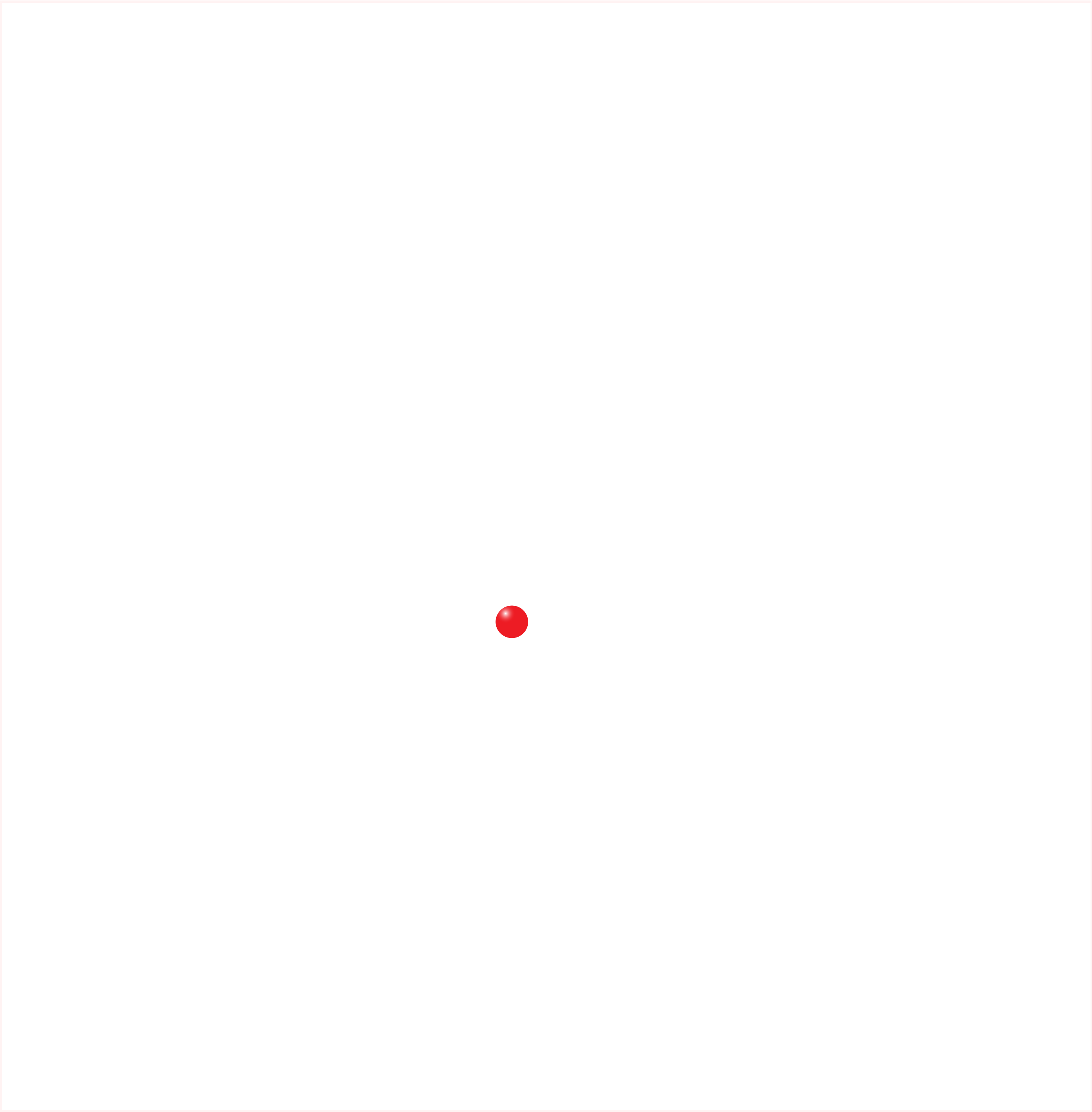
generative components theoretical frameworks

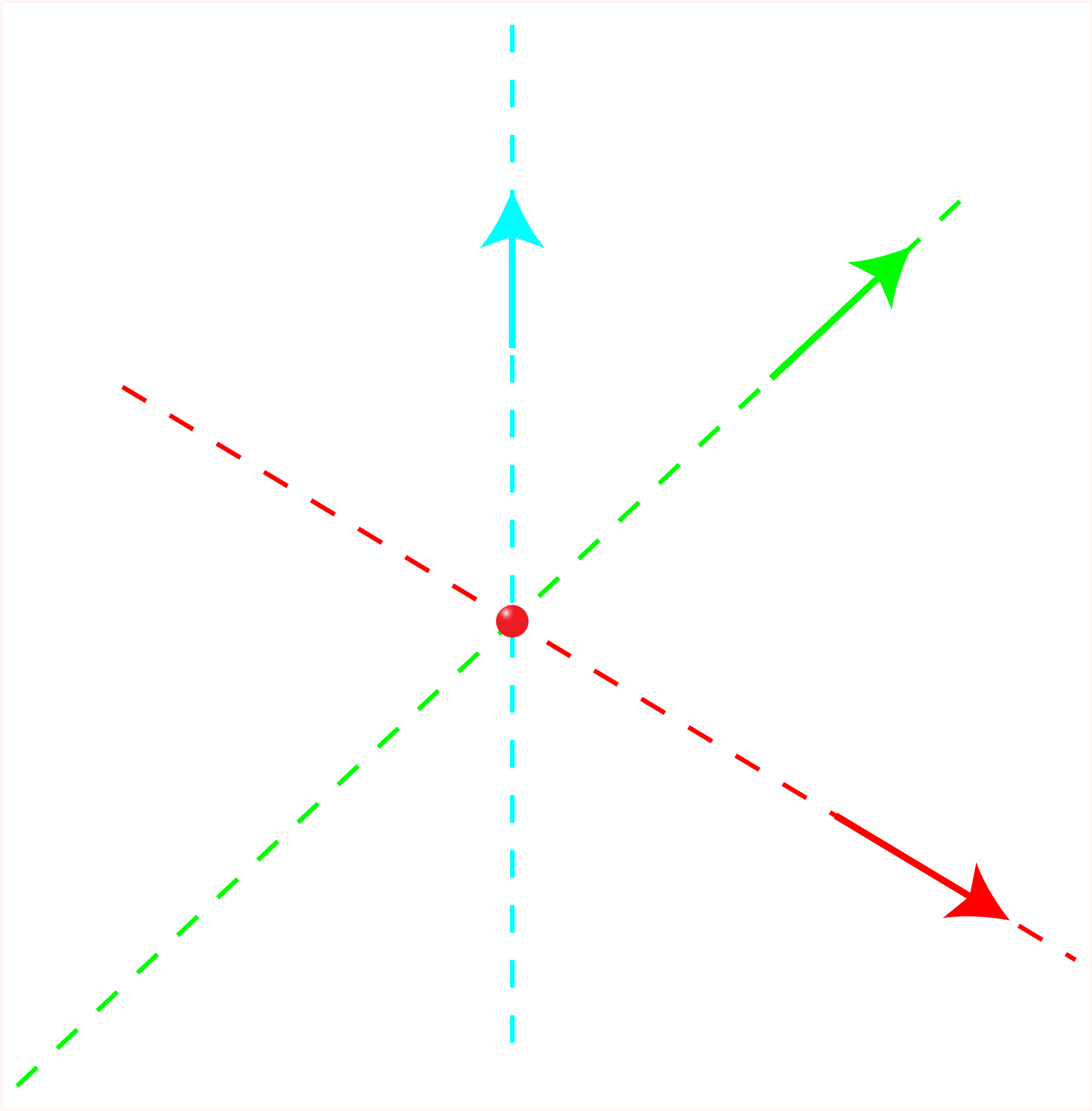
the stuff you *need* to know

grounding in space



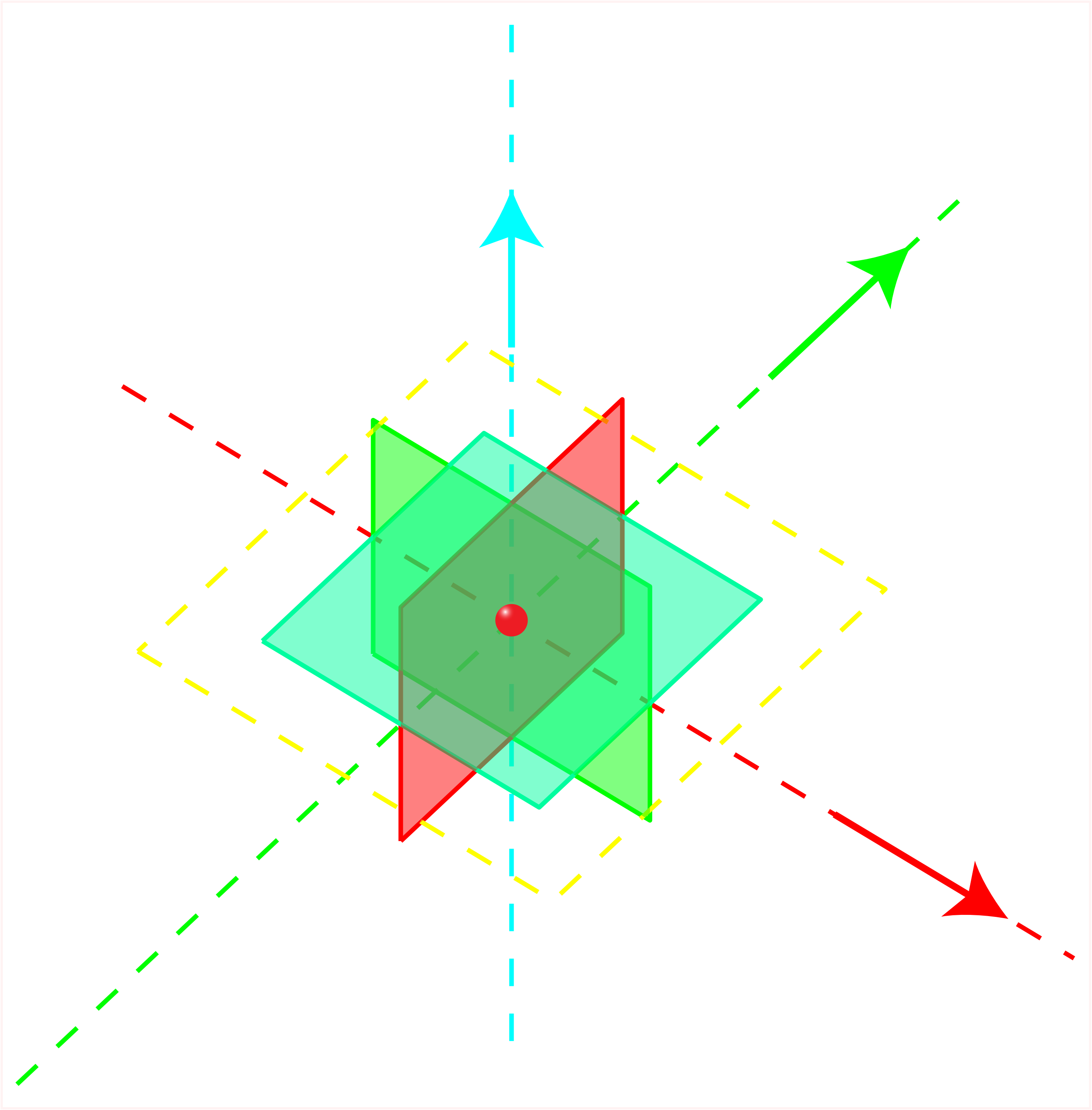
Theoretical geometry gives us an infinite universe to work in. There is no concept of up, or of where we are in a finite sense.

To make this work for us, we pick a spot an call it the 'origin'



The way of describing space that is most common is the Cartesian grid ({x,y,z} triples)

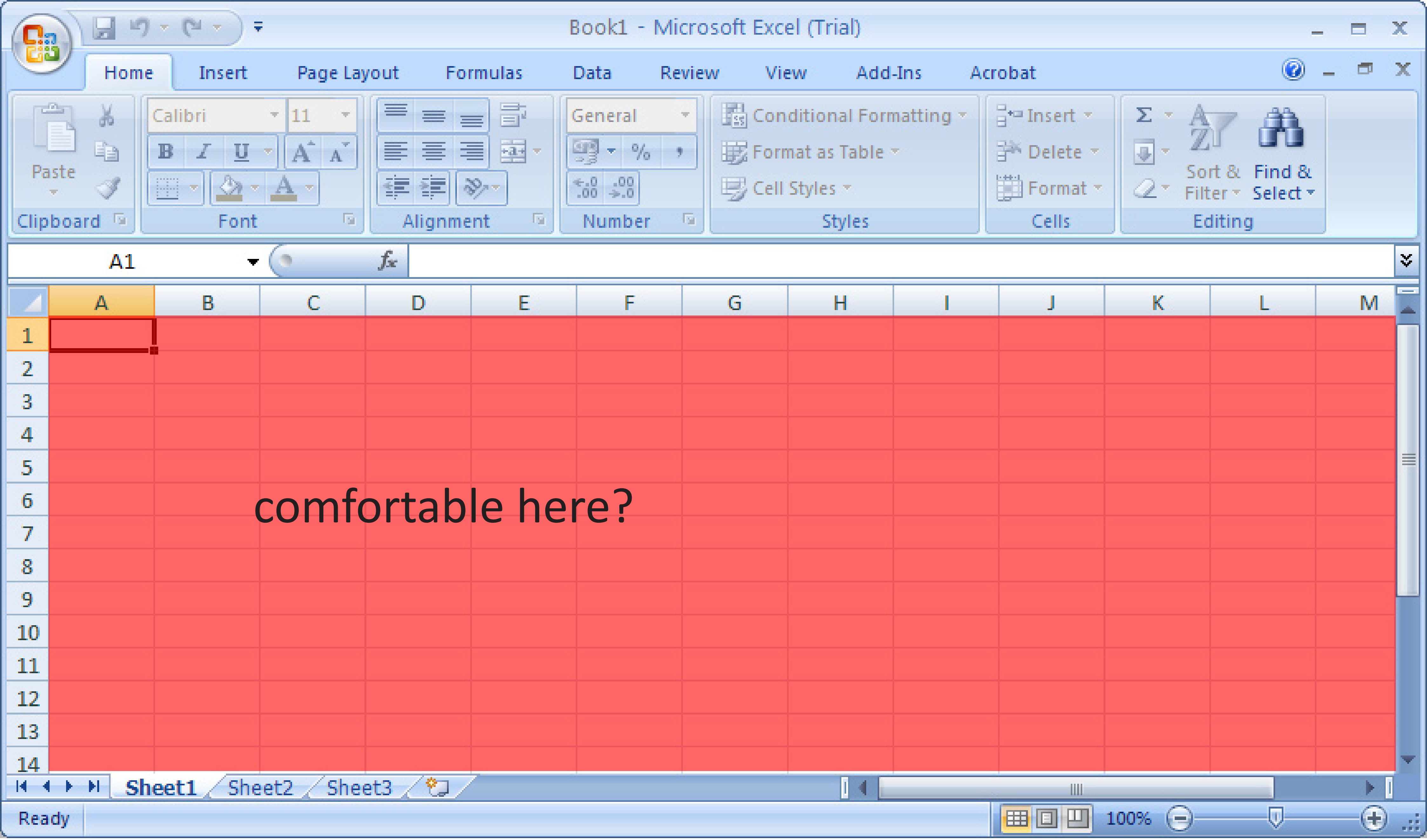
The positive part of the Z axis can be considered ‘up’ generally.



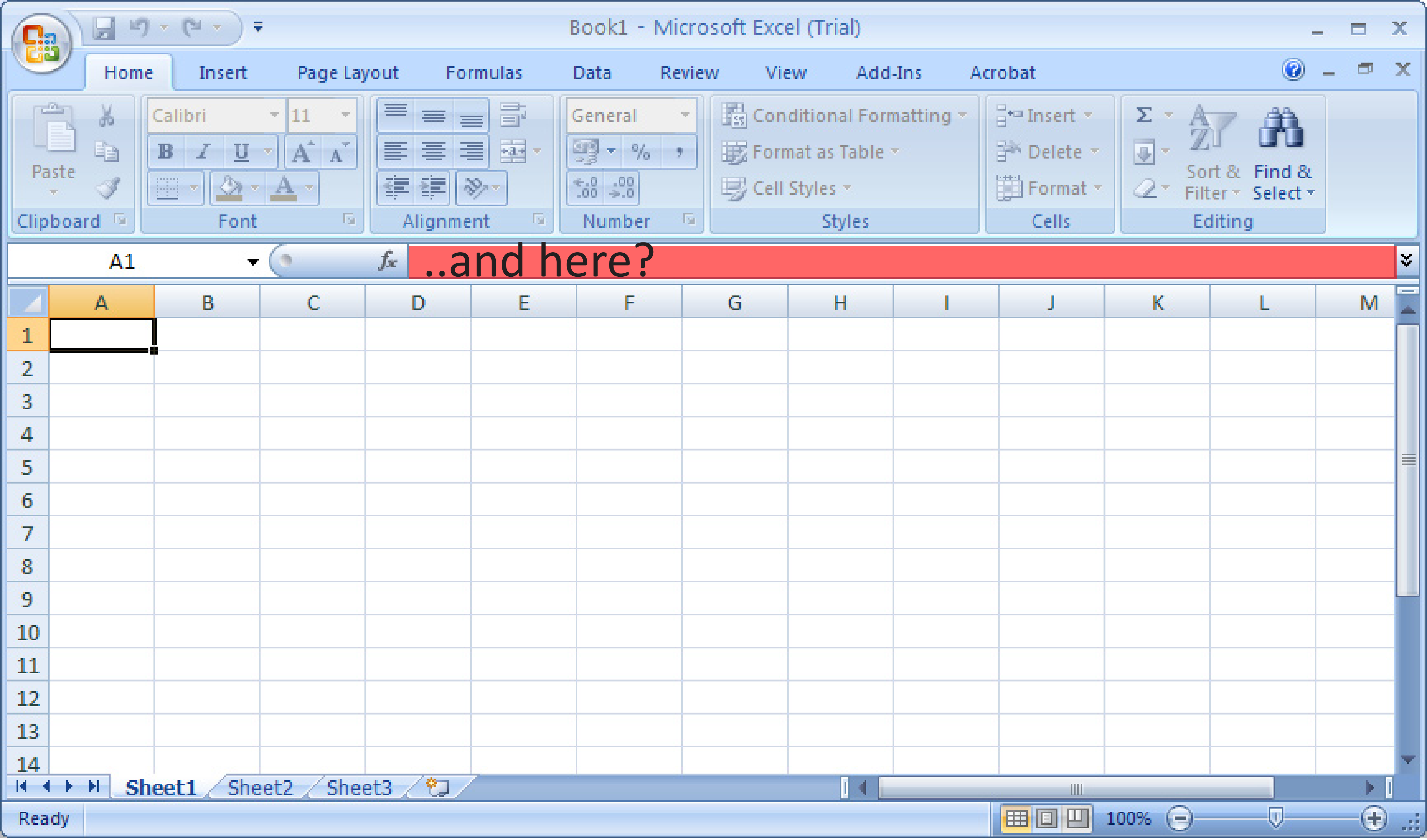
As we have 3 axis and an origin that defines 3 planes that are all at 90 degrees to each other.

(The Yellow line indicates the currently active plane)

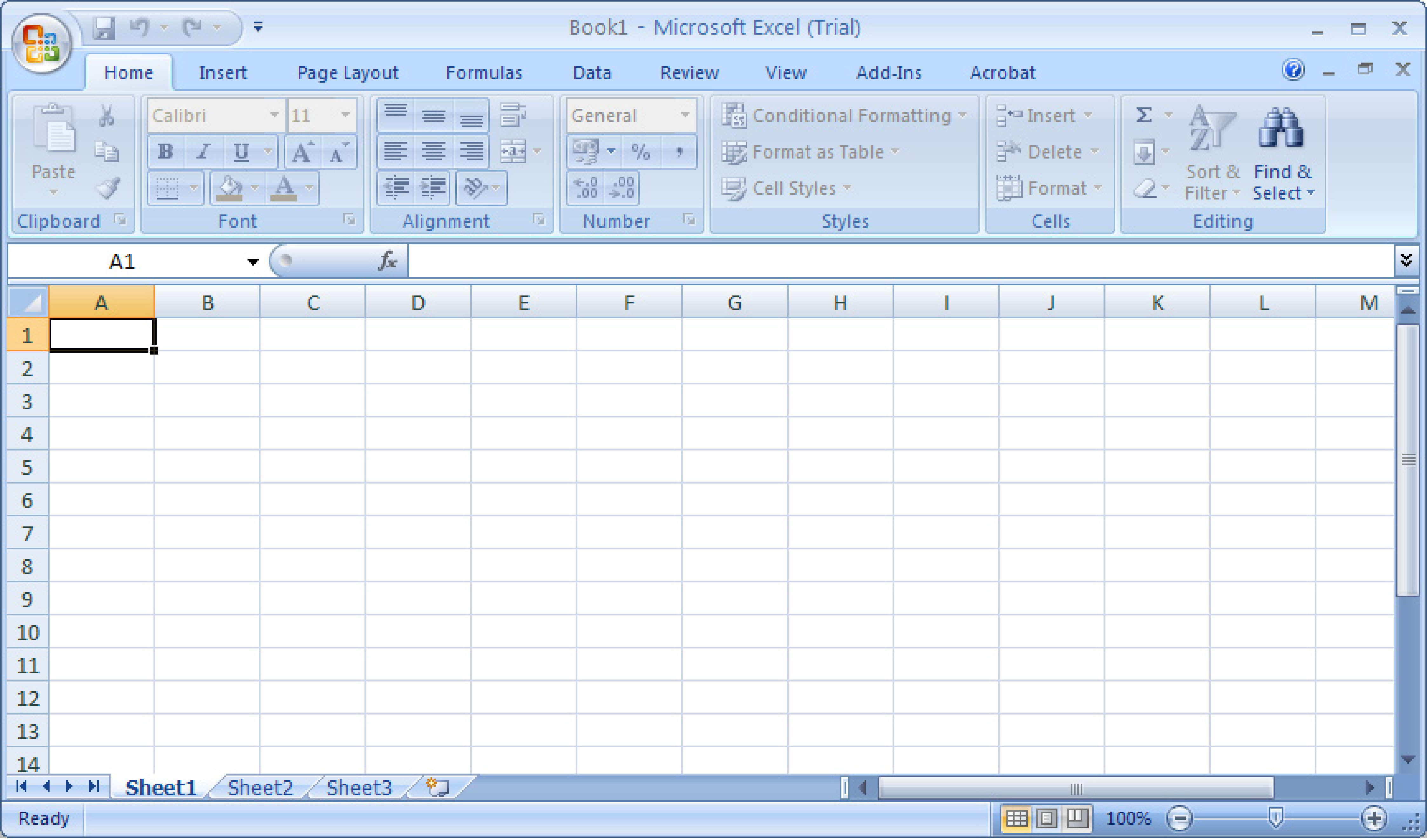
inputs



comfortable here?

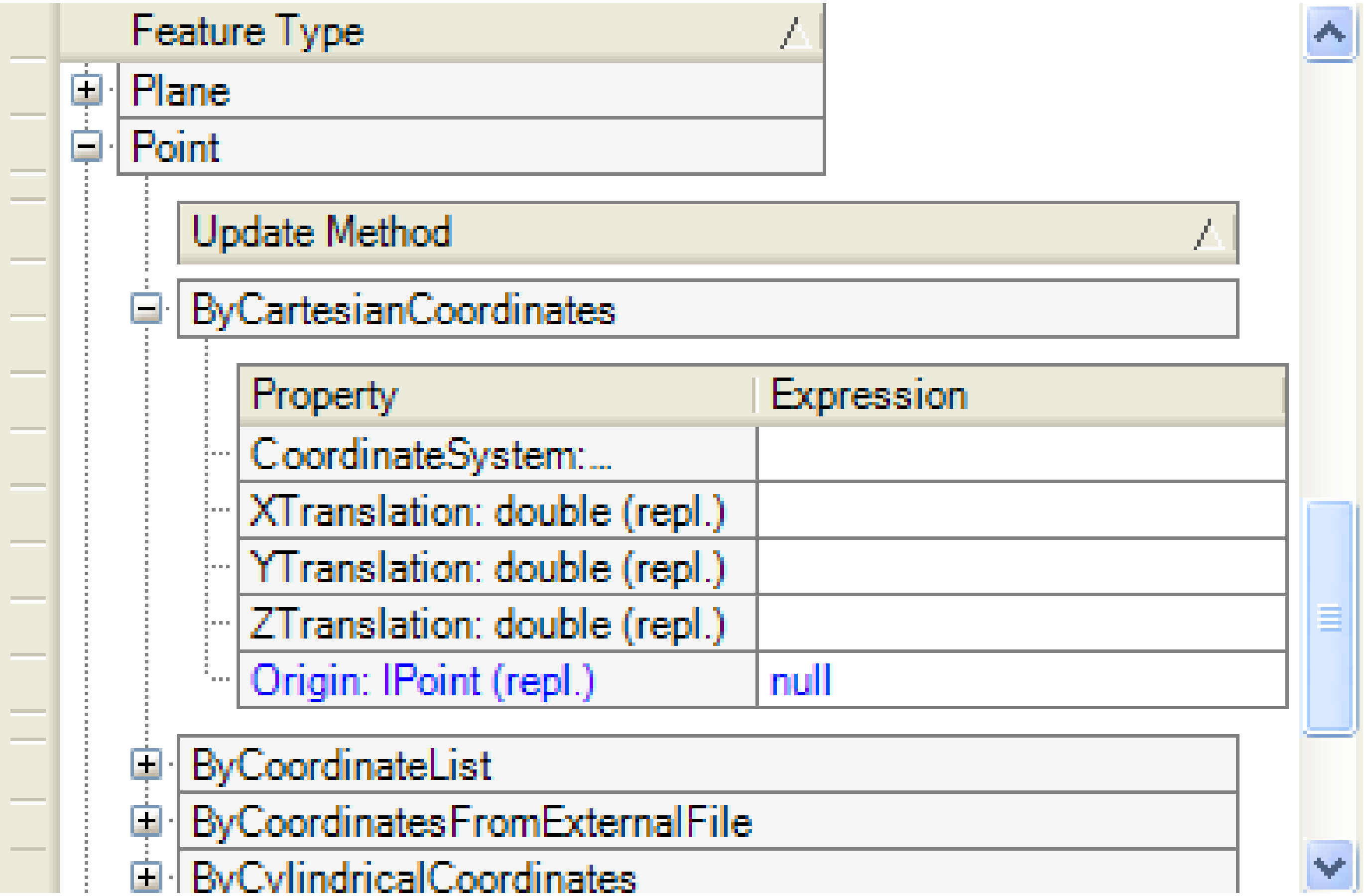

Book1 - Microsoft Excel (Trial)
Home
Insert
Page Layout
Formulas
Data
Review
View
Add-Ins
Acrobat
Paste
Clipboard
Calibri
11
Font
Alignment
General
Number
Conditional Formatting
Format as Table
Cell Styles
Styles
Insert
Delete
Format
Cells
Sort & Filter
Find & Select
Editing
A1
A
B
C
D
E
F
G
H
I
J
K
L
M
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Sheet1
Sheet2
Sheet3
Ready
100%

..and here?


Book1 - Microsoft Excel (Trial)
Home
Insert
Page Layout
Formulas
Data
Review
View
Add-Ins
Acrobat
Paste
Calibri
11
General
Conditional Formatting
Format as Table
Cell Styles
Insert
Delete
Format
Sort & Filter
Find & Select
Clipboard
Font
Alignment
Number
Styles
Cells
Editing
A1
A B C D E F G H I J K L M
Sheet1
Sheet2
Sheet3
Ready
100%

thought so!

so what about here?



Feature Type

Plane

Point

Update Method

ByCartesianCoordinates

Property	Expression
CoordinateSystem:...	
XTranslation: double (repl.)	
YTranslation: double (repl.)	
ZTranslation: double (repl.)	
Origin: IPoint (repl.)	null

ByCoordinateList

ByCoordinatesFromExternalFile

ByCylindricalCoordinates

they are pretty much the same!

pretty much anything can go into this box

5

nice and easy

single values are
easy to understand

5

nice and easy

5+2

still easy

very simple
equations are easy
too

5	nice and easy
5+2	still easy
Sin(5)	getting scary

scientific calculator stuff can be found in the function list

f_x

5	nice and easy
5+2	still easy
Sin(5)	getting scary
(1/Sin(5))+90	pretty frightening

compound statements follow BODMAS, no magic here folks.

B	Brackets first
O	Orders (ie Powers and Square Roots, etc.)
DM	Division and Multiplication (left-to-right)
AS	Addition and Subtraction (left-to-right)

$6 \times (5 + 3) \quad = \quad 6 \times 8 \quad = \quad 48$ ✓

$6 \times (5 + 3) \quad = \quad 30 + 3 \quad = \quad 33$ ✗

$6 \times 5 + 3 \quad = \quad 30 + 3 \quad = \quad 33$ ✓

5	nice and easy
5+2	still easy
Sin(5)	getting scary
(1/Sin(5))+90	pretty frightening
dave	eh? must be a variable

5	nice and easy
5+2	still easy
Sin(5)	getting scary
(1/Sin(5))+90	pretty frightening
dave	eh? must be a variable

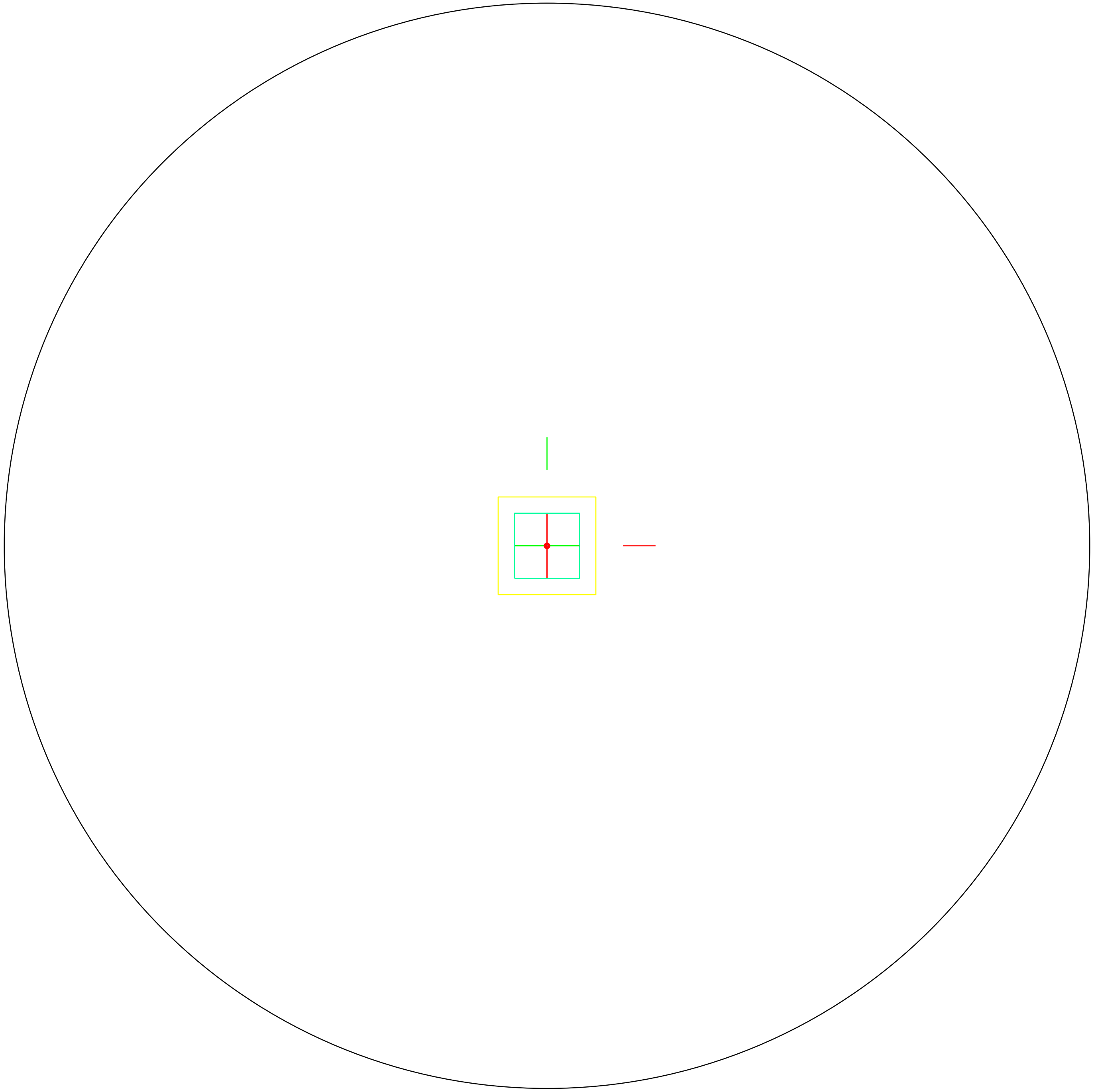
dave = 8

once a variable is defined (named) it can be used in place of a value

5	nice and easy
5+2	still easy
Sin(5)	getting scary
(1/Sin(5))+90	pretty frightening
dave	eh? must be a variable
dave*2	simple again

```
dave = 8
```

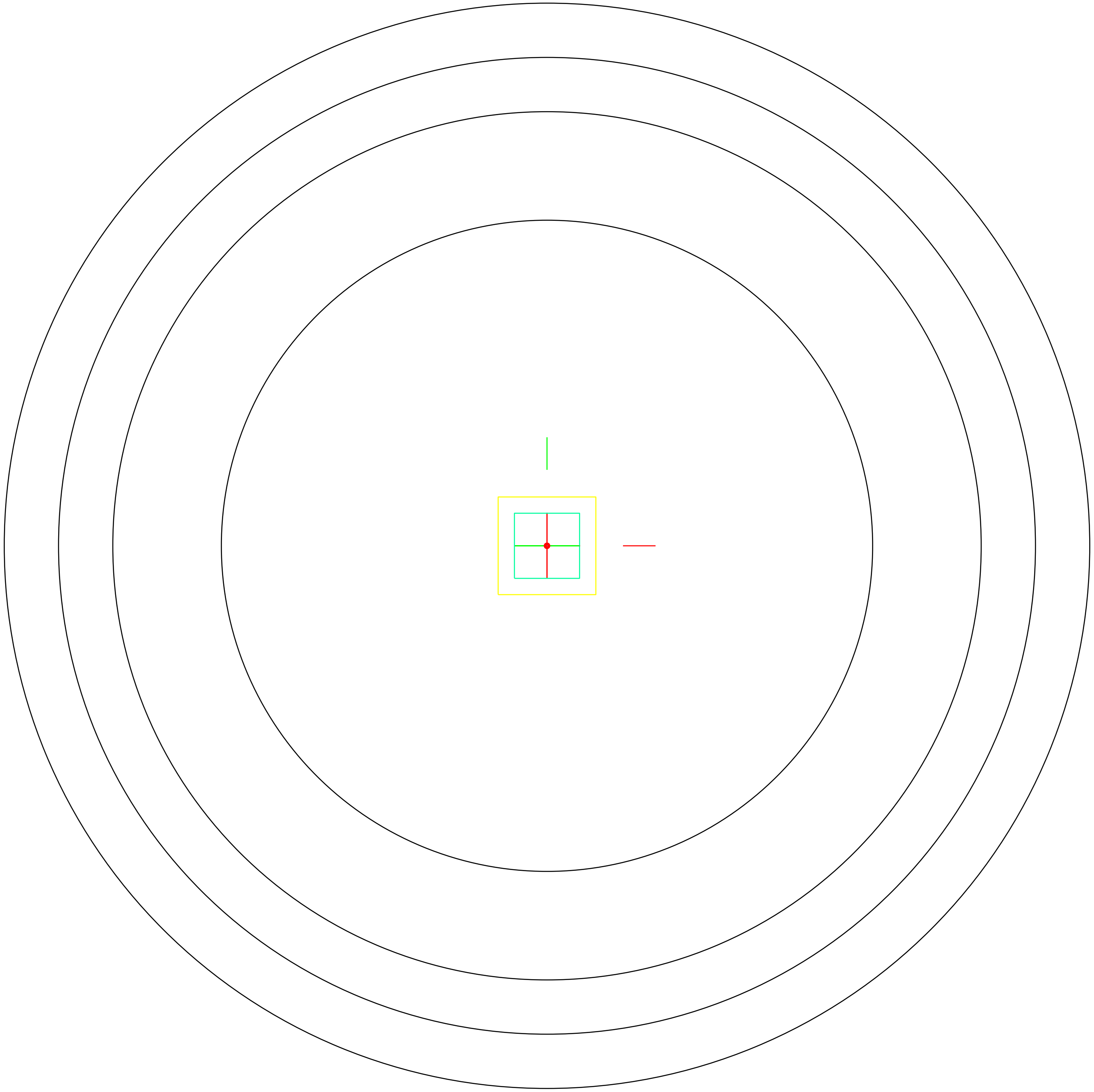
once a variable is defined (named) it can be used in place of a value



This circle’s radius is defined using a single value.

That’s how you’d expect it to work from experience.

5



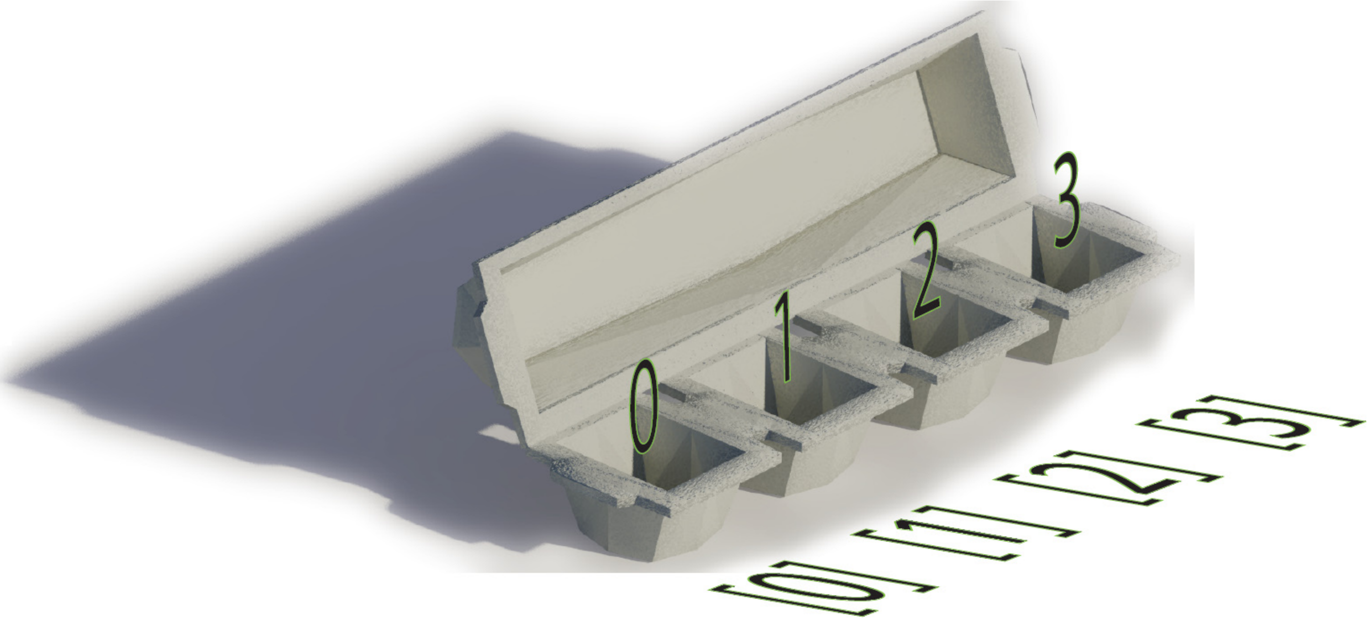
This circle’s radius is defined using a *list*.

Lists are really where the power of GC kicks in.

{3, 4, 4.5, 5}

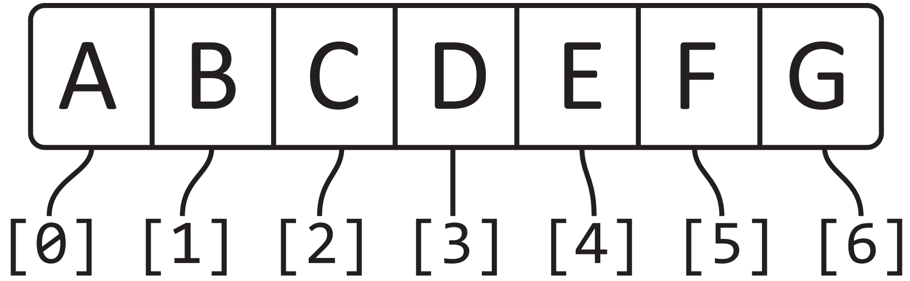
{ , , , }

Type ‘**Curly Braces**’ to define a list.



Things in a list are *indexed* from 0

dave =



A B C D E F G

[0] [1] [2] [3] [4] [5] [6]

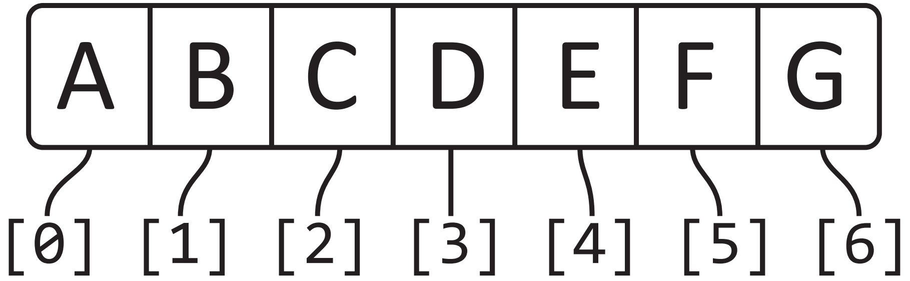
dave[4] =

If we *declare* a variable called ‘dave’ as a list having the contents {A,B,C,D,E,F}

we can refer to the contents of that list individually by their index.

remember to count indices from 0

dave =



A	B	C	D	E	F	G
[0]	[1]	[2]	[3]	[4]	[5]	[6]

dave[4] = ‘E’

If we *declare* a variable called ‘dave’ as a list having the contents {A,B,C,D,E,F}

we can refer to the contents of that list individually by their index.

remember to count indices from 0

walking up
and down the
dimensional
ladder

"O day and night, but this is wondrous strange"

FLATLAND

A ROMANCE OF MANY DIMENSIONS

By A Square

(Edwin A. Abbott)

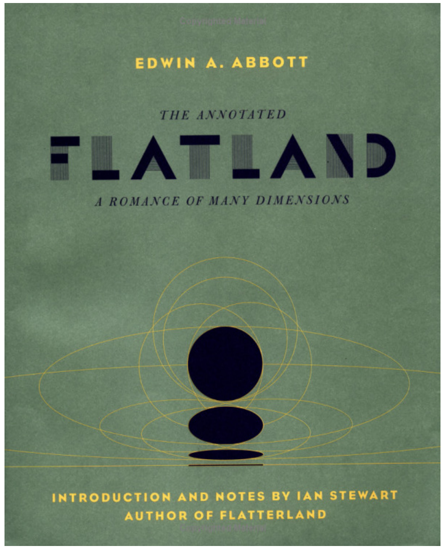
"And therefore as a stranger give it welcome."

BASIL BLACKWELL · OXFORD

Price Seven Shillings and Sixpence net

having a read of Edwin A. Abbott's Flatland will make all this dimensionality stuff much simpler to understand.

I'd recommend the annotated version.



EDWIN A. ABBOTT

THE ANNOTATED

FLATLAND

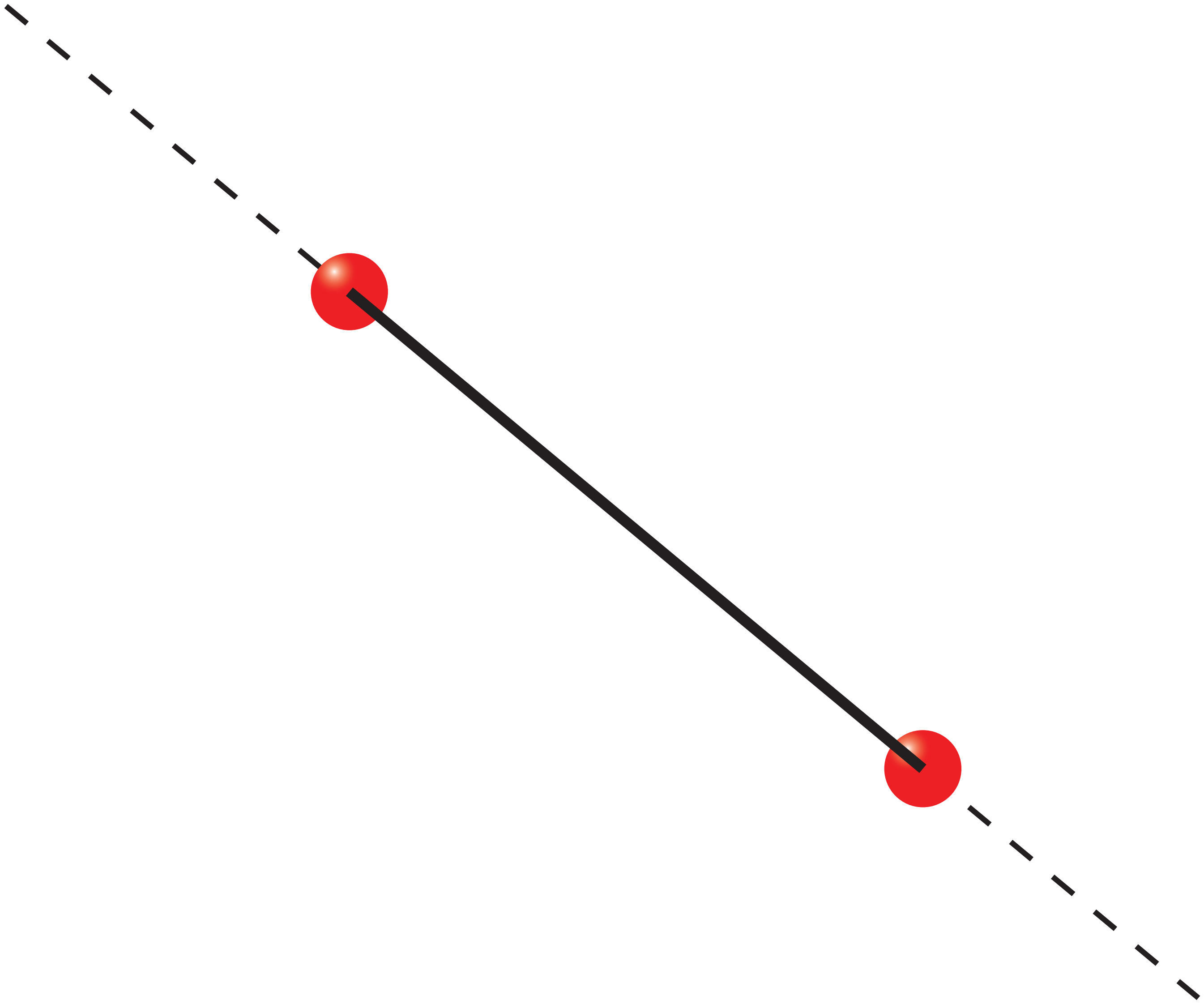
A ROMANCE OF MANY DIMENSIONS

INTRODUCTION AND NOTES BY IAN STEWART
AUTHOR OF FLATTERLAND

points

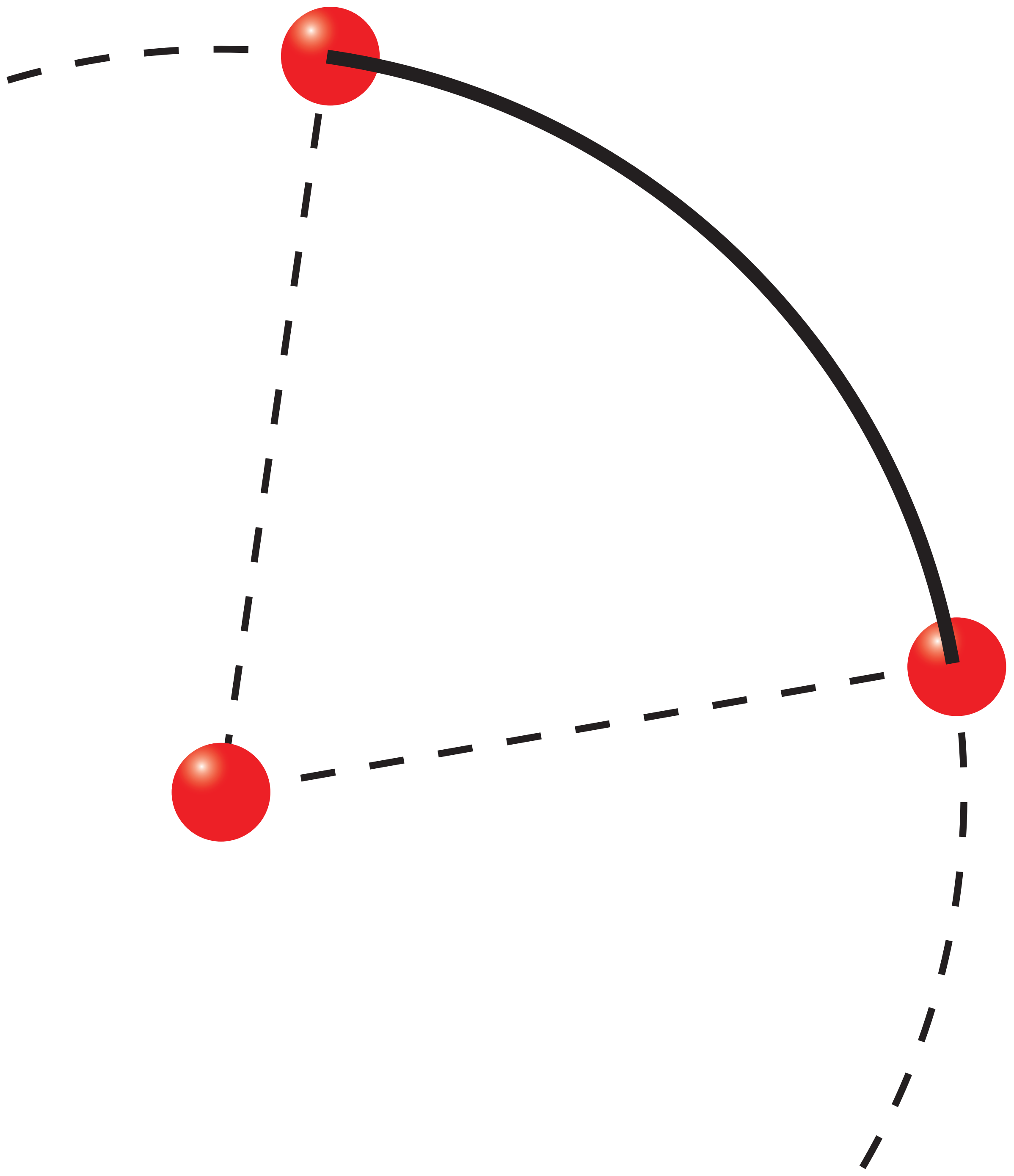
points are 0 dimensional.

They have no size, volume, nothing



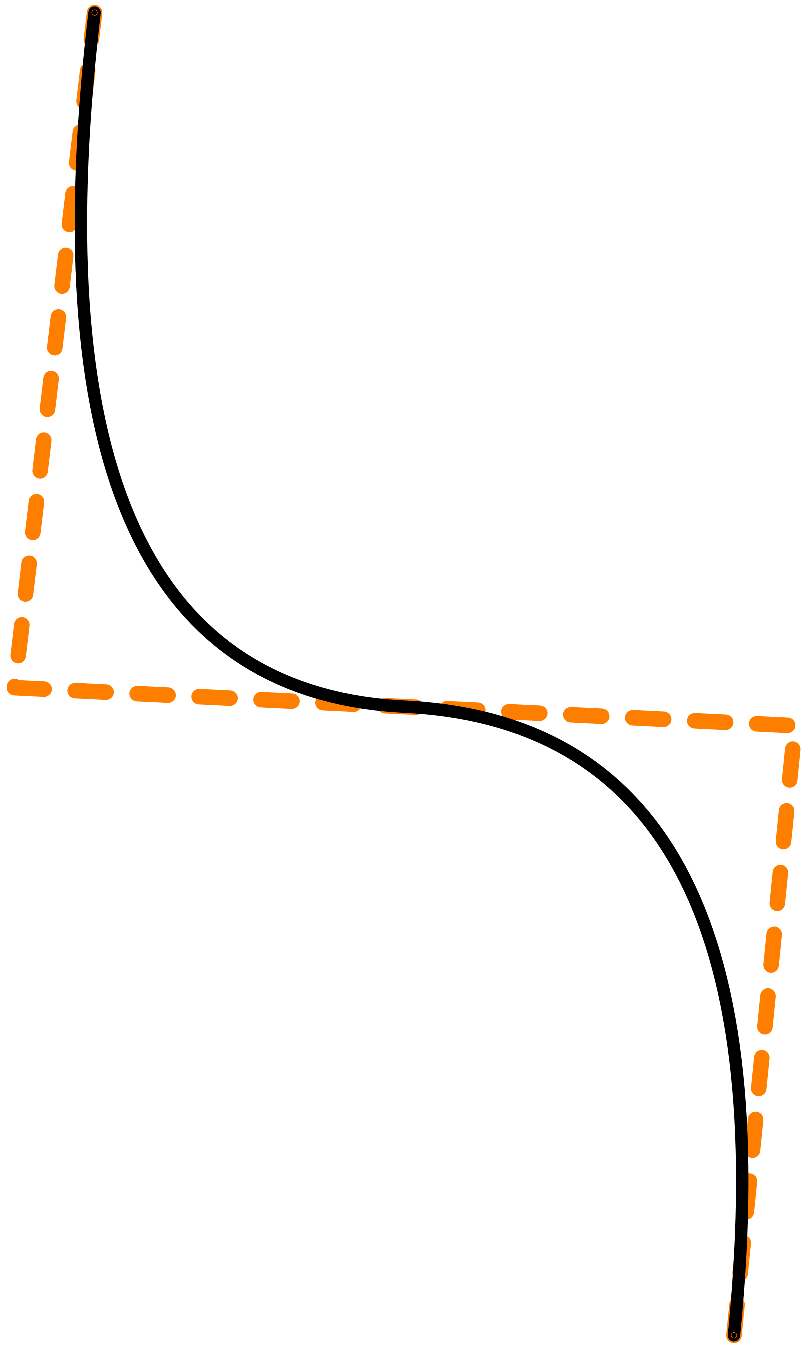
If we have 2 points, there is a line that runs through them.

Strictly *lines* are infinite, and *line segments* are bounded, but common usage means that we refer to bounded segments as lines.

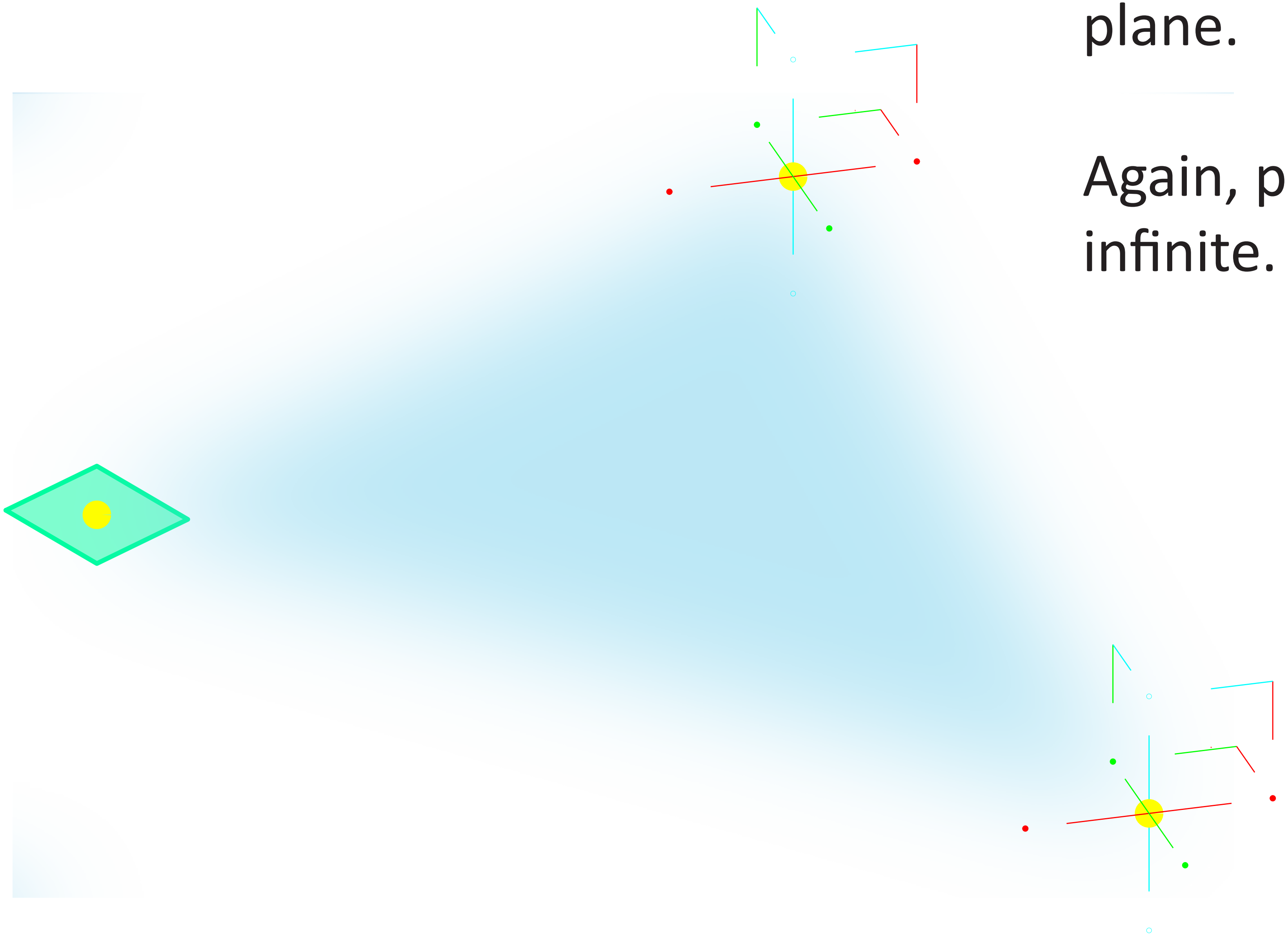


arcs and circles
are bit more
complicated to
define.

..and so it begins...

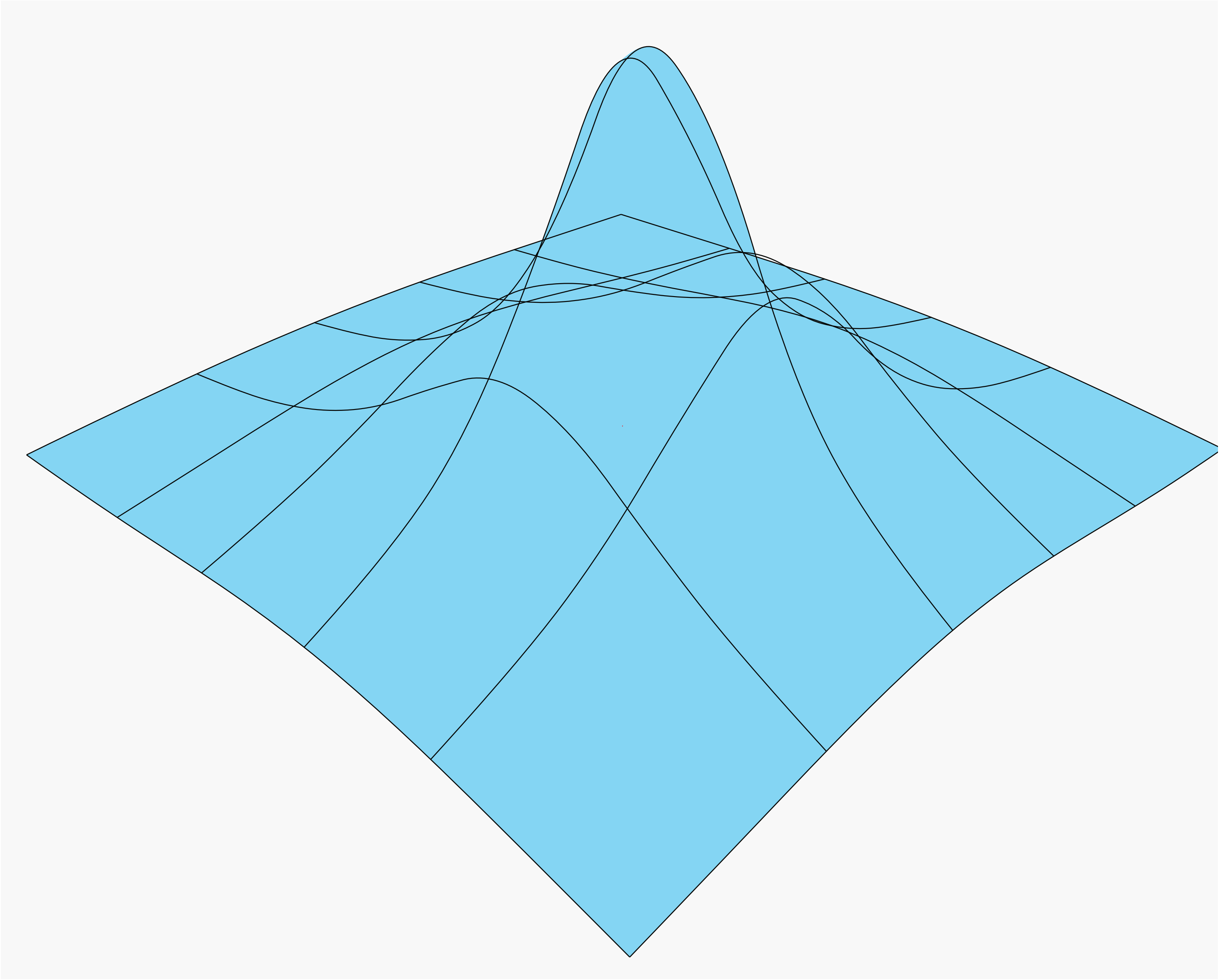


the maths
behind splines is
beyond me, but
the geometric
description is
actually quite easy -
more on that later

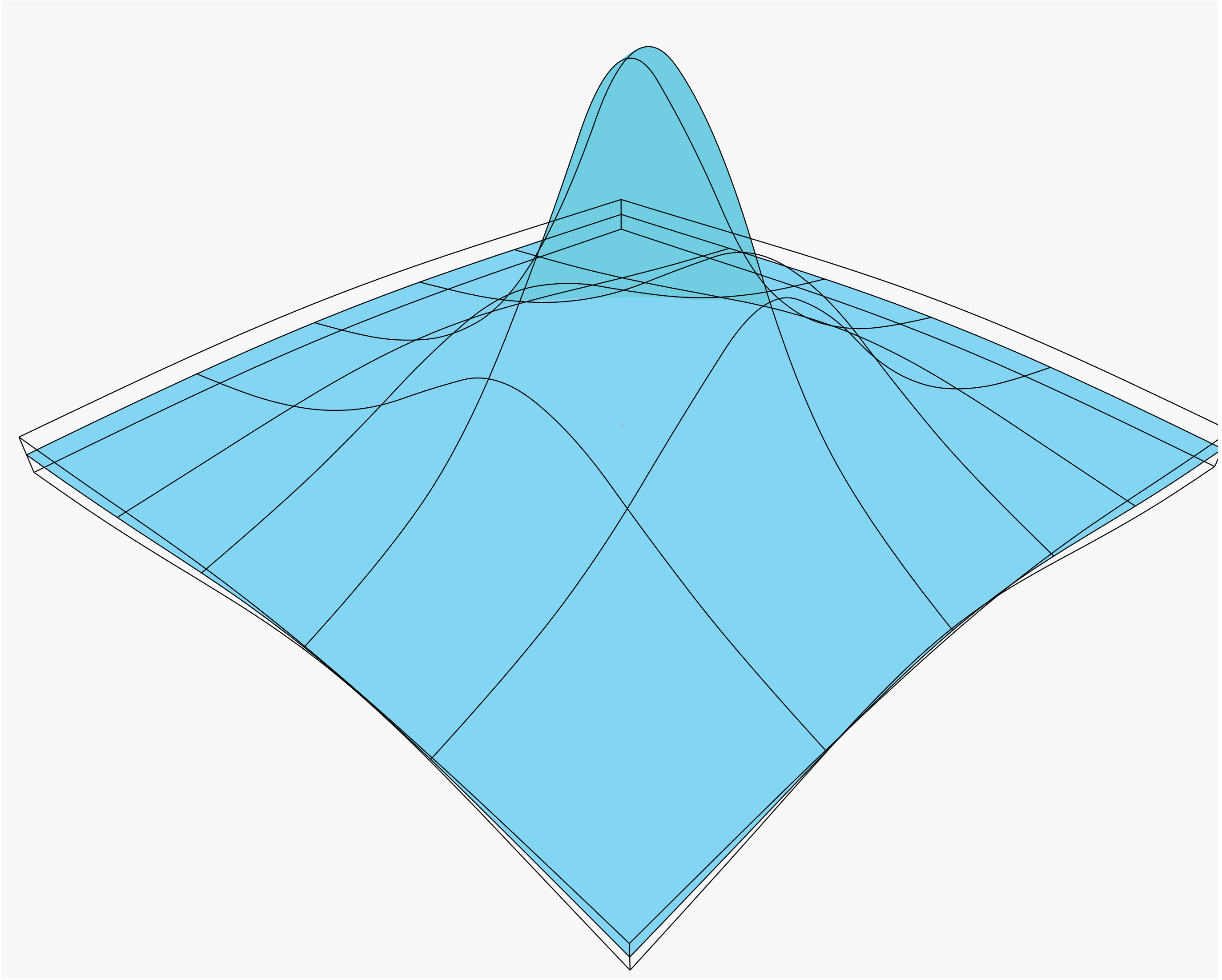


3 points define a plane.

Again, planes are infinite.

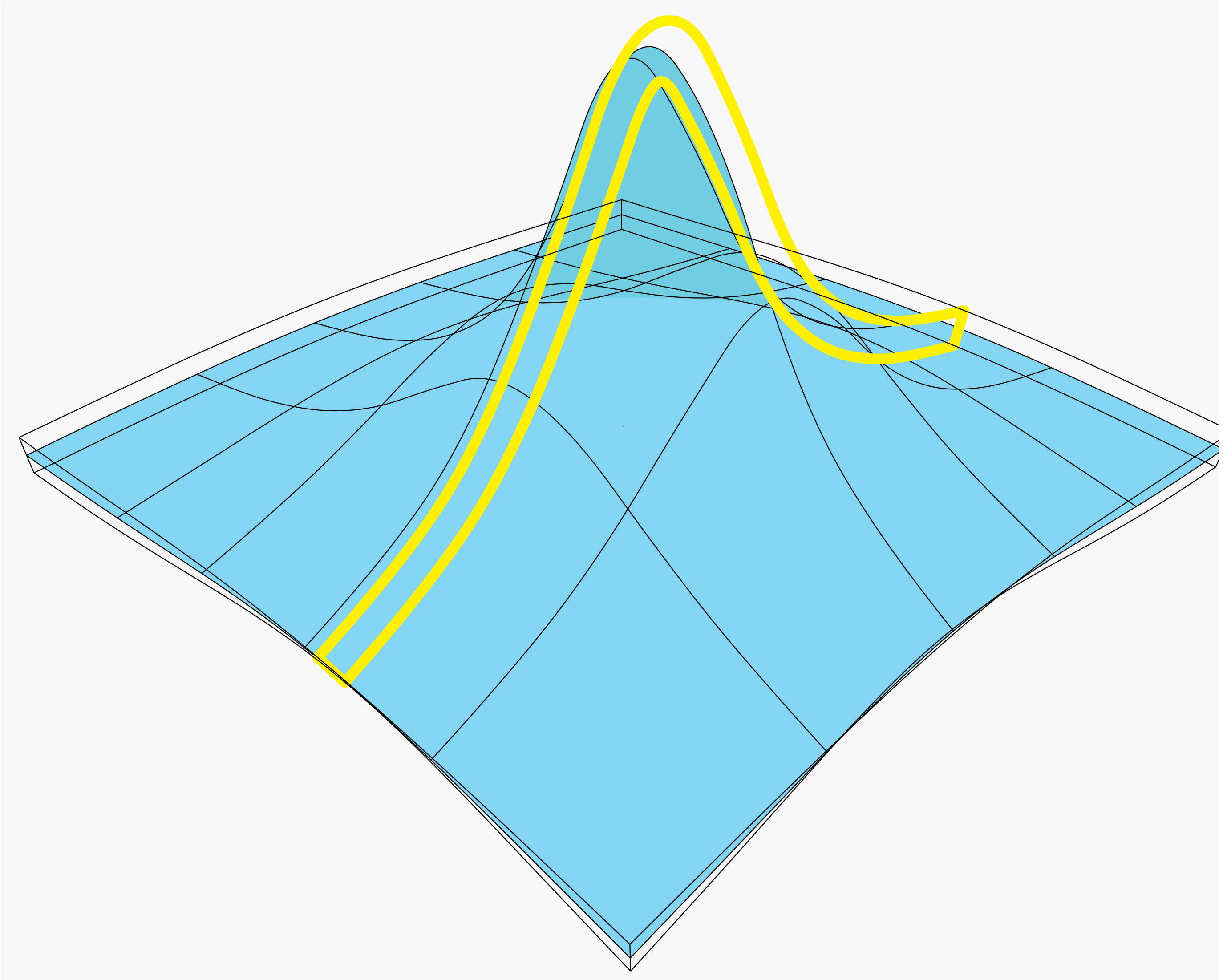


surfaces are in the class of 2d objects, even though they need to be in a 3d space, but again, more on that later



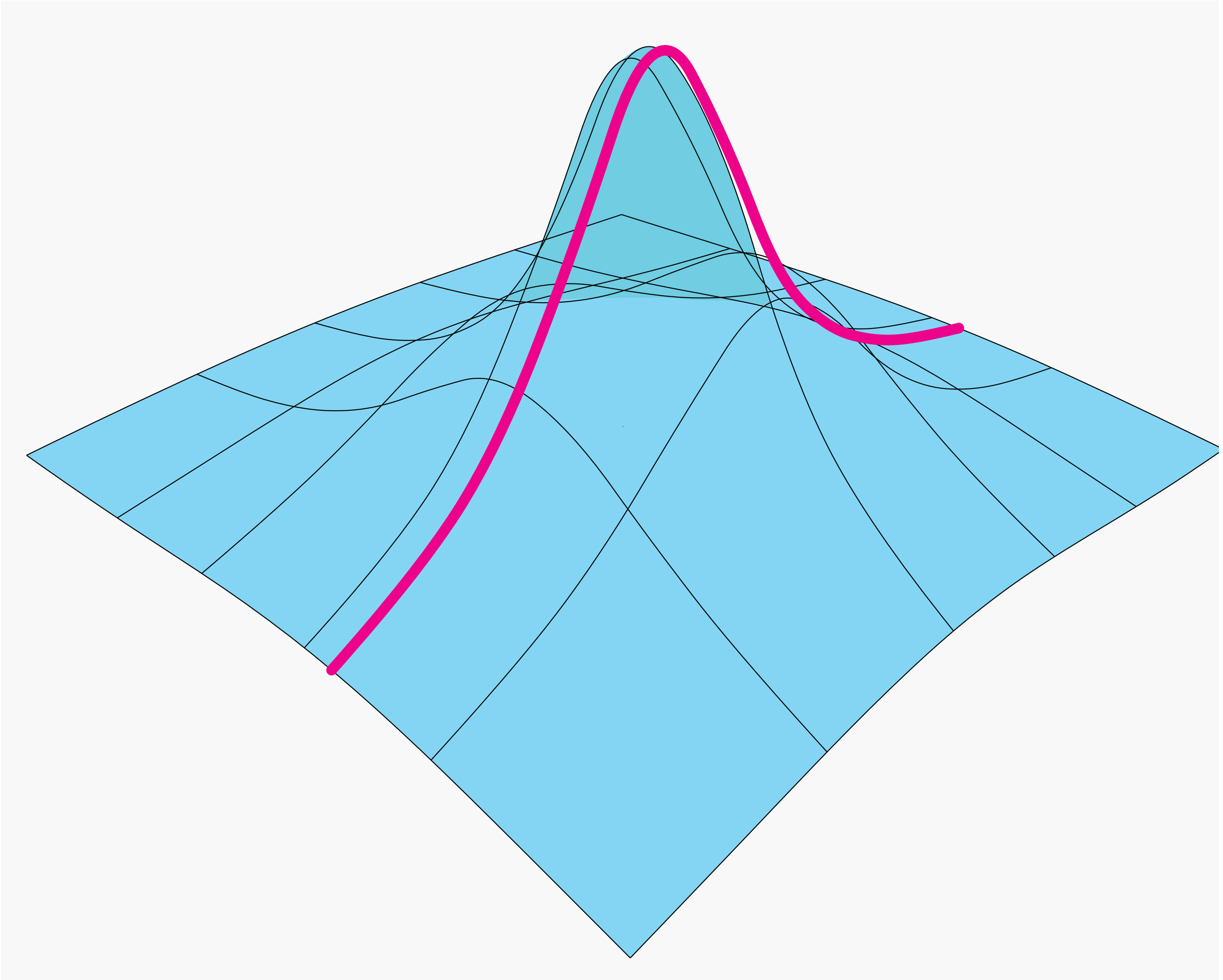
there are loads of ways of making solids, but they are the only truly 3d objects in GC, as they have a volume.

That is not to say that the rest of the things aren’t 3d, it’s just a technical geometry distinction. These sort of things come up a lot.

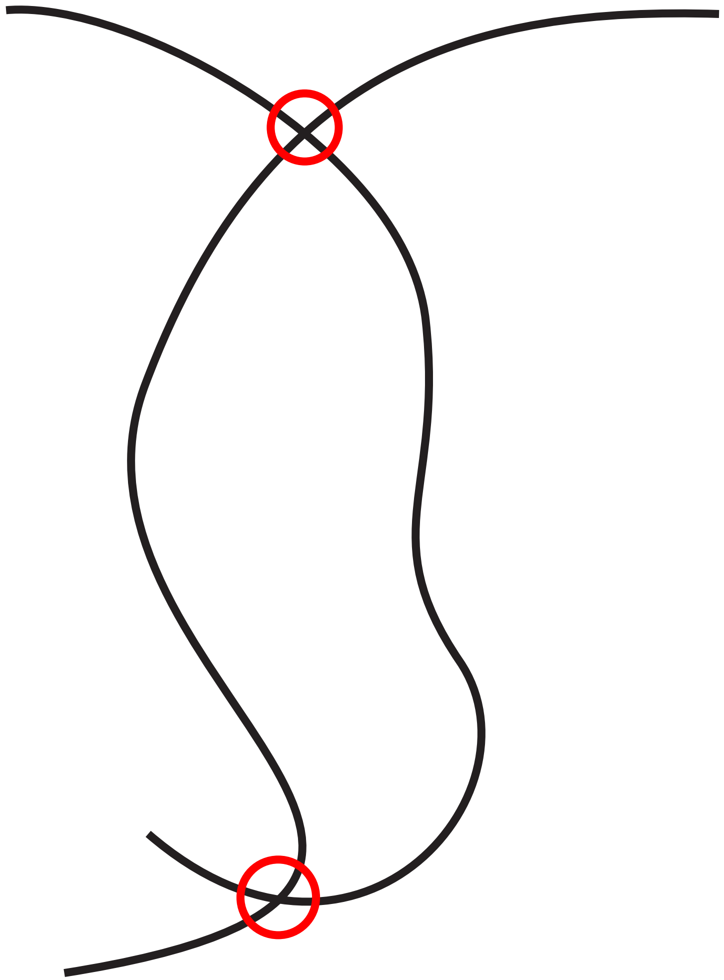


we can step
back down the
dimensional ladder
again too

solids intersected
with a plane or
surface produce a
closed curve



surfaces intersected with a plane or surface produce an open curve (usually)

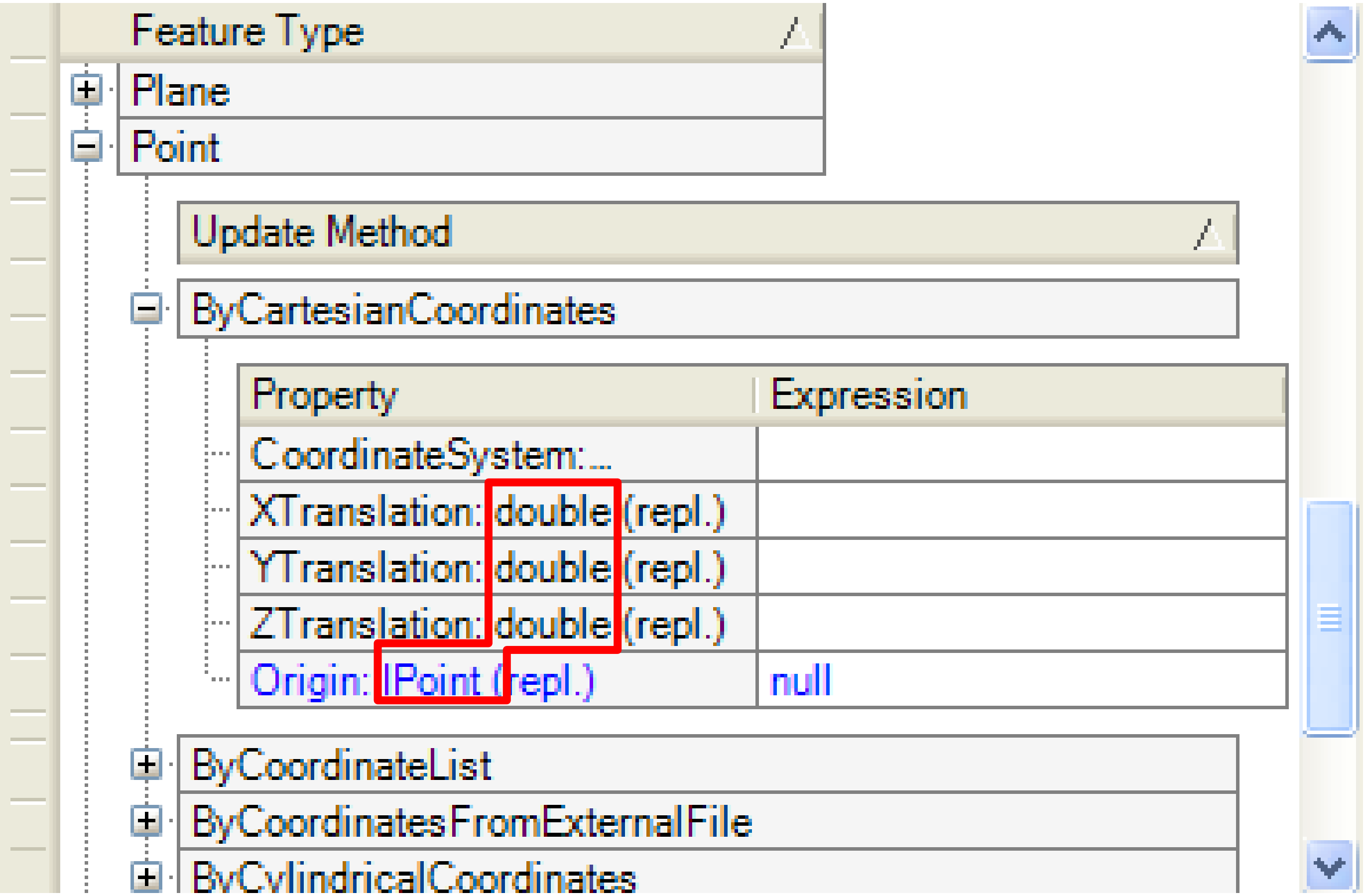


curve curve intersections produce points.

be careful of the extra point!
Circles are classic for this problem.

types

how do you tell what to put in each box?



Feature Type

- Plane
- Point

Update Method

- ByCartesianCoordinates

Property	Expression
CoordinateSystem:...	
XTranslation: double (repl.)	
YTranslation: double (repl.)	
ZTranslation: double (repl.)	
Origin: IPoint (repl.)	null

- ByCoordinateList
- ByCoordinatesFromExternalFile
- ByCylindricalCoordinates

The *type* is the biggest clue

inputName:type

so what’s a type?

data comes in different flavours.

Computers are picky, they only eat what they feel like.

So a *type* is a kind, a breed, a species, a flavour of data.

the most common types are coming up

int Counting numbers (0, 5, -4, 1000, -500)

int	Counting numbers (0, 5, -4, 1000, -500)
double	Real numbers(0.5, -7.8 ,15.0, 1598.5434)

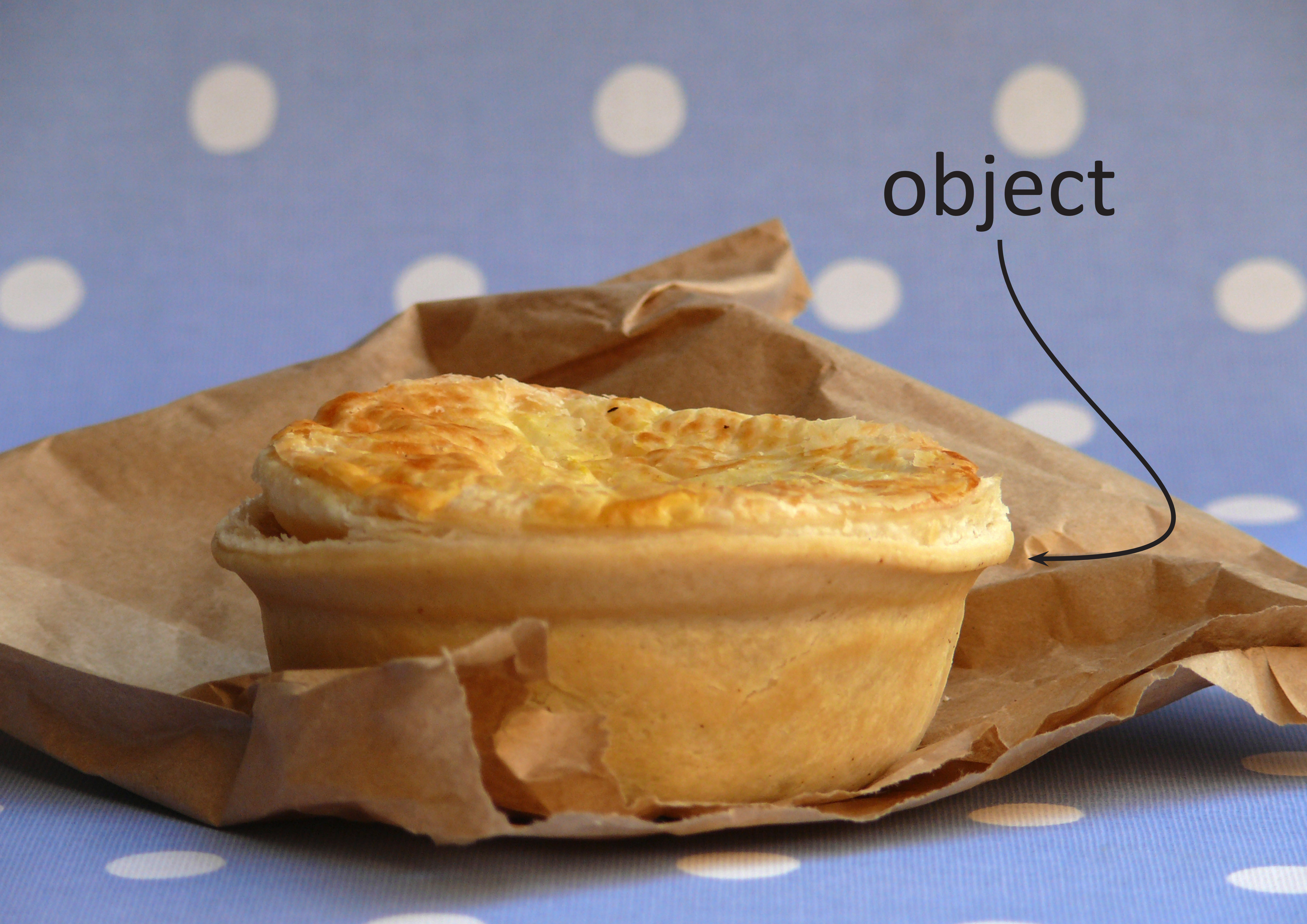
int	Counting numbers (0, 5, -4, 1000, -500)
double	Real numbers(0.5, -7.8 ,15.0, 1598.5434)
boolean	Answer to a logical question (true, false)

int	Counting numbers (0, 5, -4, 1000, -500)
double	Real numbers(0.5, -7.8 ,15.0, 1598.5434)
boolean	Answer to a logical question (true, false)
string	Some text(“hello world”, “450”, “dave”)

int	Counting numbers (0, 5, -4, 1000, -500)	generic types
double	Real numbers(0.5, -7.8 ,15.0, 1598.5434)	
boolean	Answer to a logical question (true, false)	
string	Some text(“hello world”, “450”, “dave”)	
IPoint	GC’s special point	GC specific types
IDirection	GC’s special part of a vector	
ICurve	CG’s own curve, includes lines, arcs, bsplines	
ISurface	CG’s own surface	
ISolid	CG’s own Solids	
User defined	You can define your own types in C#	

you can sometimes stuff one thing into another slot (casting) but the type is generally a good hint as to what is required.

properties


object

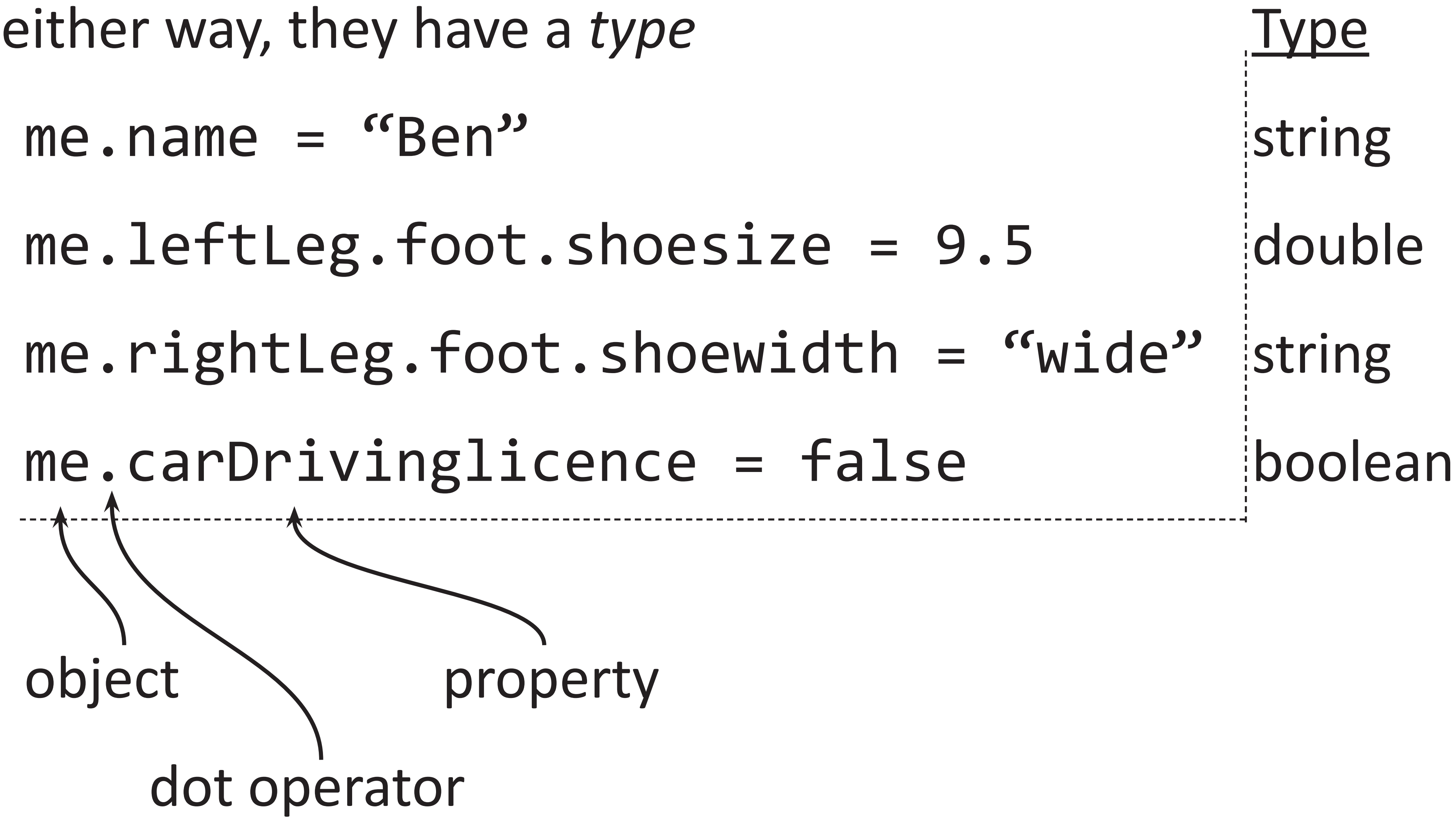
properties

dot operator

objects have *properties*

they can be values, or sub-objects

either way, they have a *type*



	Type
`me.name = “Ben”`	string
`me.leftLeg.foot.shoesize = 9.5`	double
`me.rightLeg.foot.shoewidth = “wide”`	string
`me.carDrivinglicence = false`	boolean

relationships

generally we are much less interested in numeric descriptions of where things are, and how big they are.

We are just into relationships

the coffee is constrained *in* the cup

in a normal cad program, even if we put the coffee in the place that is inside the cup, it's just numerically defined as being there.

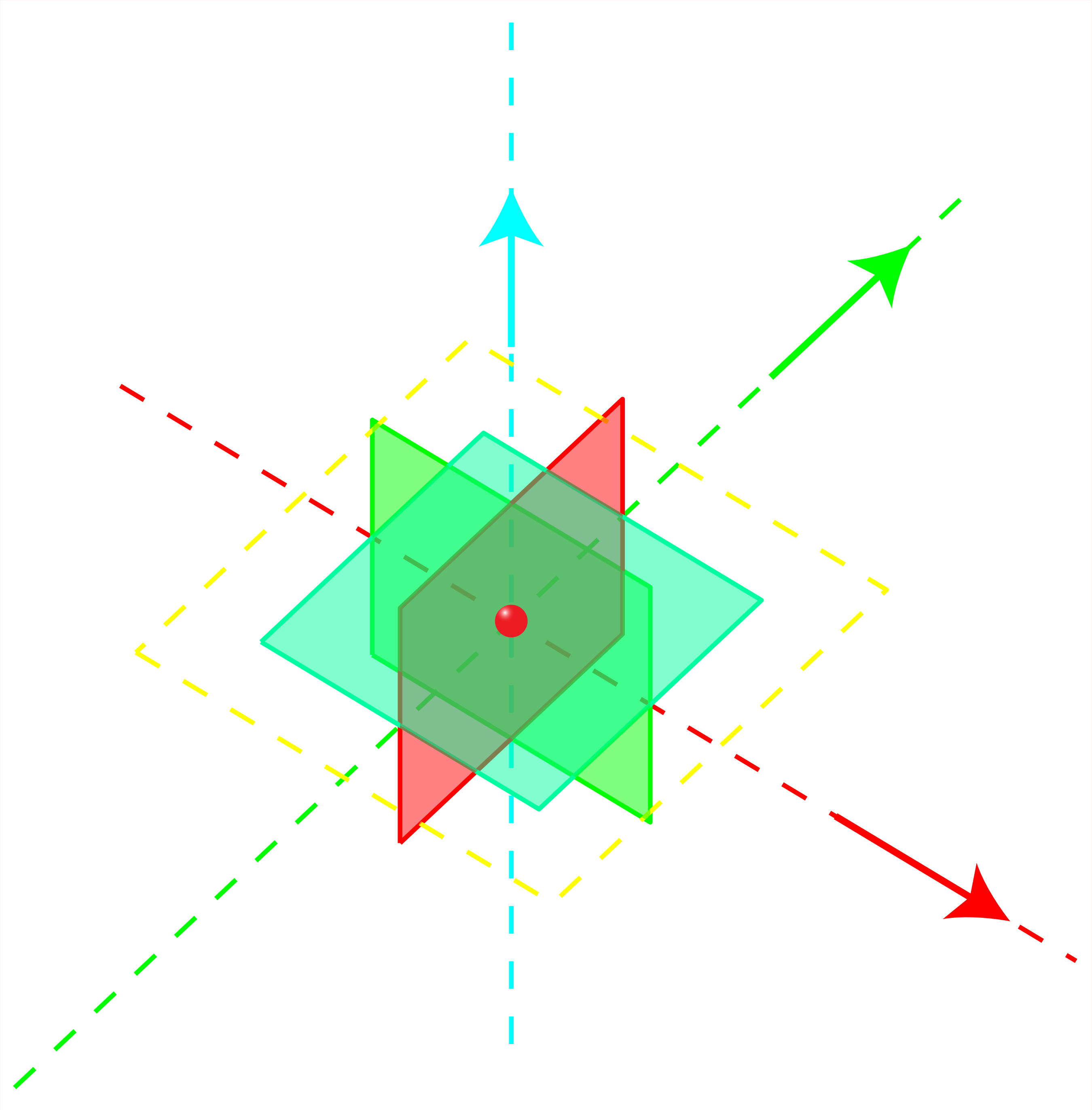
if we move the cup then it's still where is started.

in a relational
system, when we
move the cup, the
coffee moves.

in a relational
system we build
relationships
and behaviours.

i.e. **generals**
not specifics

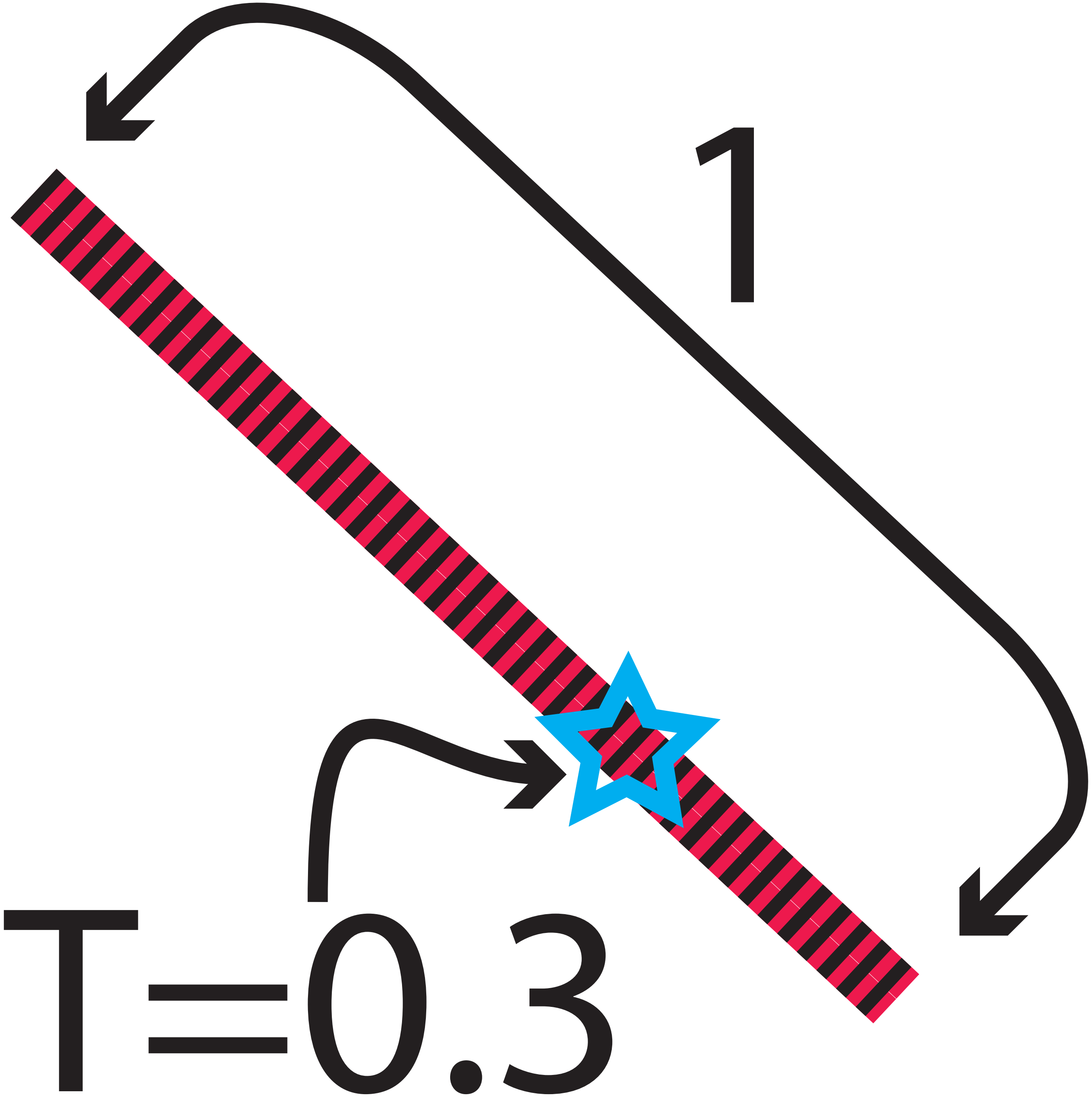
spaces



Cartesian space

Unless you are a quantum physicist or a theoretical mathematician, Cartesian 3 space is all you'll ever need (almost)

If anyone ever points at a building and tells you that it is non-euclidean then they are just plain wrong.

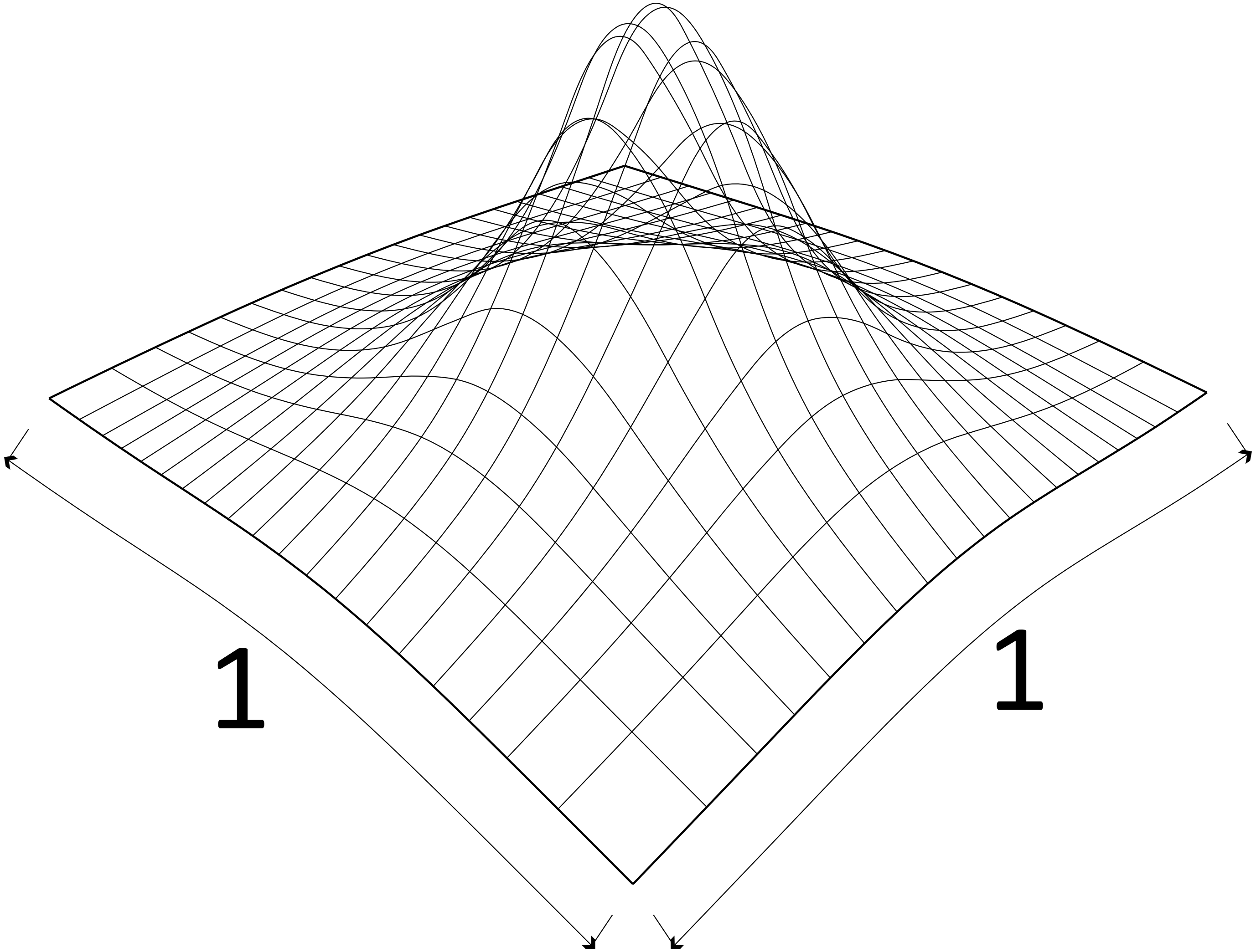


Parameter space

This is an *embeded* space.

from within the line the universe only extends as far as the end of the line.
so the space is **1** long, regardless of its size externally.

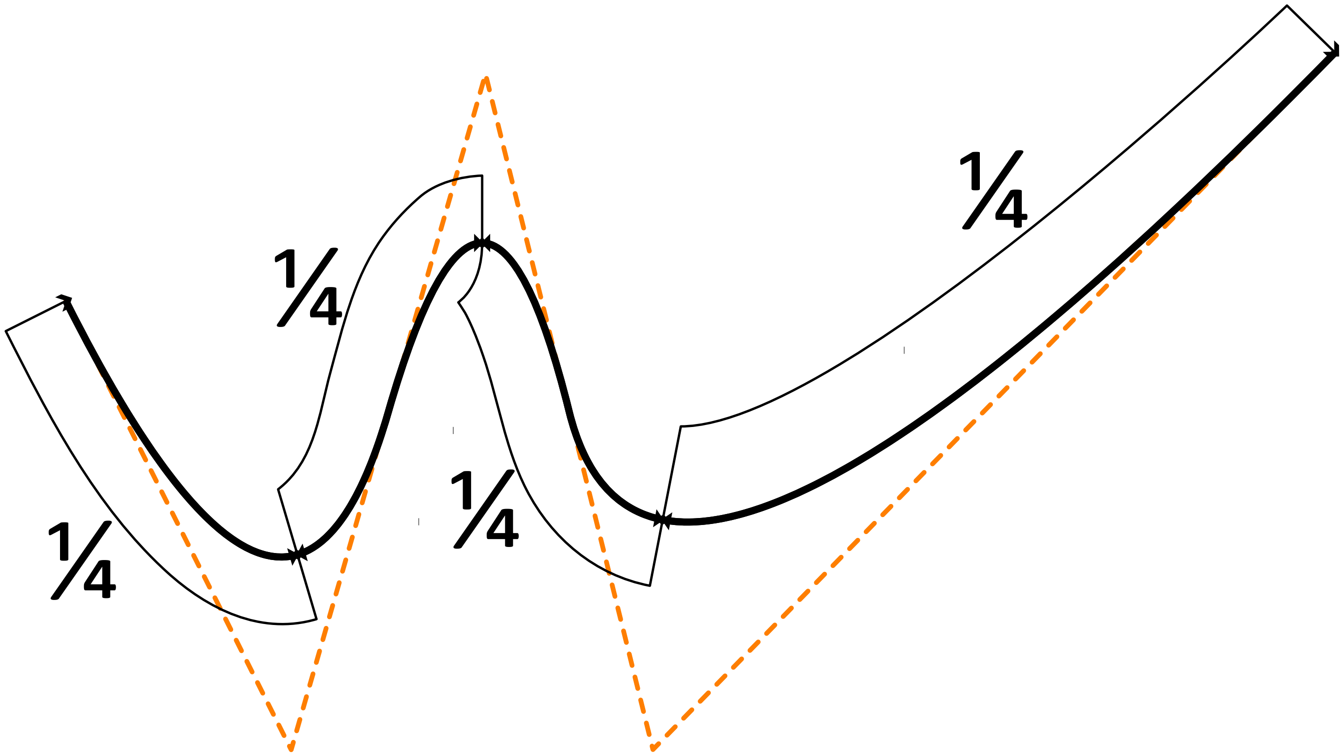
this is the **T value**



the same is true for surfaces, the surface is always considered to be a 1 by 1 square

instead of XYZ coords
it is UVD coords

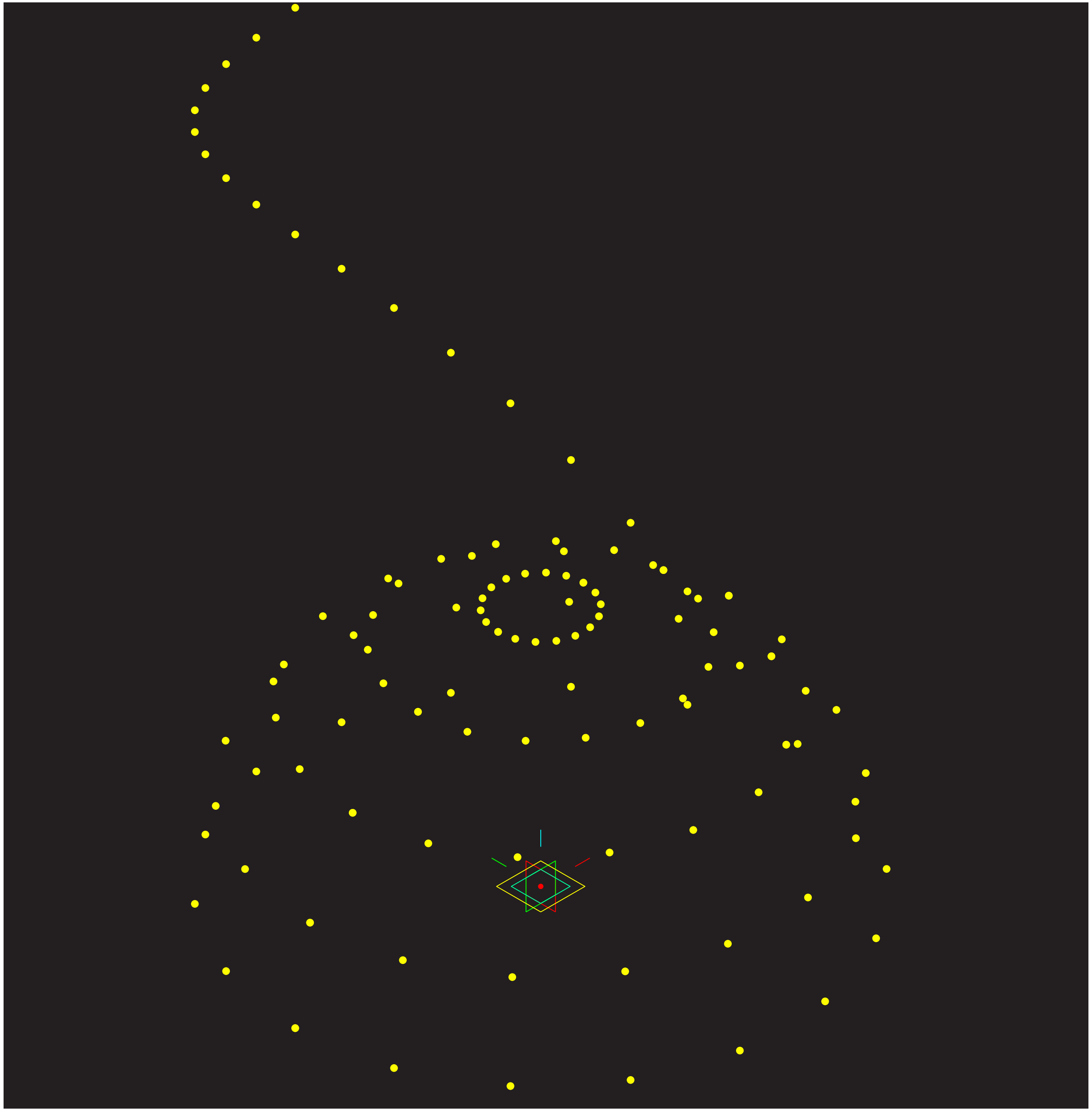
the easiest way to think about how it deals with distortion is to draw a grid on a balloon and then blow it up & squidge it about a bit. the grid changes shape, but the *relationships* stay the same.



This inconsistency when viewed from an external viewpoint can cause problems if you aren't ready for it.

T = 0.5 isn't the geometric centre, it's the parametric centre.

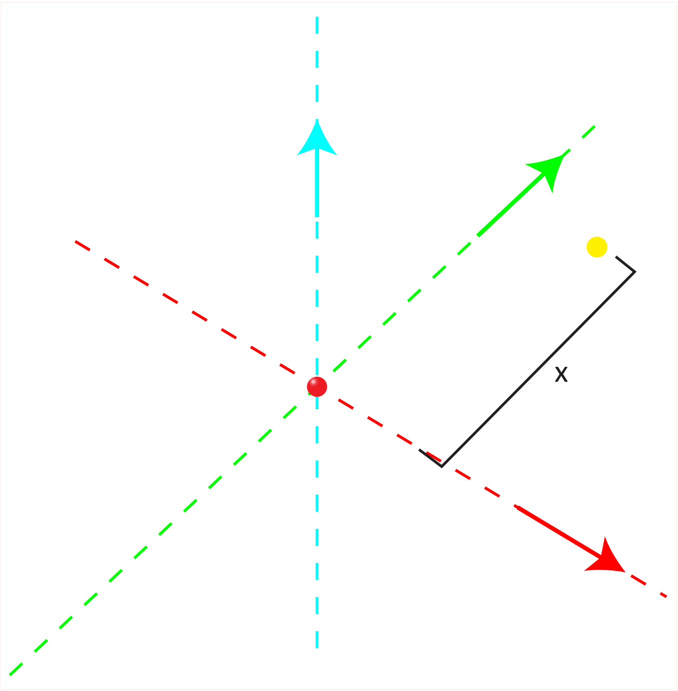
parametric distances between control points are equal



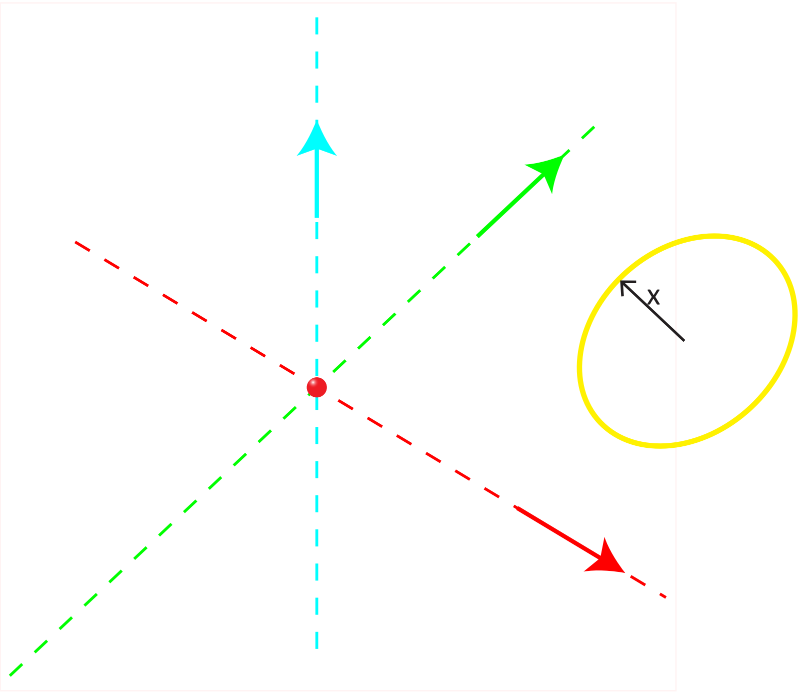
there are also cylindrical coordinate systems and spherical coordinate systems to play about with.

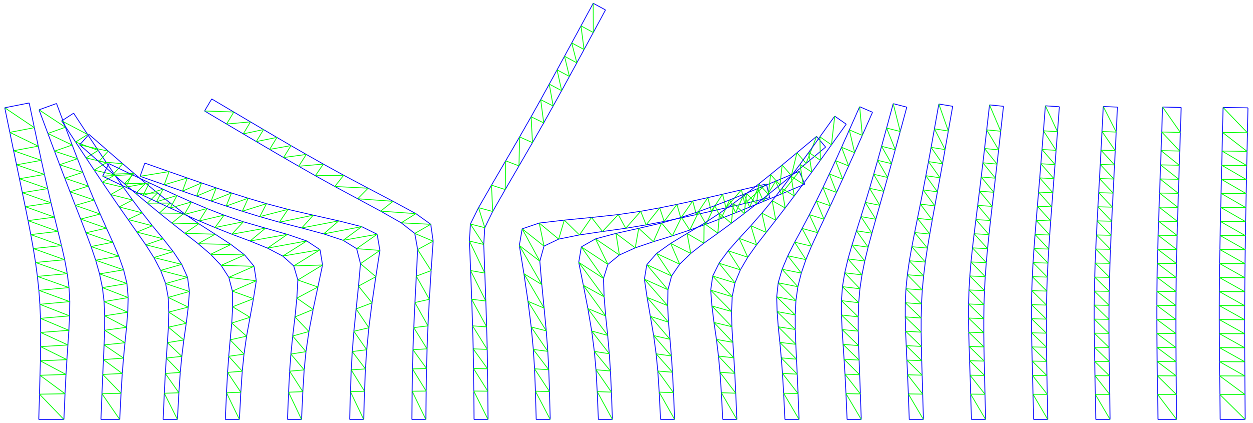
These are handy for cylindrical and spherical things, but also for survey data.

multiple spaces

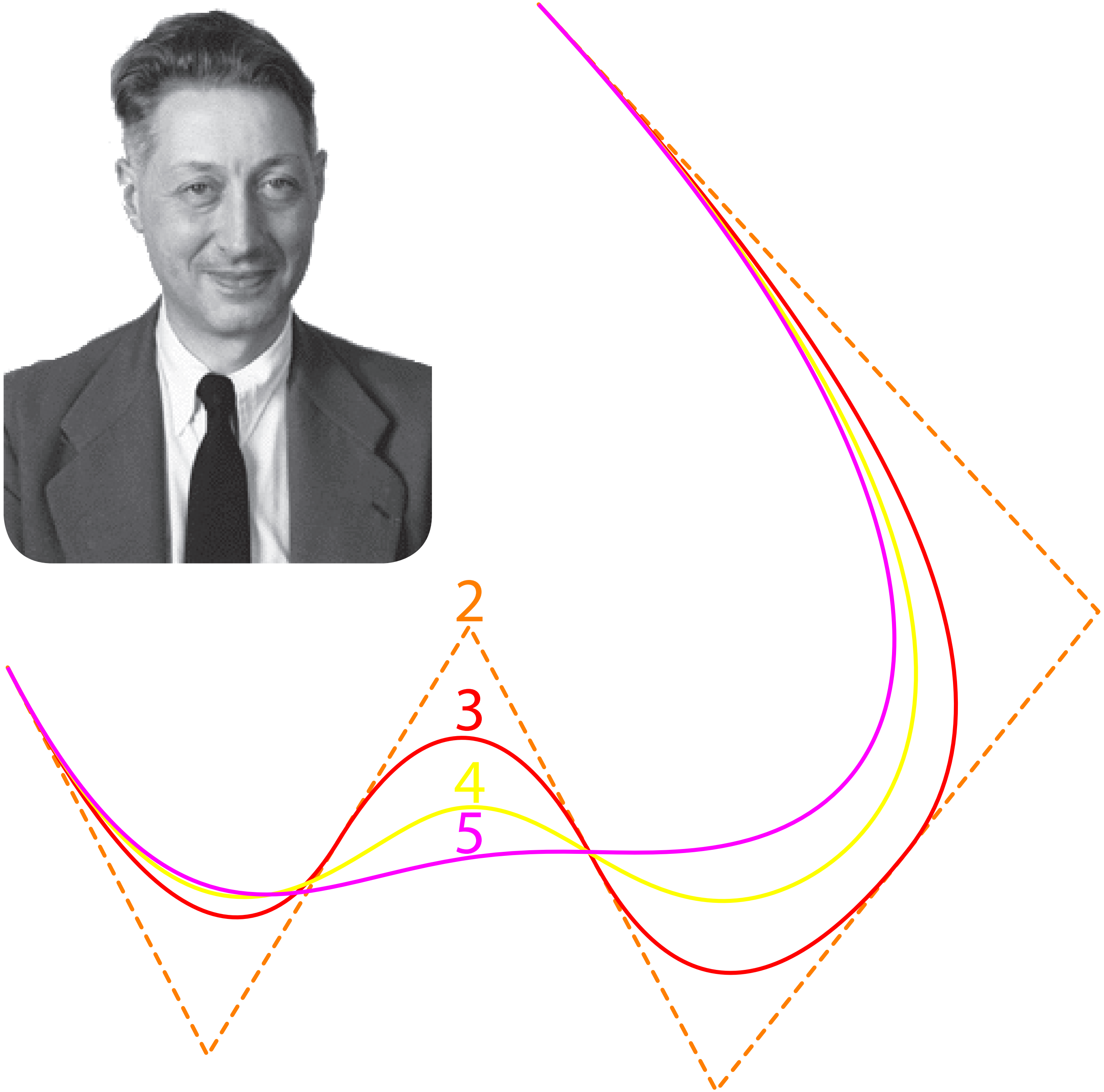


by using a parallel representations of your geometry it’s possible to build control rigs, analysis dashboards etc.





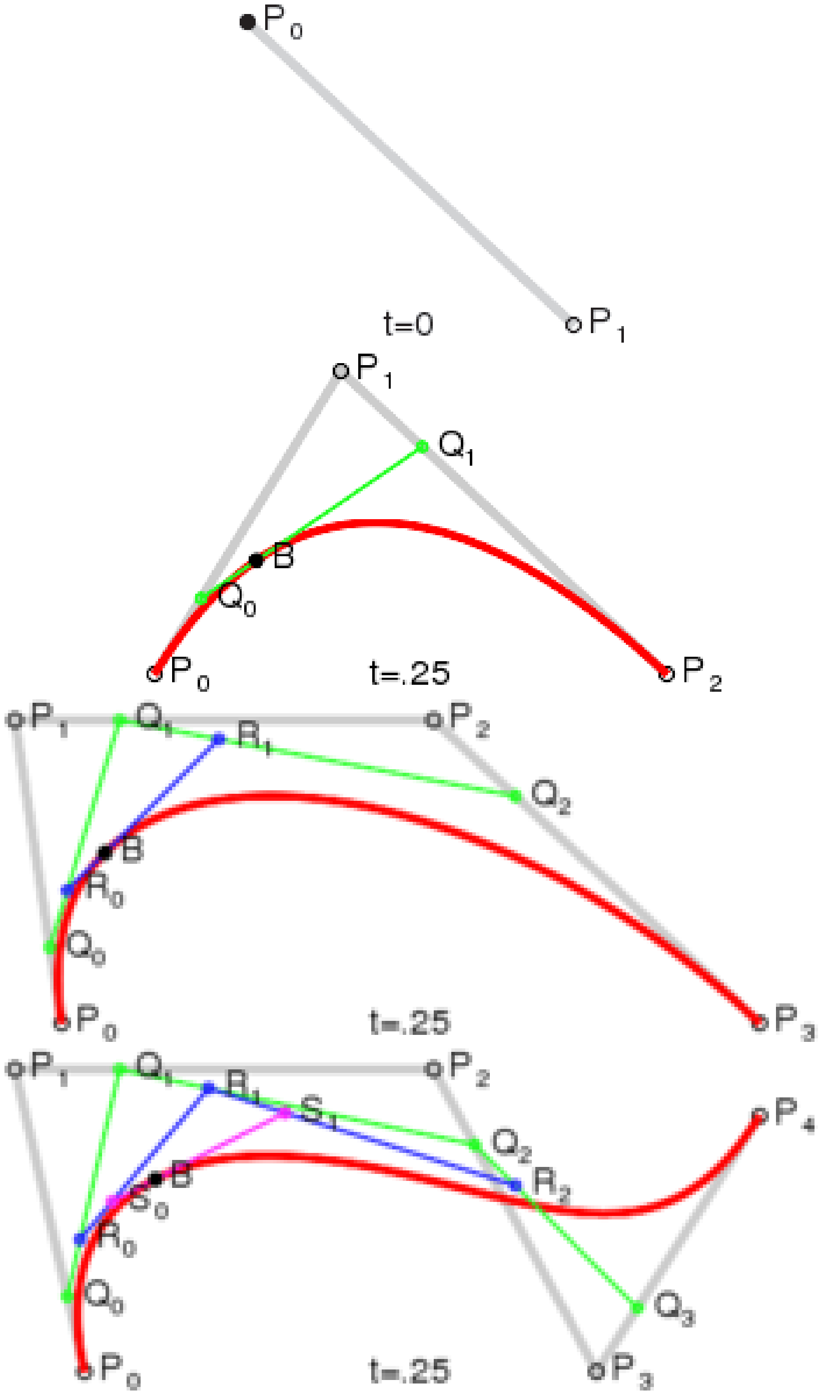
spline geometry



Splines are very ‘cool’, but they aren’t very constructible because contractors and manufacturers are a bit scared of them.

They were developed independently by a pair of French automotive engineers – Pierre Étienne Bézier at Renault and Paul de Casteljau at Citroën – working on early CAD systems back in the 1960s

They take some understanding to do them right!



linear
order = 2

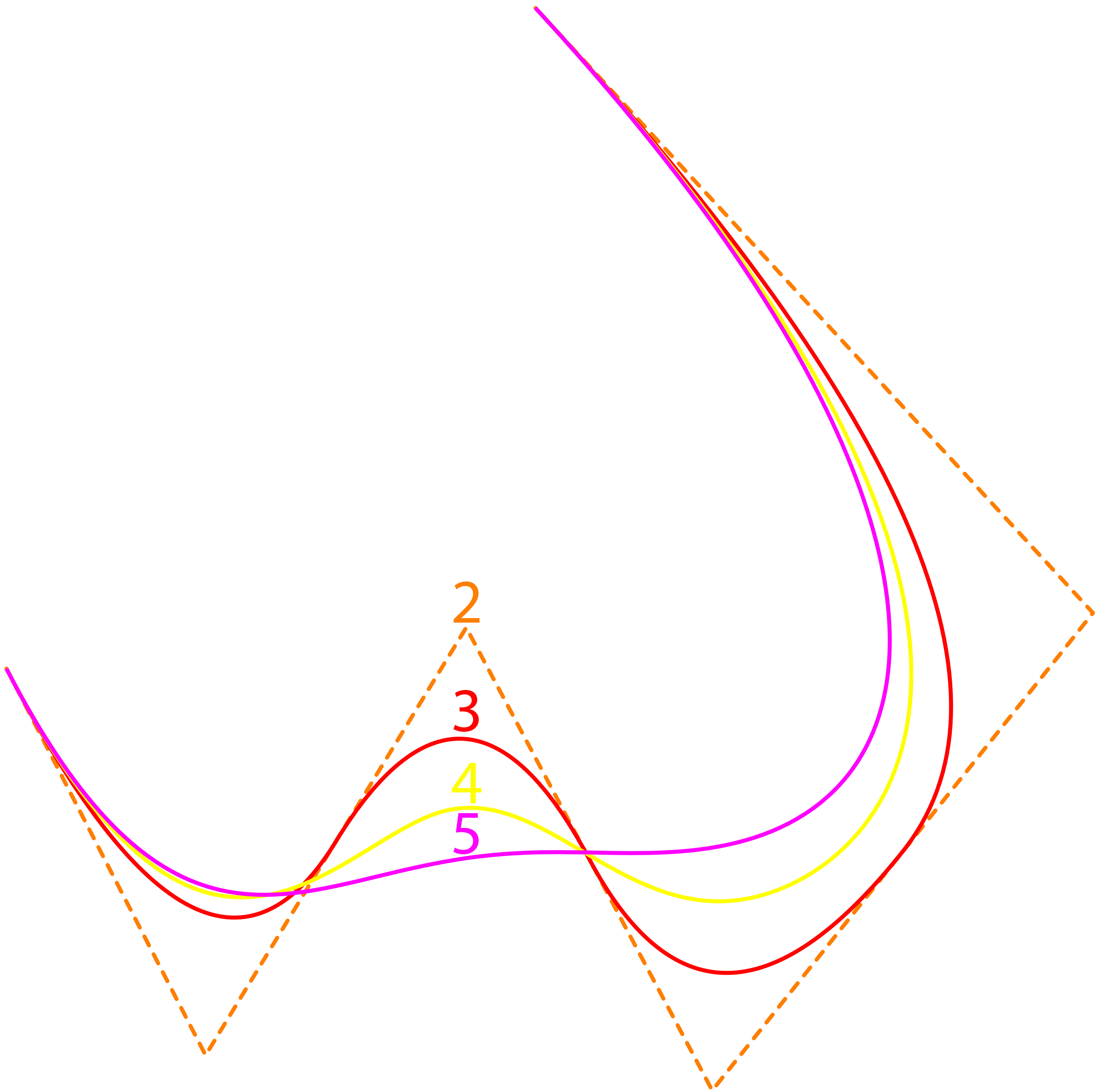
quadratic
order = 3

cubic
order = 4

quartic
order = 5

The order of a spline refers to how many control points each segment looks to for it’s shape.
An *order 2* spline is a straight line.

These diagrams explain the construction of spline curves.

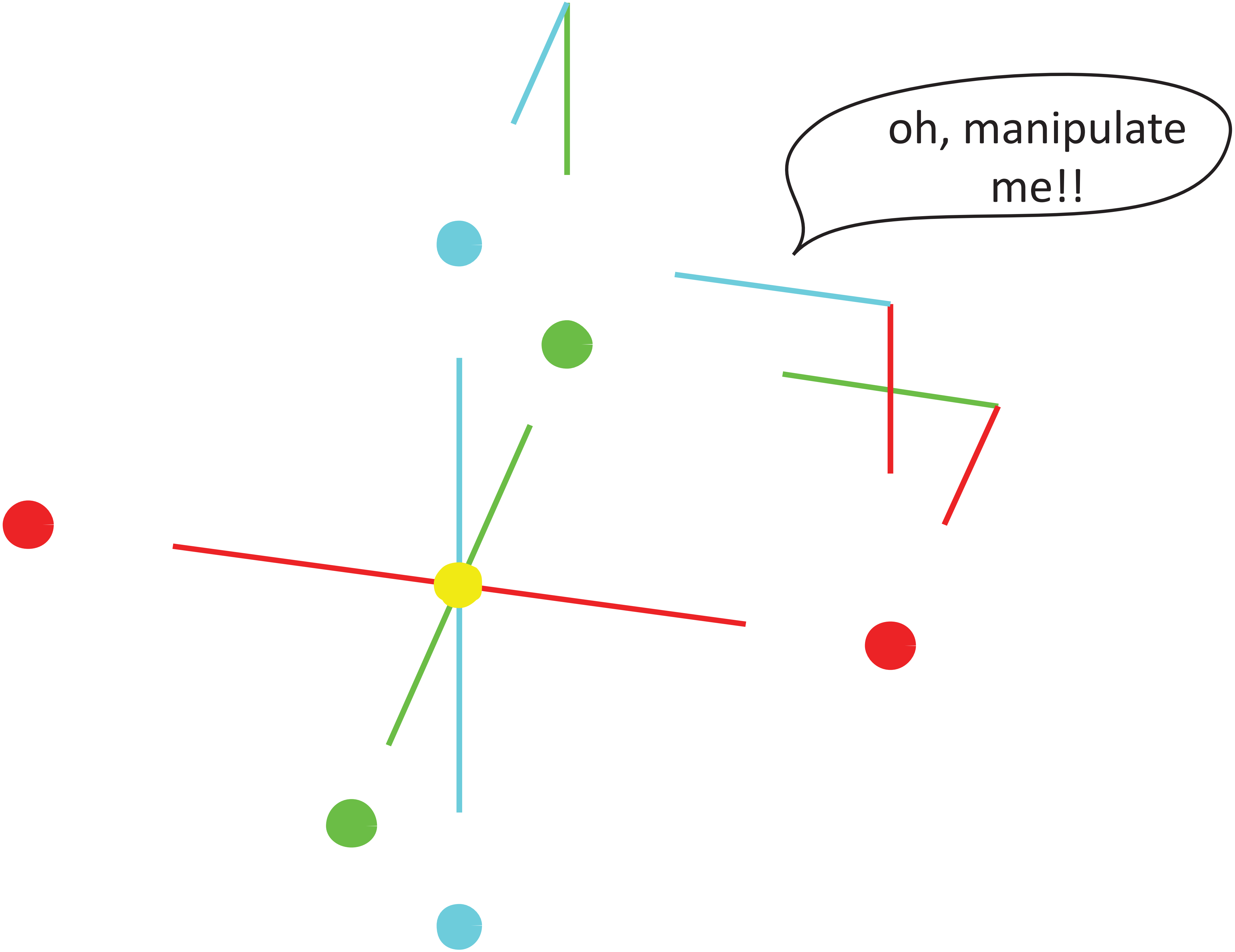


The third order spline is tangent to all of the segments of the control frame.

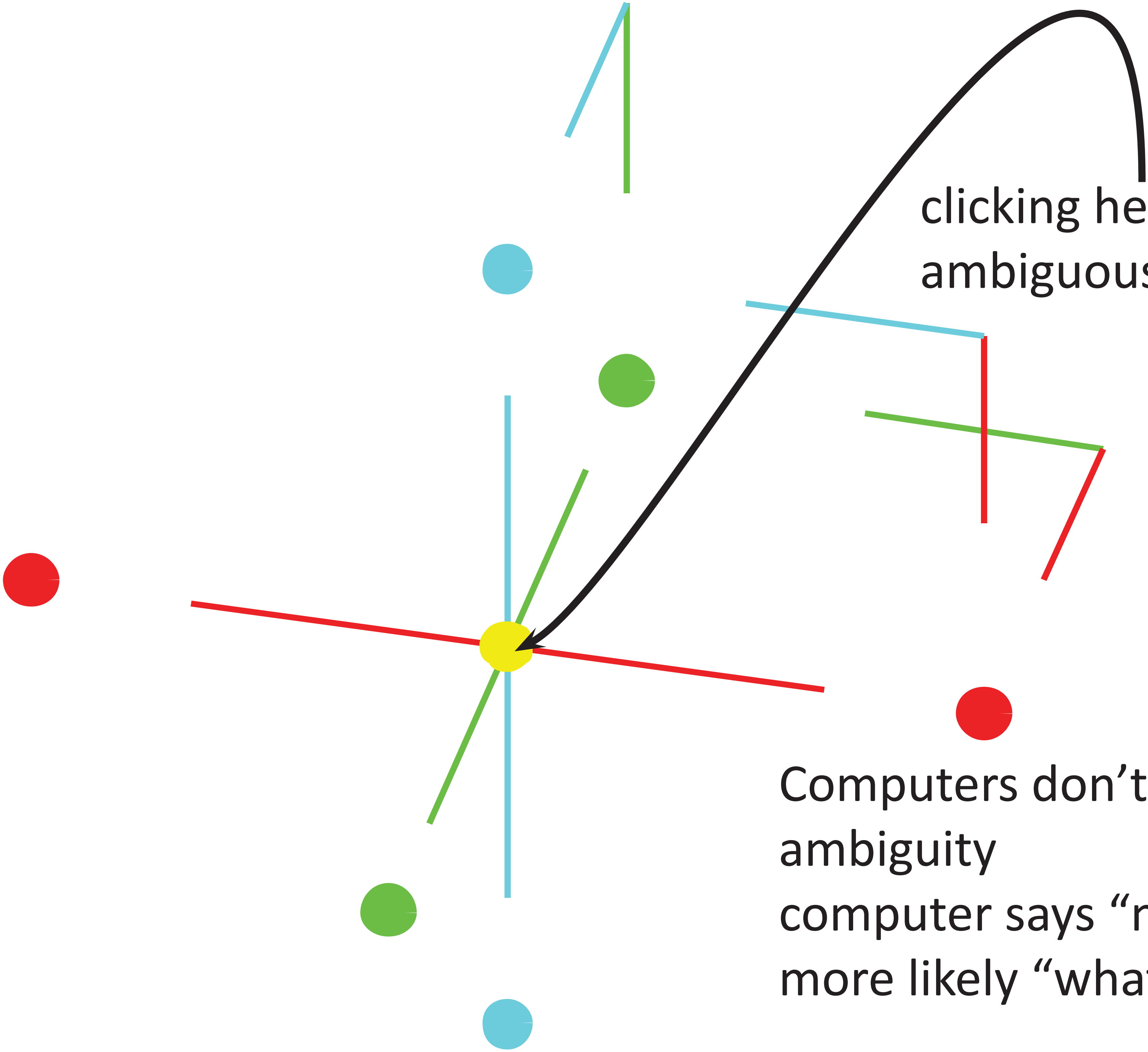
4th and 5th are less easy to visualise their construction.

unambiguous

selection

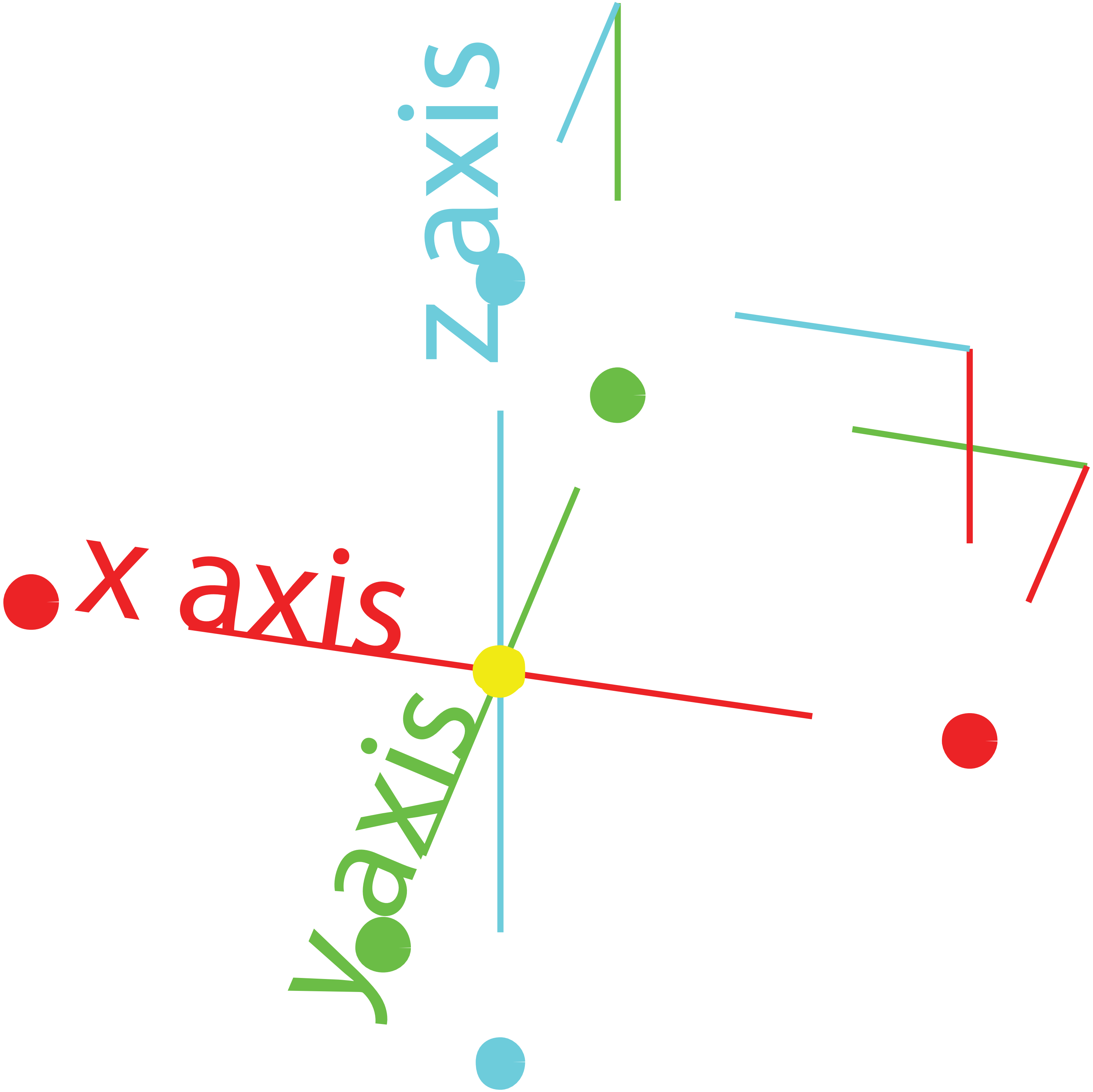


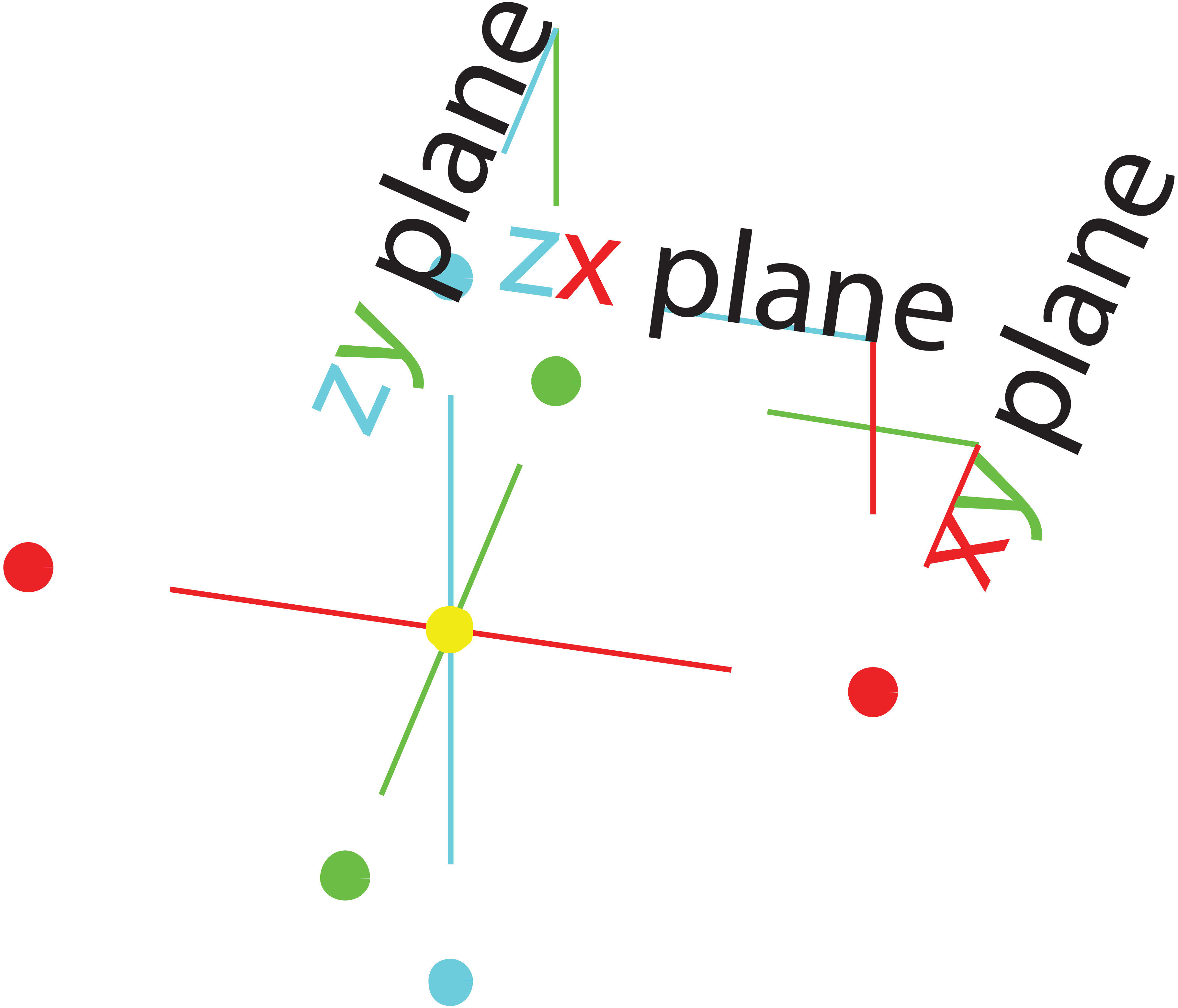
oh, manipulate me!!

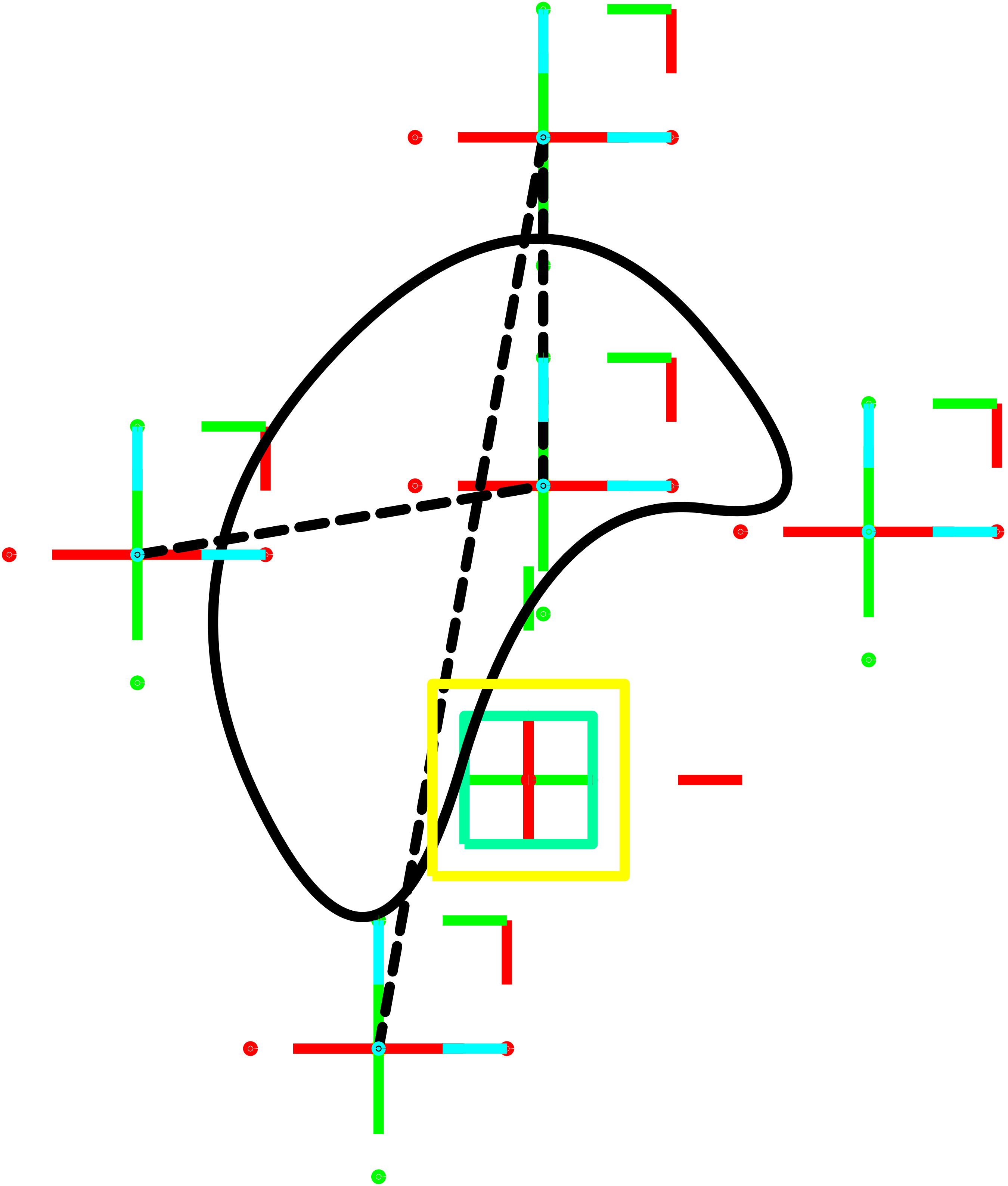


clicking here is ambiguous

Computers don’t like ambiguity
computer says “no”, or more likely “whatever”


z axis
x axis
y axis

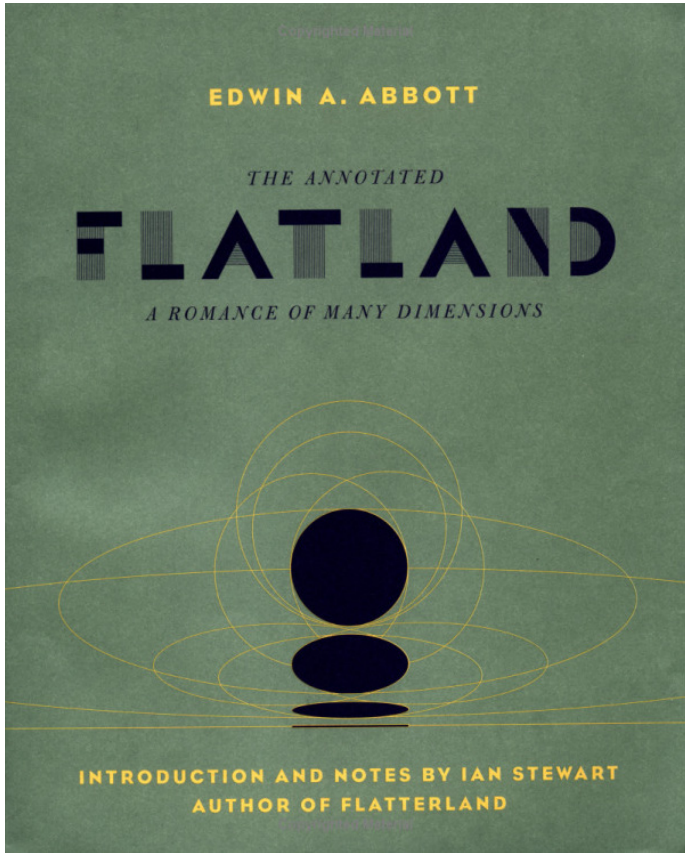

zy plane
zx plane
xy plane

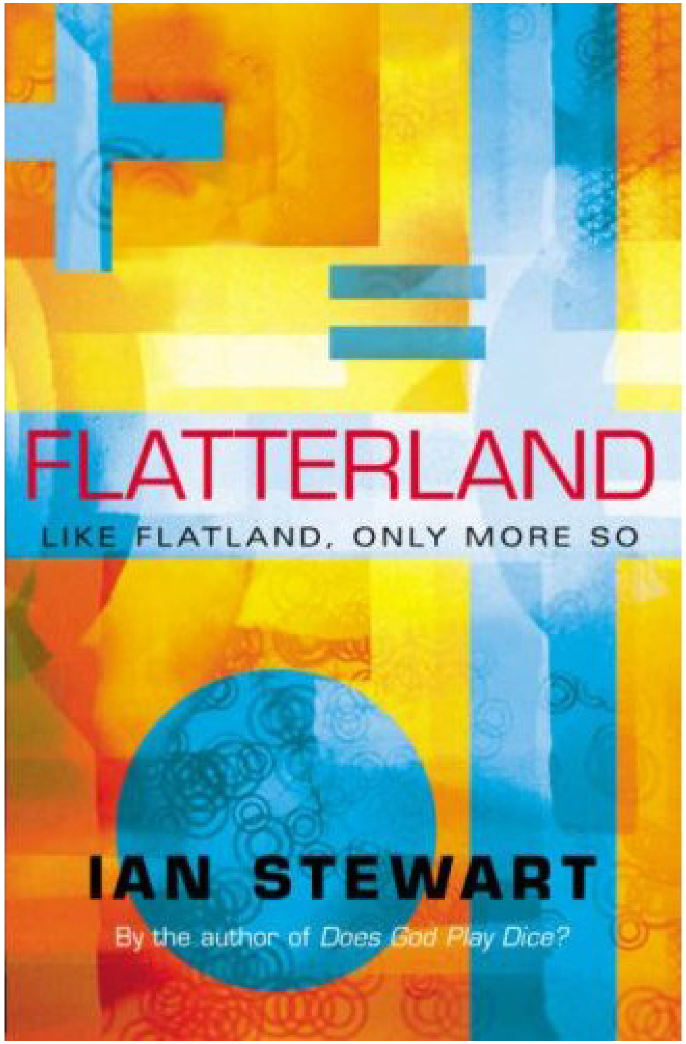


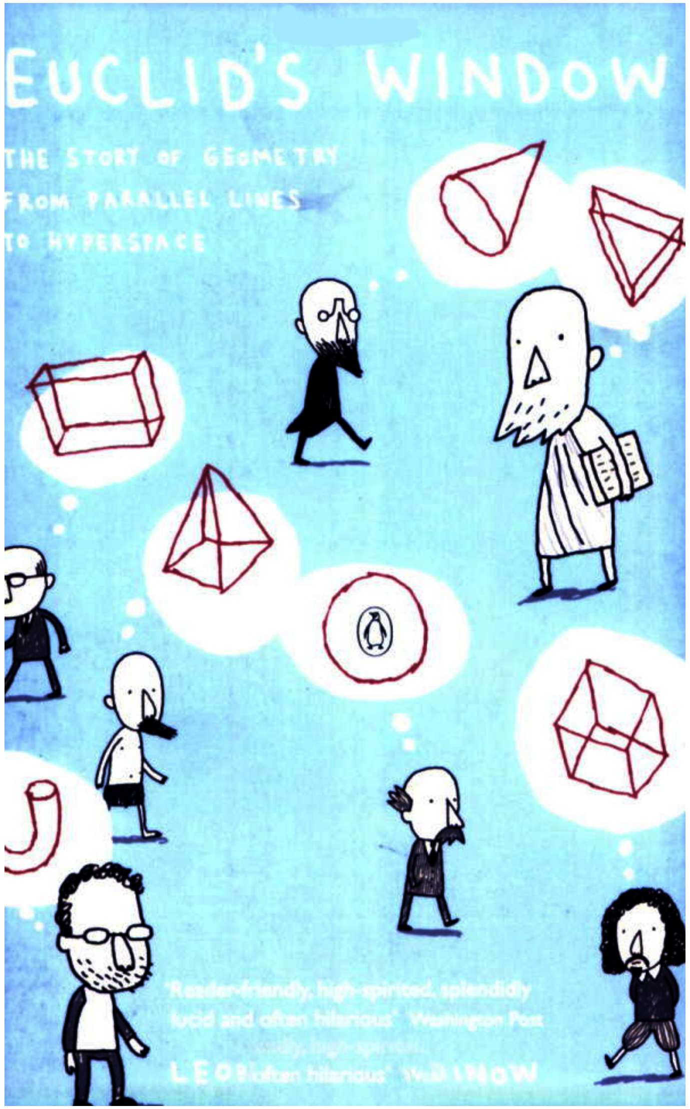
making selections in the least ambiguous position will make your like much simpler.

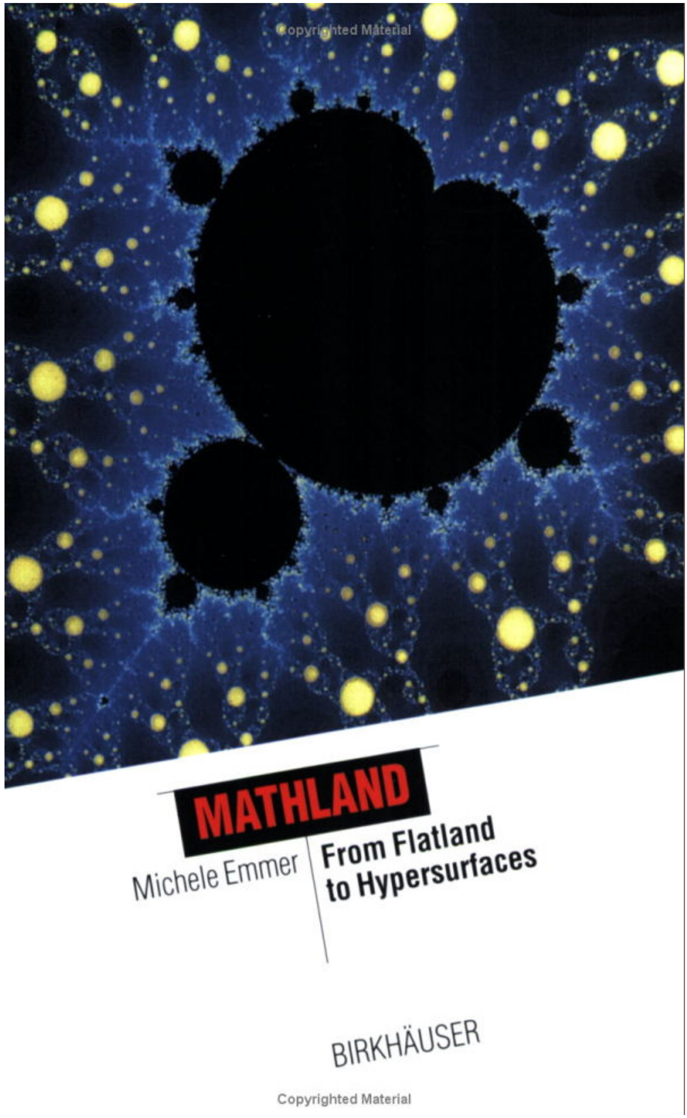
generally the less crowded a place is, the better it is as a position to select.

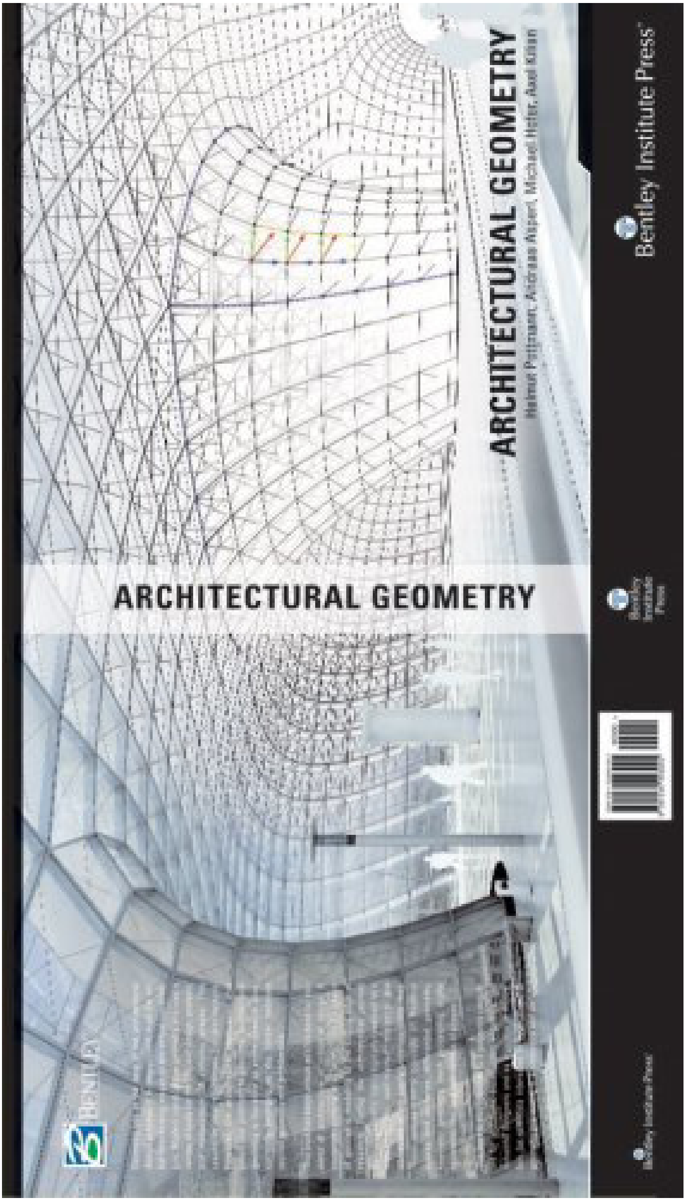
recommended reading

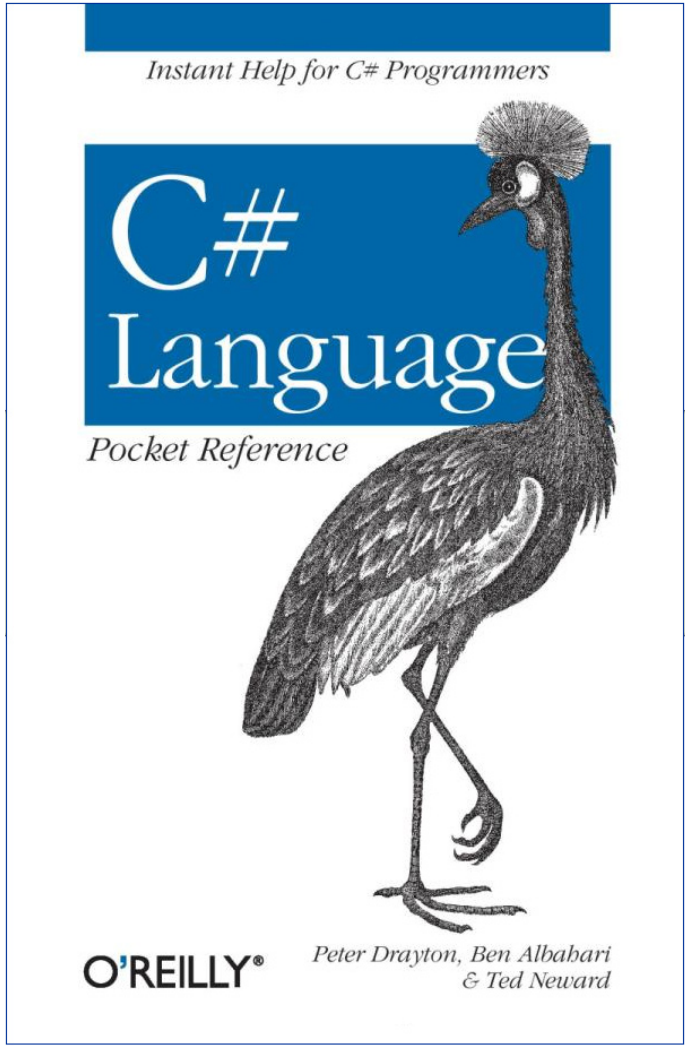

EDWIN A. ABBOTT
THE ANNOTATED
FLATLAND
A ROMANCE OF MANY DIMENSIONS
INTRODUCTION AND NOTES BY IAN STEWART
AUTHOR OF FLATTERLAND

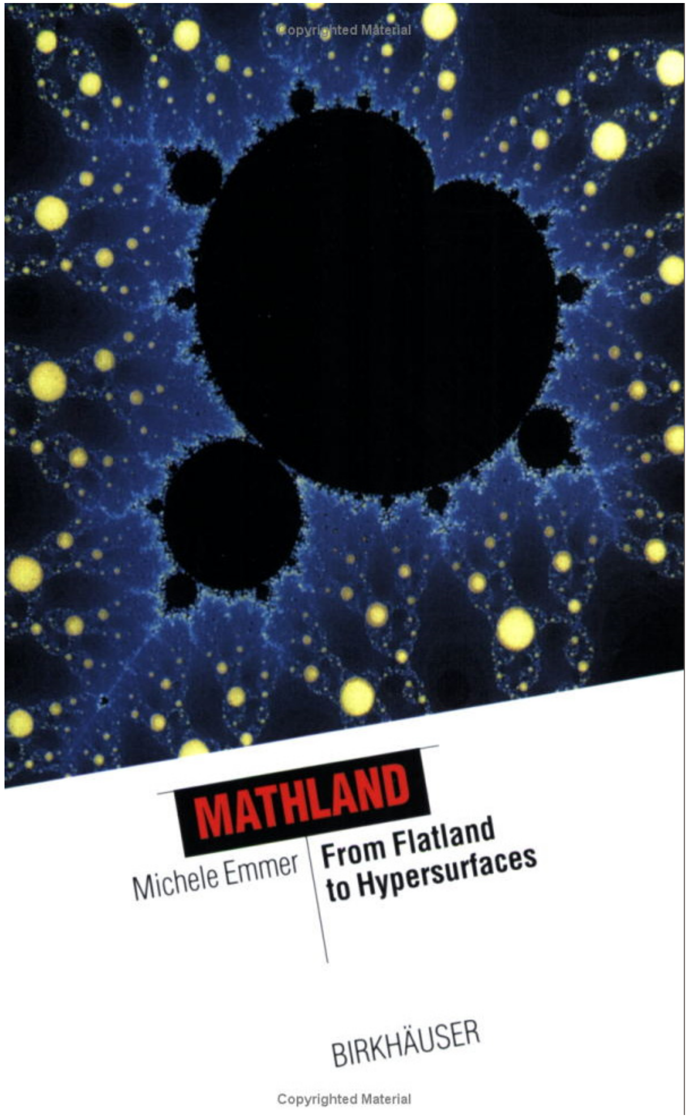

FLATTERLAND
LIKE FLATLAND, ONLY MORE SO
IAN STEWART
By the author of Does God Play Dice?

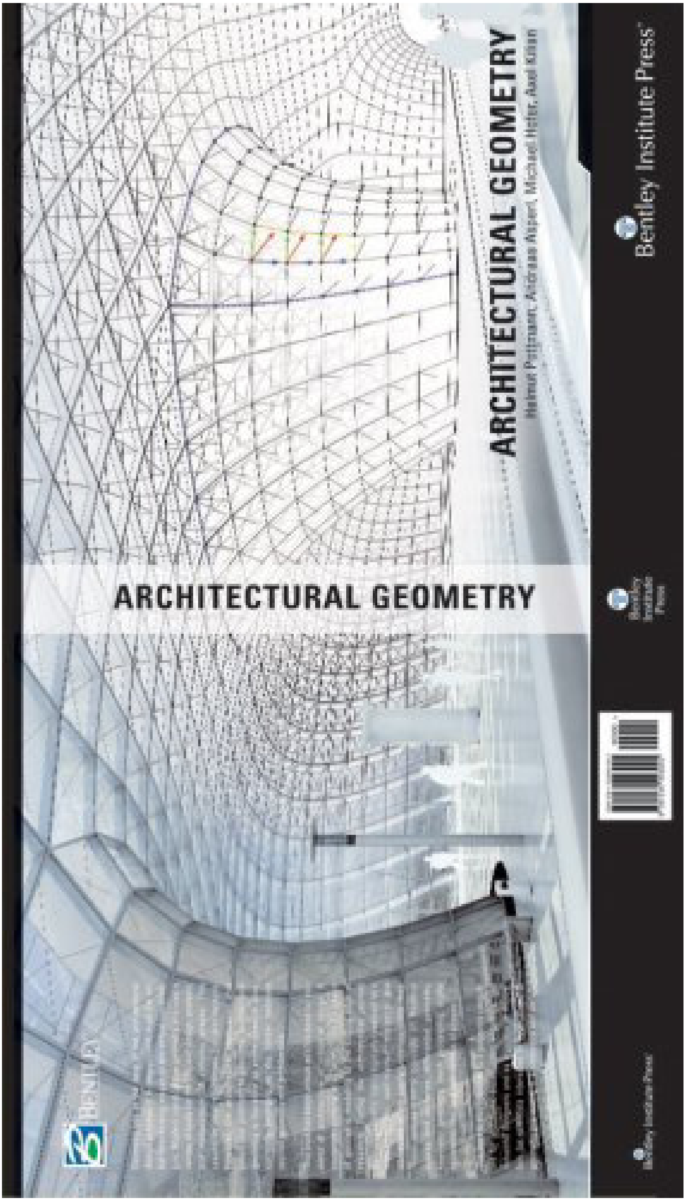

EUCLID'S WINDOW
THE STORY OF GEOMETRY
FROM PARALLEL LINES
TO HYPERSPACE
'Reader-friendly, high-spirited, splendidly lucid and often hilarious' Washington Post

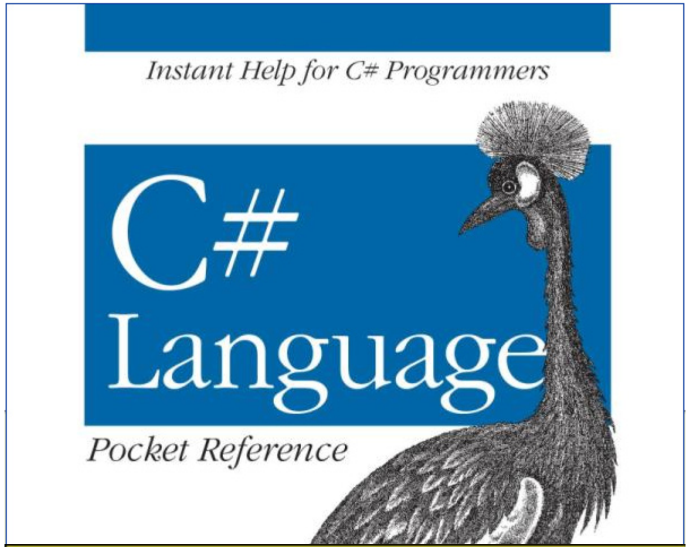

Copyrighted Material
MATHLAND
Michele Emmer
From Flatland
to Hypersurfaces
BIRKHÄUSER
Copyrighted Material

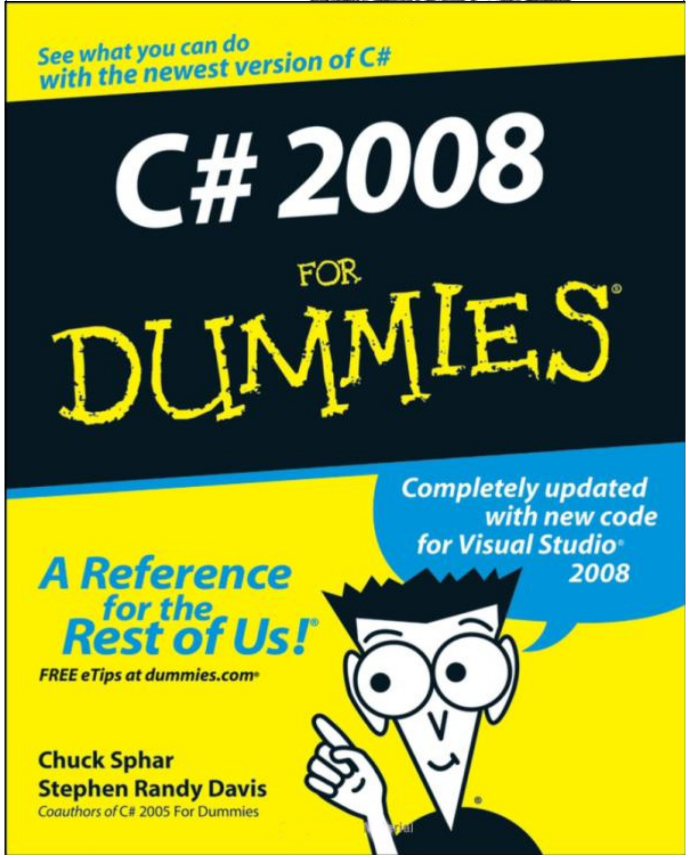

ARCHITECTURAL GEOMETRY
ARCHITECTURAL GEOMETRY
Helmut Pottmann, Andreas Asperl, Michael Hofer, Axel Kilian
Bentley Institute Press
Bentley Institute Press
BENTLEY


Instant Help for C# Programmers
C#
Language
Pocket Reference
O'REILLY®
Peter Drayton, Ben Albahari
& Ted Neward

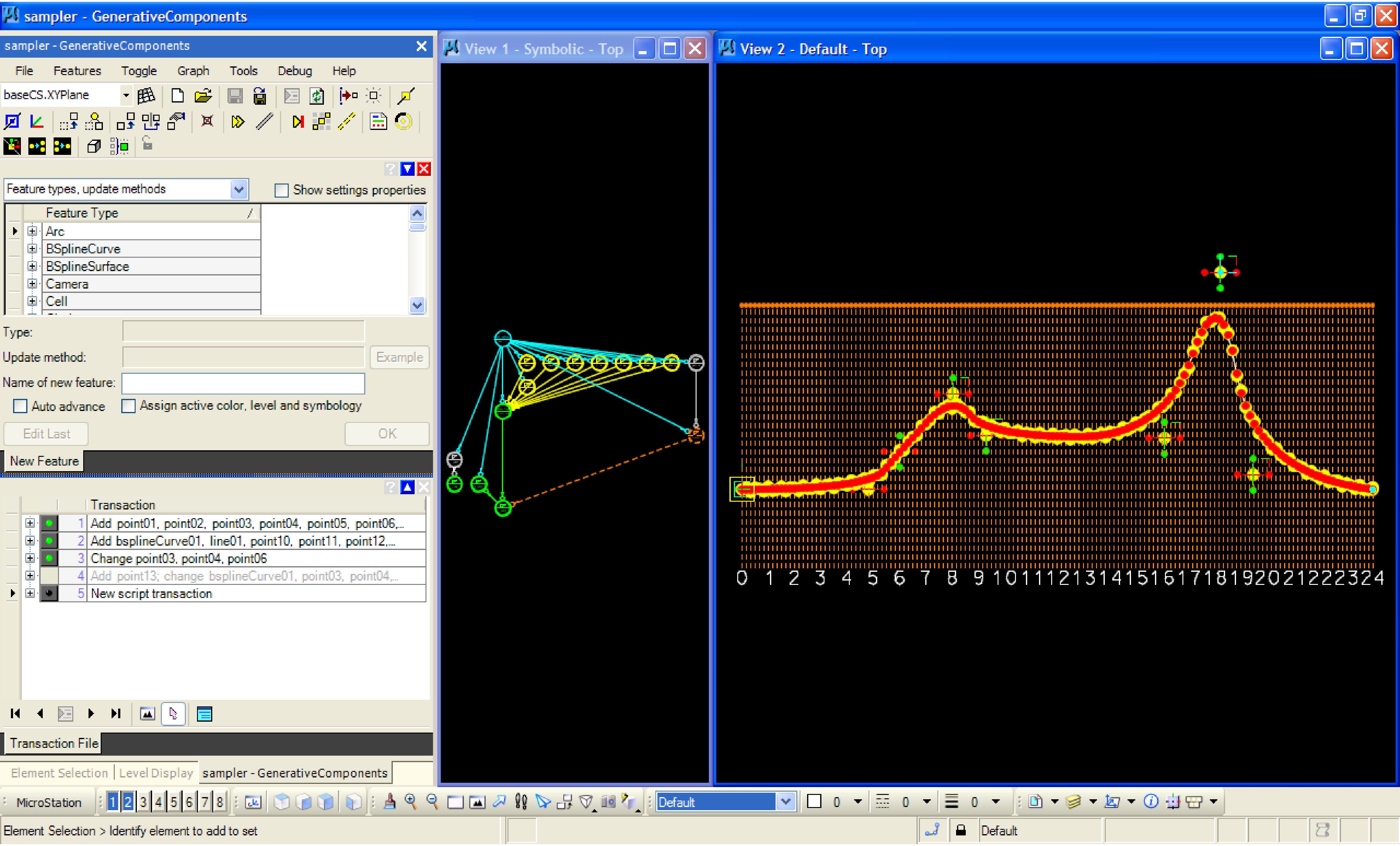

Copyrighted Material
MATHLAND
Michele Emmer
From Flatland to Hypersurfaces
BIRKHÄUSER
Copyrighted Material


ARCHITECTURAL GEOMETRY
Helmut Pottmann, Andreas Asperl, Michael Hofer, Axel Kilian
Bentley Institute Press
ARCHITECTURAL GEOMETRY


Instant Help for C# Programmers
C# Language
Pocket Reference


See what you can do with the newest version of C#
C# 2008
FOR DUMMIES
Completely updated with new code for Visual Studio 2008
A Reference for the Rest of Us!
FREE eTips at dummies.com
Chuck Sphar
Stephen Randy Davis
Coauthors of C# 2005 For Dummies

GC workspace



baseCS.XYPlane

Feature types, update methods

☐ Show settings properties

Feature Type
Arc
BSplineCurve
BSplineSurface
Camera
Cell

Type:

Update method:

Name of new feature:

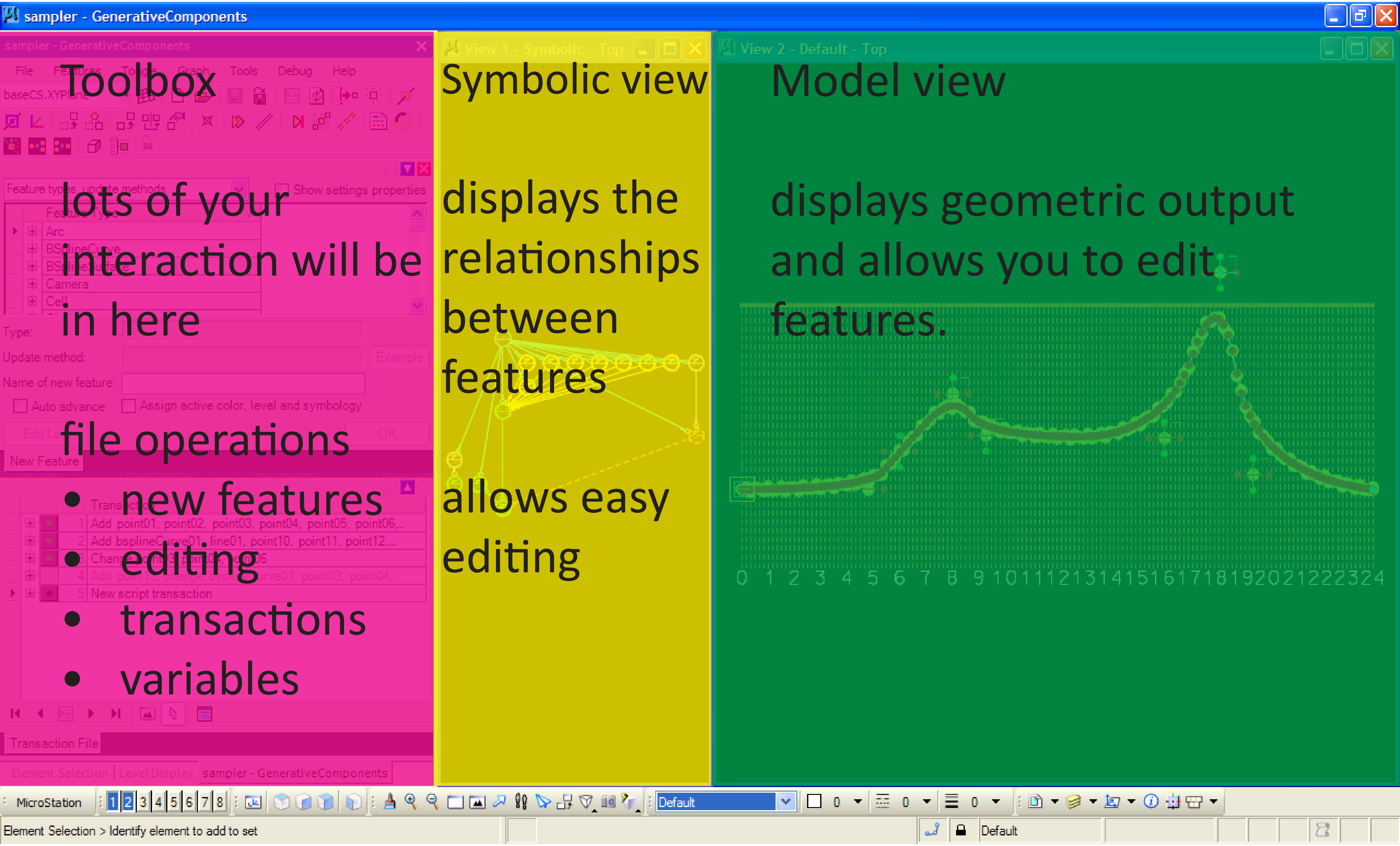
☐ Auto advance ☐ Assign active color, level and symbology

	Transaction
1	Add point01, point02, point03, point04, point05, point06,...
2	Add bsplineCurve01, line01, point10, point11, point12,...
3	Change point03, point04, point06
4	Add point13; change bsplineCurve01, point03, point04,...
5	New script transaction

Default 0 0 0

Element Selection > Identify element to add to set

Default



Toolbox

lots of your interaction will be in here

file operations

- new features
- editing
- transactions
- variables

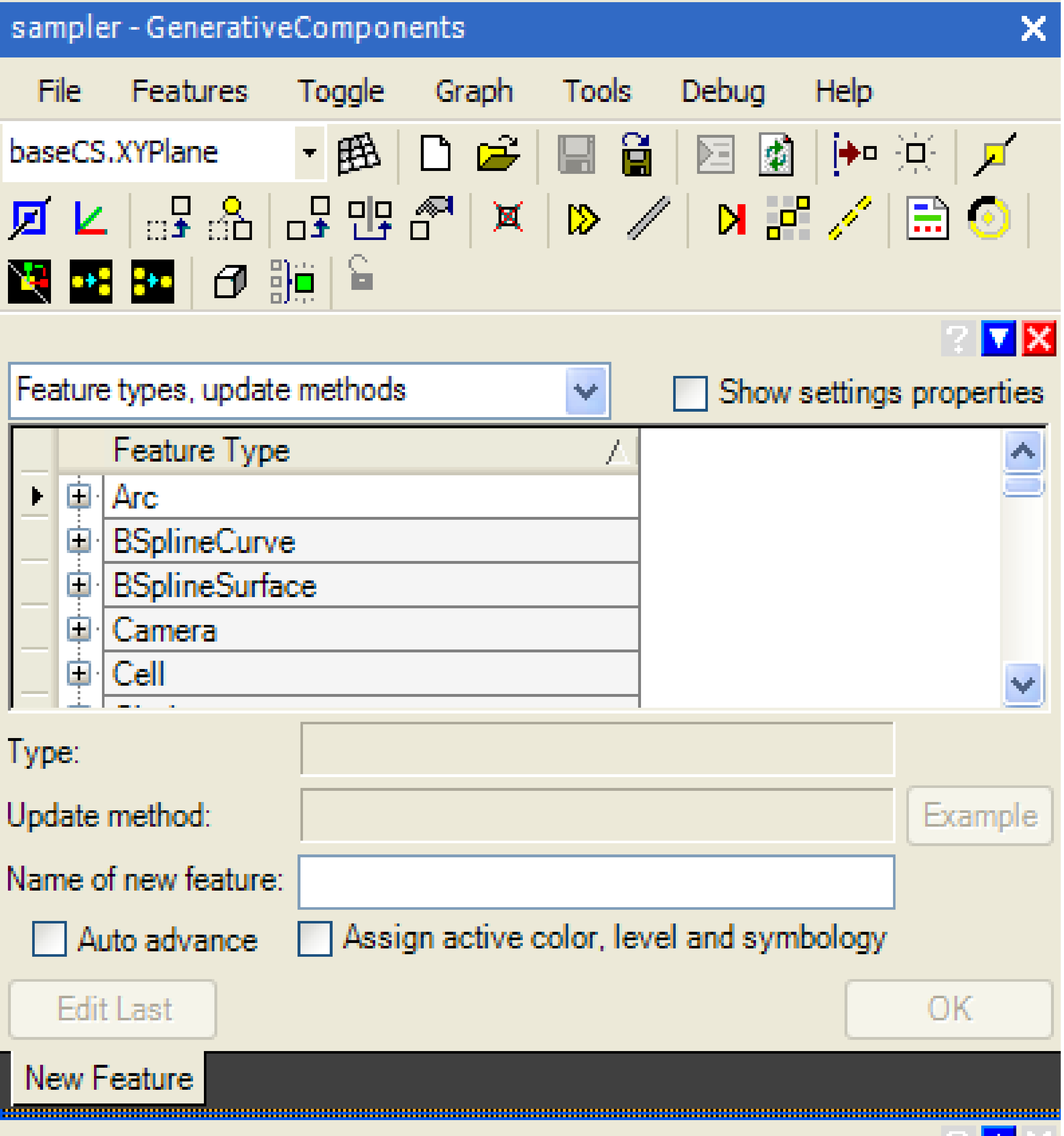
Symbolic view

displays the relationships between features

allows easy editing

Model view

displays geometric output and allows you to edit features.



baseCS.XYPlane

Feature types, update methods

□ Show settings properties

Feature Type
Arc
BSplineCurve
BSplineSurface
Camera
Cell

Type:

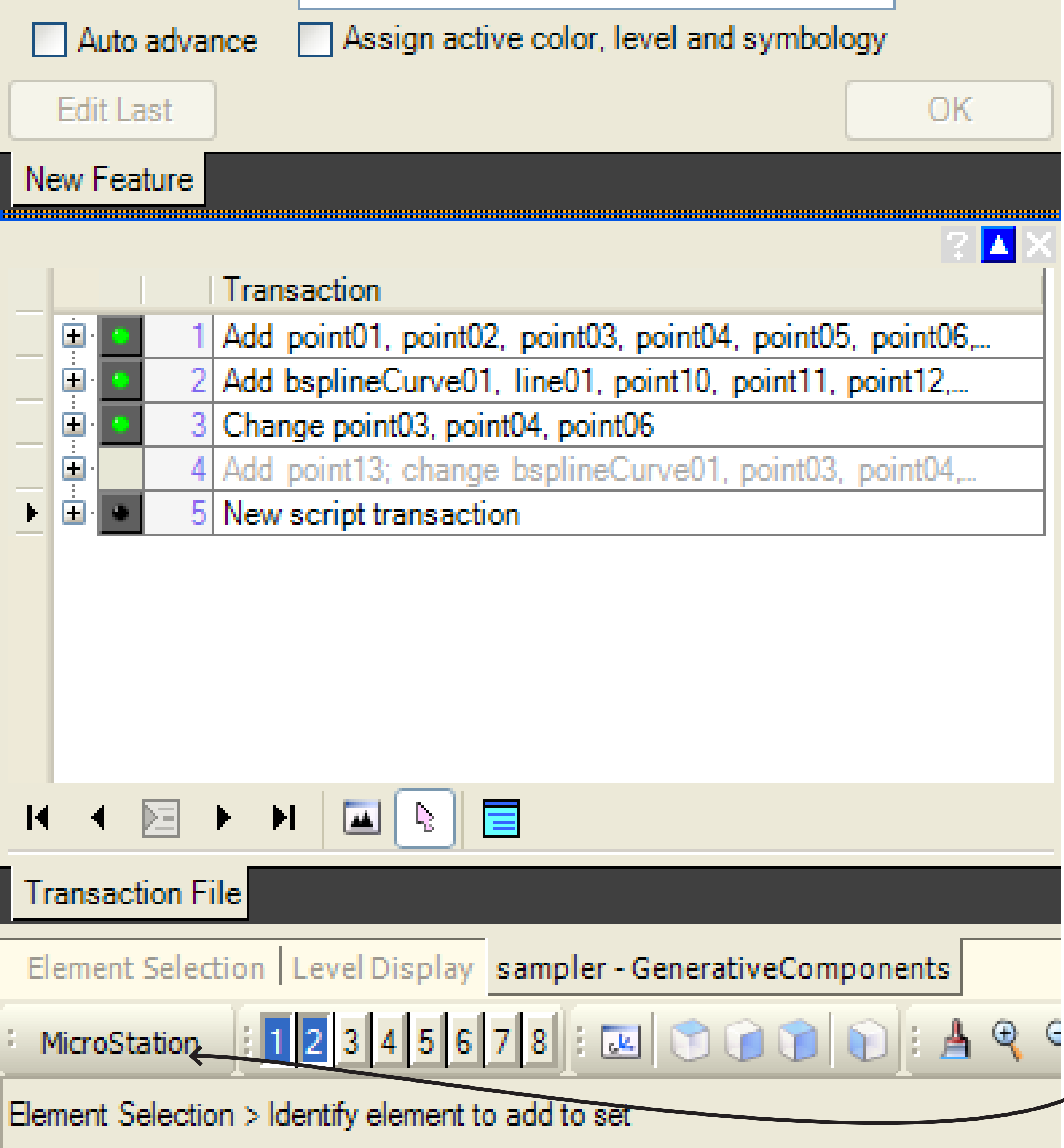
Update method:

Name of new feature:

□ Auto advance □ Assign active color, level and symbology

The tool box has everything you need to open and save files, and a lot of useful tools for editing your models.

The new feature box is where you will make all your new objects in your model.



□ Auto advance □ Assign active color, level and symbology

Edit Last OK

New Feature

	Transaction
1	Add point01, point02, point03, point04, point05, point06,...
2	Add bsplineCurve01, line01, point10, point11, point12,...
3	Change point03, point04, point06
4	Add point13; change bsplineCurve01, point03, point04,...
5	New script transaction

Transaction File

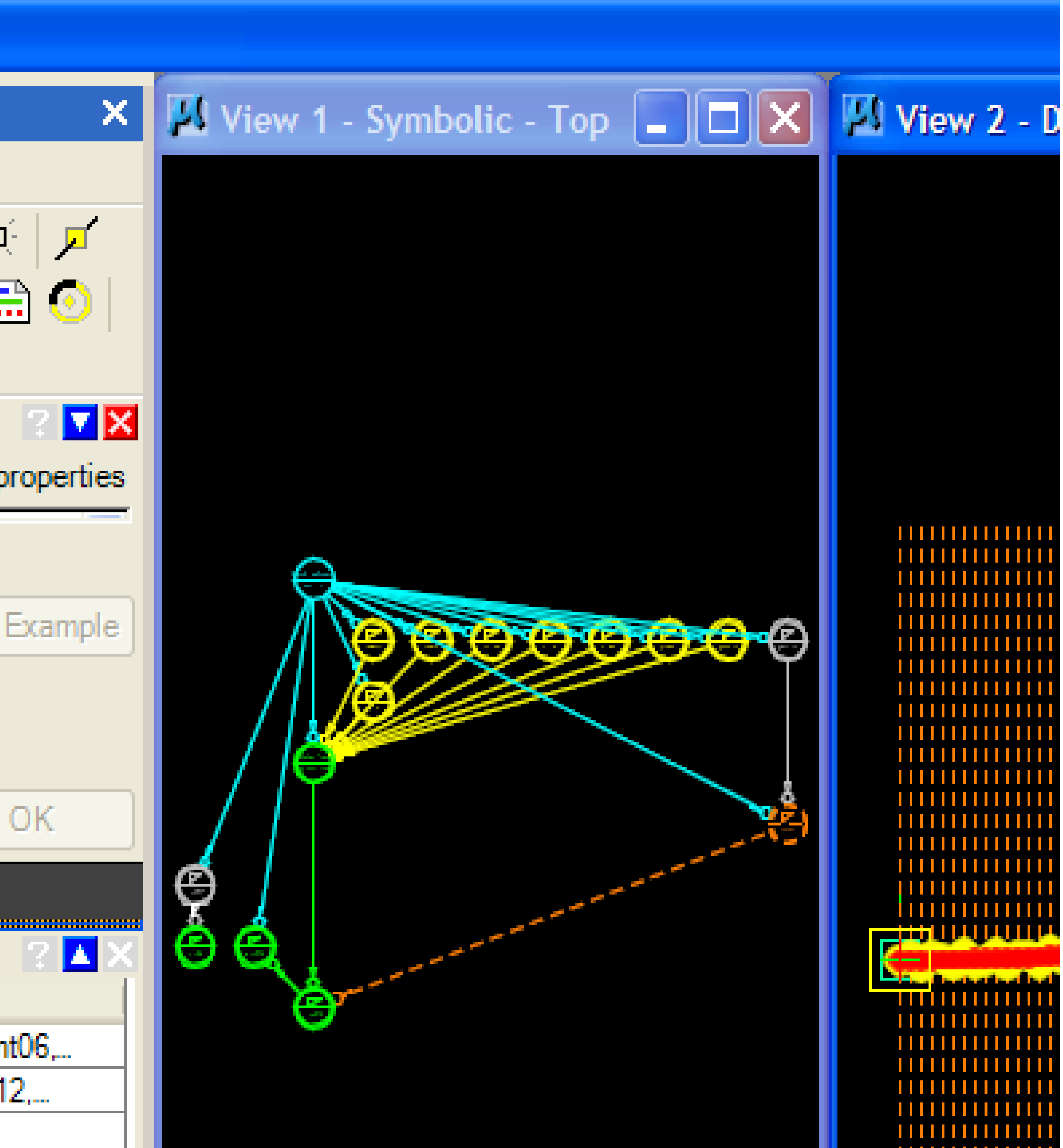
Element Selection | Level Display | sampler - GenerativeComponents

MicroStation 1 2 3 4 5 6 7 8

Element Selection > Identify element to add to set

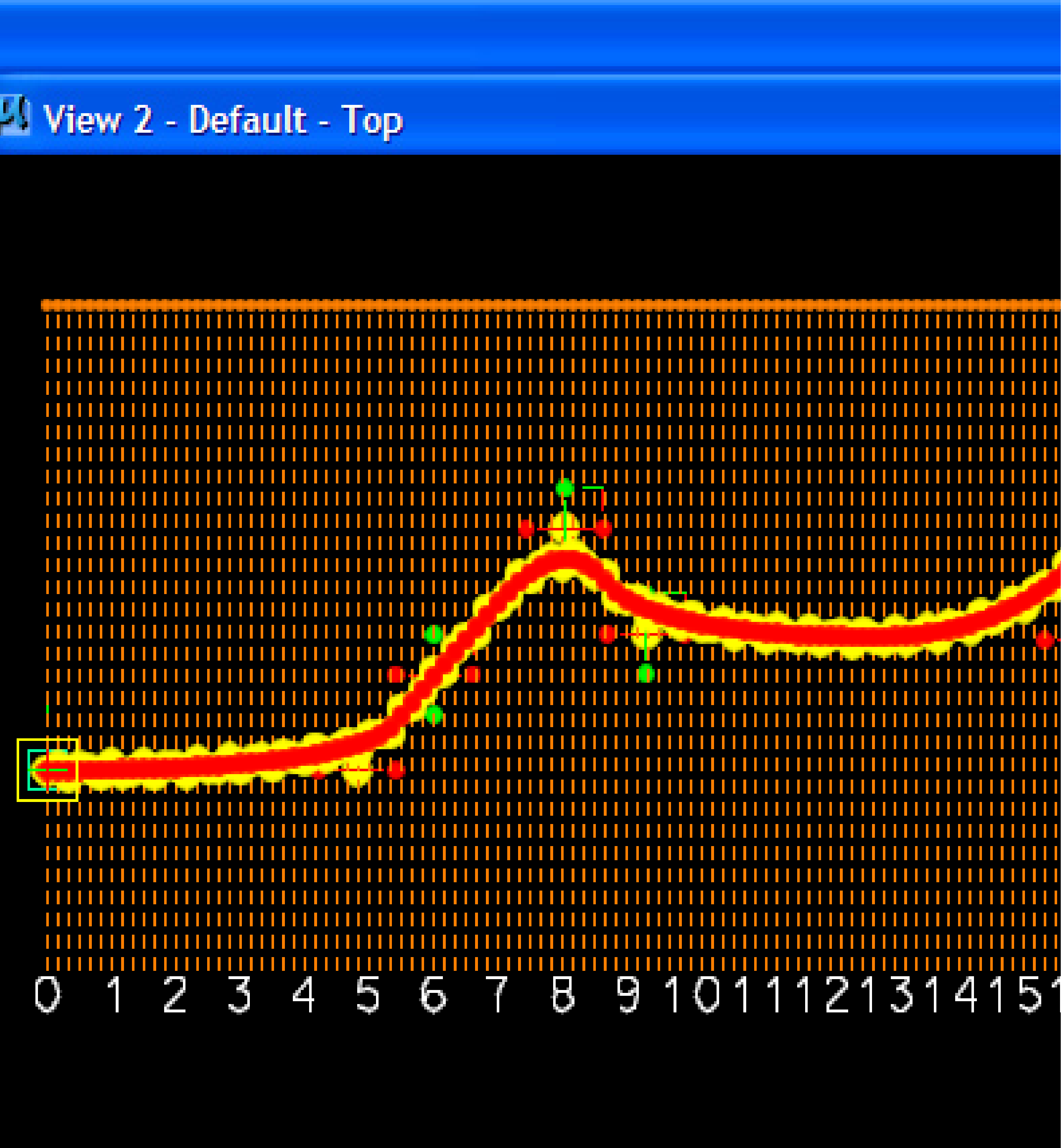
The transaction player saves the current state of the model. This allows you to explain the design process in an incremental way.

The normal Microstation menus hide in here

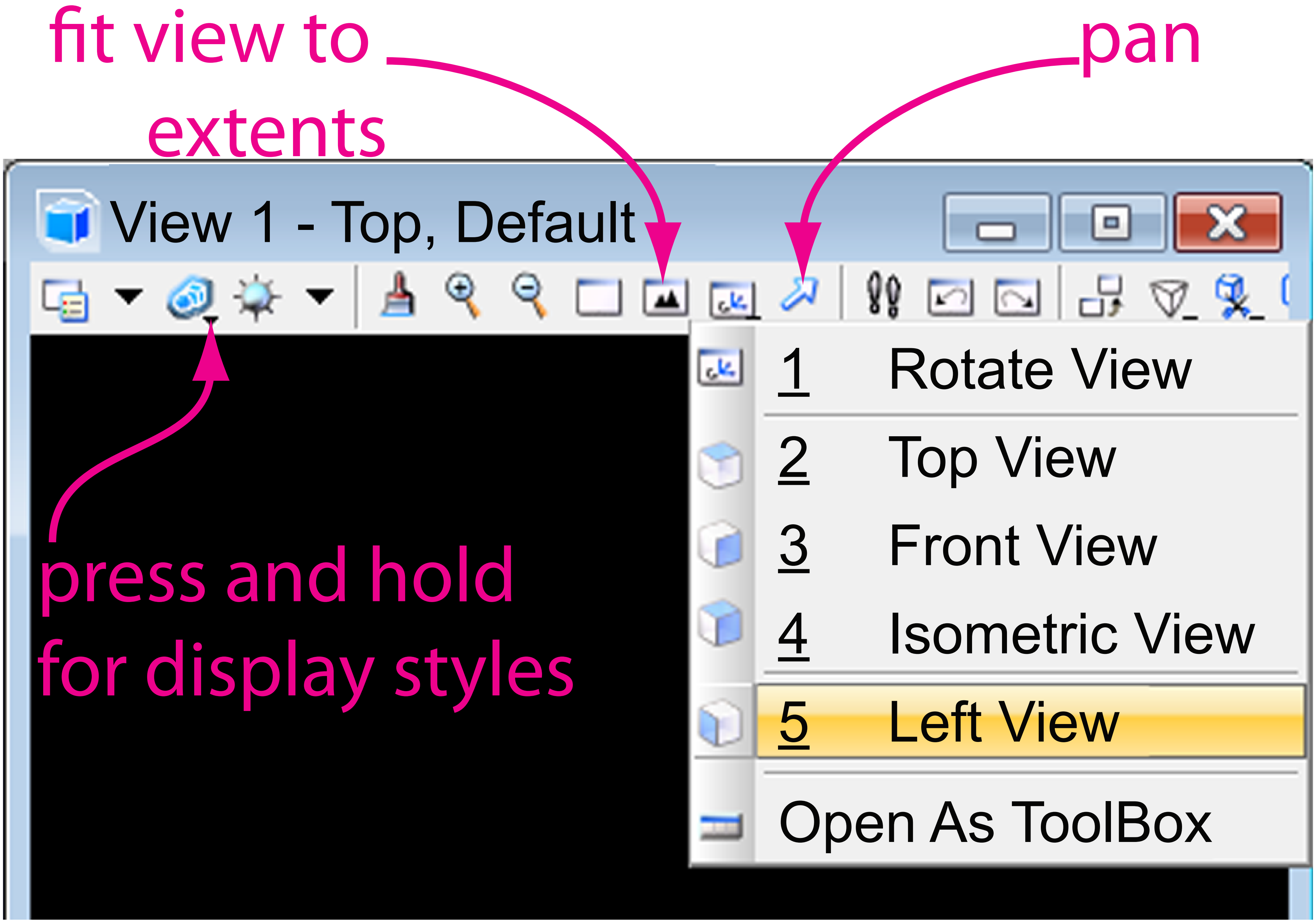


The symbolic view shows how the relationships between the features (both geometric and information based) work.

It allows you to show hidden parts, and to edit hard to reach features.



The default/model view shows you all the geometry. This is the most impressive bit visually, but isn’t always the best place to be working as it can get very busy.



fit view to extents

pan

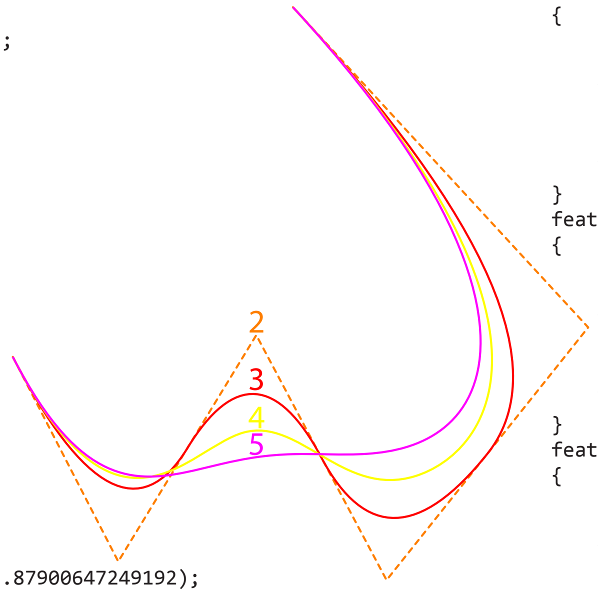
press and hold for display styles

files

```
// Bentley GenerativeComponents Transaction File -- File structure version 1.03.
(Please do not delete or change this line.)

environment
{
    GCVersion                 = “08.09.05.50”;
    MSVersion                 = “08.09.04.51”;
    MSProject                 = “GC_Default”;
    MSDesignFile              = “C:\\Documents and Settings\\Ben\\Local Set-
tings\\Application Data\\Bentley\\MicroStation\\8.9\\s0_tIuz1-SOIbXHdlUtOuQ\\GC\\
workdgn\\$gcworkdgn.tmp”;
}

transaction modelBased “Add bsplineCurve01, point01, point02, point03, point04,
point05, point06”
{
    feature bsplineCurve01 GC.BSplineCurve
    {
        Poles                     = {point01,point02,point03,point04,point05,point06};
        Order                     = {3,4,5};
        SymbolXY                  = {100, 102};
        SymbolicModelDisplay      = null;
        Color                     = {3,4,5};
        ConstructionsVisible      = true;
        FillColor                 = -1;
        Free                      = true;
        IsConstruction            = true;
        Level                     = 0;
        LevelName                 = “Default”;
        LineStyle                 = 0;
        LineStyleName             = “0”;
        LineWeight                = 0;
        MaximumReplication        = true;
        PartFamilyName            = null;
        PartName                  = null;
        RoleInExampleGraph        = null;
        Transparency              = 0.0;
    }
    feature point01 GC.Point
    {
        CoordinateSystem          = baseCS;
        XTranslation              = <free> (-5.87900647249192);
        YTranslation              = <free> (7.77042858980672);
        ZTranslation              = <free> (0.0);
        HandlesVisible            = true;
        Visible                   = false;
    }
```



```
feature point02 GC.Point
{
    CoordinateSystem          = baseCS;
    XTranslation              = <free> (-2.89228802588997);
    YTranslation              = <free> (1.97318303987809);
    ZTranslation              = <free> (0.0);
    HandlesVisible            = true;
    Visible                   = false;
}
feature point03 GC.Point
{
    CoordinateSystem          = baseCS;
    XTranslation              = <free> (1.01679935275081);
    YTranslation              = <free> (8.3852879663143);
    ZTranslation              = <free> (0.0);
    HandlesVisible            = true;
    Visible                   = false;
}
feature point04 GC.Point
{
    CoordinateSystem          = baseCS;
    XTranslation              = <free> (4.70627508090615);
    YTranslation              = <free> (1.4461607171573);
    ZTranslation              = <free> (0.0);
    HandlesVisible            = true;
    Visible                   = false;
}
feature point05 GC.Point
{
    CoordinateSystem          = baseCS;
    XTranslation              = <free> (10.4161779935275);
    YTranslation              = <free> (8.60488060078129);
    ZTranslation              = <free> (0.0);
    HandlesVisible            = true;
    Visible                   = false;
}
feature point06 GC.Point
{
    CoordinateSystem          = baseCS;
    XTranslation              = <free> (2.05446440129452);
    YTranslation              = <free> (17.6740564042681);
    ZTranslation              = <free> (0.0);
    HandlesVisible            = true;
    Visible                   = false;
}
}
```

This is the entire file for generating these splines.