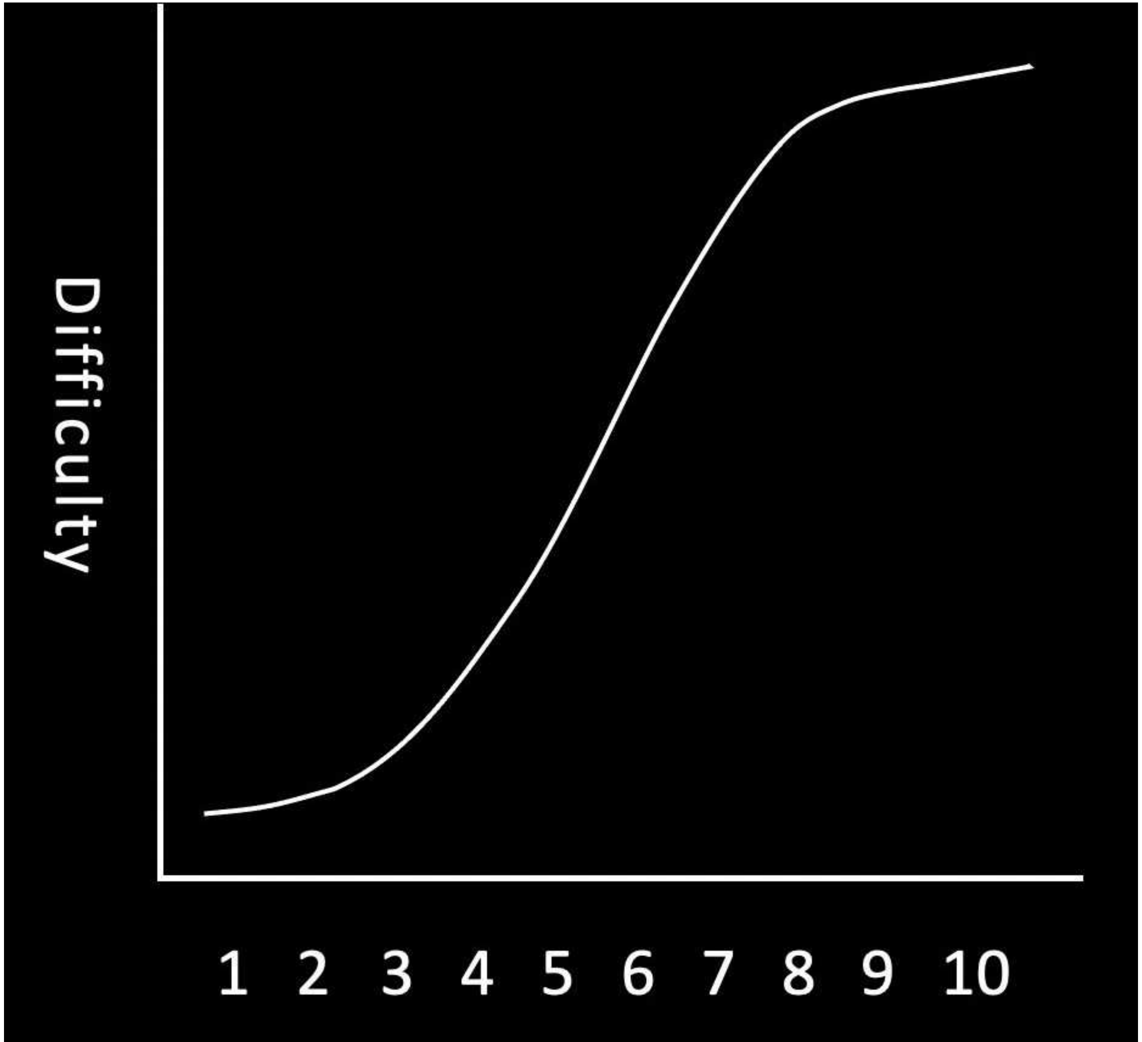
COMS30020 - Computer Graphics

Week 3 Briefing

Dr Simon Lock

Where are we ?



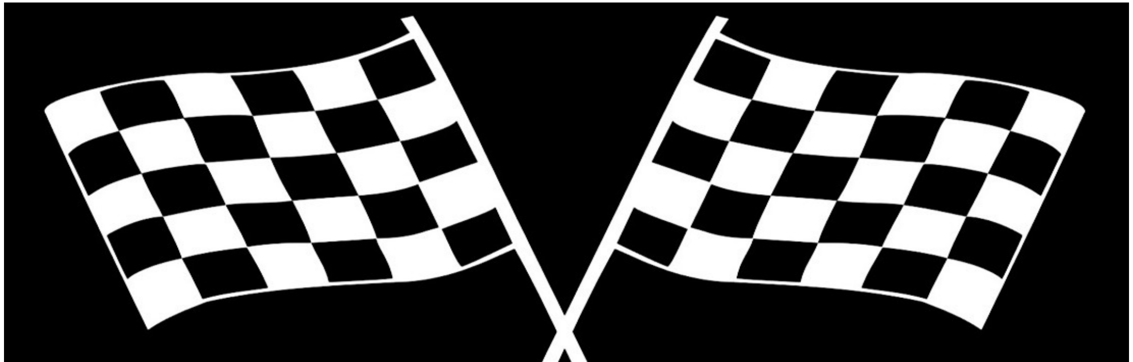
This week on "Computer Graphics"

We actually start doing some structured drawing !

Our focus will be on a key drawing primitive:

△ △ △ "The Triangle" △ △ △

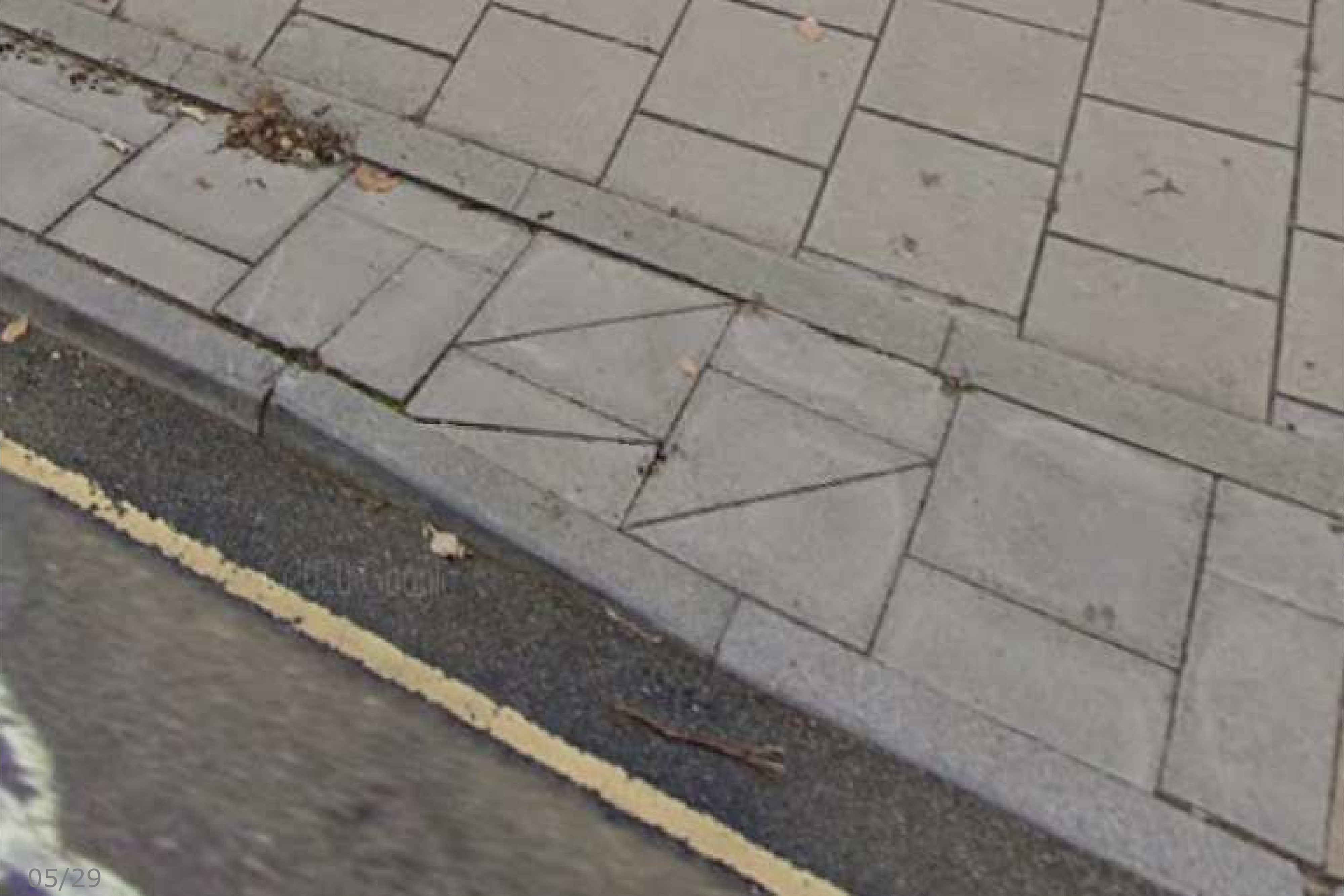
The primary building block for the rest of this unit !!!



Powerful Things

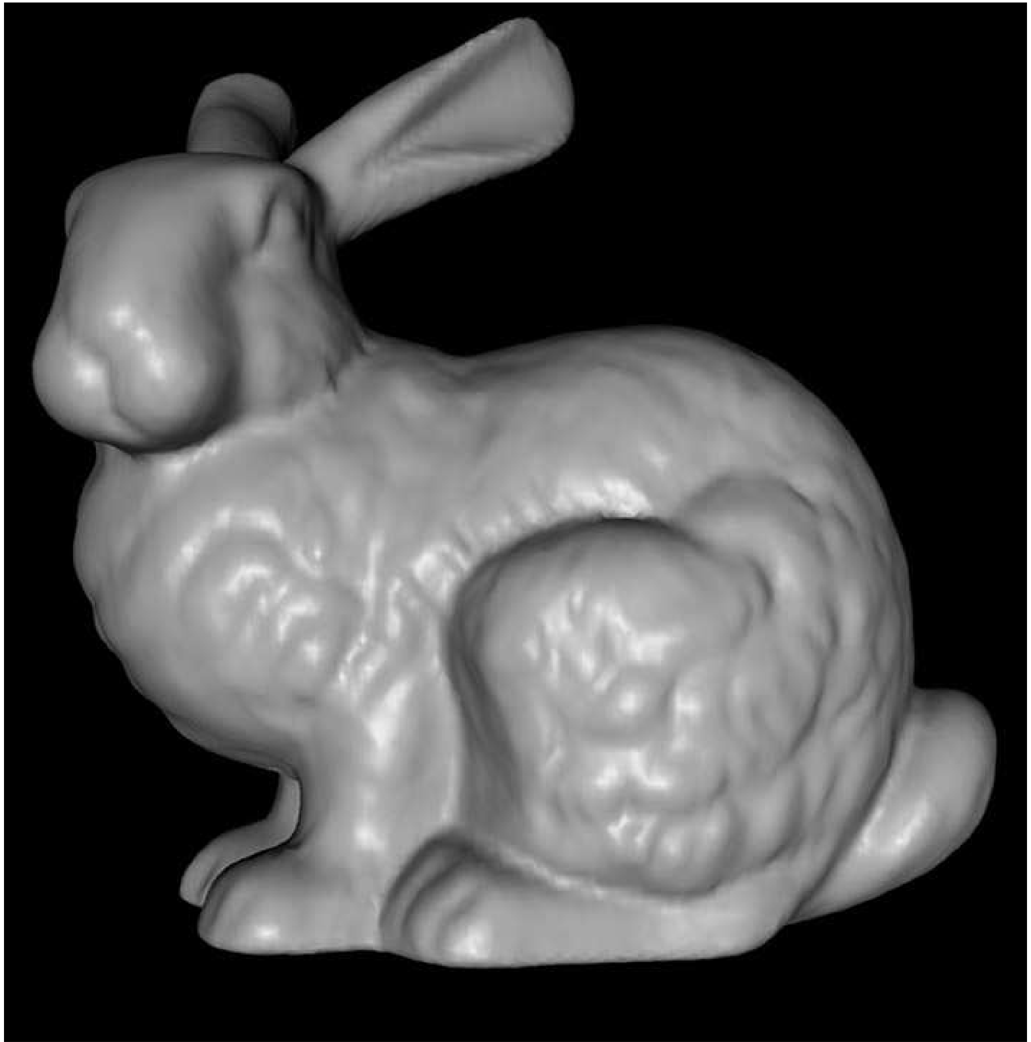
Although simple, Triangles are VERY powerful

A convenient structure to cover ANY surface

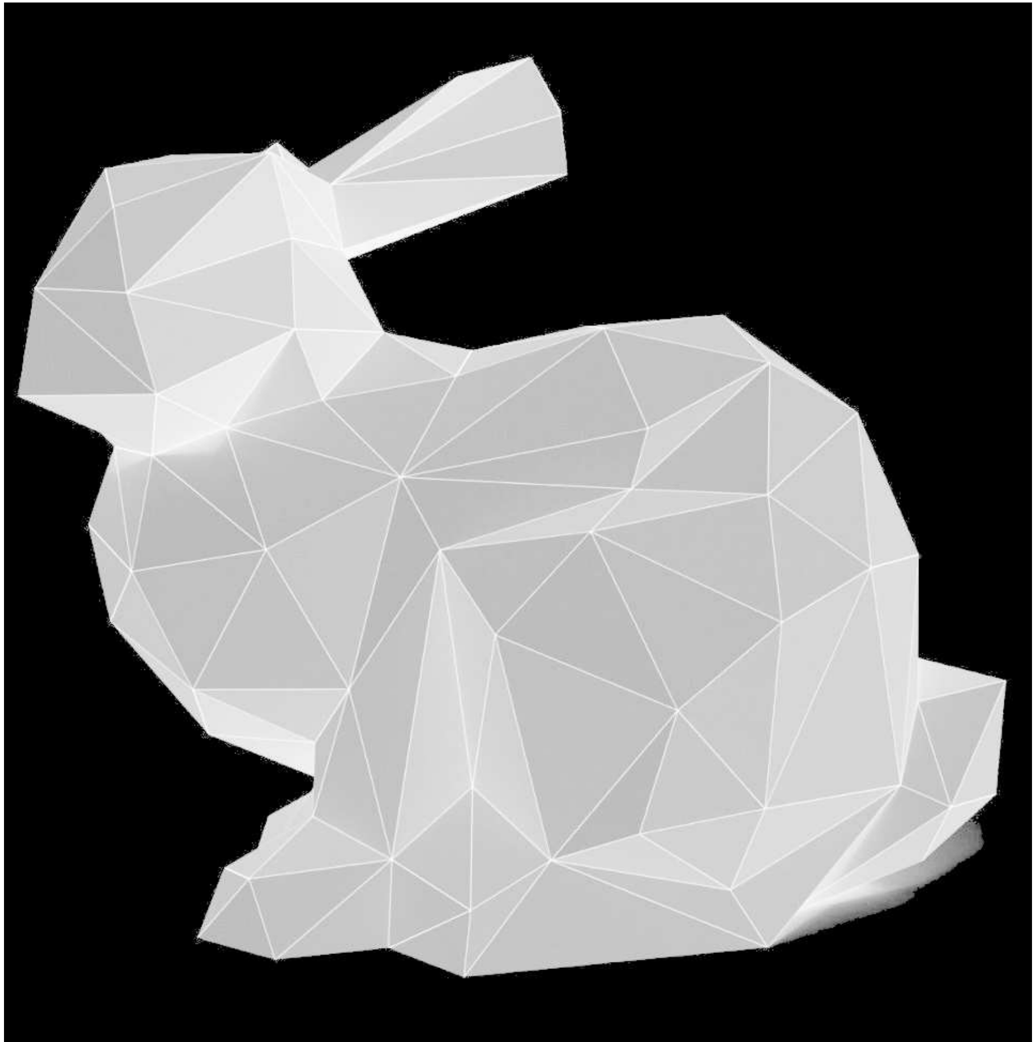


Using MANY triangles, we can create complex shapes

The Stanford Bunny (70k triangles)



Low Poly Equivalent



What kinds of Triangle ?

This week we'll draw various types of triangle:

- Unfilled (also known as "stroked") triangles
- Filled triangles (with a choice of colours !)
- Composite triangles (filled AND stroked !!!)

<u>THE CHALLENGE</u>

We won't use an existing `drawTriangle` function
We will instead be drawing our own triangles
This is however non-trivial (the devil is in the detail)
There is hidden complexity that we must deal with !

Challenges

Surely a "stroked" triangle is easy ?
It's just three straight lines ! △

Sure, currently we don't have a `drawLine` function
But we can just use the `setPixelColour` function...
Call it lots of times to draw a sequence of pixels

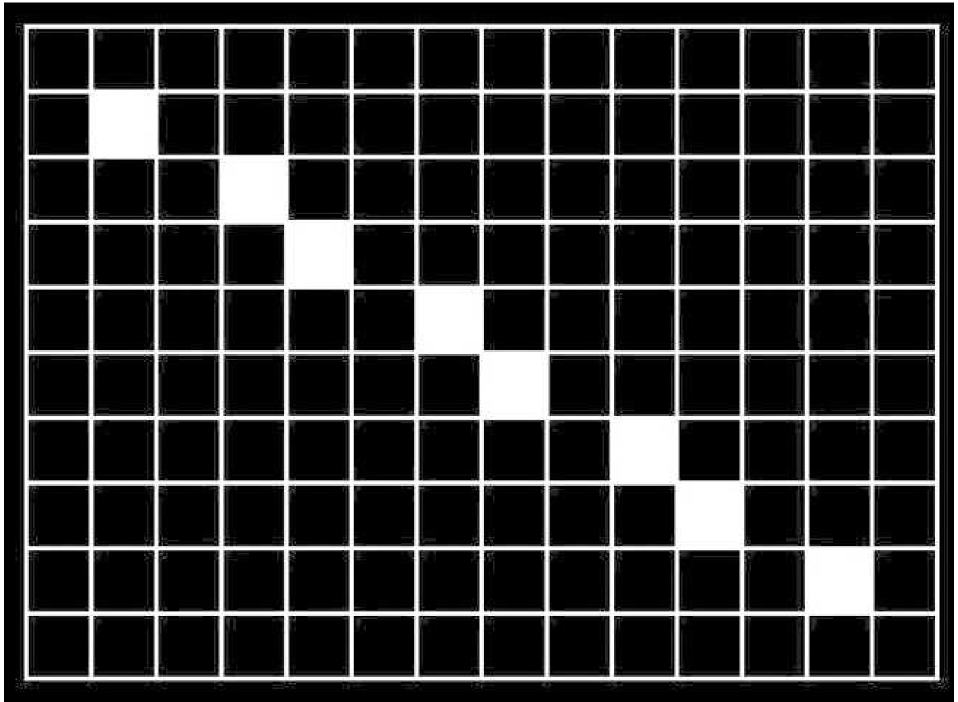
We'll just need to calculate X and Y
for each point along the line
(with a nice bit of interpolation)

Not that easy !

If we just loop through each ROW like this:

```
for(int y=1; y<=8 ;y++) {
```

We could end up drawing something like this:

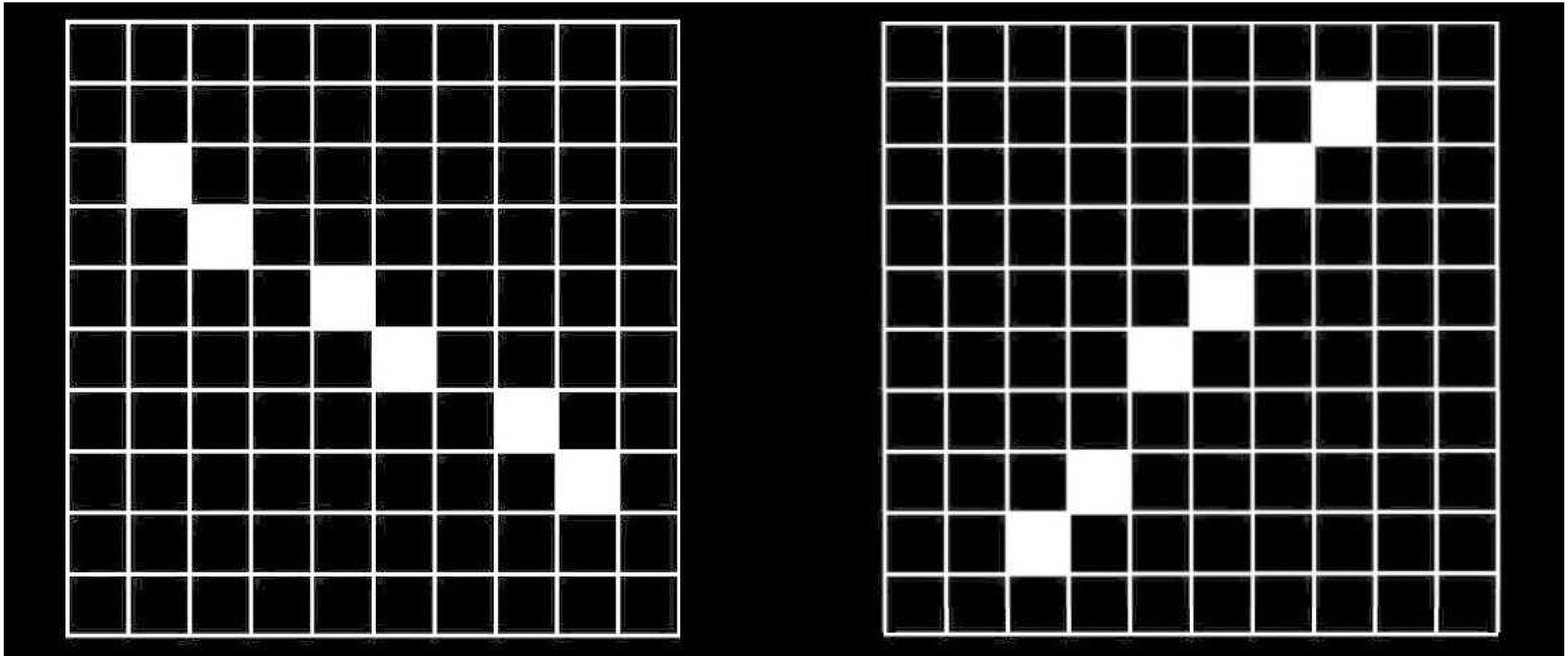


Needs Careful Handling

Neither can we ALWAYS just loop through columns

We must consider gradient of line when drawing pixels

Workbook provides details on how to deal with this



Filled Triangles - Any Easier ?

Filled triangles must surely be easier then ?
All we need to do is draw some horizontal lines
Starting and ending at the correct x positions

We just draw from "the left-hand side"...
All the way to "the right-hand side"
One row at a time
From top to bottom
Easy ?

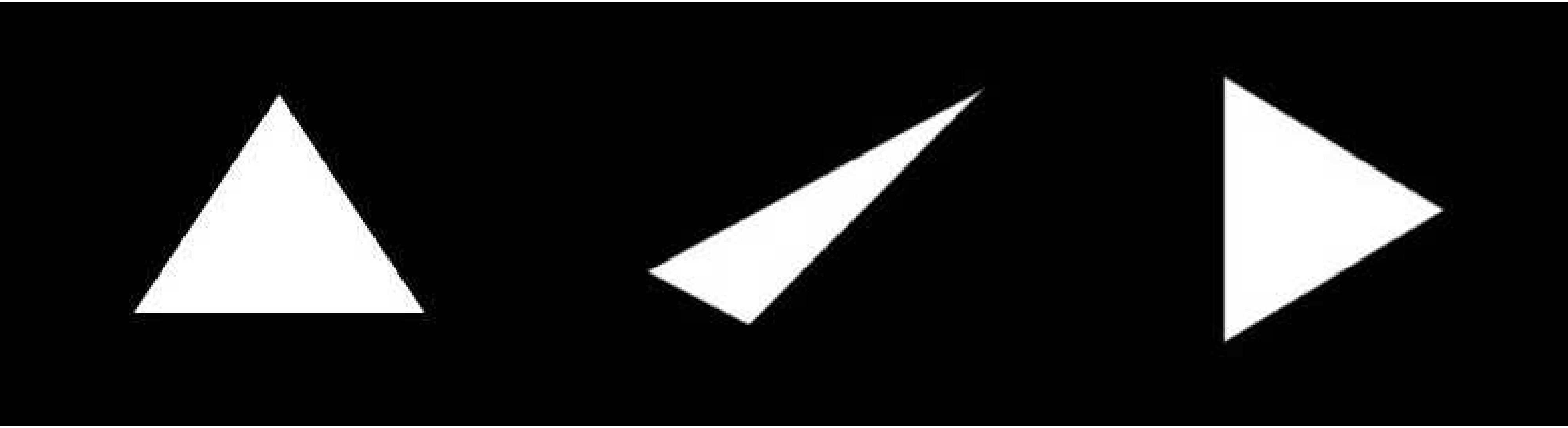
Different types of triangle

The first triangle below (a "flat bottomed") is easy !

However, non flat top/bottom triangles are harder

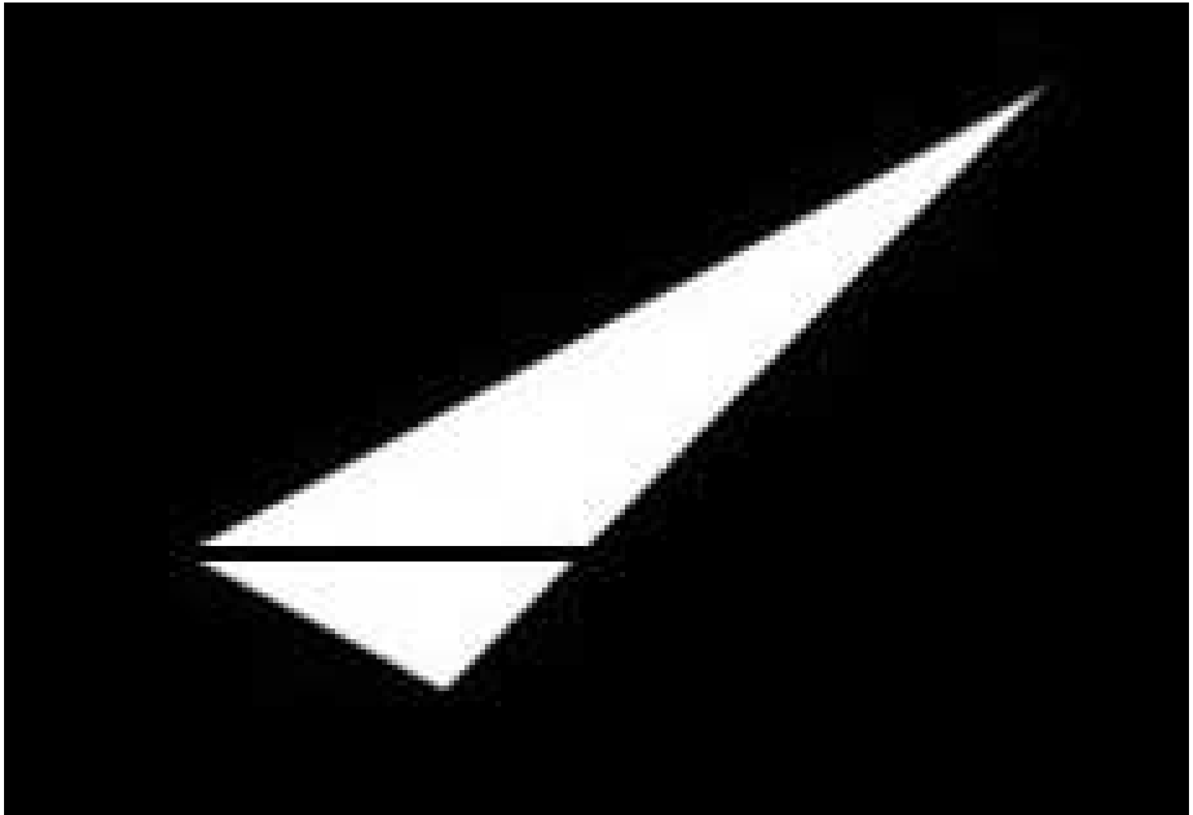
There isn't a "left-hand side" and "right-hand side"

This is because each triangle has THREE sides...



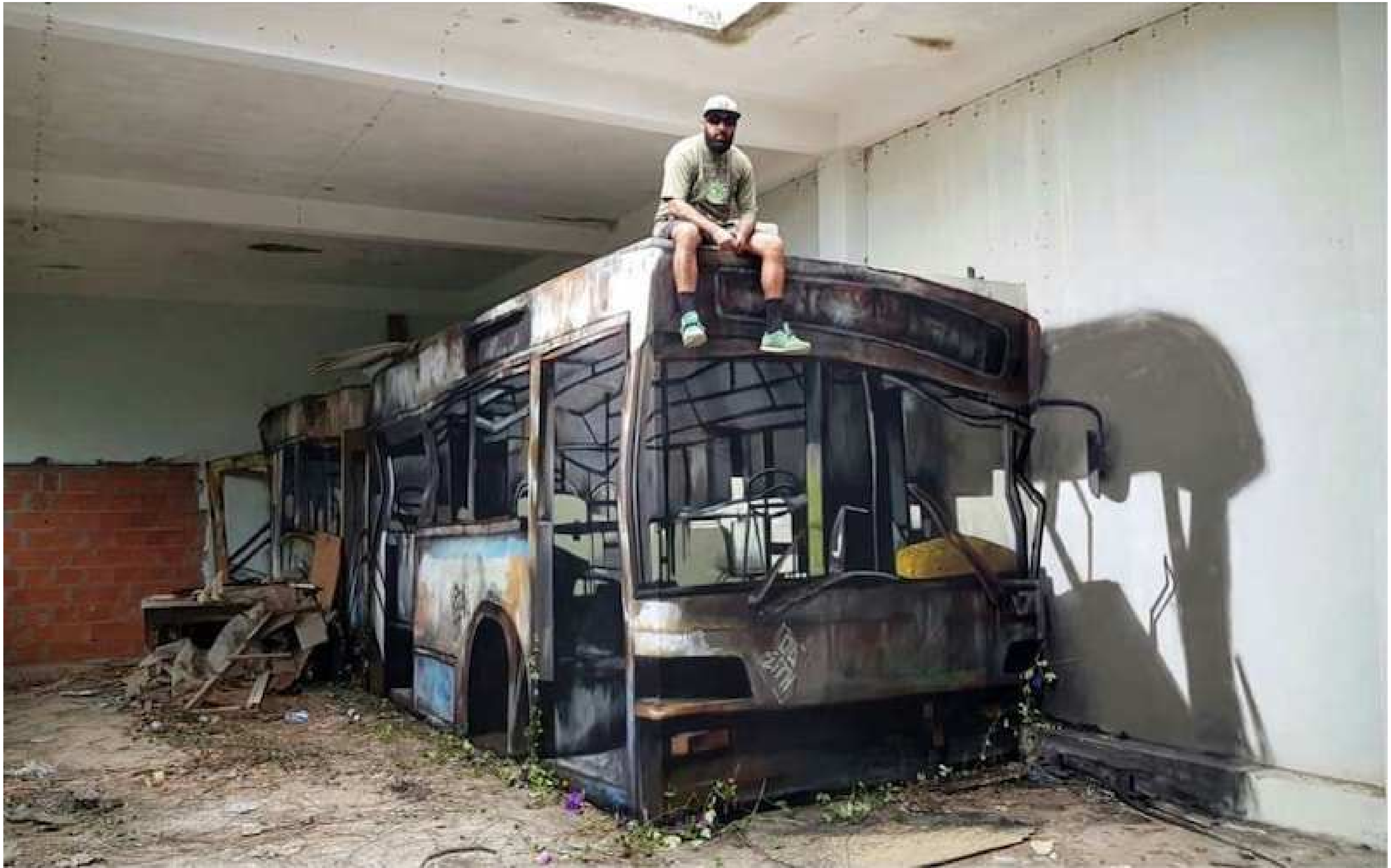
Workbook explains a technique for dealing with this !

(We basically split each triangle into 2 easier ones)

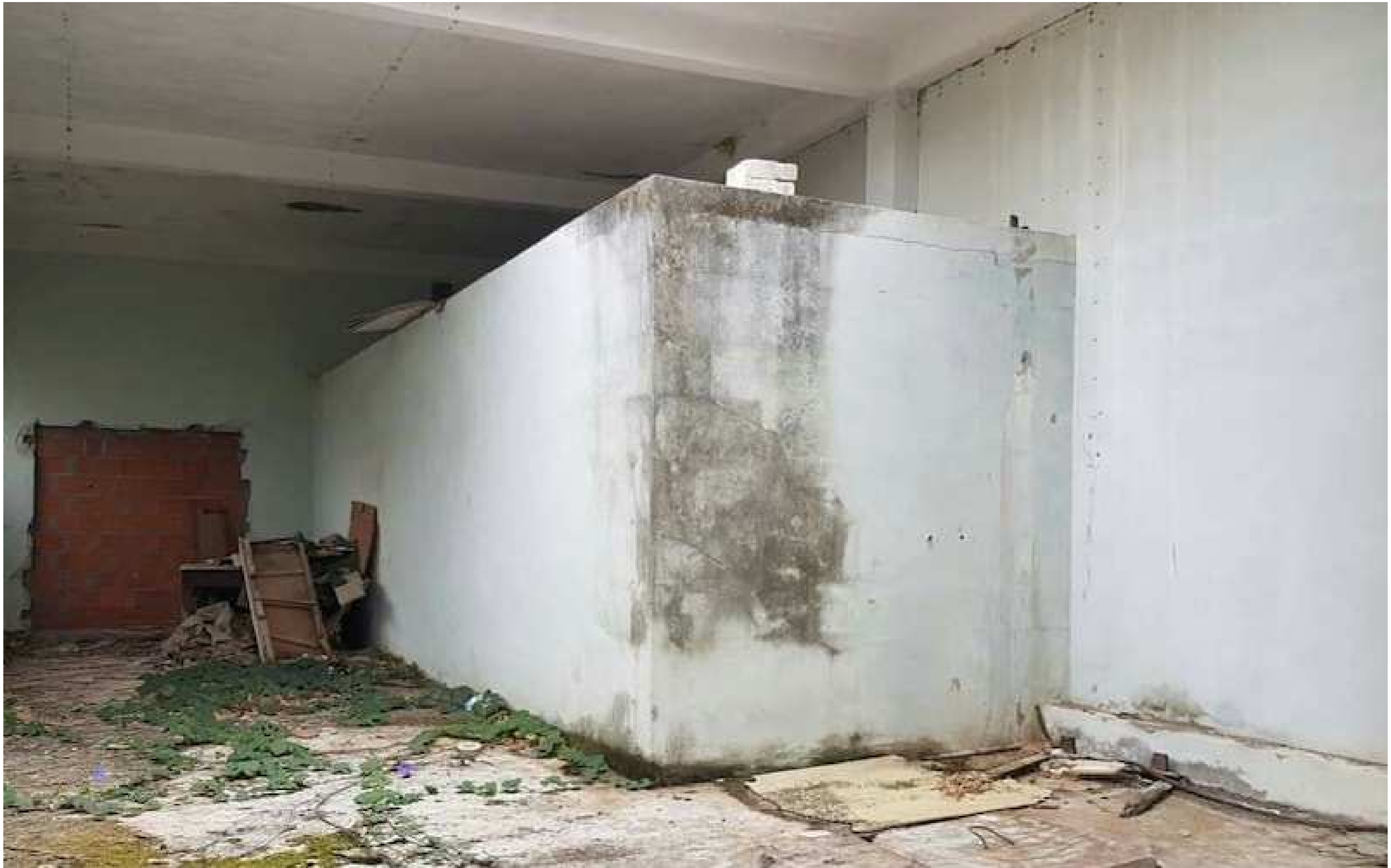


And if all that wasn't exciting enough...

Sergio Odeith

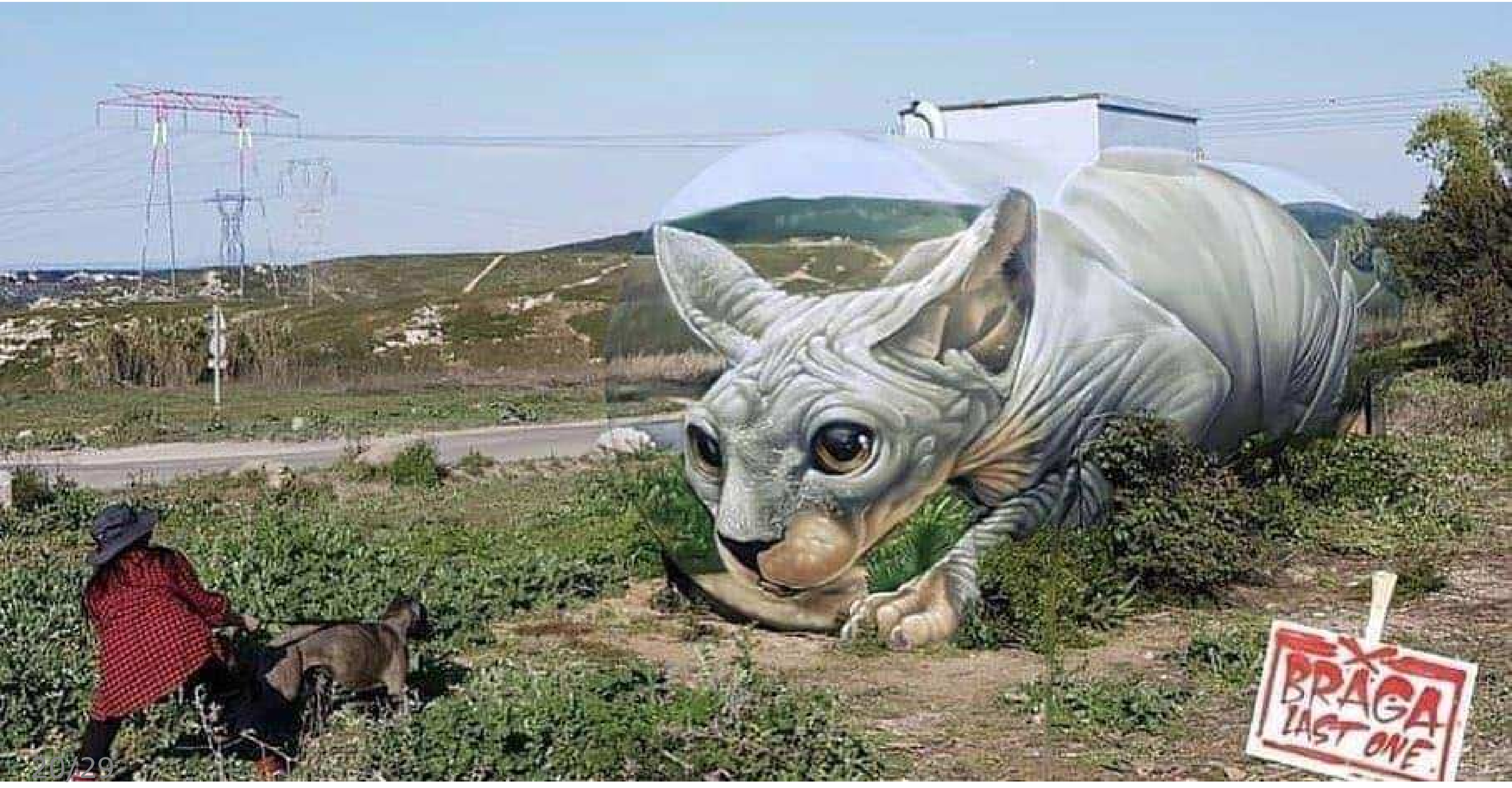


Sergio Odeith



Tom Bragado Blanco

Tom Bragado Blanco

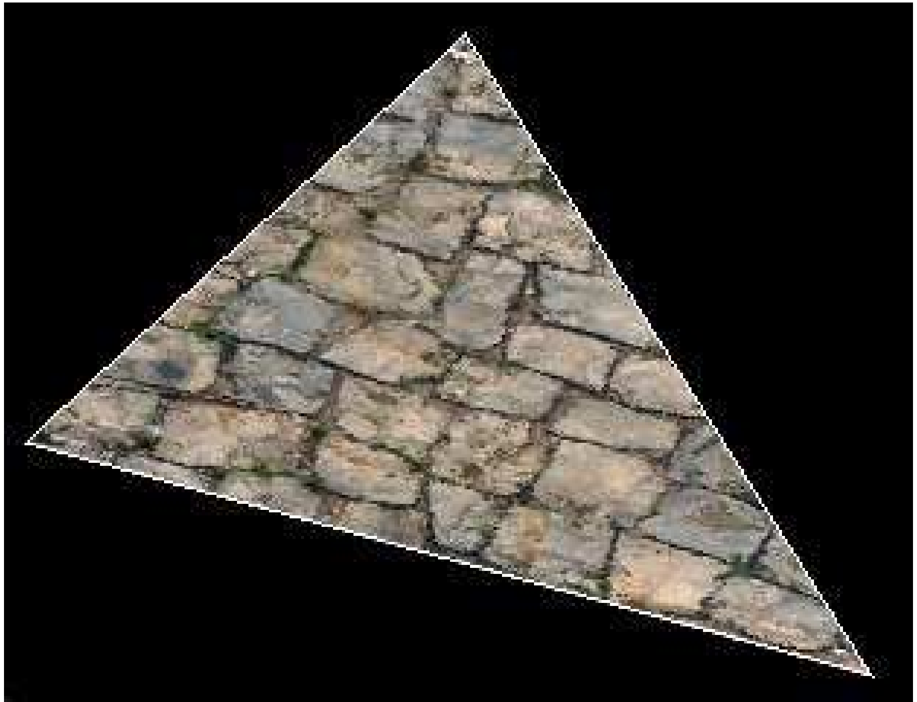


Your Objective

A texture is provided for you in the workbook

A reference image illustrates your final objective

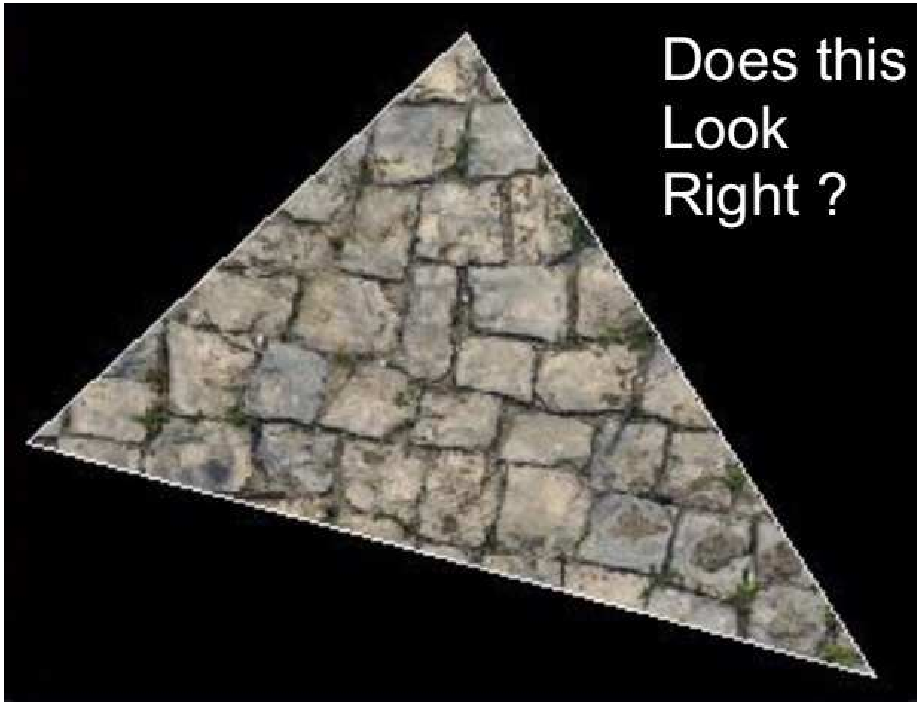
This allows you to "visually verify" your success



Implicit Feedback

Reference image provides passive/implicit feedback

You can gauge how well you are currently doing and what aspects of your work need improving



Approach to Texture Mapping

Texture mapping is a non-trivial activity

(as with many things in Computer Graphics !)

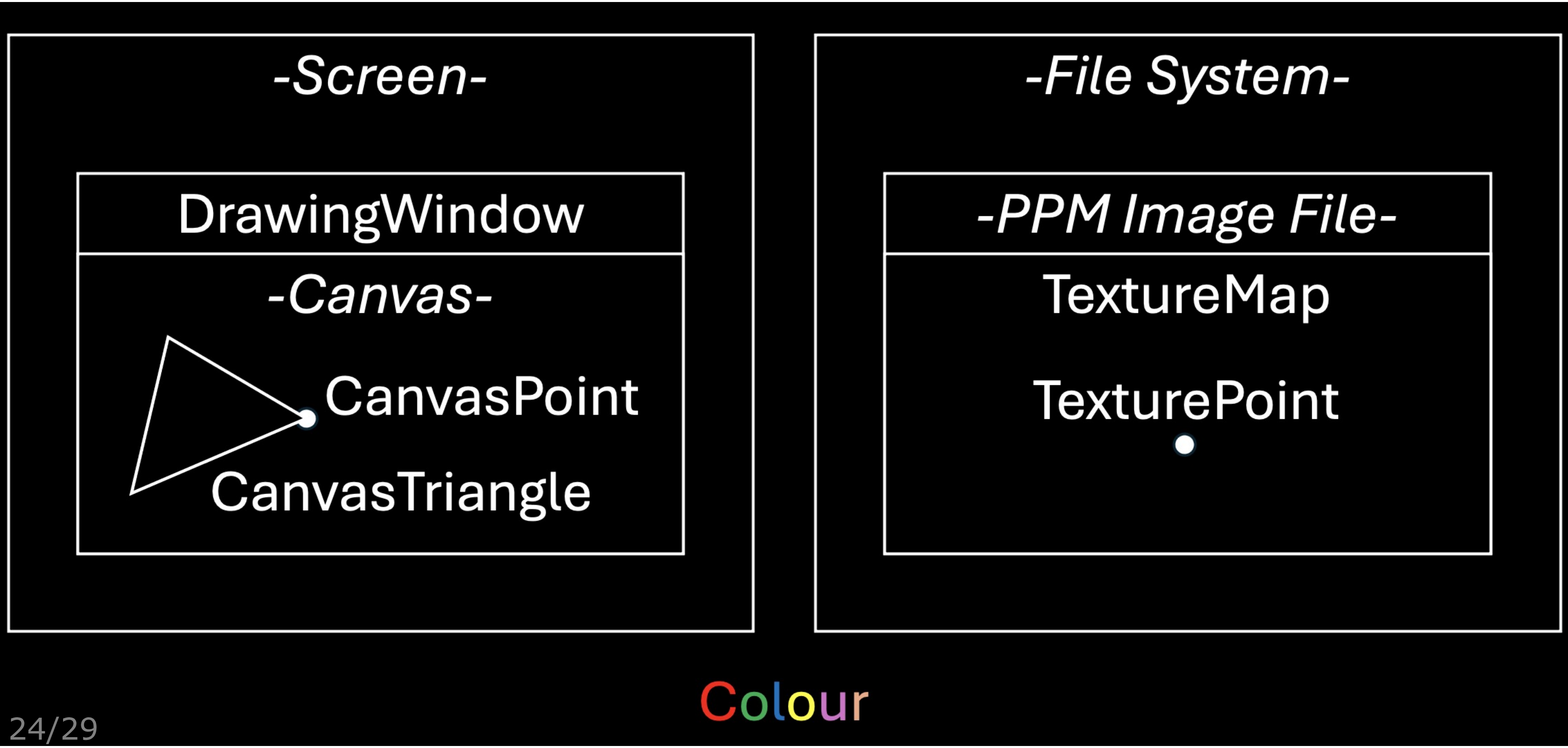
There is some hidden complexity to deal with

Let's explore an animation for more detail...

TextureMapper

Key Concepts and Provided Classes

All class source files can be found in libs/sdw



-Screen-

DrawingWindow

-Canvas-

CanvasPoint

CanvasTriangle

-File System-

-PPM Image File-

TextureMap

TexturePoint

Colour

Top Tips

You are about to embark on some complex coding

Just because it is "graphical" and "mathematical"
You shouldn't forget good programming practice

DON'T try to write it as one monolithic function
Remember to "Divide and Conquer"

Try to write modular reusable functions...

Potentially Useful Reusable Functions

Given a start and end point of a line on a CANVAS...
Return a vector of ALL pixel positions along that line

Given a start and end point of a line on a CANVAS...
Return the position of a SINGLE pixel on that line...
A specified PROPORTIONAL DISTANCE along the line

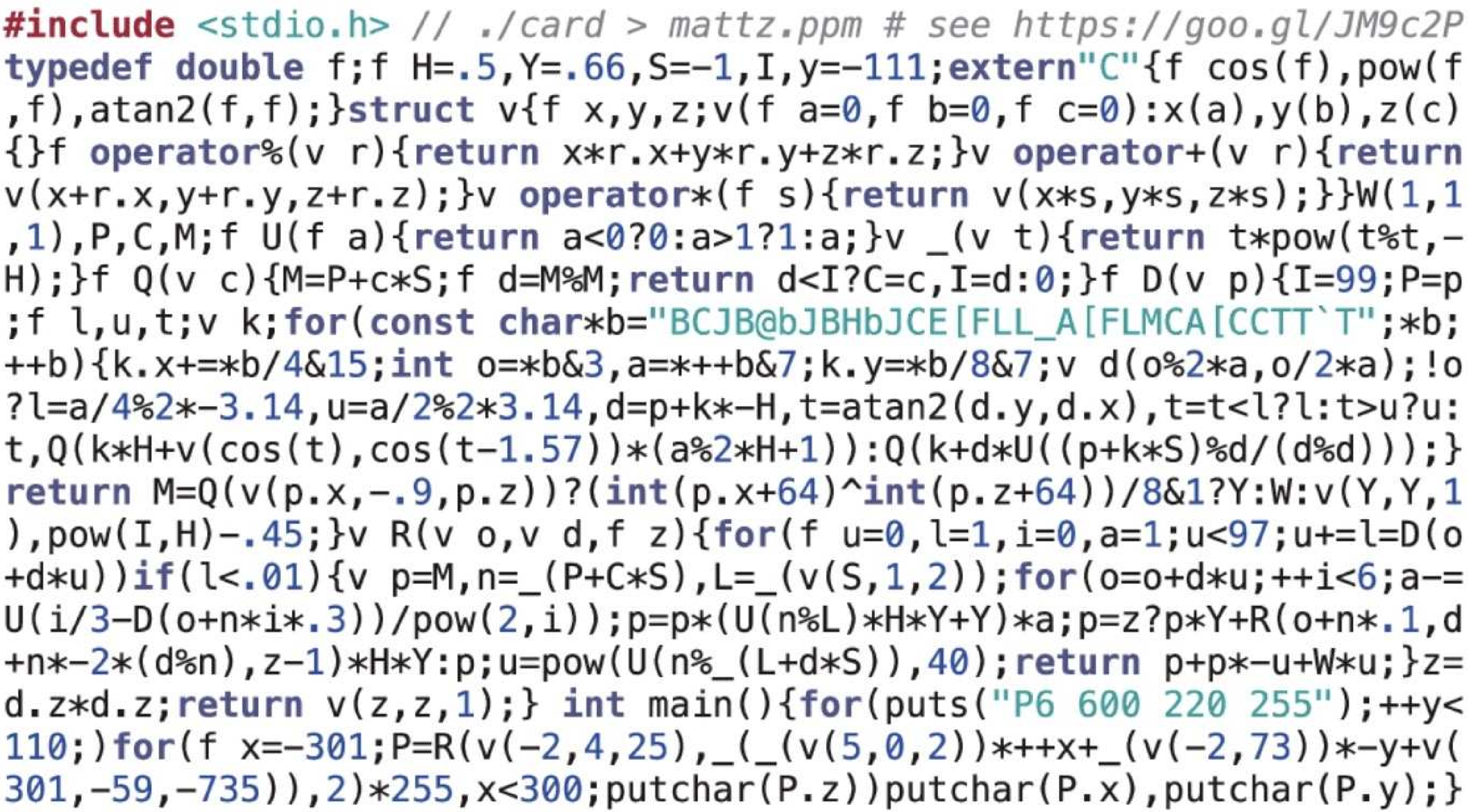
Given a start and end point of a line on a TEXTURE...
Return the position of a SINGLE pixel on that line...
A specified PROPORTIONAL DISTANCE along the line

Meaningful Names

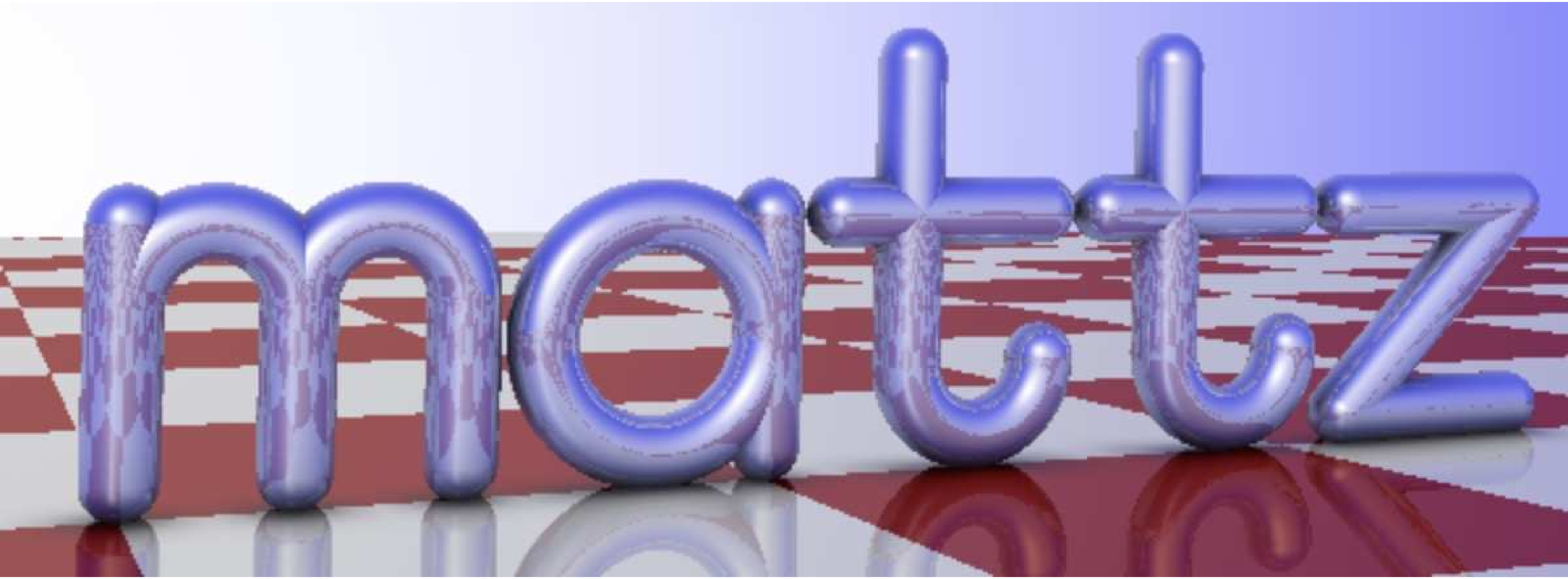
Mathematicians use very short names [x,y,z]
This is fine for tightly constrained problems

Complex and extensive code is very different
We soon get into difficulty if all names are short

Choose useful and meaningful names for everything
Makes everyone's life much easier when debugging



```
#include <stdio.h> // ./card > mattz.ppm # see https://goo.gl/JM9c2P
typedef double f;f H=.5,Y=.66,S=-1,I,y=-111;extern"C"{f cos(f),pow(f
,f),atan2(f,f);}struct v{f x,y,z;v(f a=0,f b=0,f c=0):x(a),y(b),z(c)
{}f operator%(v r){return x*r.x+y*r.y+z*r.z;}v operator+(v r){return
v(x+r.x,y+r.y,z+r.z);}v operator*(f s){return v(x*s,y*s,z*s);}}W(1,1
,1),P,C,M;f U(f a){return a<0?0:a>1?1:a;}v _(v t){return t*pow(t%t,-
H);}f Q(v c){M=P+c*S;f d=M%M;return d<I?C=c,I=d:0;}f D(v p){I=99;P=p
;f l,u,t;v k;for(const char*b="BCJB@bJBHbJCE[FLL_A[FLMCA[CCTT`T";*b;
++b){k.x+=*b/4&15;int o=*b&3,a=*++b&7;k.y=*b/8&7;v d(o%2*a,o/2*a);!o
?l=a/4%2*-3.14,u=a/2%2*3.14,d=p+k*-H,t=atan2(d.y,d.x),t=t<l?l:t>u?u:
t,Q(k*H+v(cos(t),cos(t-1.57))*(a%2*H+1)):Q(k+d*U((p+k*S)%d/(d%d)));}
return M=Q(v(p.x,-.9,p.z))?(int(p.x+64)^int(p.z+64))/8&1?Y:W:v(Y,Y,1
),pow(I,H)-.45;}v R(v o,v d,f z){for(f u=0,l=1,i=0,a=1;u<97;u+=l=D(o
+d*u))if(l<.01){v p=M,n=_(P+C*S),L=_(v(S,1,2));for(o=o+d*u;++i<6;a-=
U(i/3-D(o+n*i*.3))/pow(2,i));p=p*(U(n%L)*H*Y+Y)*a;p=z?p*Y+R(o+n*.1,d
+n*-2*(d%n),z-1)*H*Y:p;u=pow(U(n%_(L+d*S)),40);return p+p*-u+W*u;}z=
d.z*d.z;return v(z,z,1);} int main(){for(puts("P6 600 220 255");++y<
110;)for(f x=-301;P=R(v(-2,4,25),_(_(v(5,0,2))*++x+_(v(-2,73))*-y+v(
301,-59,-735)),2)*255,x<300;putchar(P.z))putchar(P.x),putchar(P.y);}
```



mattz