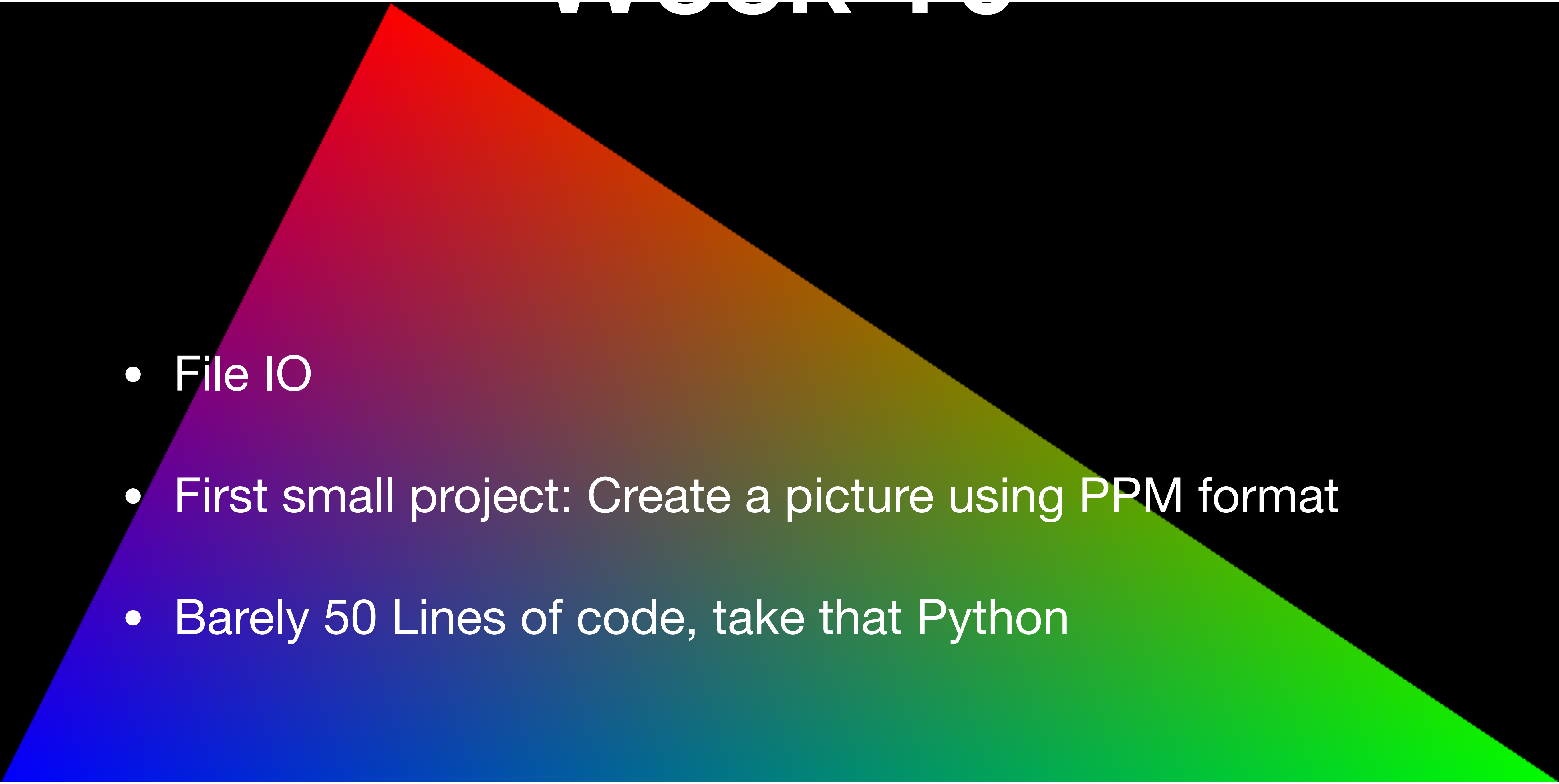
Week 10



- File IO
- First small project: Create a picture using PPM format
- Barely 50 Lines of code, take that Python

What did we not use so far?

- File IO
- Everything in Linux is a file
- Yes. Everything.
- We all love Linux

How do you work with files?

- Open
- read/write/append
- close

File Modes

- r
- w
- a
- +
- x
- Everything clear?

r

- Open File Read Only

w

- Create a file to write into it

a

- Append to the end of a file

+

- Update Mode
- Read and Write allowed
- Need to flush in between reading/writing to file

x

- Don’t override file if it already exists

Methods:

- fopen(path, mode);
- fprintf(filepointer, text, variables_ifany);
- fclose(filepointer);

Example

```
#include <stdio.h>
int main(void){
  FILE *fp;
  fp = fopen("/tmp/test.txt", "w+");
  fprintf(fp, "Today i allocated %i memory\n", 5);
  fclose(fp);
}
```

Reading a file

- fscanf(filepointer, string, buffer);
- fgets(buffer, maxLength, filepointer);
- All just read up to end of file/line

Example

```c
#include <stdio.h>
int main(void){
  FILE *fp;
  char buff[255];
  fp = fopen("/tmp/test.txt", "r");
  fgets(buff, 255, (FILE*)fp);
  printf("1: %s\n", buff );
  fclose(fp);
}
```

How to write a image?

- Many different standards
- We use ppm because it is so incredibly convenient for our purpose

```
P3\n
# feep.ppm\n
4 4\n
255\n
 0  0  0    0  0   0    0  0   0   255 0 255  \n
 0  0  0    0 255 128   0  0   0    0  0  0\n
 0  0  0    0  0   0    0 255 128    0  0  0\n
255 0 255   0  0   0    0  0   0    0  0  0\n
```

Version

Filename

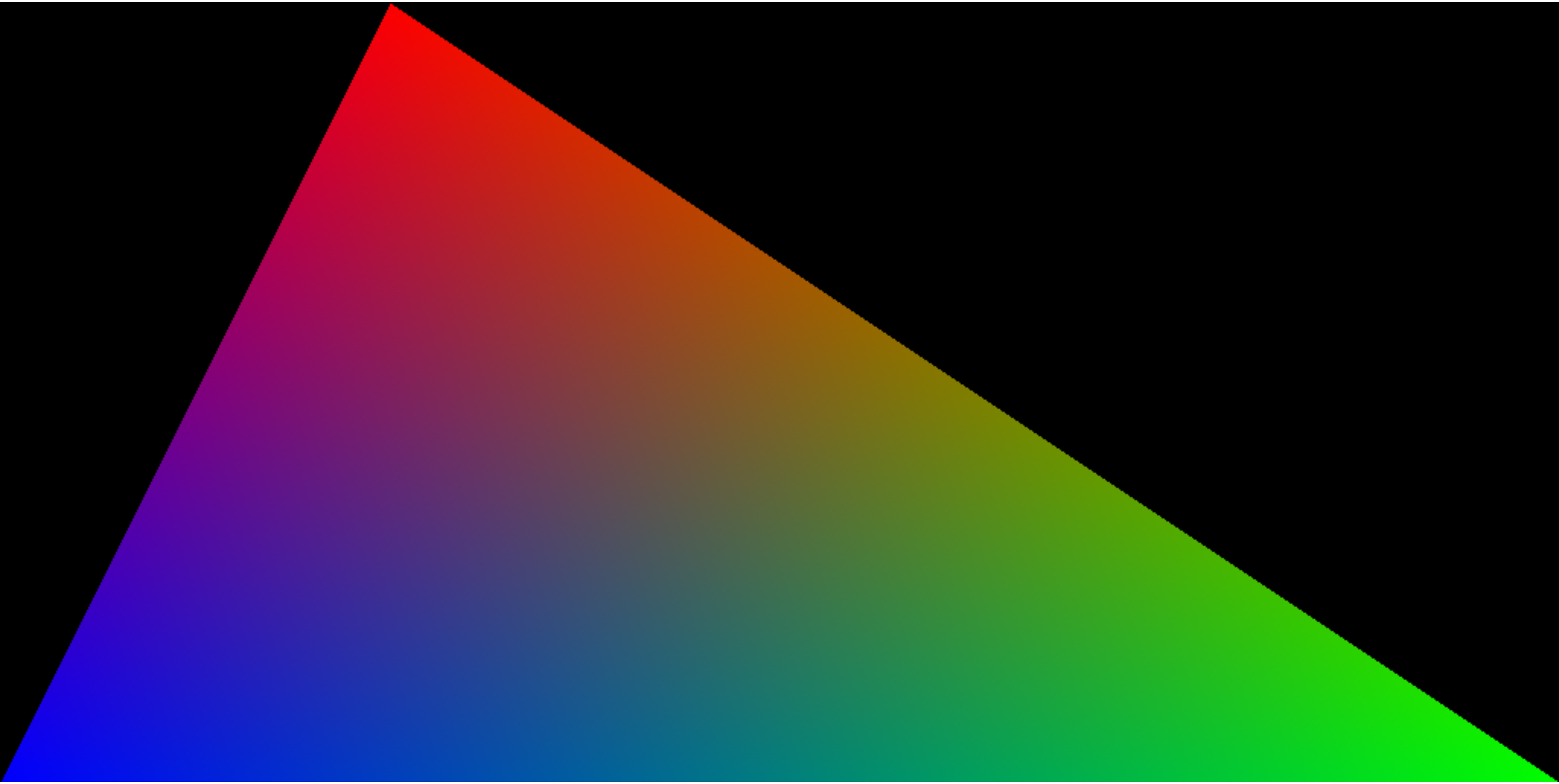
Dimensions

Max Value

Magenta Pixel

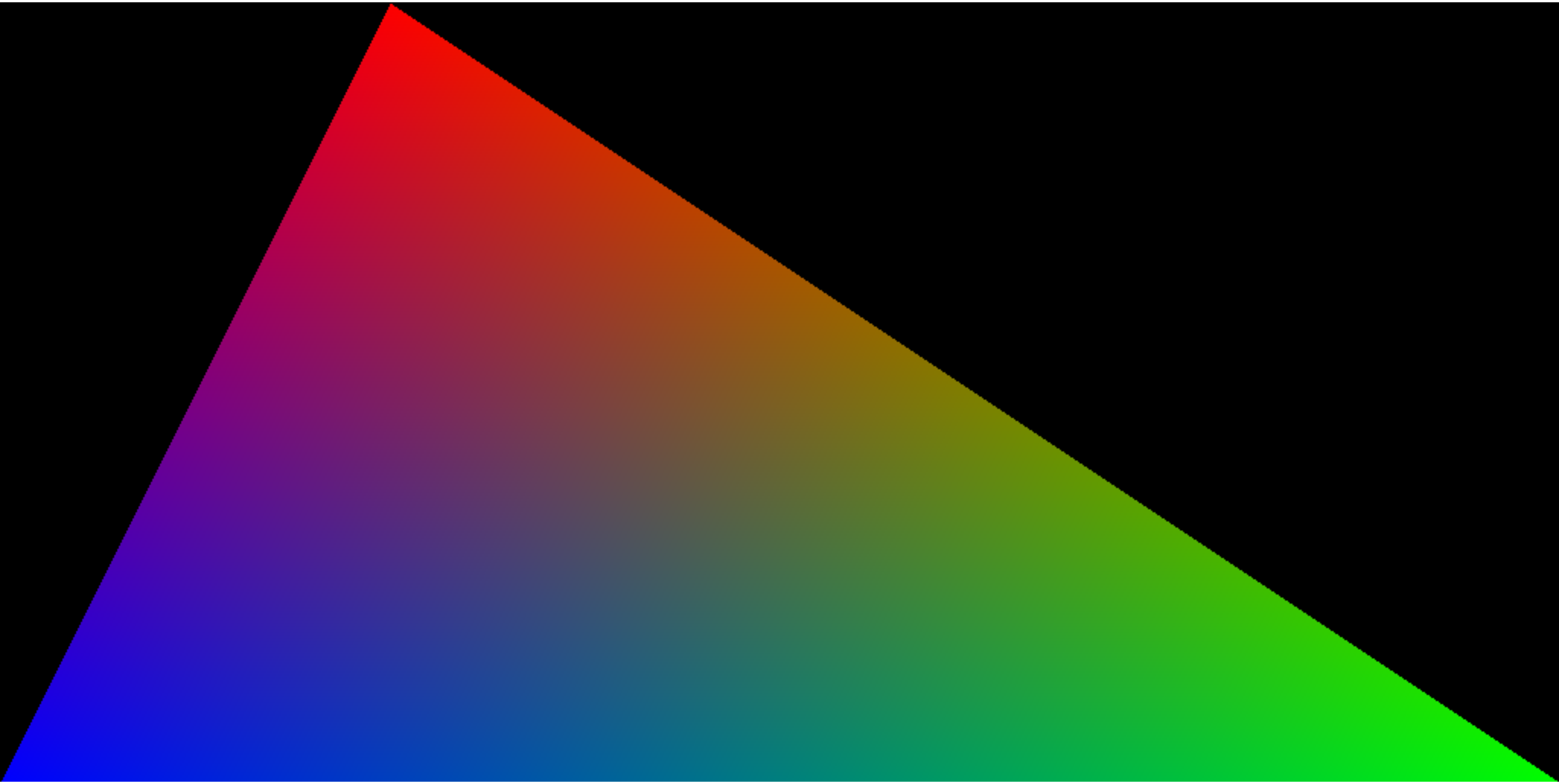
Triangles are simple

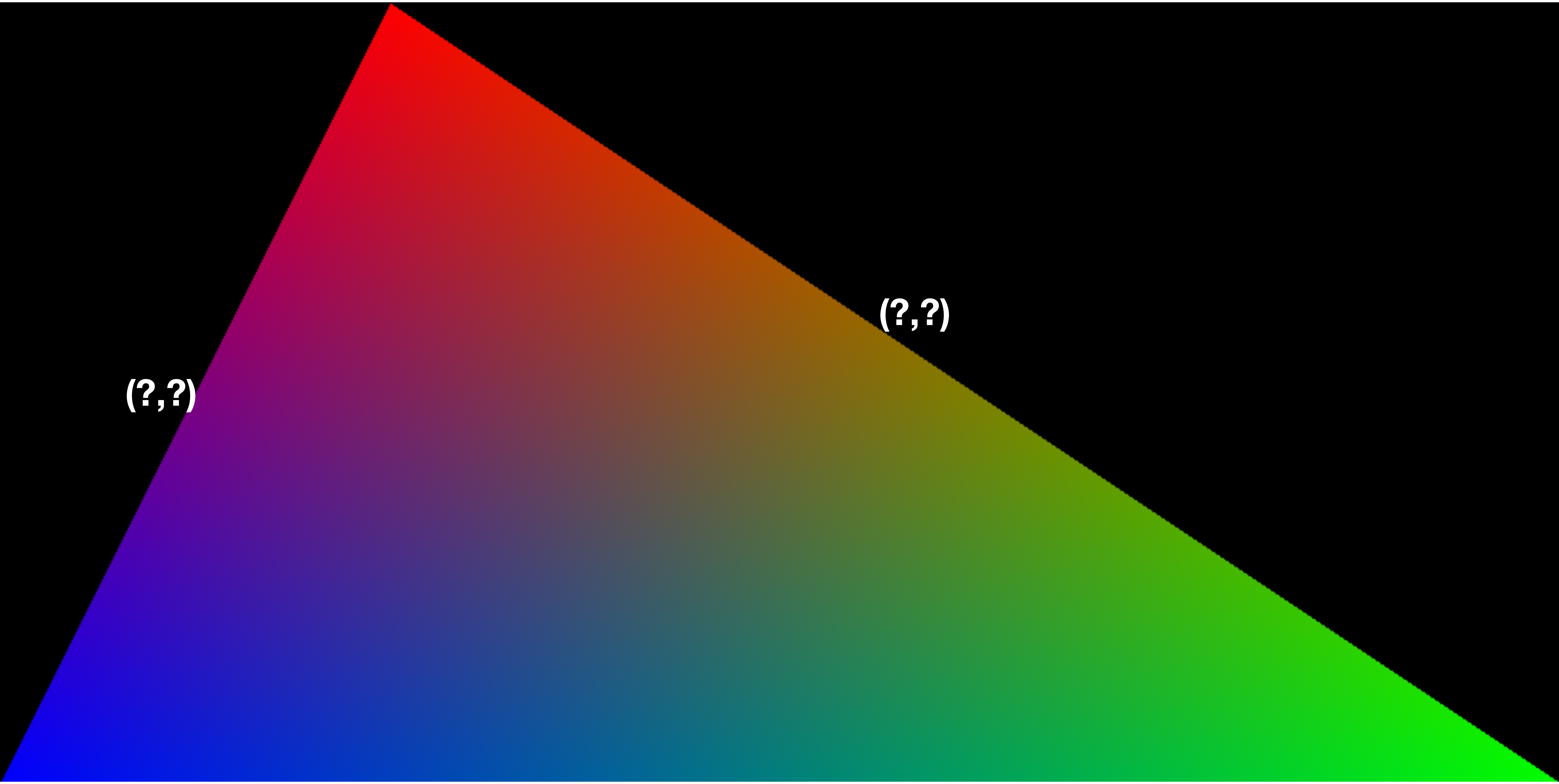
- You all know how a triangle works right?
- Most simple thing you can draw in any graphics engine
- Millions of triangles per frame, each one lit af and textured
- How hard can it be to draw a single simple triangle?

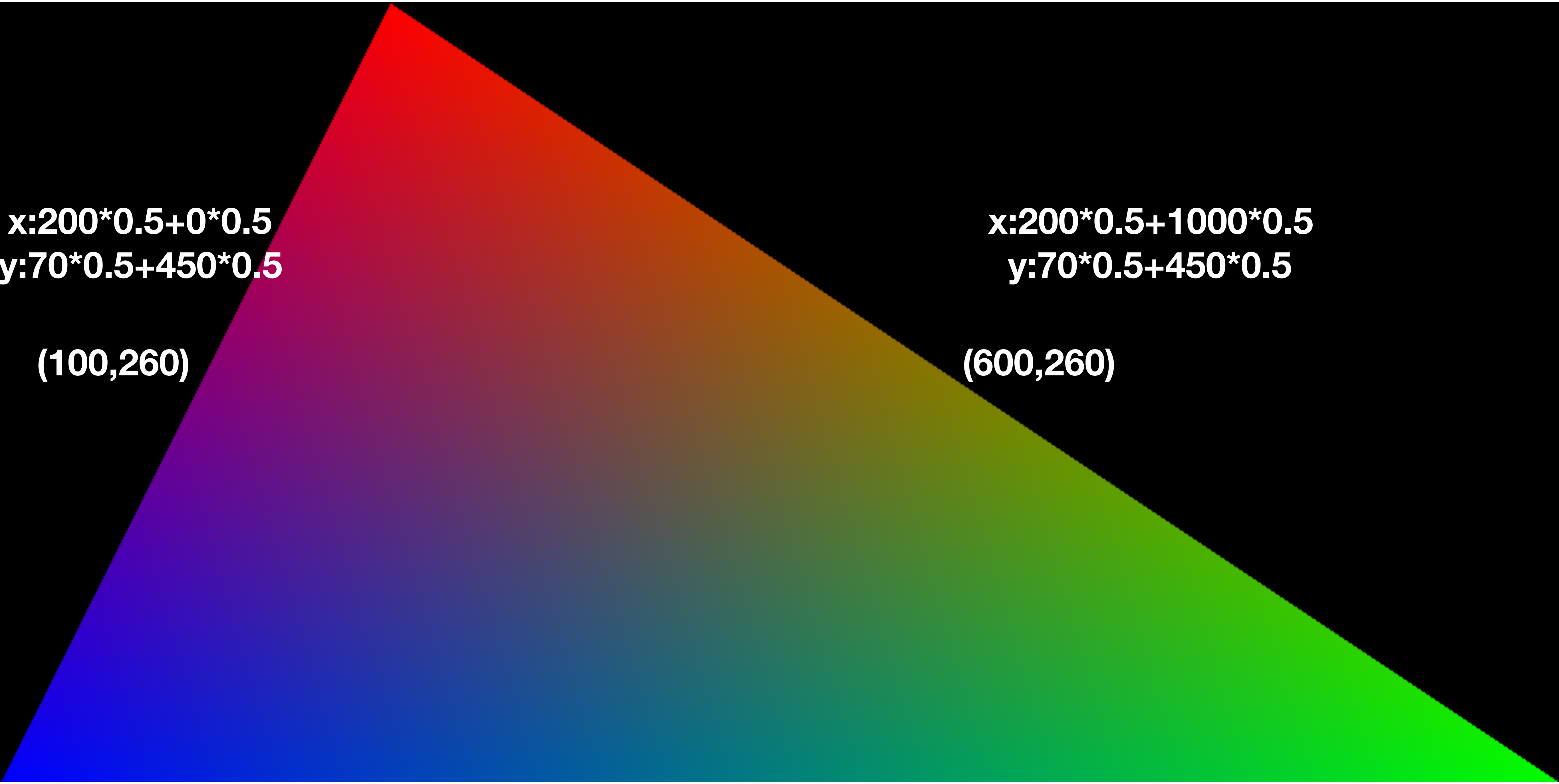


Triangles are complicated

- Three vertices in 2D/3D space per triangle for position
- Coloring attributes
- Texture attributes
- Which vertices make up a triangle?
- Huge redundancy if shared vertices are stored manyfold







x:200*0.5+0*0.5
y:70*0.5+450*0.5

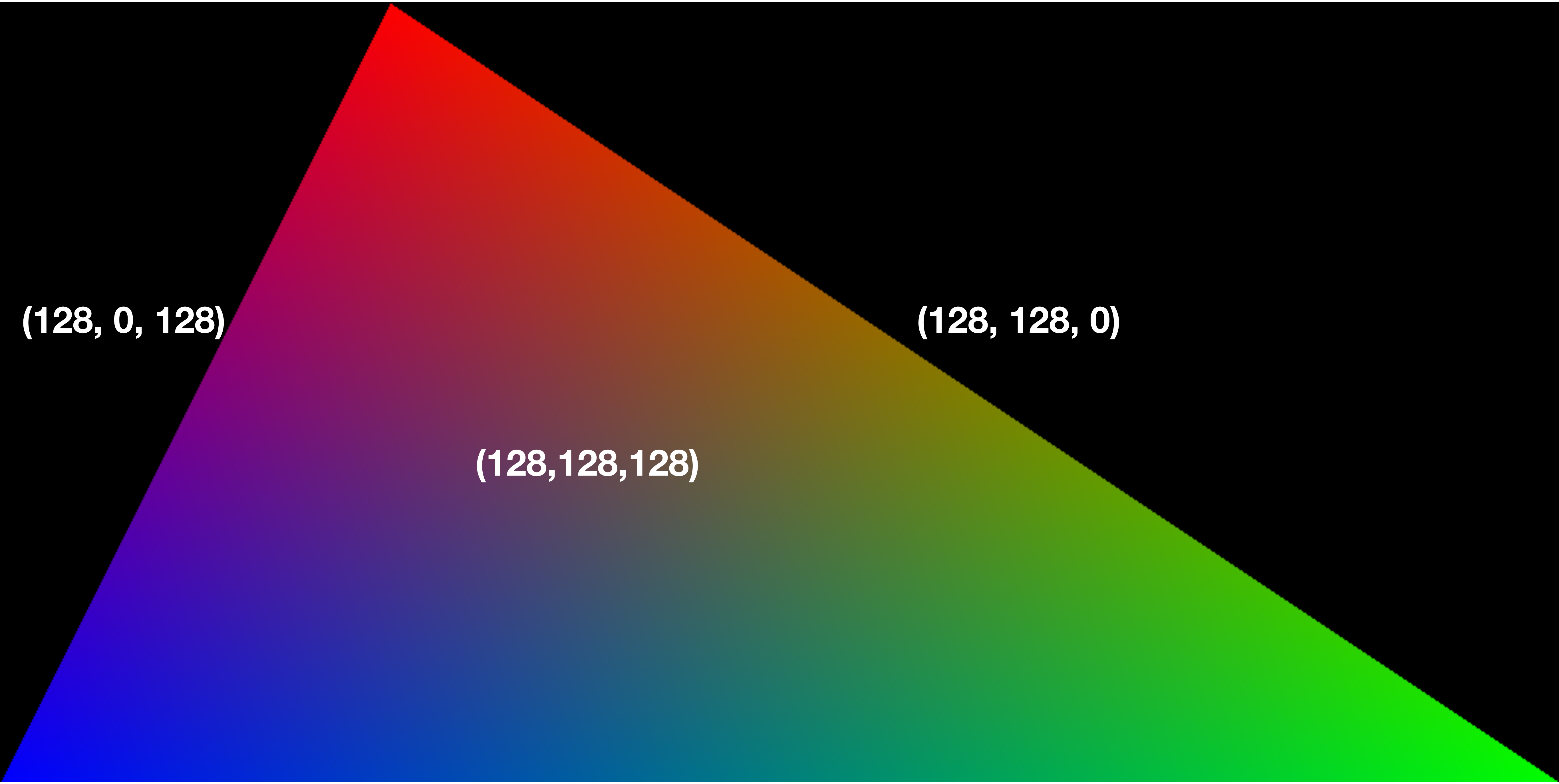
(100,260)

x:200*0.5+1000*0.5
y:70*0.5+450*0.5

(600,260)

Ok now we can calculate positions. Cool.

- We can extend it to colors as well!
- Define a color for each vertex
- Interpolate colors for each pixel

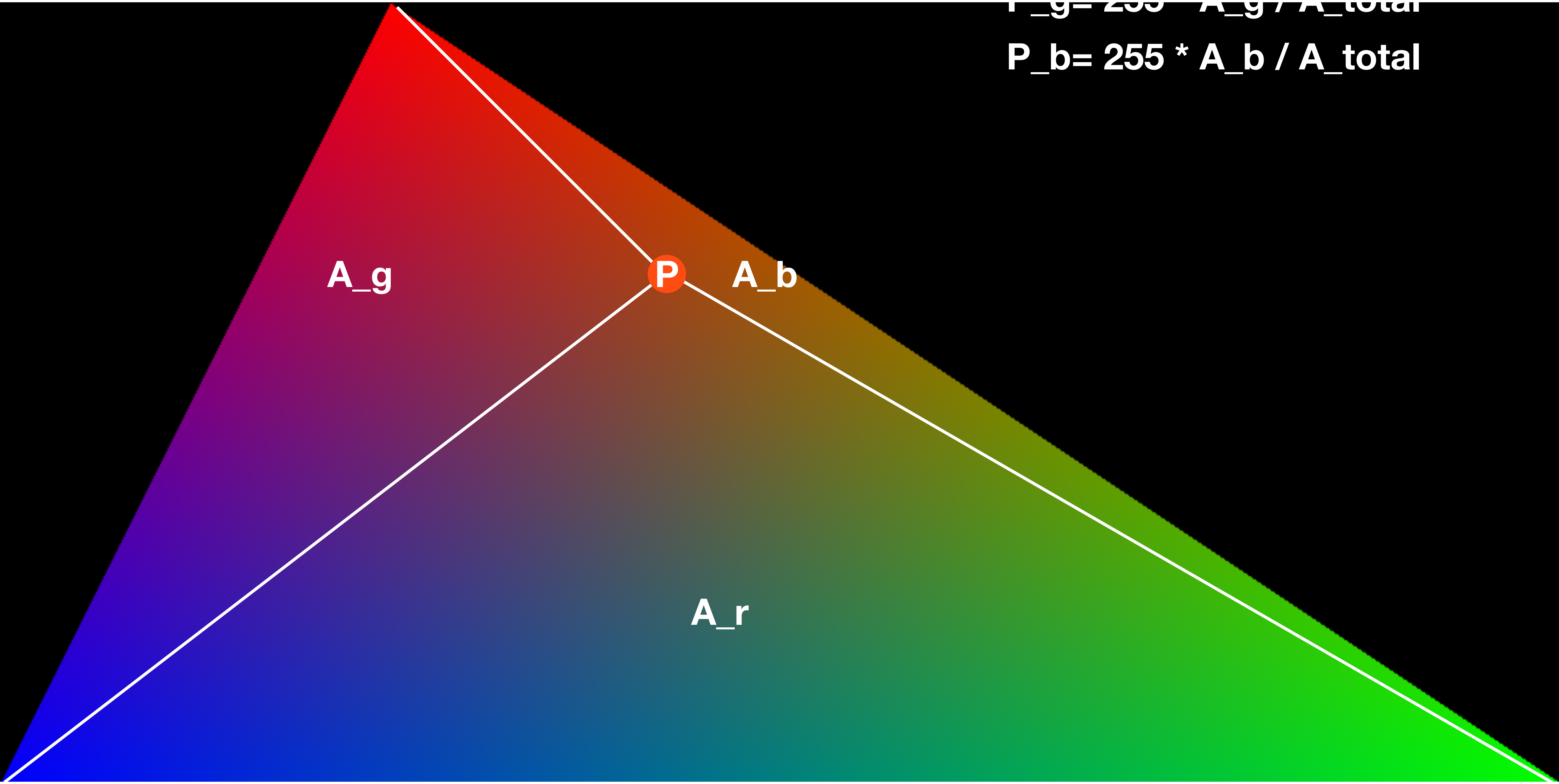

(128, 0, 128)
(128, 128, 0)
(128,128,128)

Ok but what is t?

- Needs to be calculated through maths
- Use a better approach then!

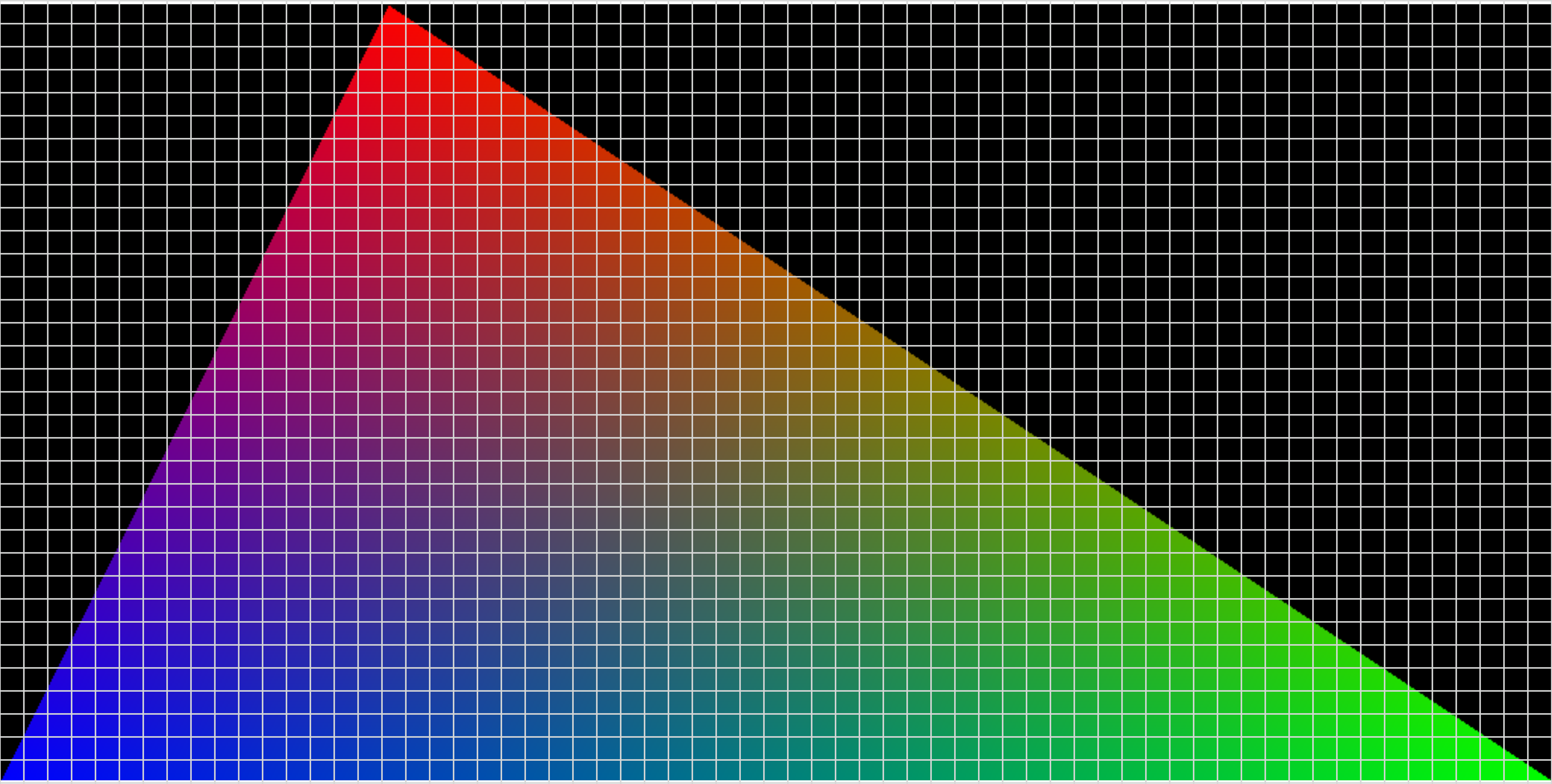
P_g= 255 * A_g / A_total

P_b= 255 * A_b / A_total



You are creating a rasterizer

- Rasterize continuous image into discrete pixels
- Compute color for each pixel
- Draw pixel
- Your graphics card is doing exactly this
- Your graphics card does it faster because it parallelizes work



```
int width = 1024, height = 512;
point top ={256,0}, left ={0, 512}, right ={1024, 512};
  for(int y = 0; y < height; ++y){
    for( int x = 0; x < width; ++x){
      point pt ={x,y};
      if(PointInTriangle(pt, top, left, right)){
        findColor(pt);
      }
    }
  }
```

How to do?

- Find out necessary algorithms (Is point in triangle? What color has point interpolated in triangle?)
- Write all data structures you need
- Write tests
- Work your way towards the goal

Thank you