ECE/CS 412
Introduction to Computer Graphics

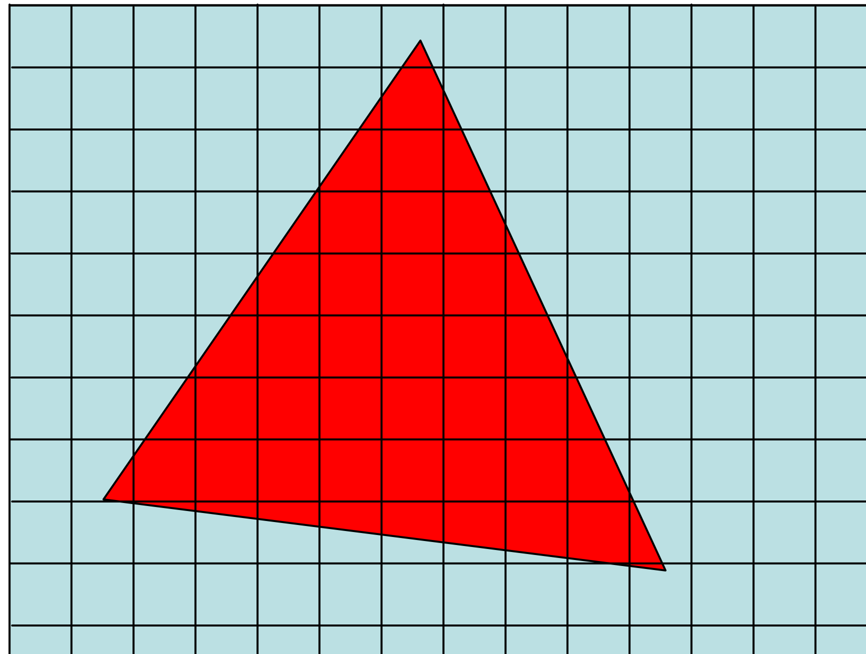
Class 13

Pradeep Sen
Advanced Graphics Lab

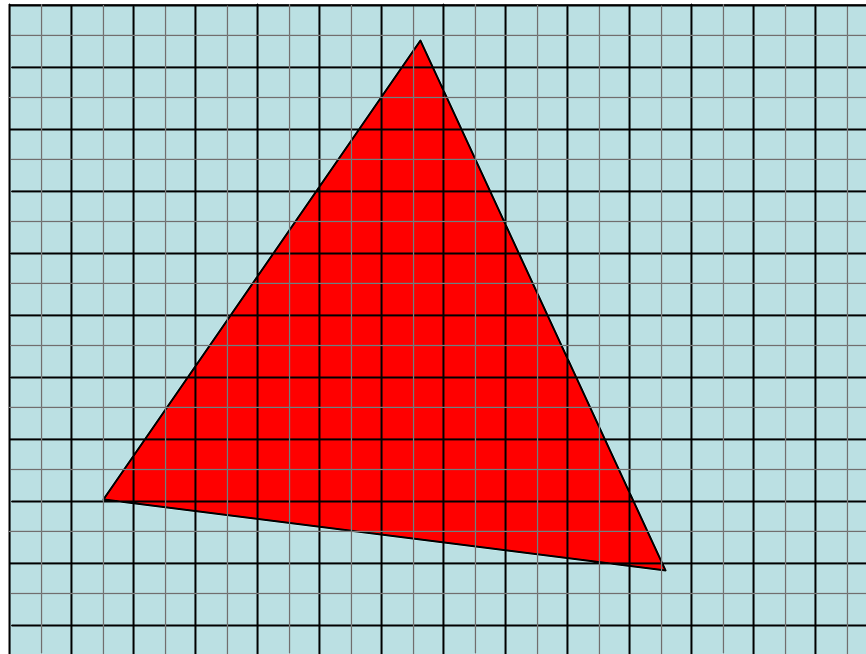
Rasterization

- The goal of rasterization is to take the mathematical representation of a triangle, given by the vertices and determine which pixels should be lit on the screen

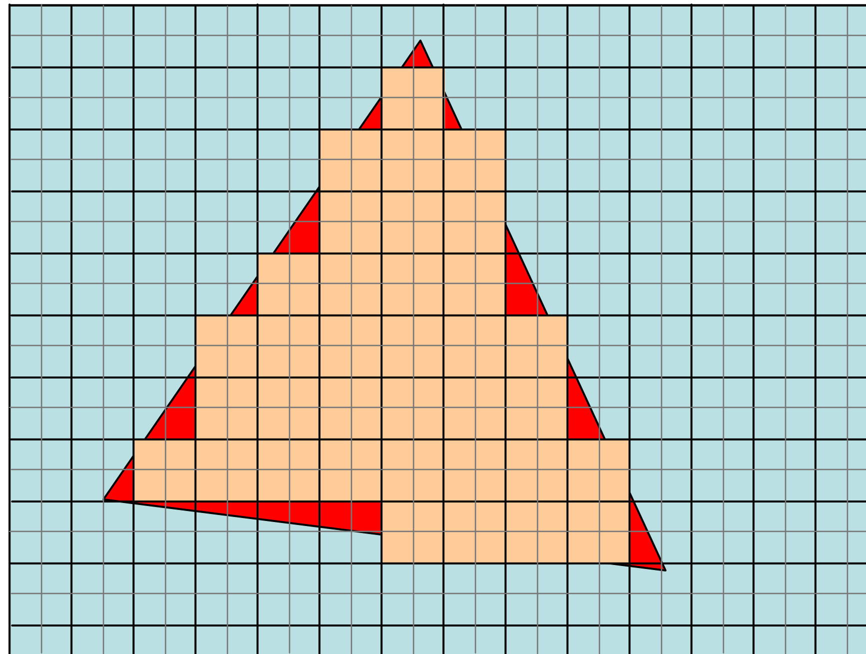
Rasterization



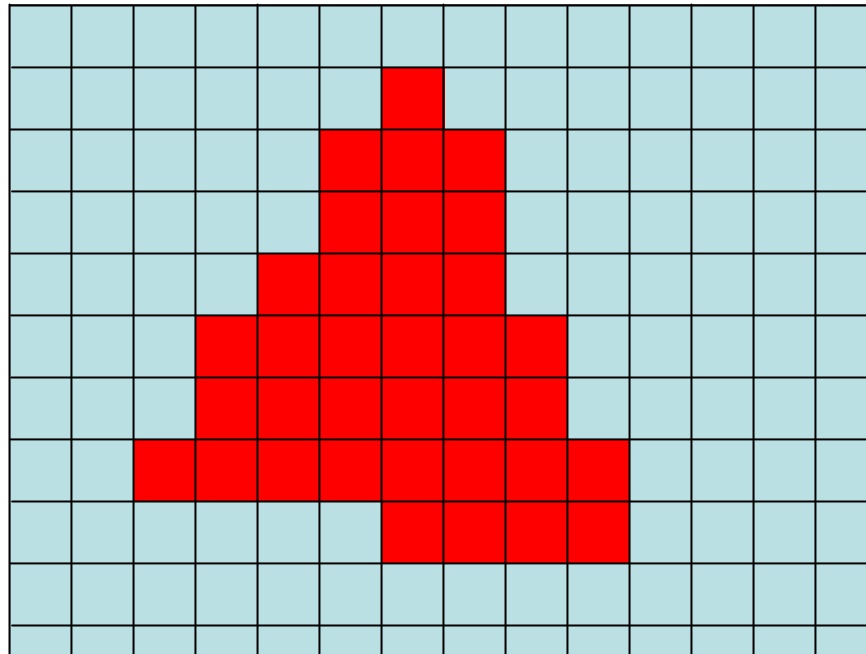
Rasterization



Rasterization



Rasterization



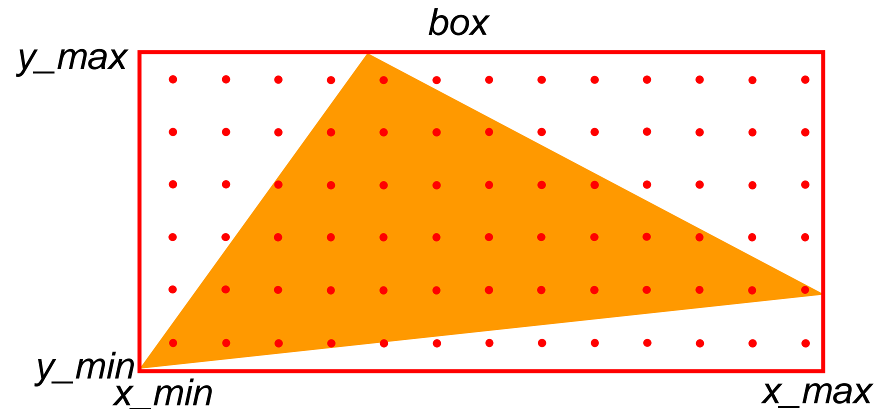
Rasterization

- There are two problems the rasterizer must solve:
  – Determine which pixels are “touched” by the triangle
  – Determine the value of interpolants at each of these vertices

Rasterization

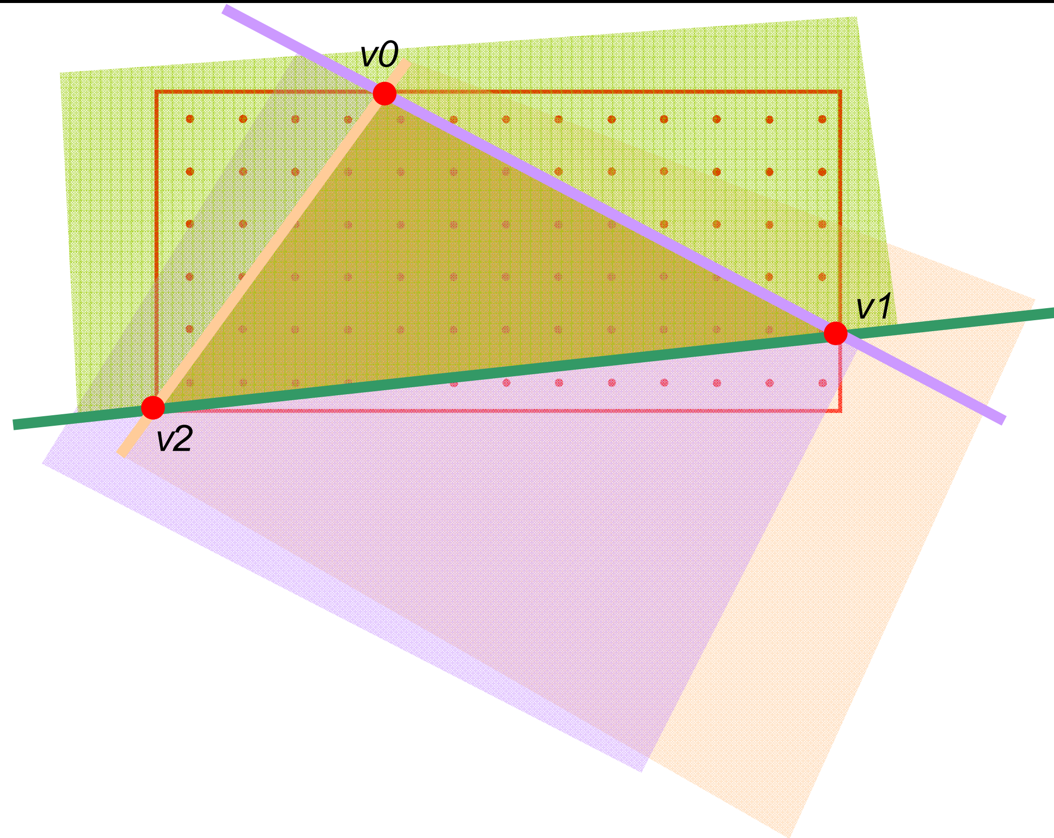
- Conversion from floating point screen space coordinates into integer pixel coordinates on the screen

Bounding box algorithm




```
void rasterizeTriangle(void) {
   for (y = y_min; y < y_max; y++) {
      for (x = x_min; x < x_max; x++) {
         if insideTriangle(x,y) {
            generateFragment(x,y);
         }
      }
   }
}
```

Inside tests



Inside tests

```
void lineEquation(float *v0, float *v1, float *f){
    crossProduct(v1, v0, f);
}

bool insideTriangle(int x,y) {
    lineEquation(v0, v1, &f0);
    lineEquation(v1, v2, &f1);
    lineEquation(v2, v0, &f2);

    Pt p(x,y,1);
    if ((dotProduct(p, f0) > 0) &&
        (dotProduct(p, f1) > 0) &&
        (dotProduct(p, f2) > 0) {
        // inside the triangle
        return true;
    }

    return false;
}
```

Inside tests

To accelerate things, you can make it incremental:

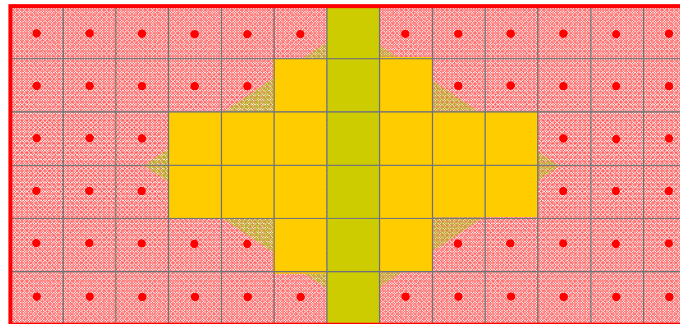
```
void rasterizeTriangle(void) {
    // compute line equations f0,f1,f2 as before
    Pt p(x_min, y_min, 1);
    e0 = dotProduct(p,f0);
    e1 = dotProduct(p,f1);
    e2 = dotProduct(p,f2);

    for (y = y_min; y < y_max; y++) {
        for (x = x_min; x < x_max; x++) {
            if (e0 >= 0) && (e1 >= 0) && (e2 >= 0) {
                generateFragment(x,y);
            }
            e0 += f0.i;   e1 += f1.i;   e2 += f2.i;
        }
        e0 += -(x_max - x_min)* f0.i + f0.j;
        e1 += -(x_max - x_min)* f1.i + f1.j;
        e2 += -(x_max - x_min)* f2.i + f2.j;
    }
}
```

Boundary cases

- We need to prevent double counting of pixels on triangle boundaries
- This is important for transparency, CSG, and efficiency
- Let's first look at a few wrong ways to handle it

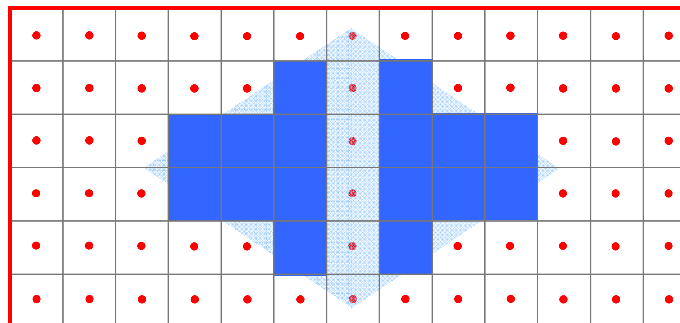
Incorrect solution #1

- Ignore the condition and double rasterize
- Imagine we are rendering semi-transparent triangles:



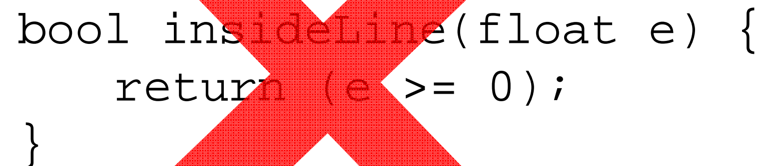
Incorrect solution #2

- Set the inside test to ‘>’ to ignore samples that lie exactly on the line edge
- No two triangles will “touch” the same pixels unless they overlap, in which case you want to rasterize both anyway
- Problem:



Shadow test

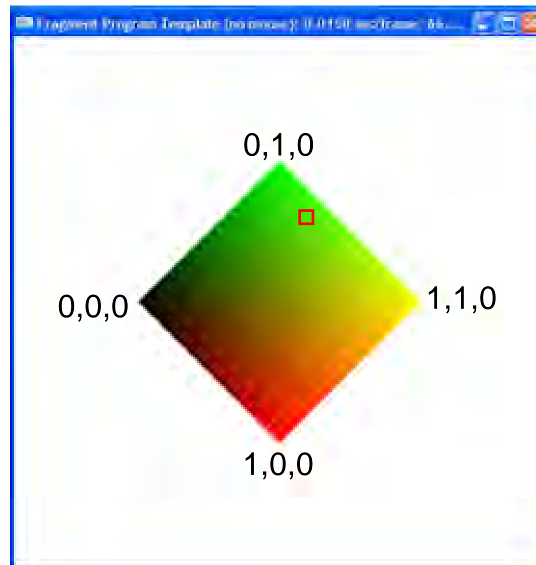
- Modify our line side test…




```
bool insideLine(float e, float *f) {
    return (e == 0) ? shadowTest(f) : (e > 0);
}

bool shadowTest(float *f) {
    return (f.i == 0) ? (f.j > 0) : (f.i > 0);
}
```

What about shading?

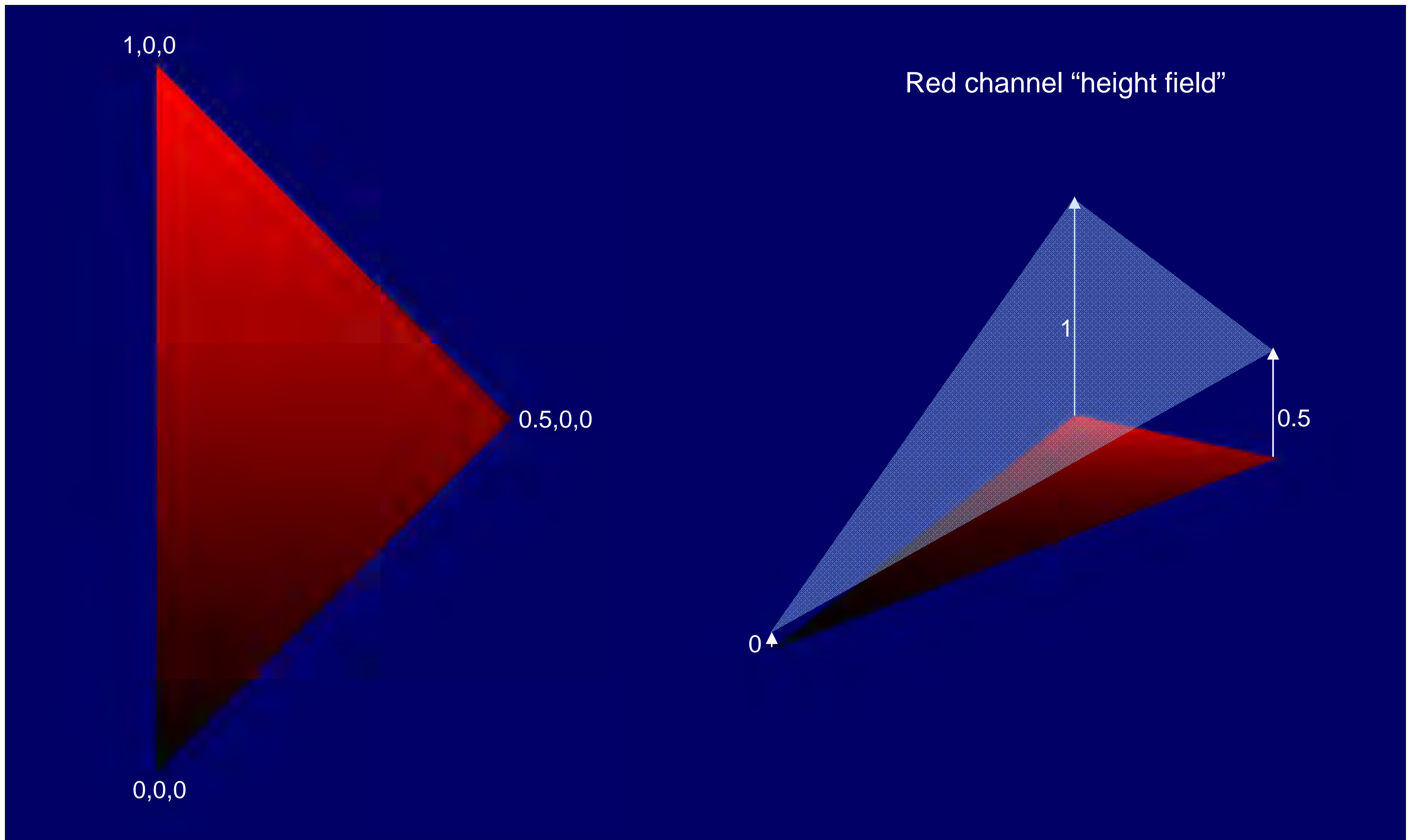
- So now we have a simple way to determine which pixels are on.
- Now how do we color the pixel?



Interpolation

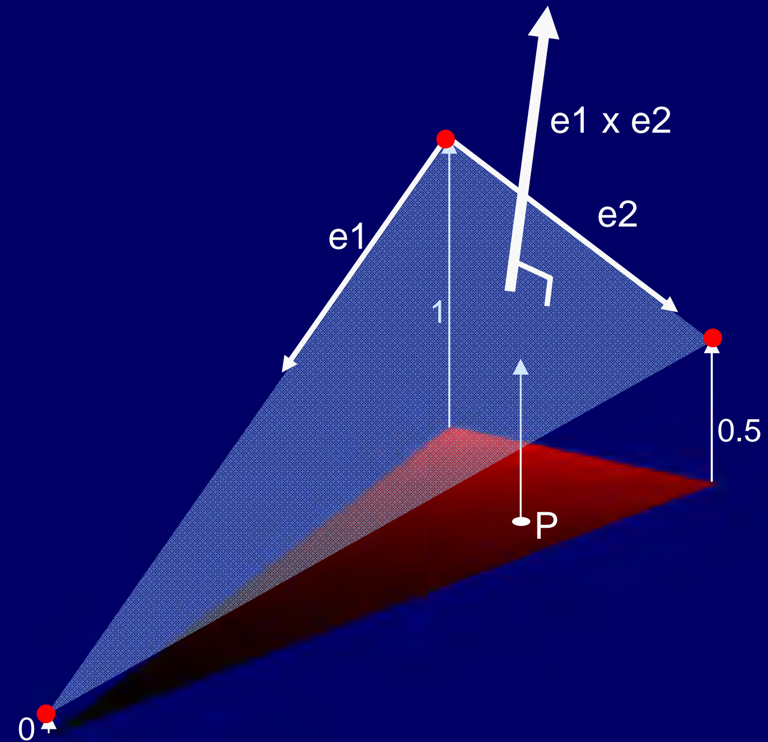
- Want to represent a function over the surface of the triangle
- Many ways to do this…
- Graphics hardware does it implicitly from the values at the corners of the triangle

Interpolation



How do you compute this plane?

- Equation of a plane



Barycentric coordinates