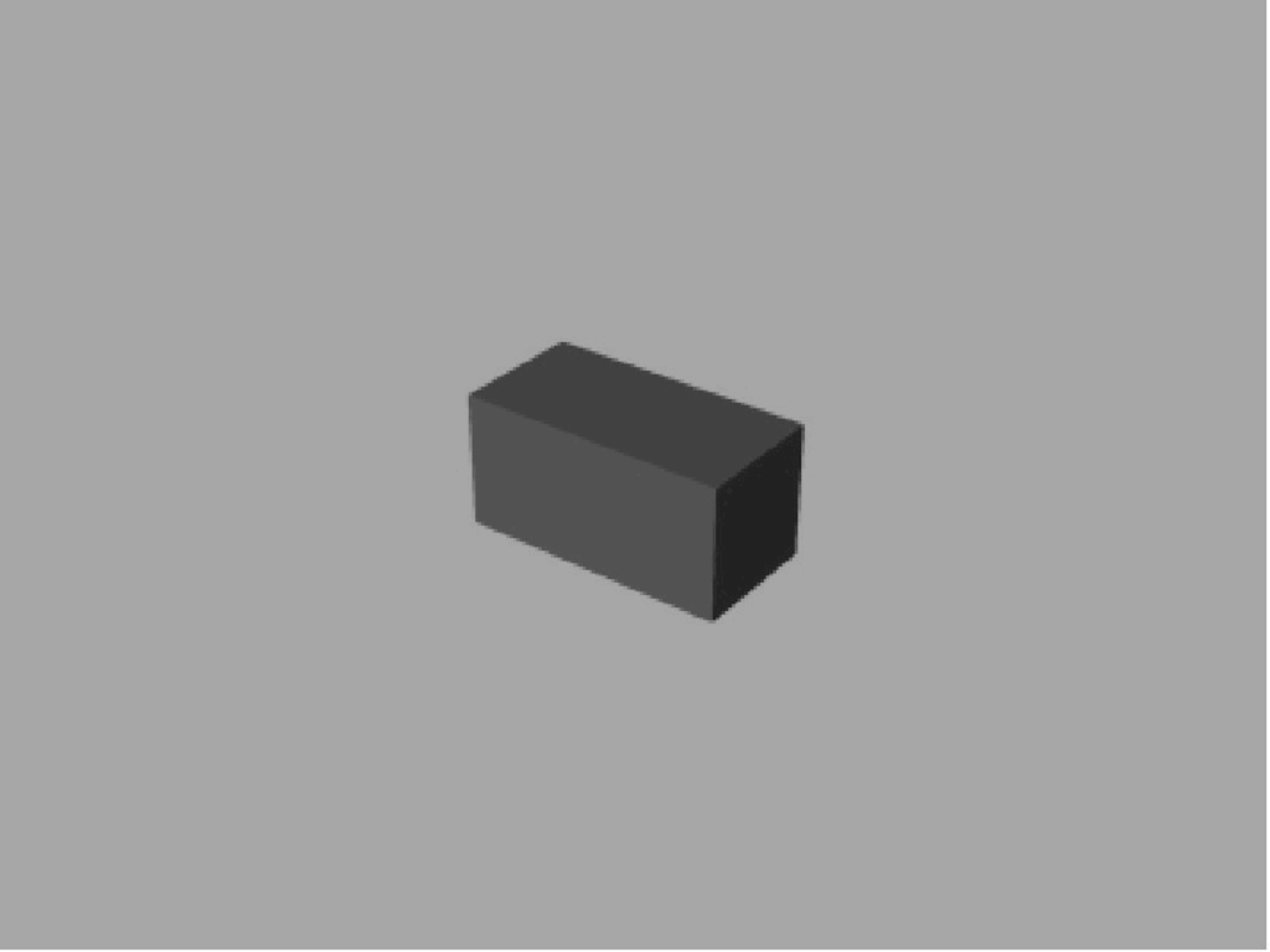
Constructive Solid Geometry

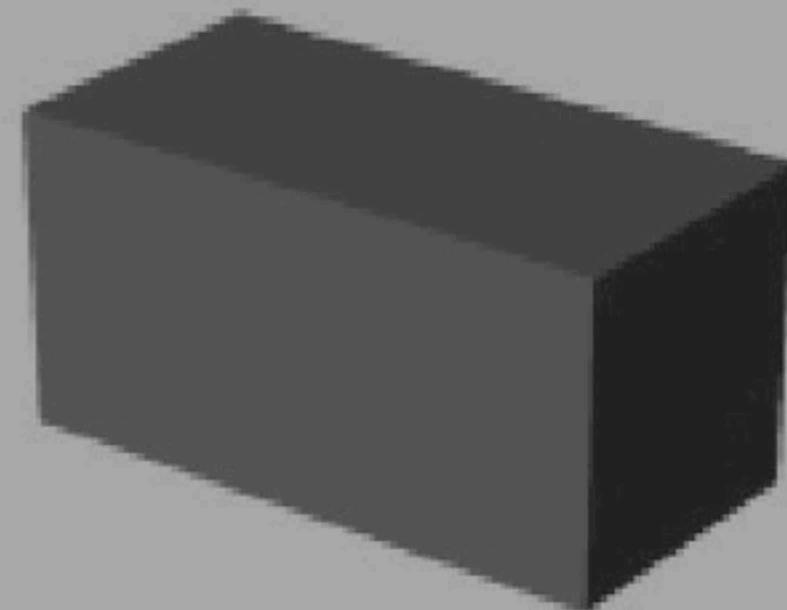


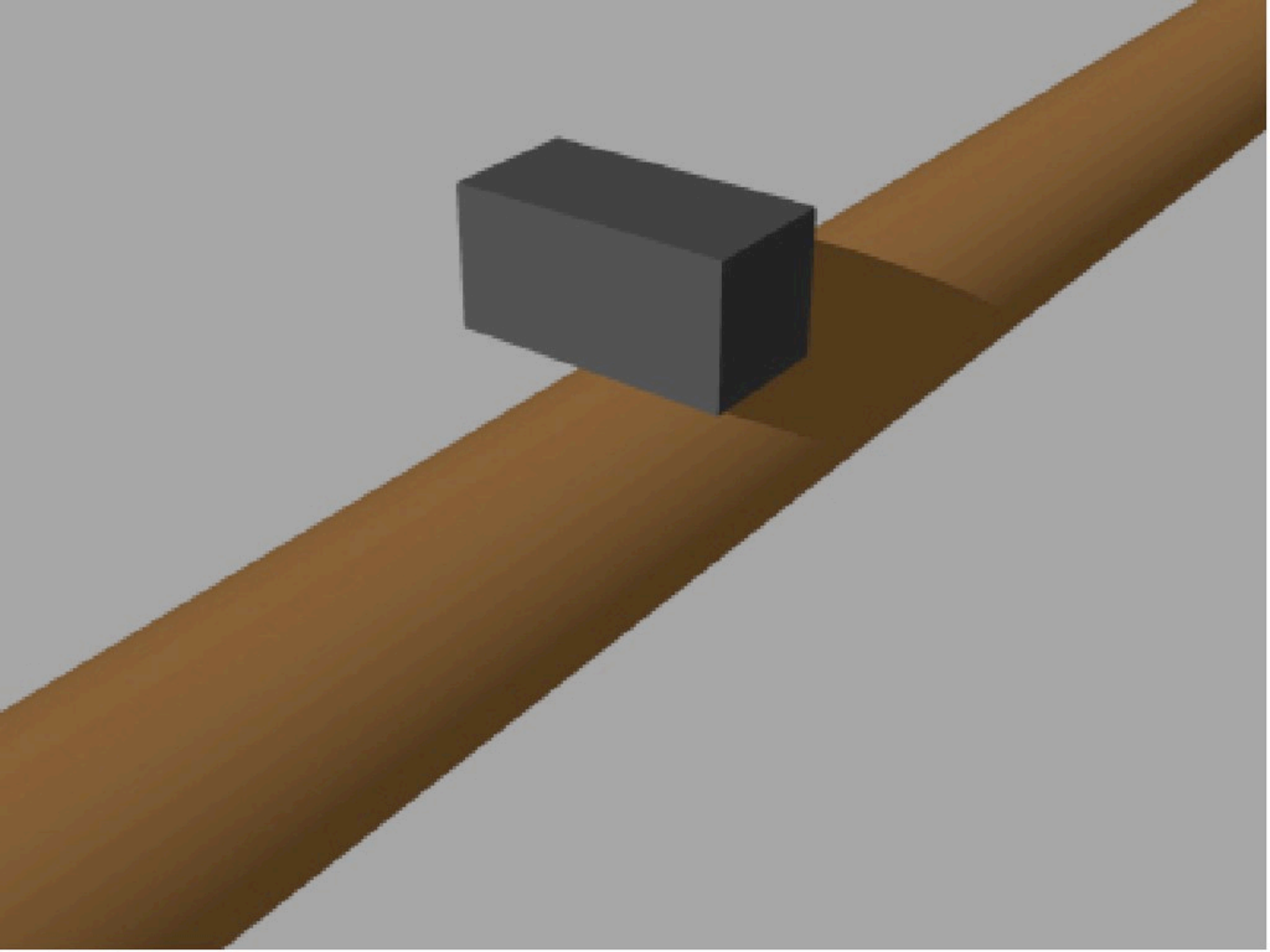
Building a hammer

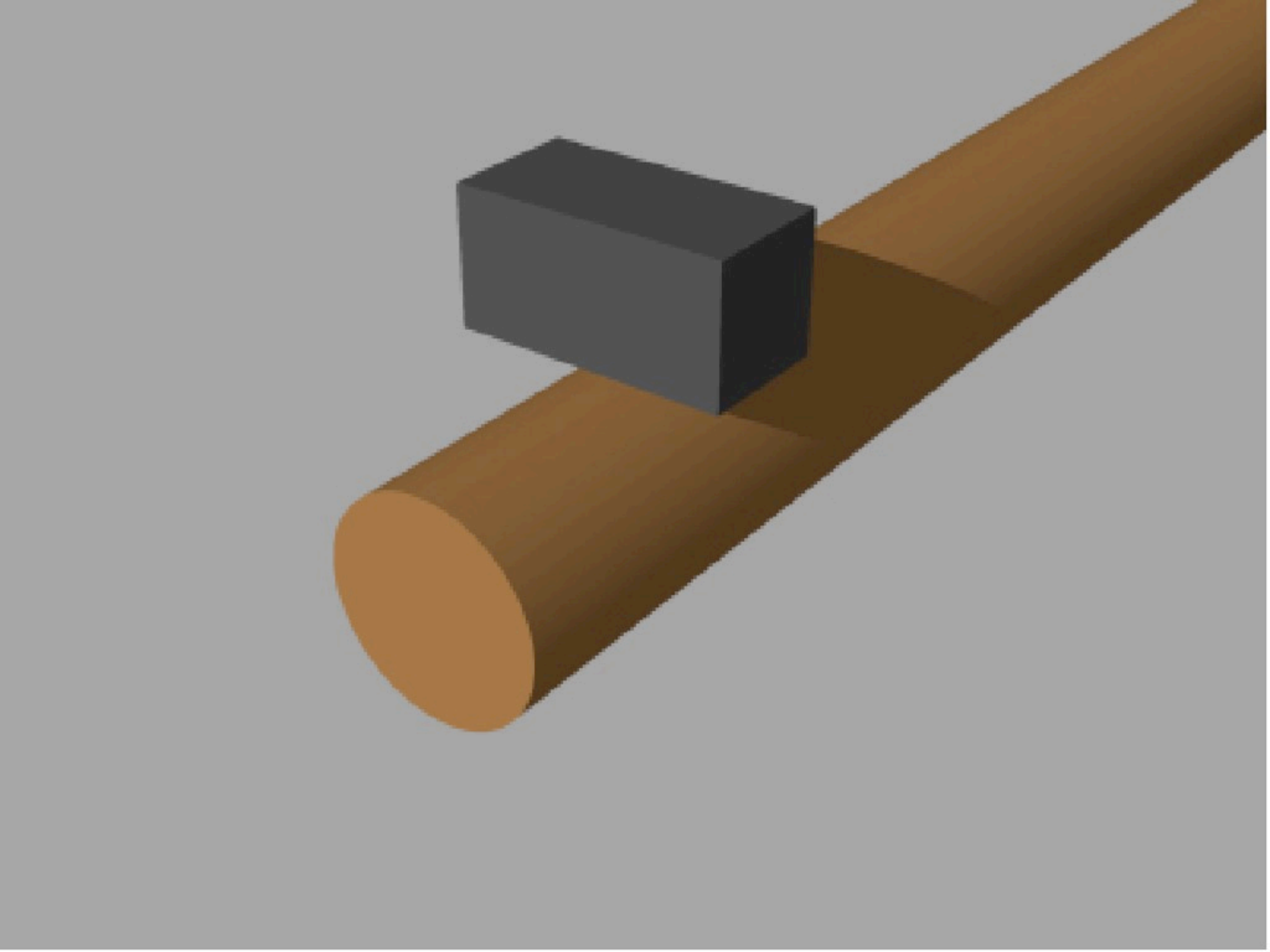
Demonstration of CSG modelling by David Mason (while a stage 4 student).

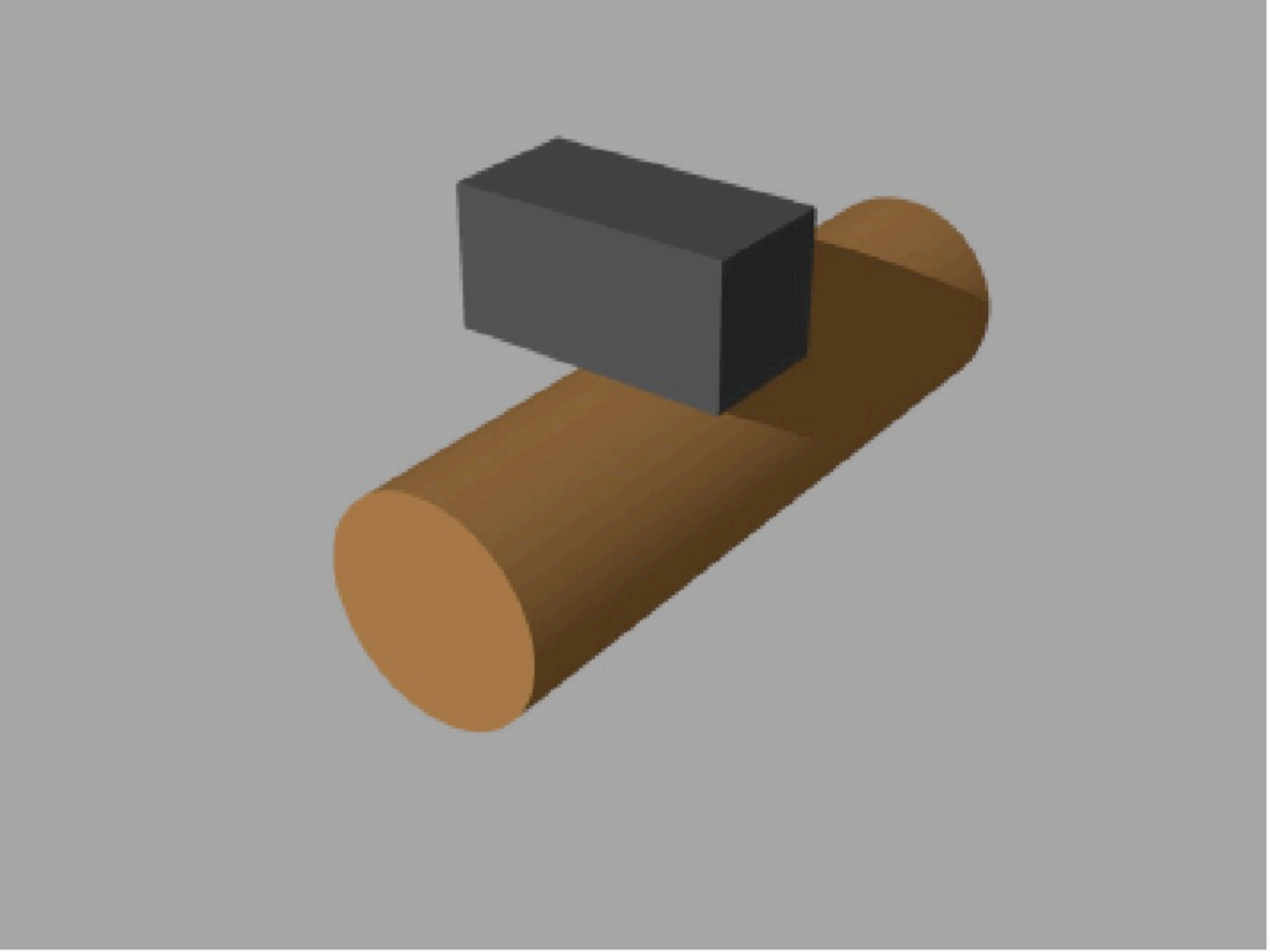


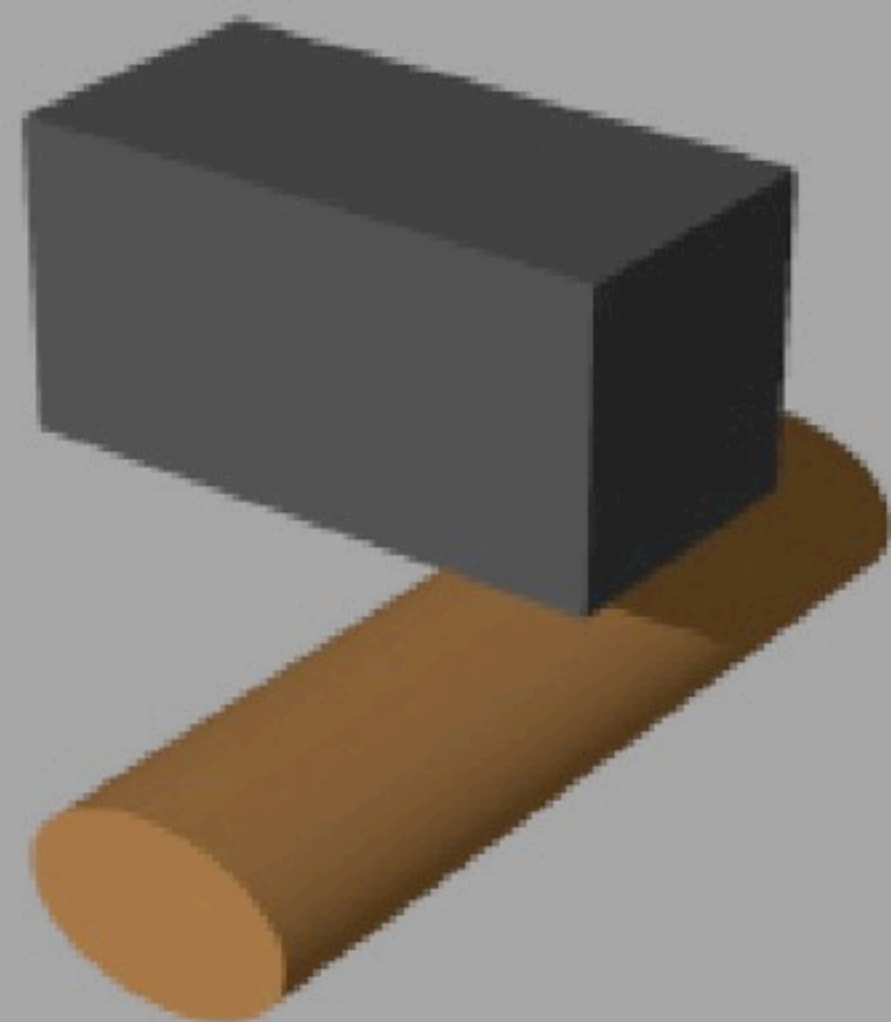


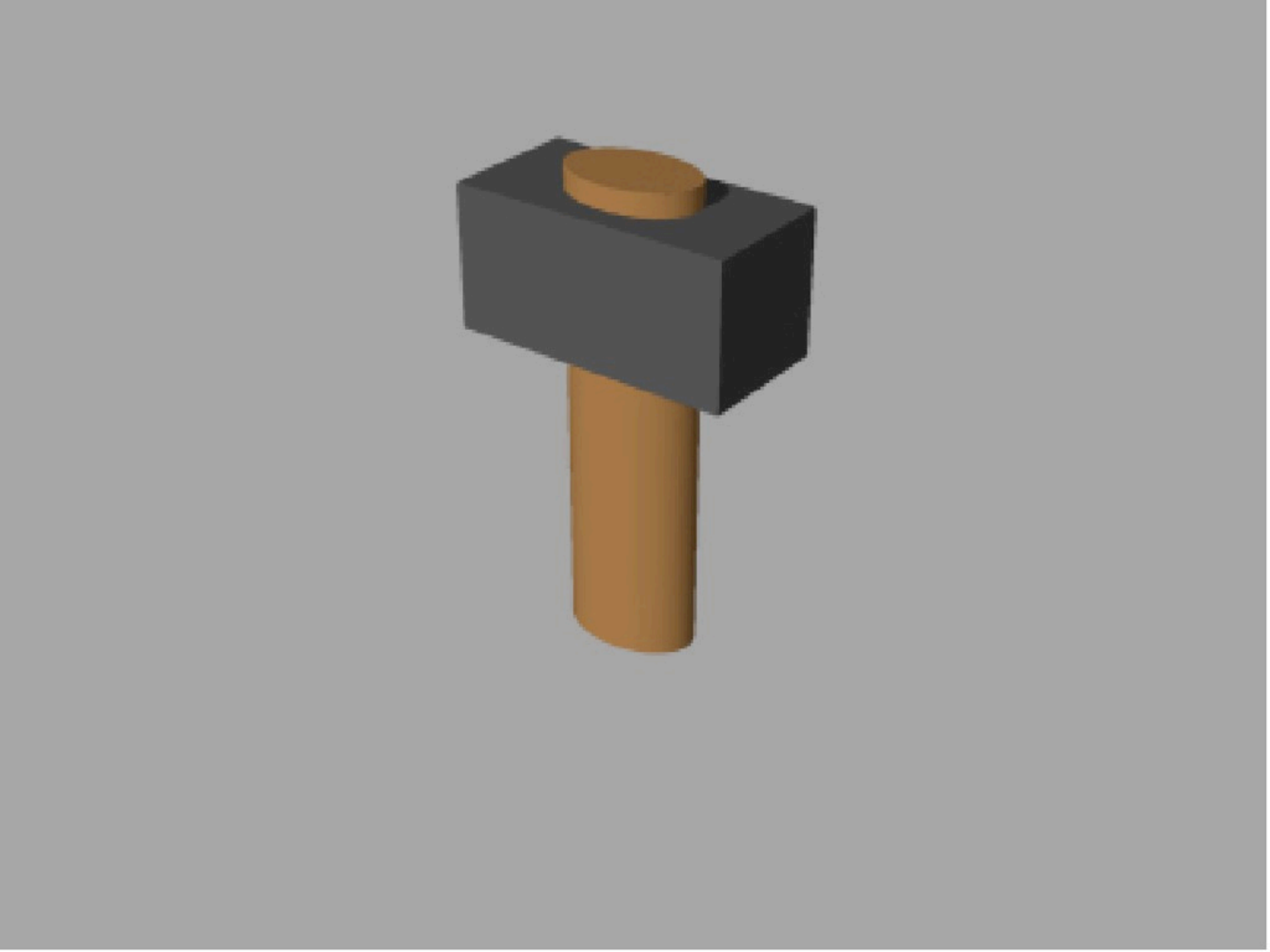




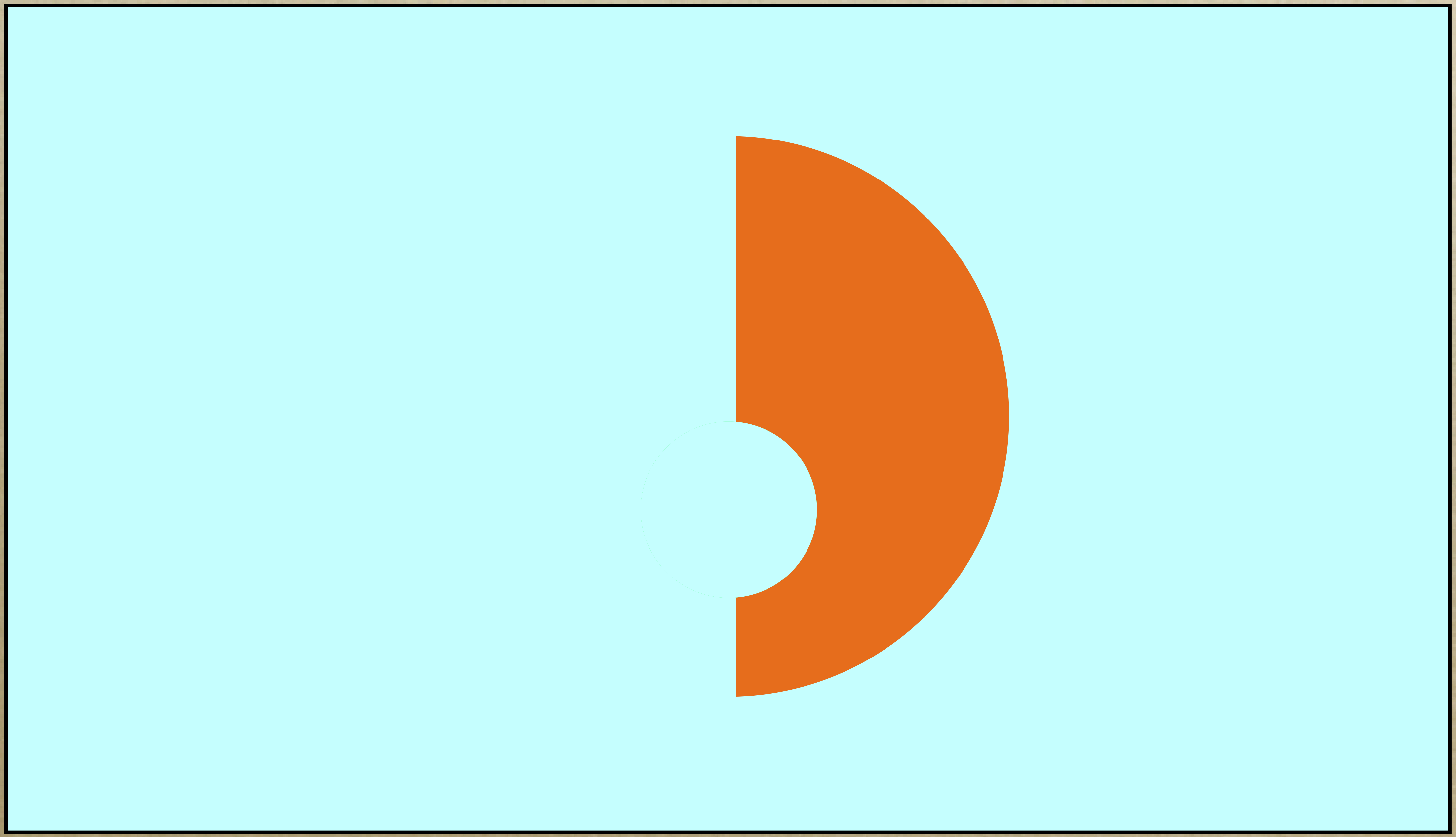




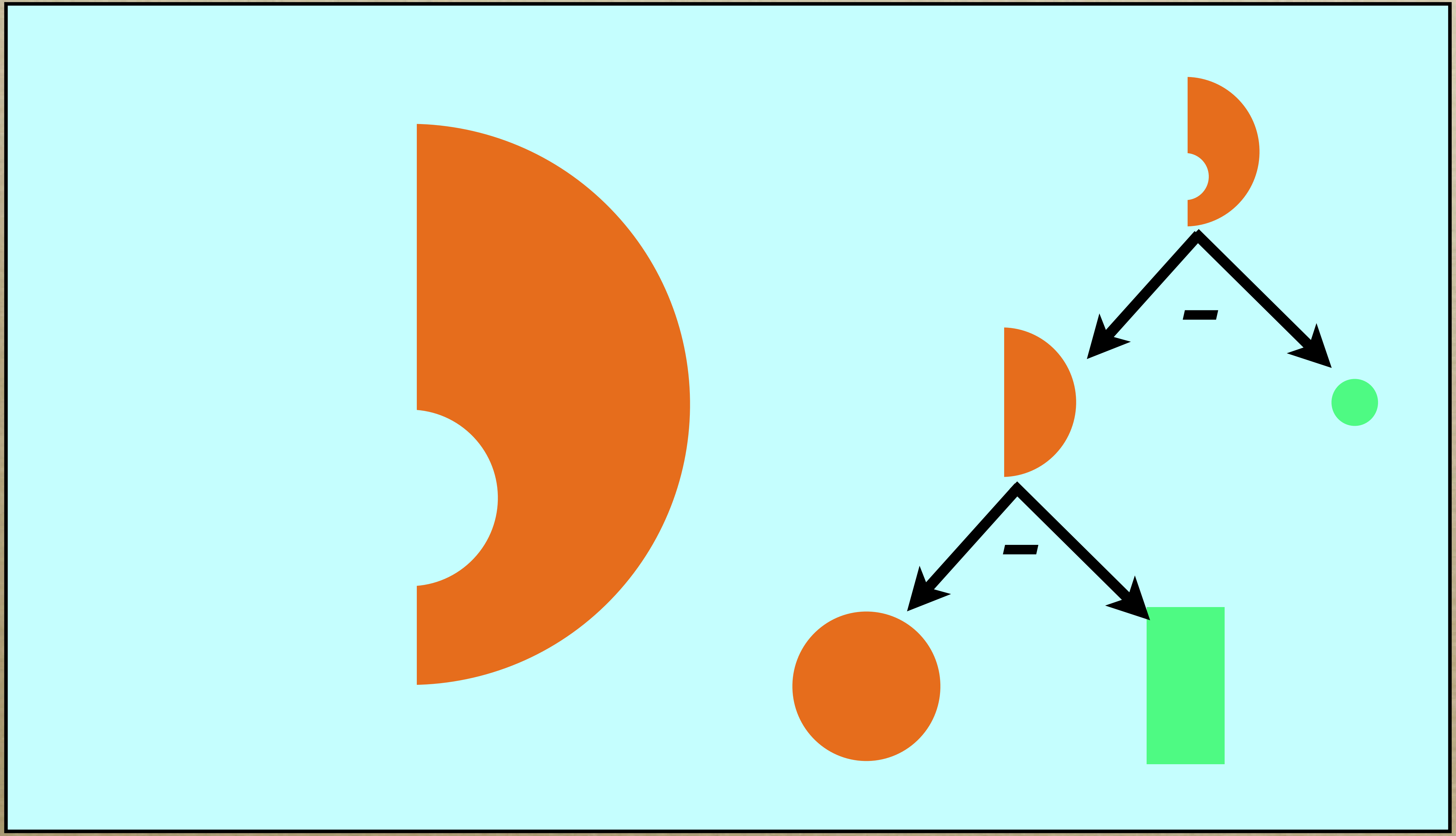




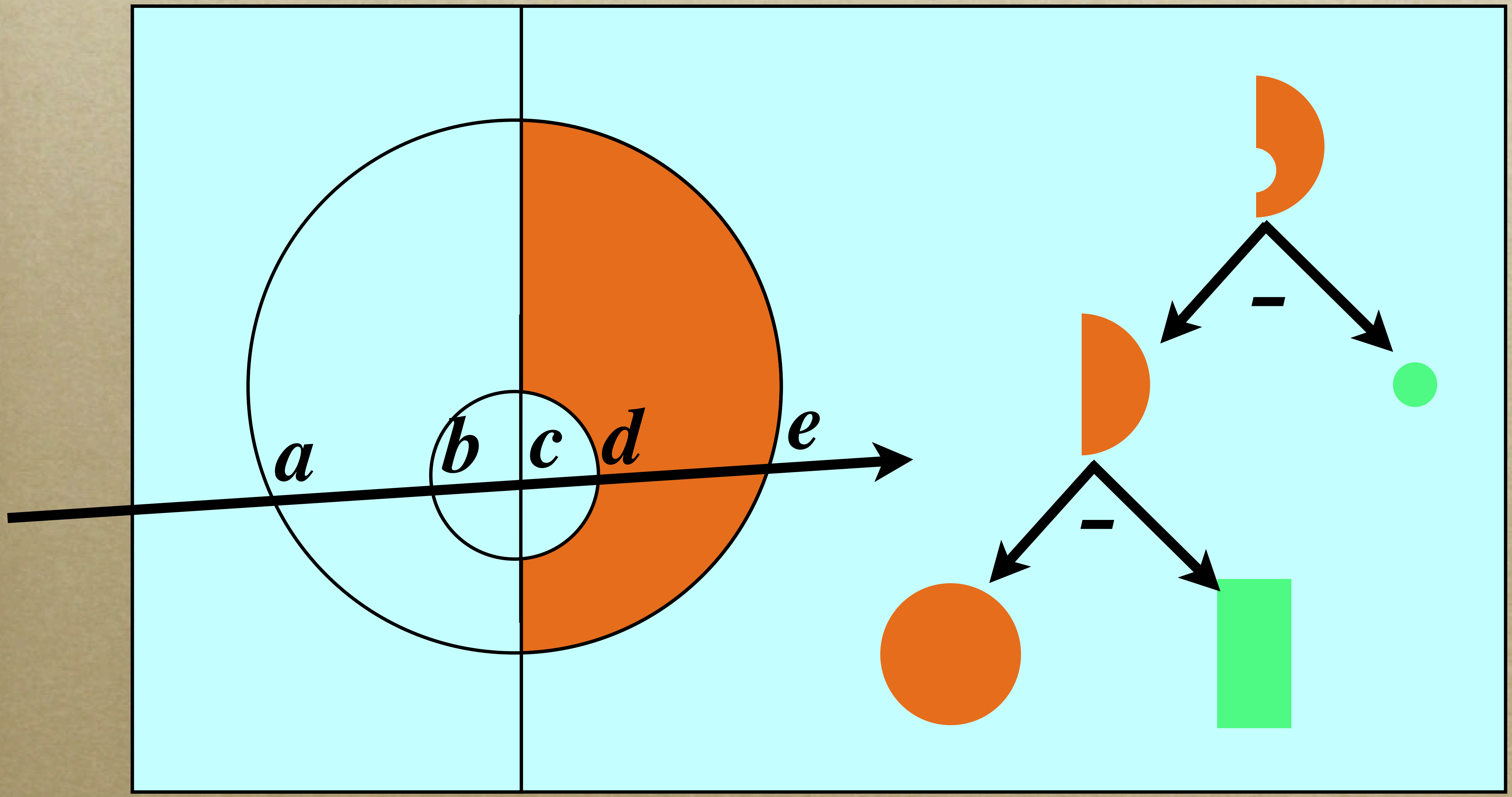
Direct ray tracing example



Object and CSG ‘tree’



Classifying the ray



Classification at *subtract* node

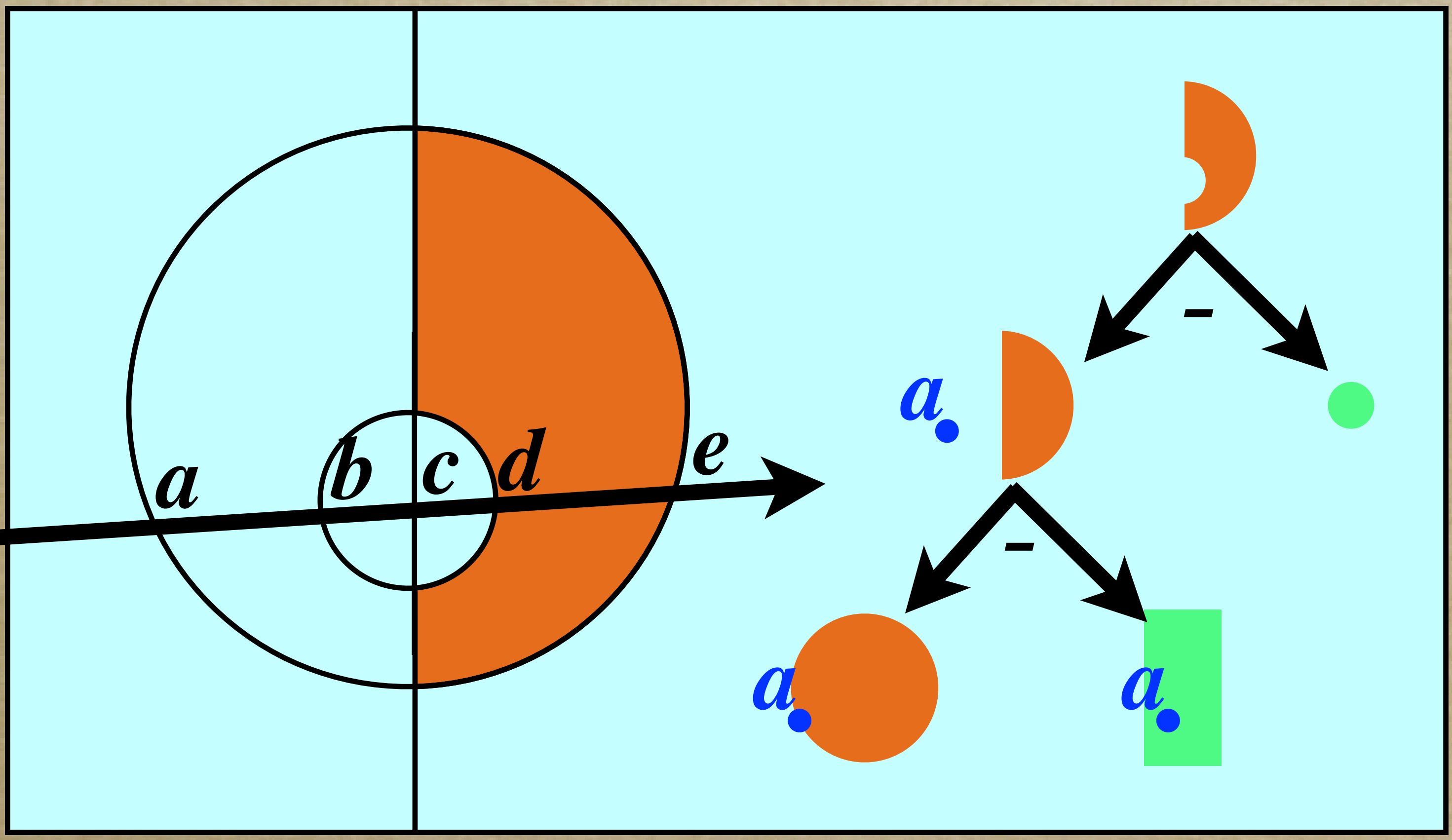
		Right tree child		
		In	**Out**	**Border**
Left tree child	**In**	*Out*	*In*	*Border*
	Out	*Out*	*Out*	*Out*
	Border	*Out*	*Border*	

Subtract nodes for our object

		Right tree child		
		In	**Out**	**Border**
Left t.c.	**In**	*Out*	*In*	*Border*
	Out	*Out*	*Out*	*Out*
	Border	*Out*	*Border*	

Consider point 'a'

... it is outside the hemisphere



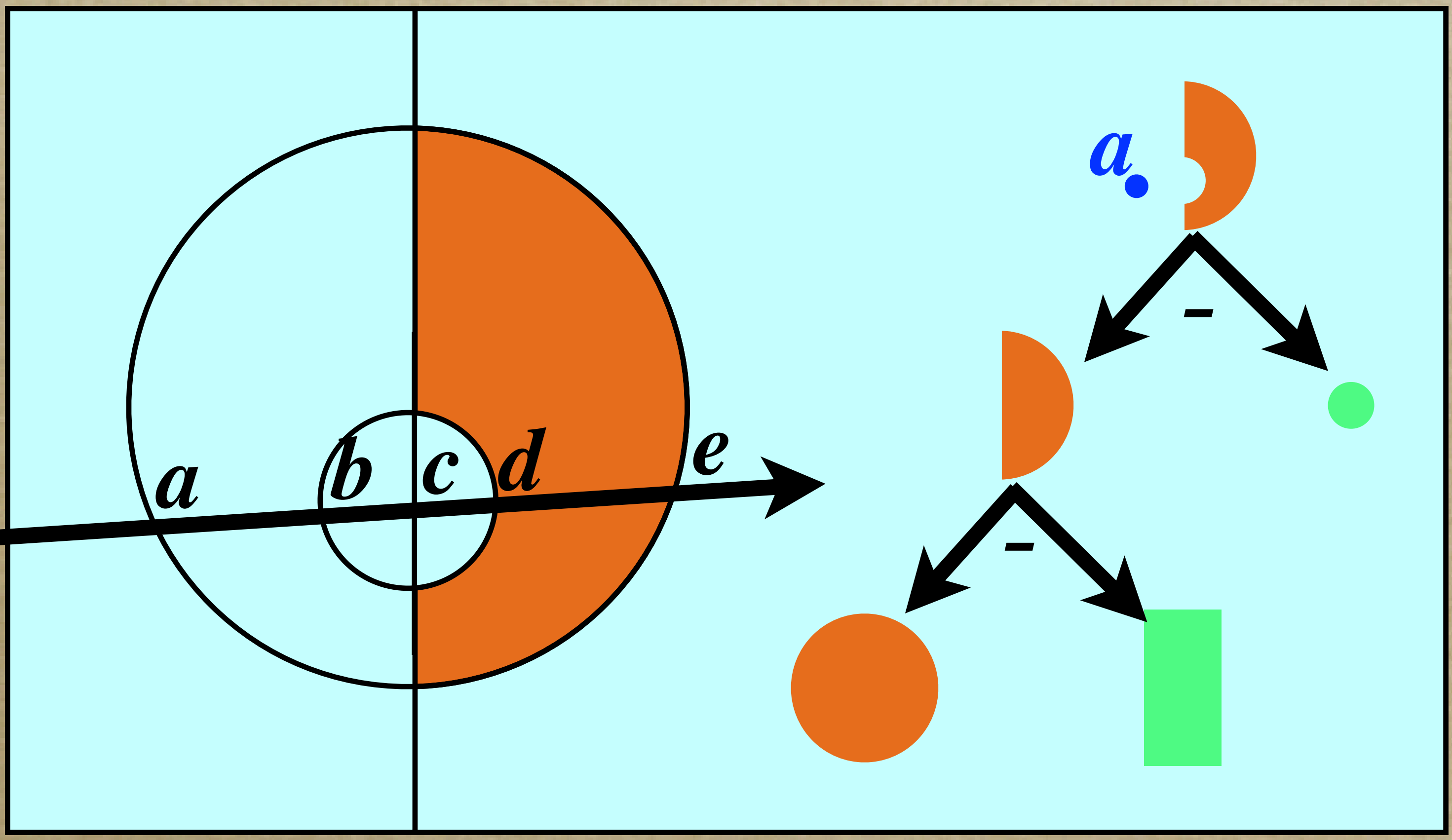
Subtract nodes for our object

		Right tree child		
		In	**Out**	**Border**
Left t.c.	**In**	*Out*	*In*	*Border*
	Out	*Out*	*Out*	*Out*
	Border	*Out*	*Border*	

Consider point 'a'

... it is outside the hemisphere

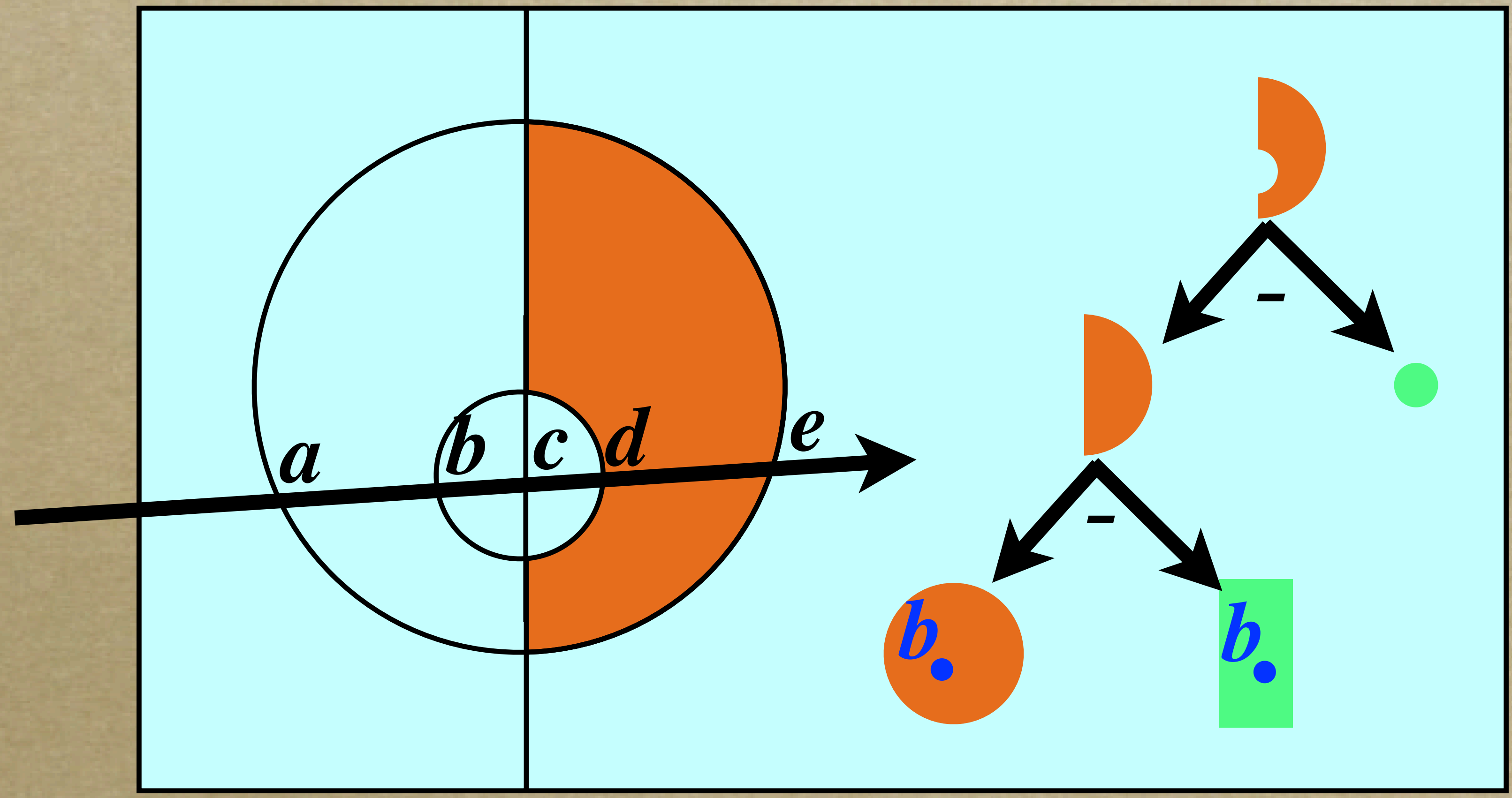
... and outside our object



Subtract nodes for our object

		Right tree child		
		In	**Out**	**Border**
Left t.c.	**In**	*Out*	*In*	*Border*
	Out	*Out*	*Out*	*Out*
	Border	*Out*	*Border*	

Point 'b'

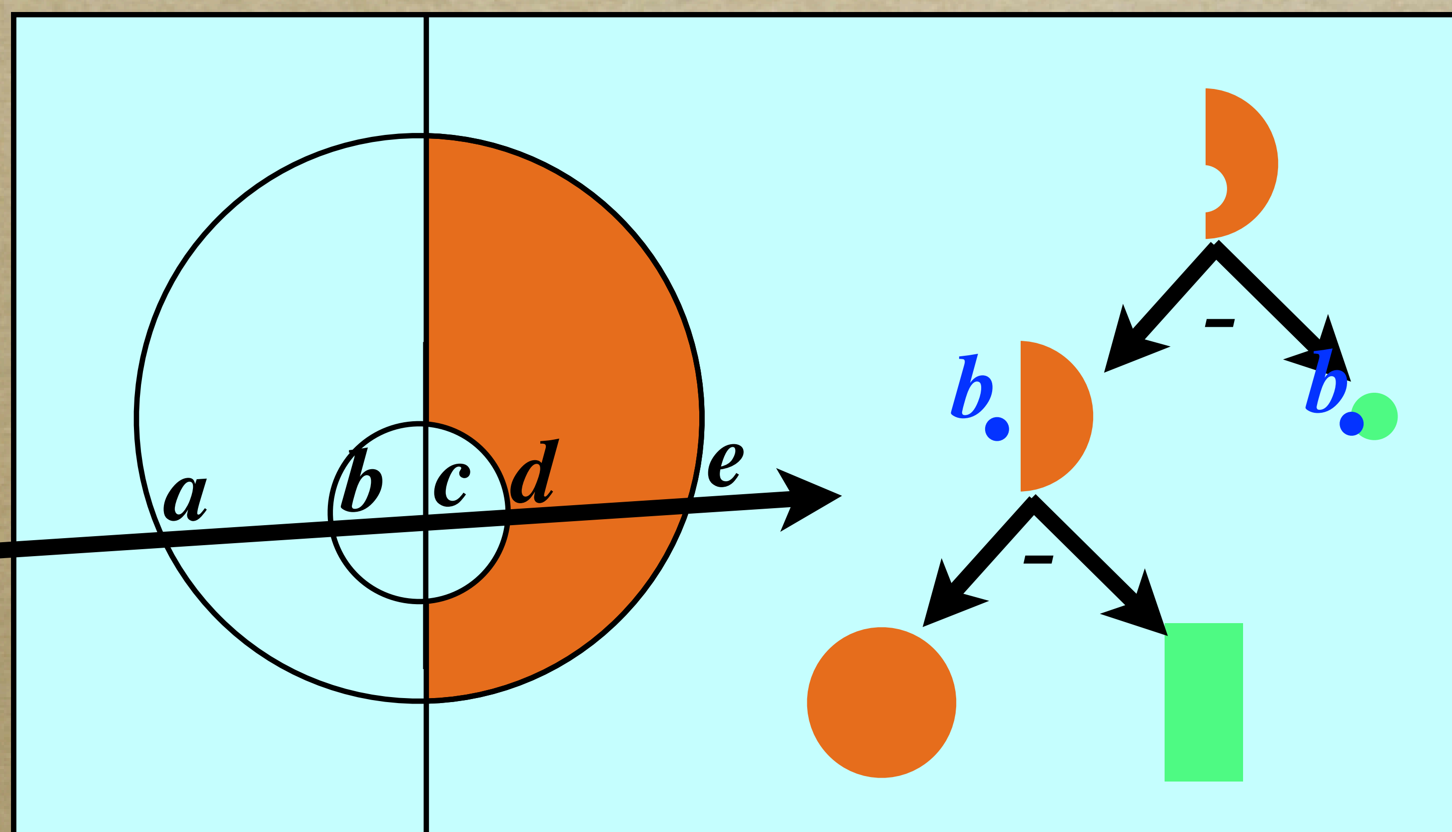


Subtract nodes for our object

		Right tree child		
		In	**Out**	**Border**
Left t.c.	**In**	*Out*	*In*	*Border*
	Out	*Out*	*Out*	*Out*
	Border	*Out*	*Border*	

Point ‘b’

… is outside the hemisphere



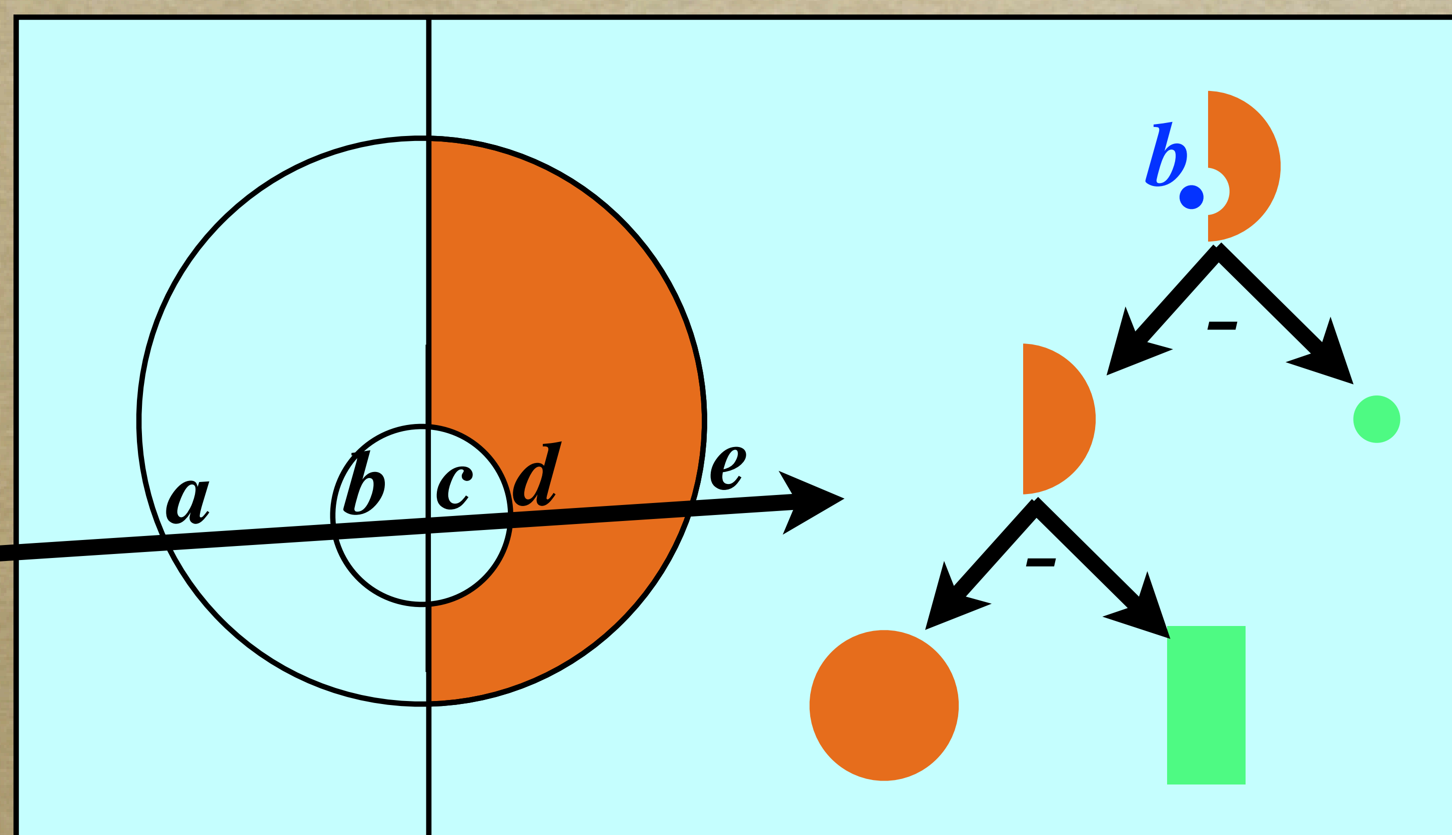
Subtract nodes for our object

		Right tree child		
		In	**Out**	**Border**
Left t.c.	**In**	*Out*	*In*	*Border*
	Out	*Out*	*Out*	*Out*
	Border	*Out*	*Border*	

Point 'b'

... is outside the hemisphere

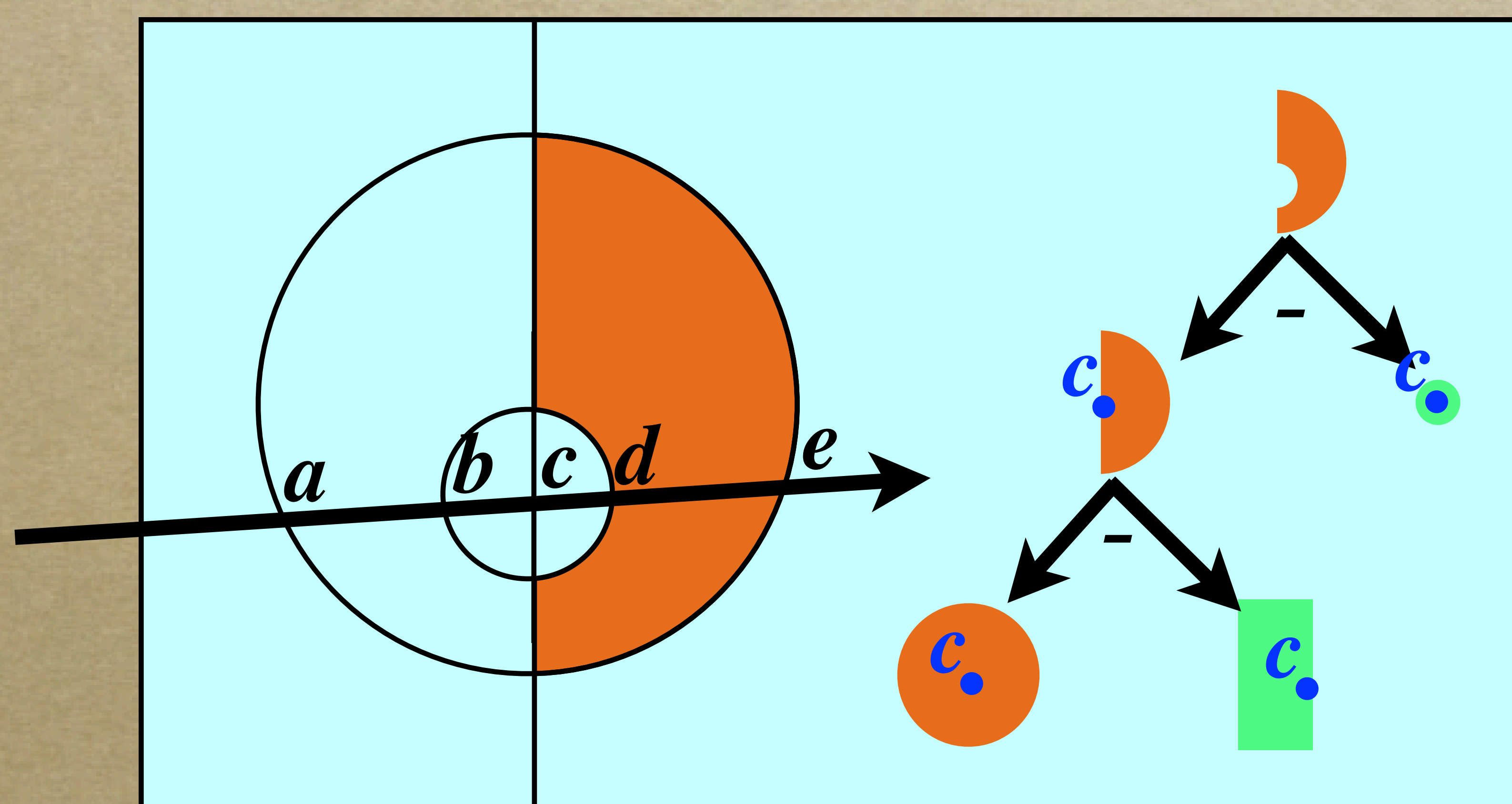
... and outside our object



Subtract nodes for our object

		Right tree child		
		In	**Out**	**Border**
Left t.c.	**In**	*Out*	*In*	*Border*
	Out	*Out*	*Out*	*Out*
	Border	*Out*	*Border*	

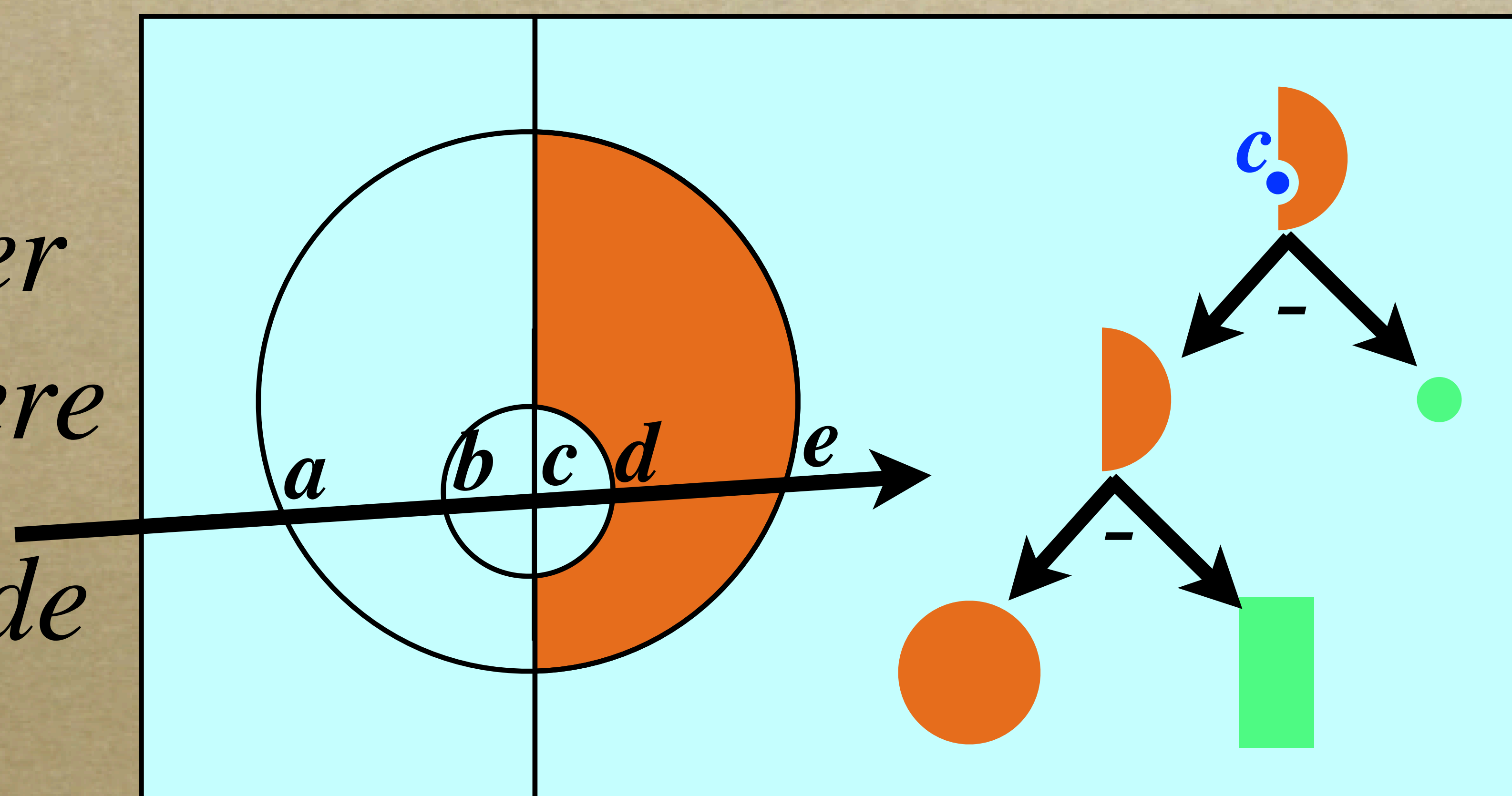
Point 'c'



Subtract nodes for our object

		Right tree child		
		In	**Out**	**Border**
Left t.c.	**In**	*Out*	*In*	*Border*
	Out	*Out*	*Out*	*Out*
	Border	*Out*	*Border*	

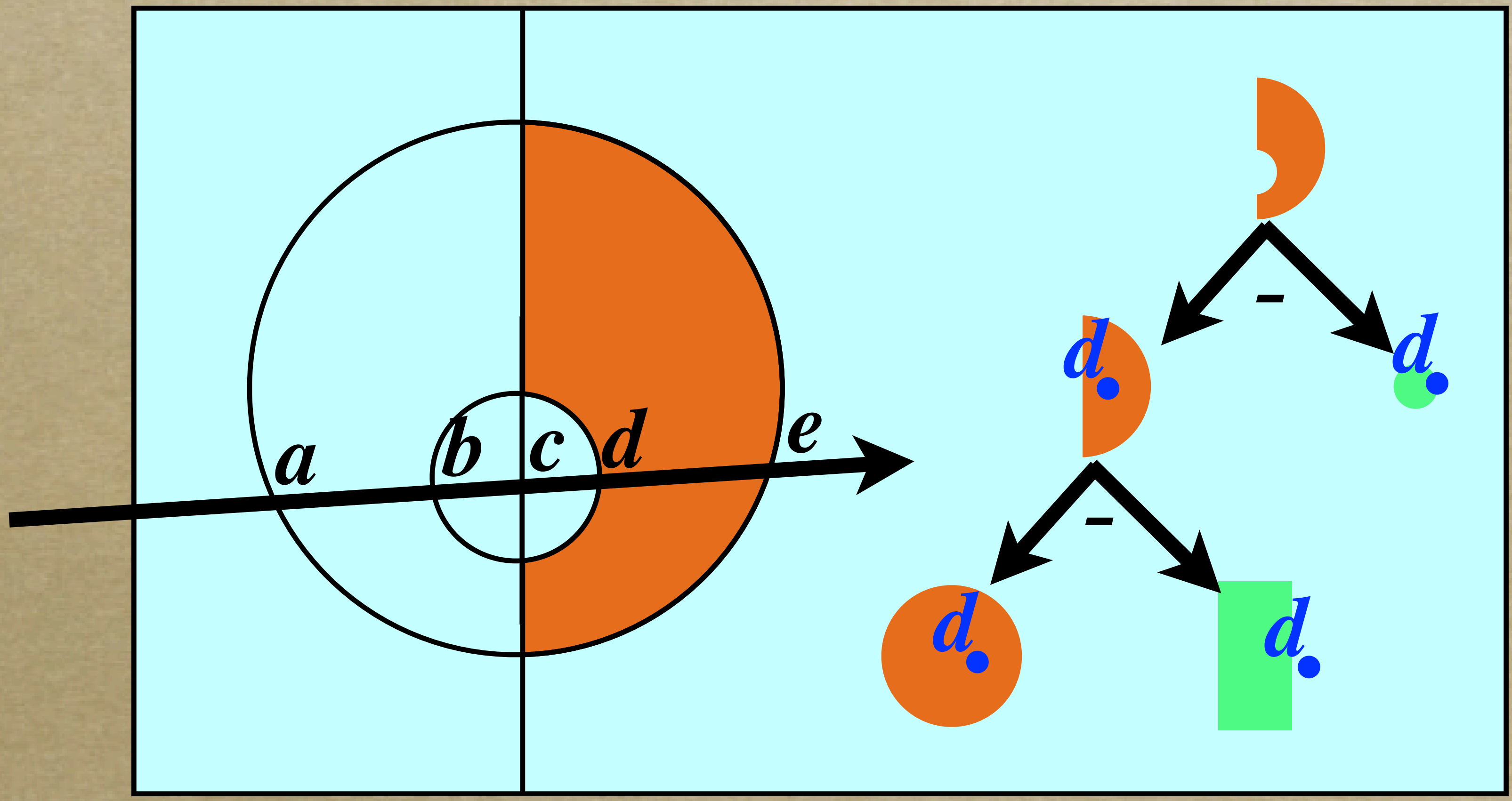
Point 'c'
… on the border of the hemisphere
… but still outside our object



Subtract nodes for our object

		Right tree child		
		In	**Out**	**Border**
Left t.c.	**In**	*Out*	*In*	*Border*
	Out	*Out*	*Out*	*Out*
	Border	*Out*	*Border*	

Point 'd'

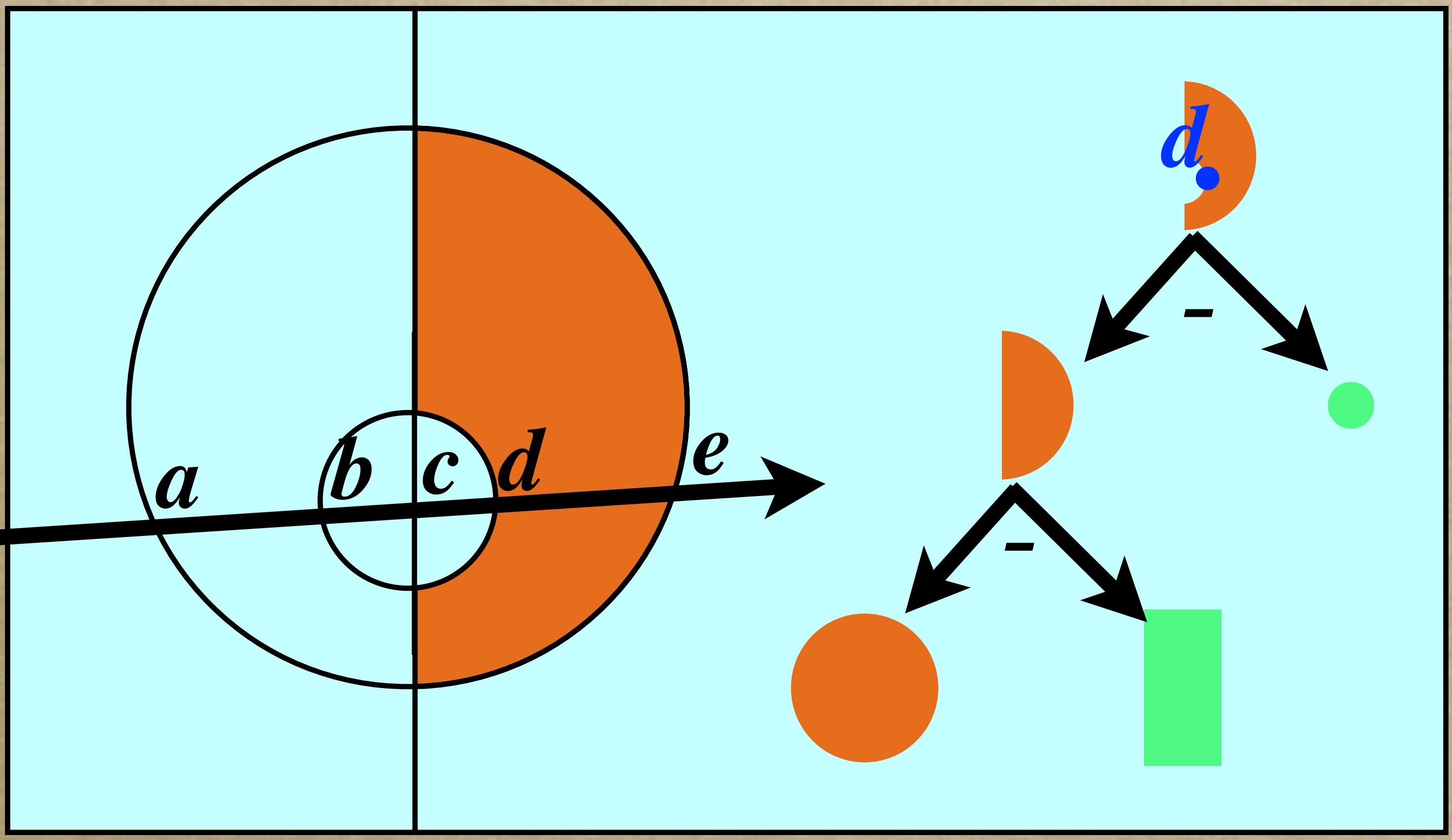


Subtract nodes for our object

		Right tree child		
		In	**Out**	**Border**
Left t.c.	**In**	*Out*	*In*	*Border*
	Out	*Out*	*Out*	*Out*
	Border	*Out*	*Border*	

Point 'd'

... is inside the hemisphere and on the border of our object.



Classification at *add* node

		Right tree child		
		In	**Out**	**Border**
Left tree child	**In**	*In*	*In*	*In*
	Out	*In*	*Out*	*Border*
	Border	*In*	*Border*	

Pros and cons

- ✓ *Create engineering objects easily*
- ✓ *Directly ray traceable*
- ✓ *Can be used for volume properties*
- ✗ *Difficult to make free form shapes*

Subdivision surfaces

- *A recursive approach to adding detail*
- *E.g. Catmull-Clark mesh subdivison: (steps are only sketched out here)*
  - *Add a point at centroid of each face*
  - *Add new 'average' edge points*
  - *'Smooth out' original vertices*
- *In the limit, a cube becomes a sphere*

Implicit surfaces

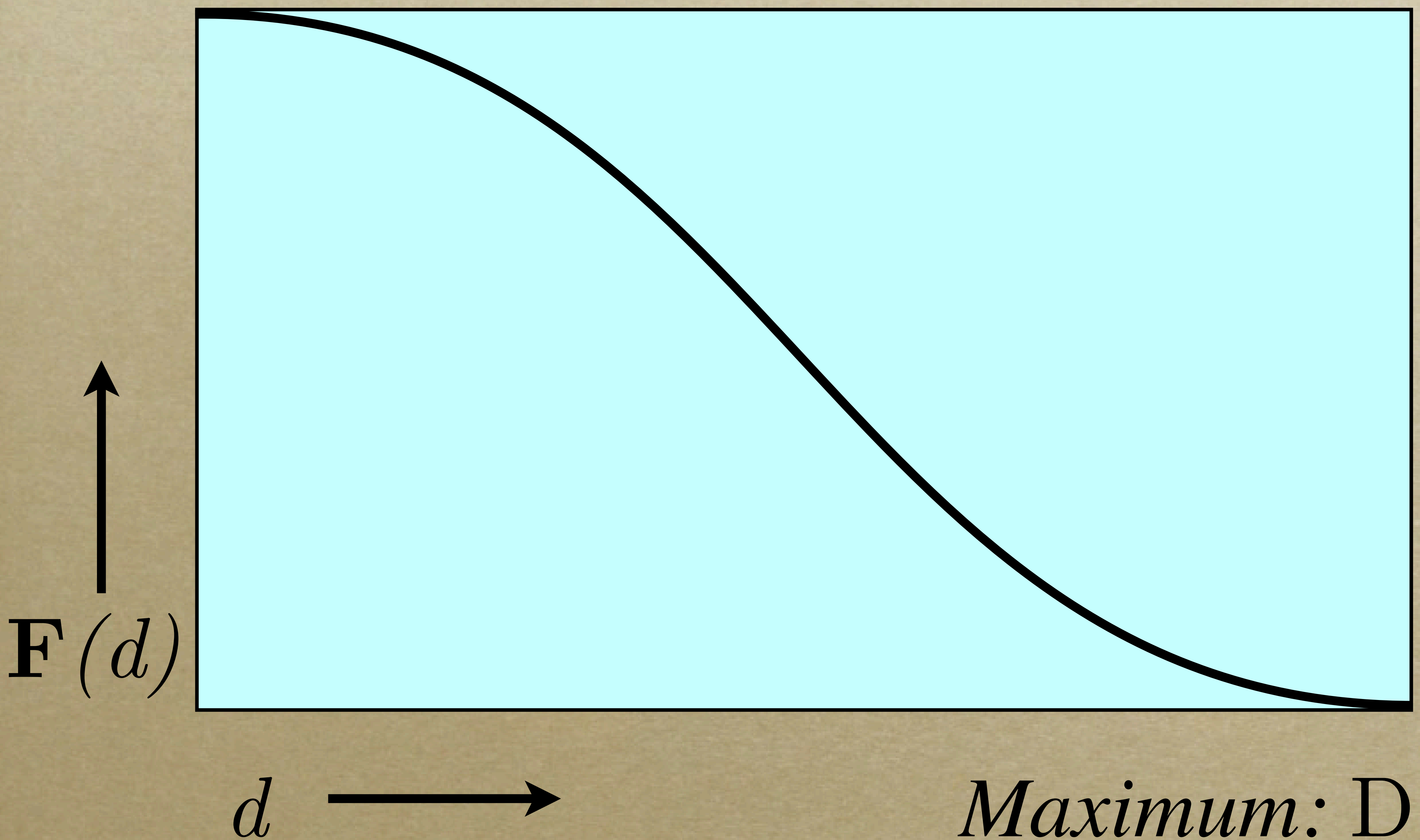
Instead of $\mathbf{p} = (\mathrm{F_x}(s,t),\ \mathrm{F_y}(s,t),\ \mathrm{F_z}(s,t))$

use $\mathbf{p}$ *such that* $\mathbf{F}(\mathbf{p}) = \mathrm{k}$

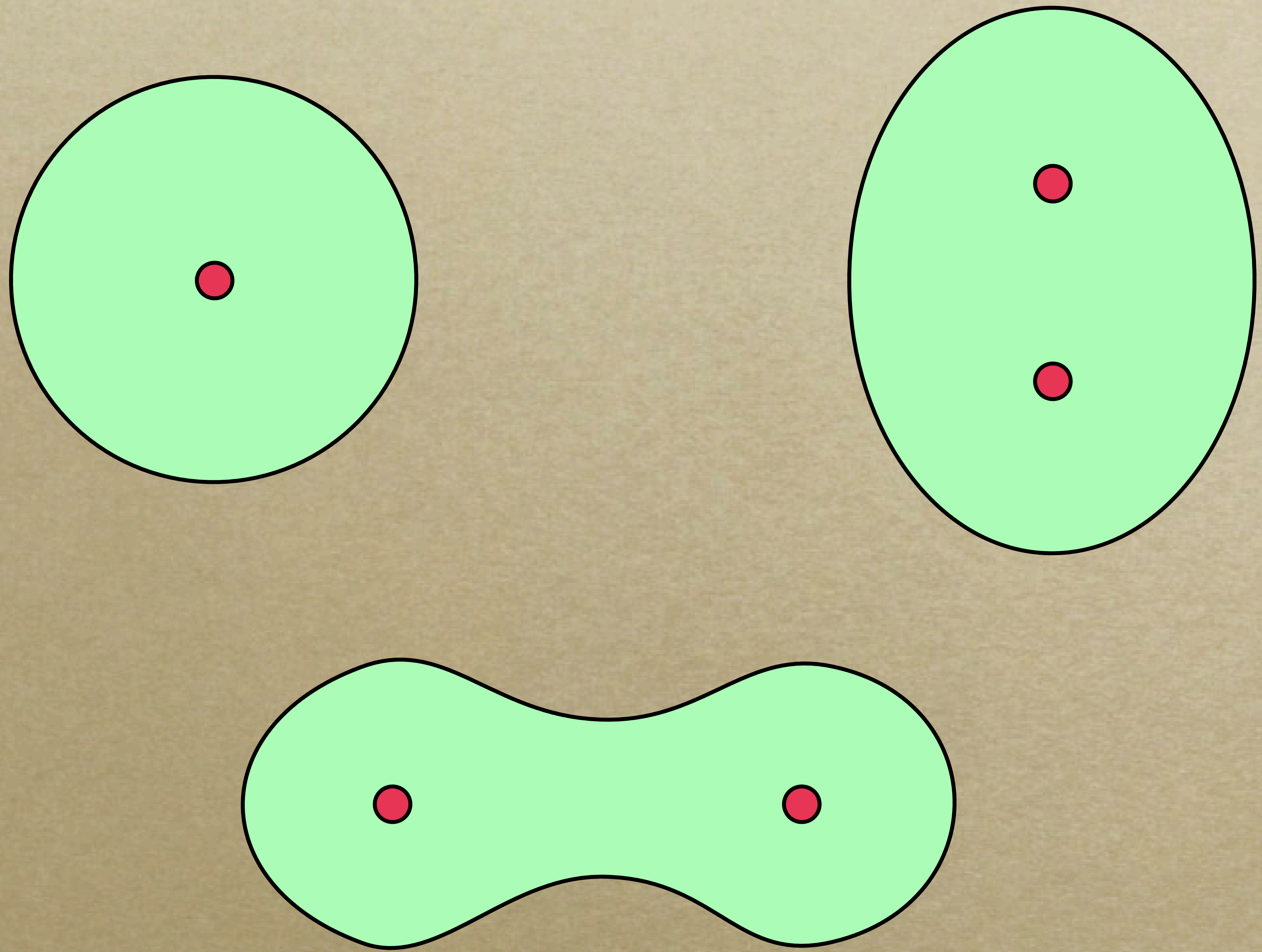
E.g: $\mathbf{p}^2 = 1$ *is a sphere*

Typically make $\mathbf{F}$ *a function of distance*

Field function

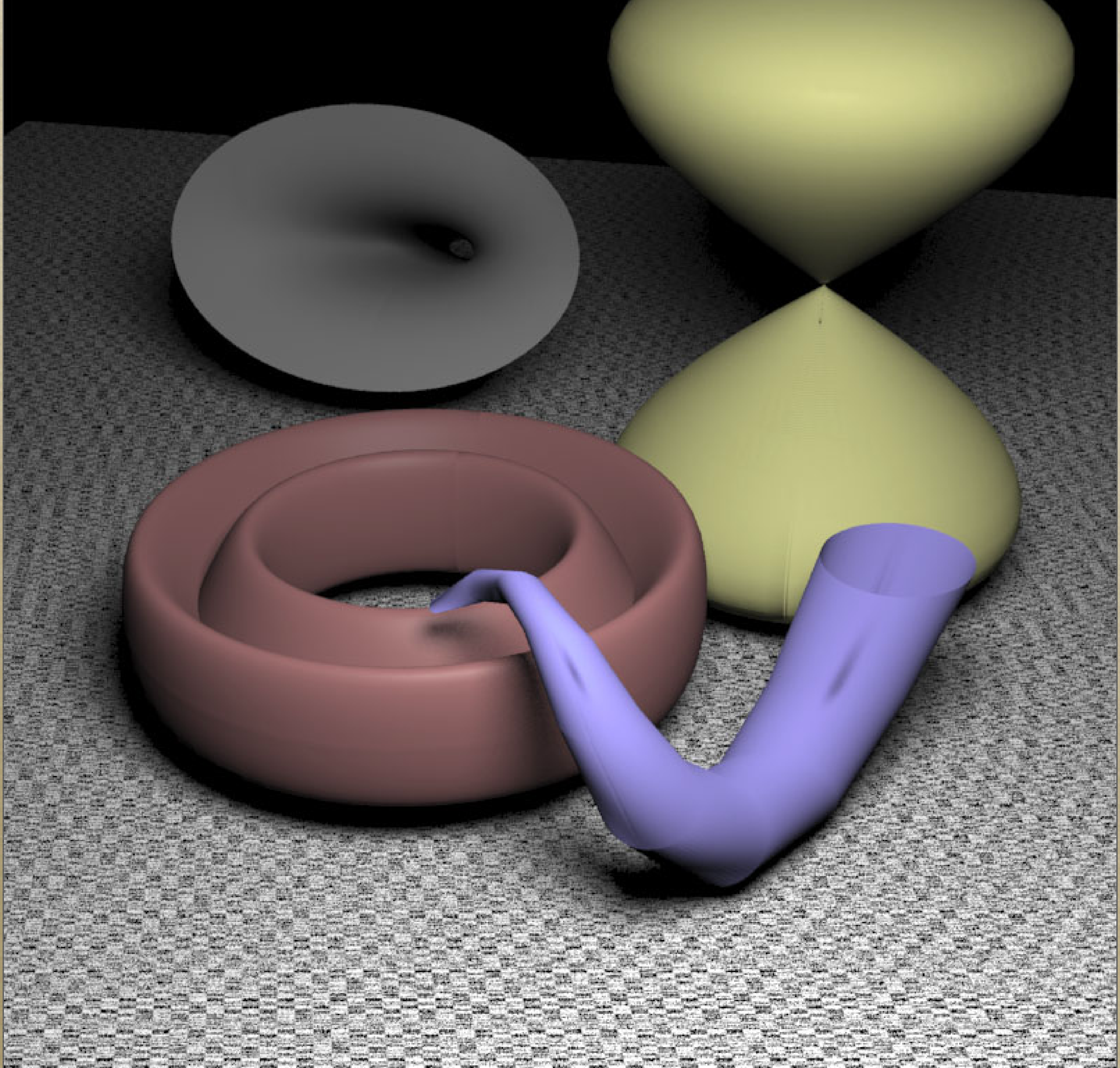


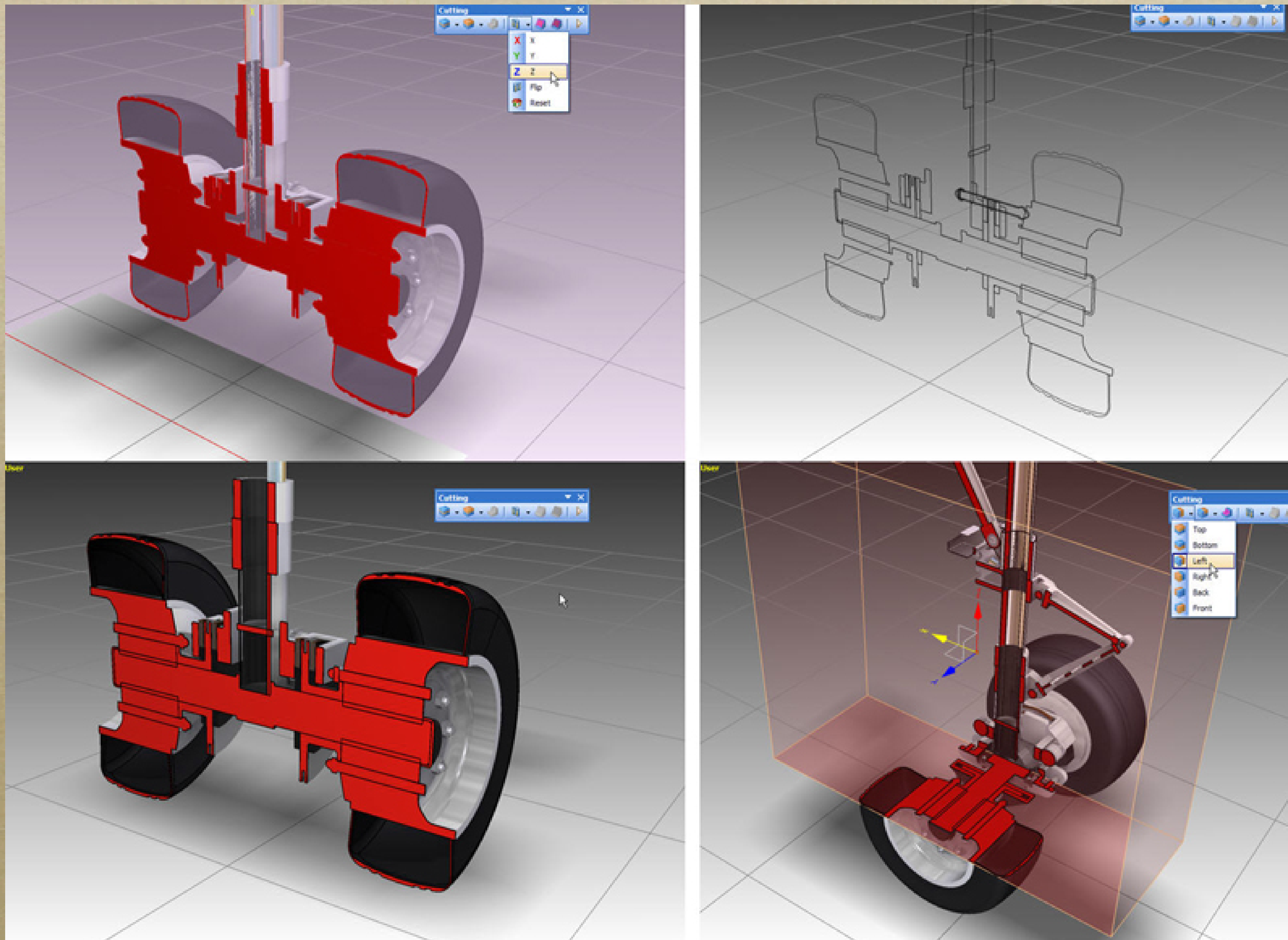
Skeleton of points

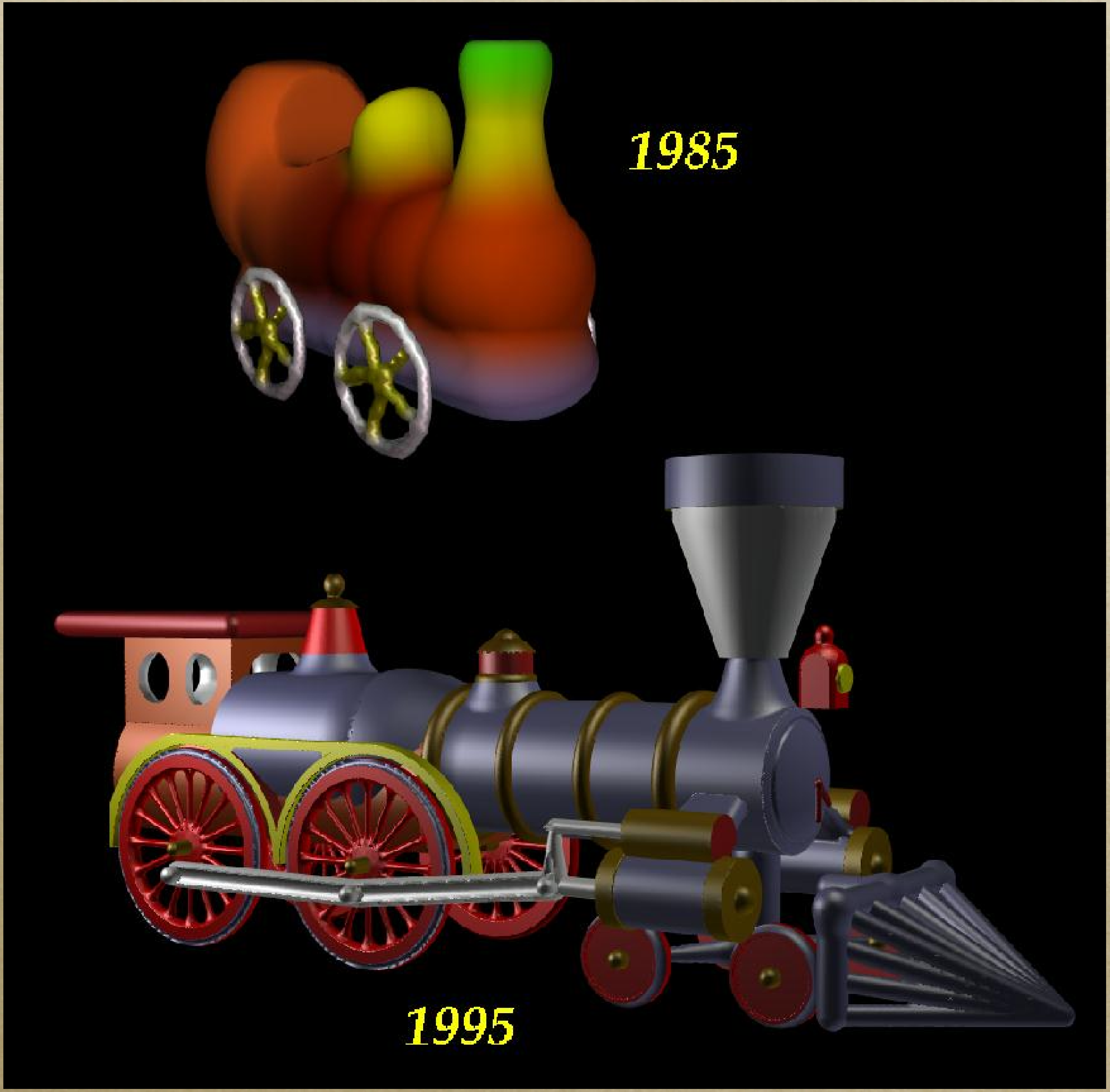


Pros and cons

- ✓ *Intuitive construction*
- ✓ *guaranteed inside/outside - no holes*
- ✓ *ray traceable*
- ✓ *change topology*
- ✗ *no obvious way to draw surface*
- ✗ *design seems to be by black art*




Cutting
X
Y
Z
Flip
Reset
Cutting
User
Cutting
User
Cutting
Top
Bottom
Left
Right
Back
Front


1985
1995