# The Story of the Teapot in HTML

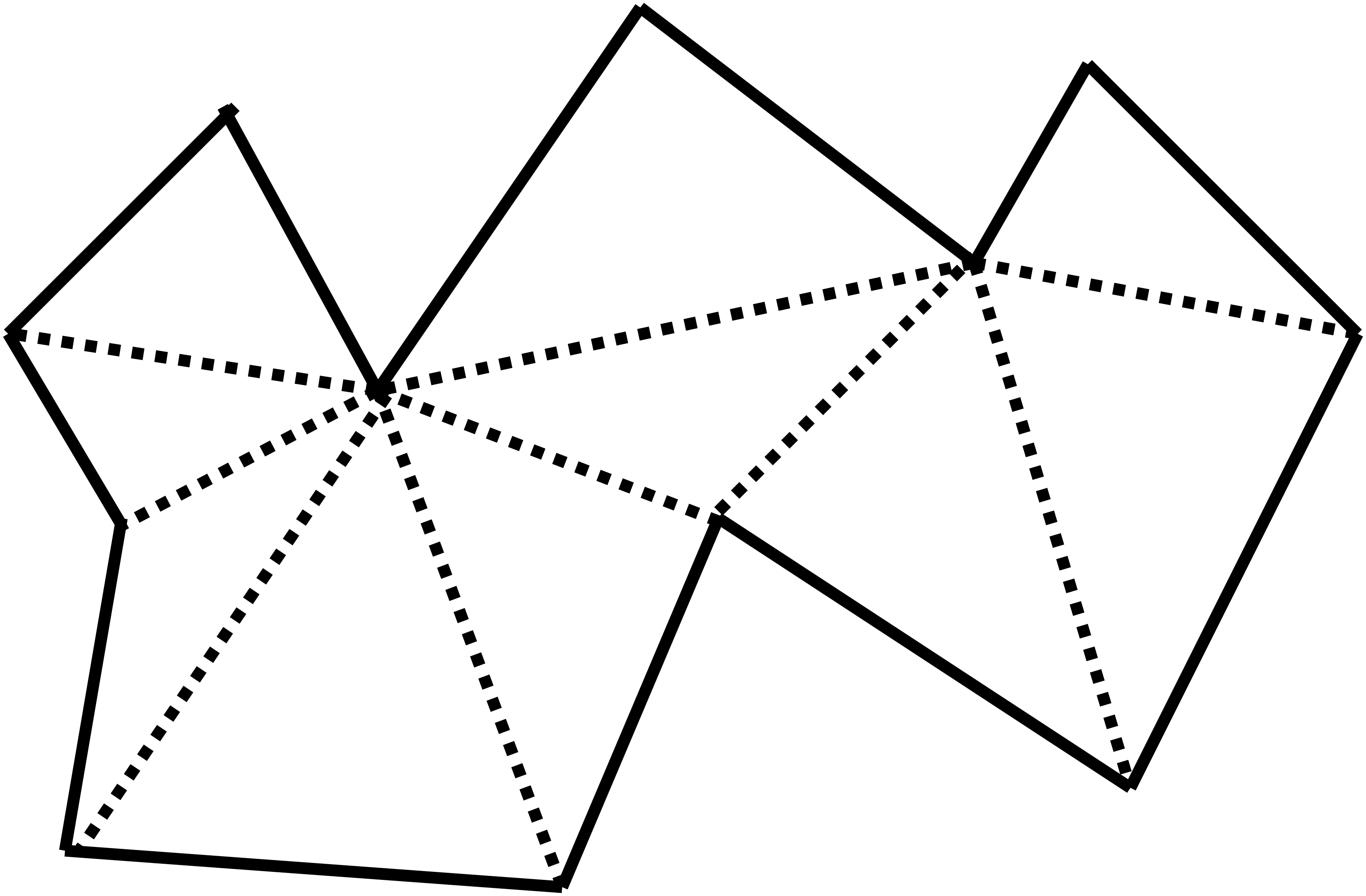
Erik Meijer
@headinthebox

Brian Beckman
http://rebcabin.wordpress.com

```
<div
style="width: 0px; height: 0px; border-style: solid; border-width: 0 0
7.481995px 79.77628px; border-color: transparent transparent #5ED355
transparent; position:absolute; left: 483.8142px; top: 325.5917px;"
>
</div>
<div
style="width: 0px; height: 0px; border-style: solid; border-width: 0
6.951904px 33.62497px 0; border-color: transparent #440E75 transparent
transparent; position:absolute; left: 483.8142px; top: 333.0737px;"
>
</div>
<div
style="width: 0px; height: 0px; border-style: solid; border-width:
33.62497px 72.82437px 0 0; border-color: #1BF7B3 transparent
transparent transparent; position:absolute; left: 490.7661px; top:
333.0737px;"
>
</div>
<div
style="width: 0px; height: 0px; border-style: solid; border-width:
7.481995px 16.20441px 0 0; border-color: #ACBB2A transparent
transparent transparent; position:absolute; left: 563.5905px; top:
325.5917px;"
>
</div>
```

Computer
Graphics
101

# Every polygon can be dissected into triangle



Can we render 3D graphics using pure HTML & CSS
(no canvas, OpenGL, …)?

⇐ { **Any polygon can be dissected into triangles** }

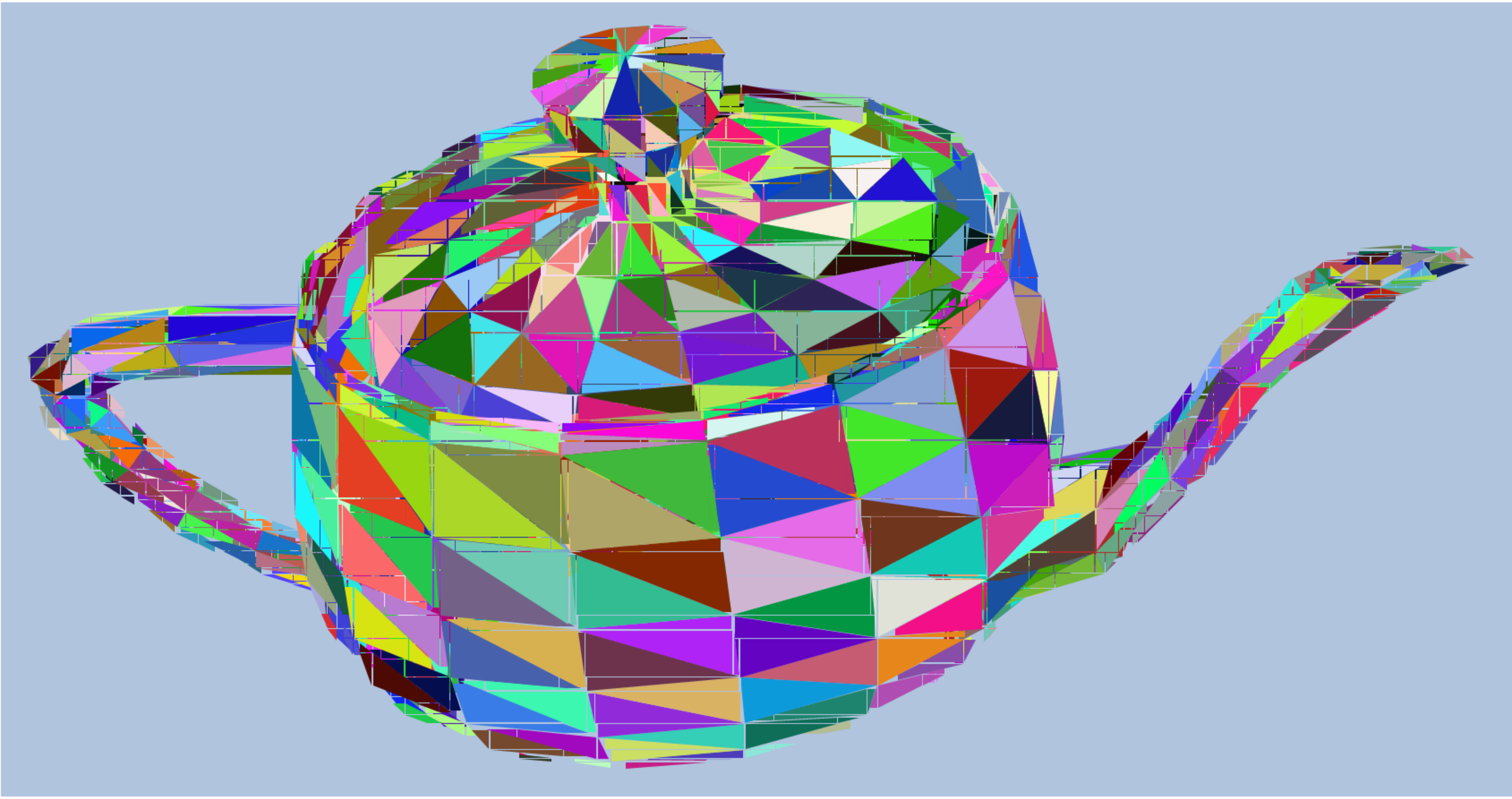
Can we render arbitrary triangle using pure HTML & CSS?

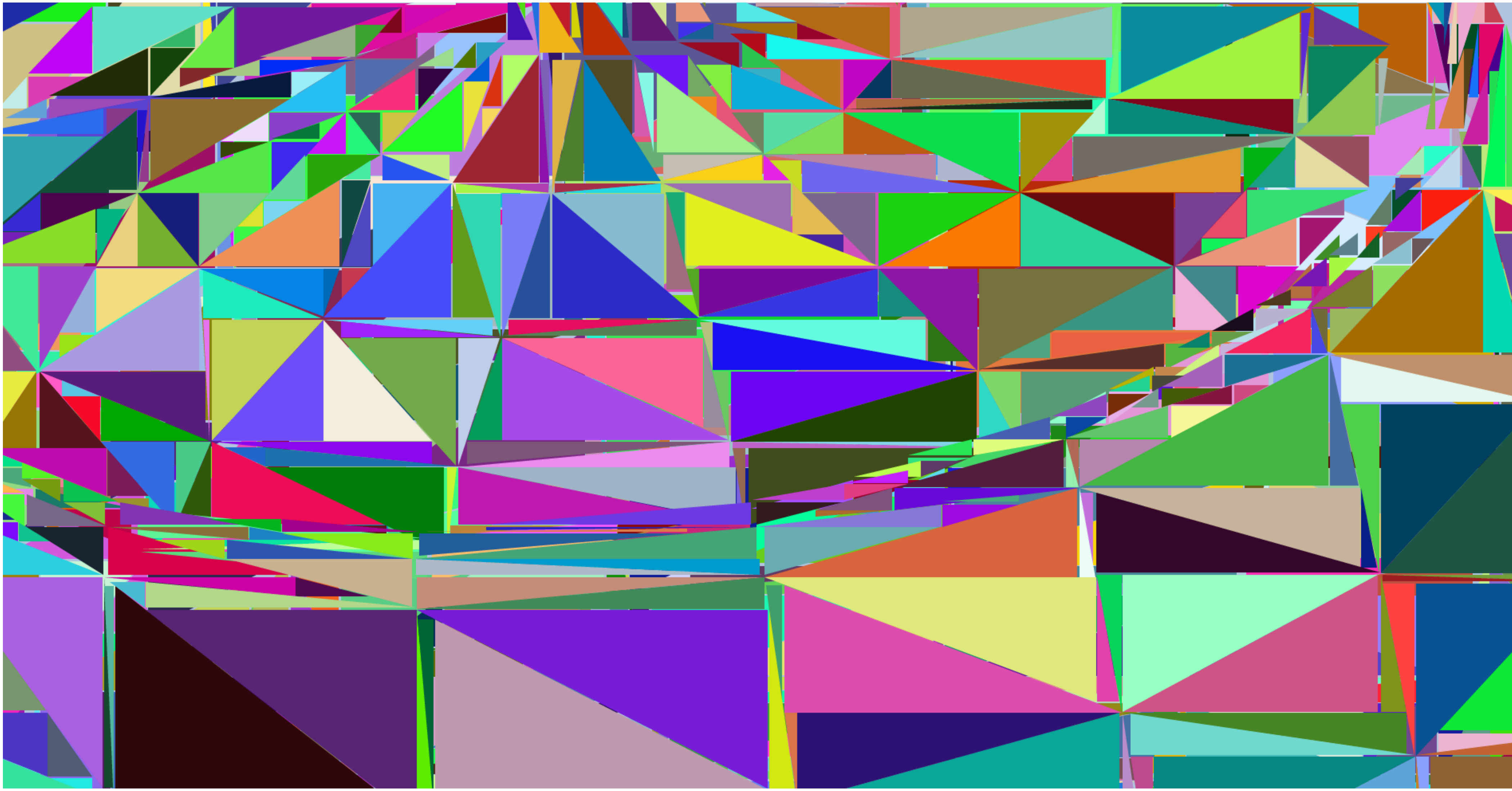
⇐ { **Any triangle can be dissected into right triangles** }

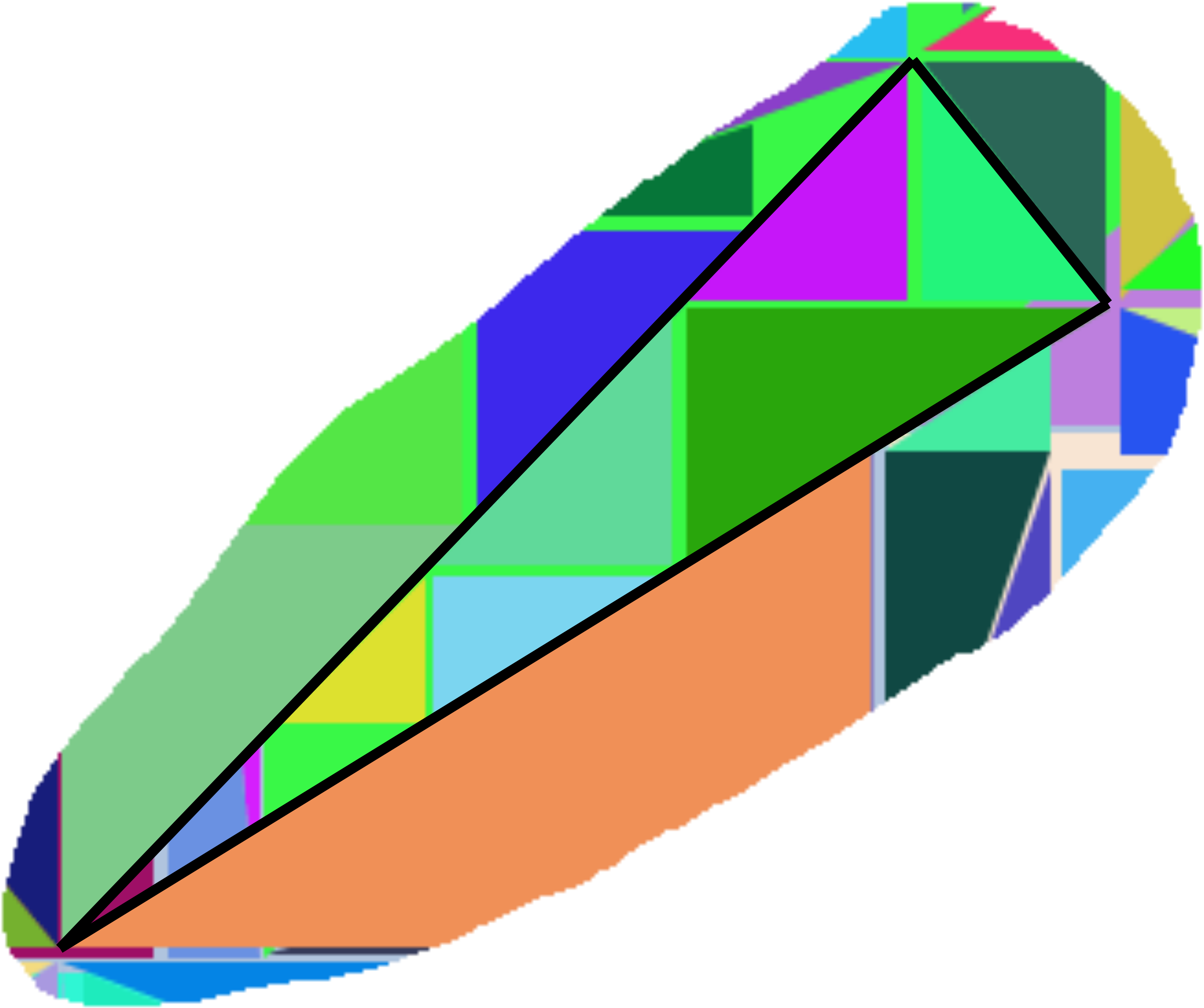
Can we render arbitrary triangle using pure HTML & CSS?

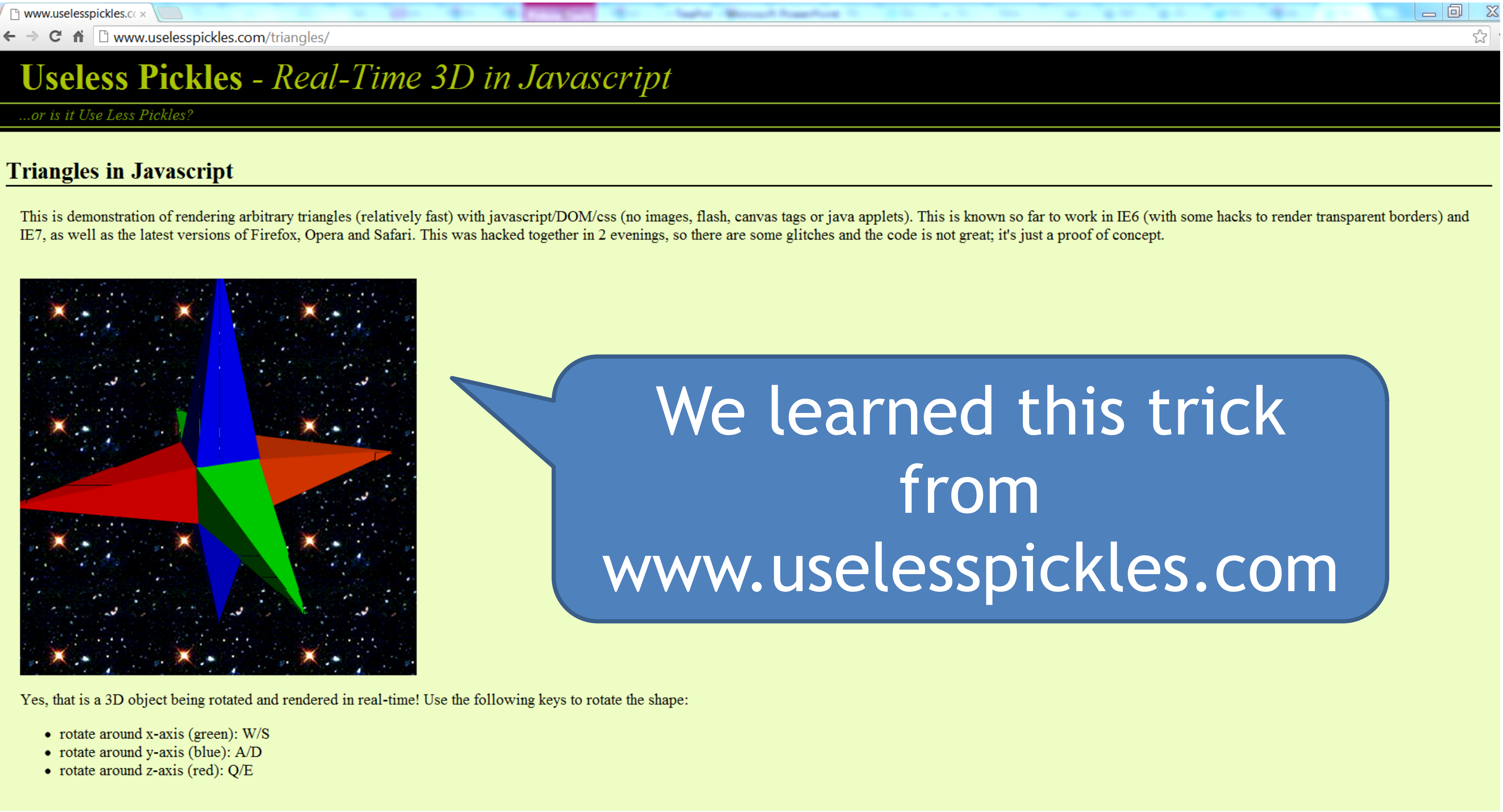
⇐ { **Any right triangle can be constructed using just <DIV>s** }

True!









**Useless Pickles** - *Real-Time 3D in Javascript*

*...or is it Use Less Pickles?*

## Triangles in Javascript

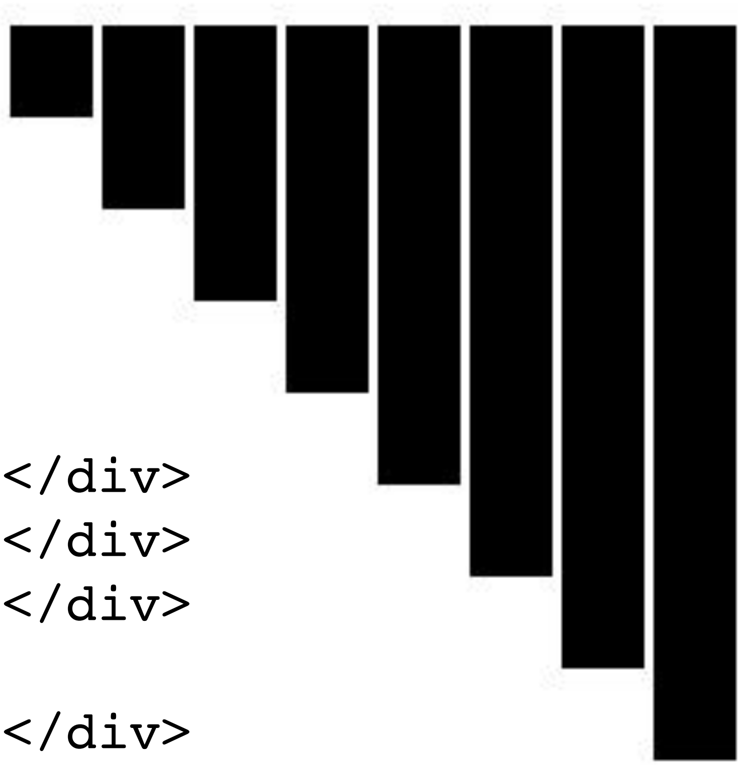
This is demonstration of rendering arbitrary triangles (relatively fast) with javascript/DOM/css (no images, flash, canvas tags or java applets). This is known so far to work in IE6 (with some hacks to render transparent borders) and IE7, as well as the latest versions of Firefox, Opera and Safari. This was hacked together in 2 evenings, so there are some glitches and the code is not great; it's just a proof of concept.

Yes, that is a 3D object being rotated and rendered in real-time! Use the following keys to rotate the shape:

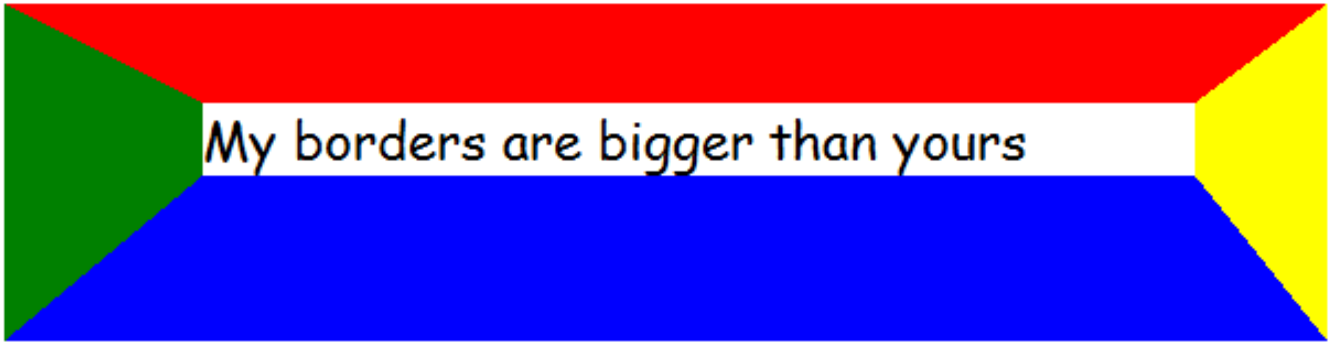
- rotate around x-axis (green): W/S
- rotate around y-axis (blue): A/D
- rotate around z-axis (red): Q/E

We learned this trick from www.uselesspickles.com

```
<style>
  div{ background:Black;
       position:absolute;
       width:9px; }
</style>
```

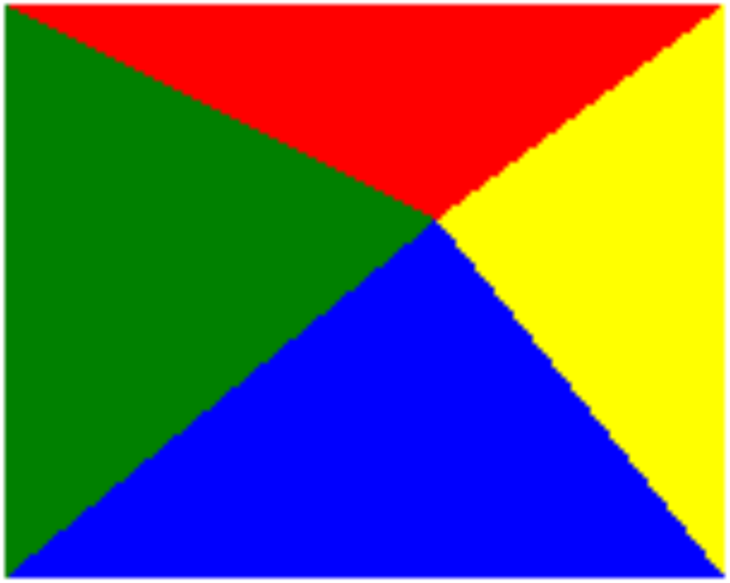


```
<div style="left:10px; height:10px;"></div>
<div style="left:20px; height:20px;"></div>
<div style="left:30px; height:30px;"></div>
...
<div style="left:80px; height:80px;"></div>
```



My borders are bigger than yours

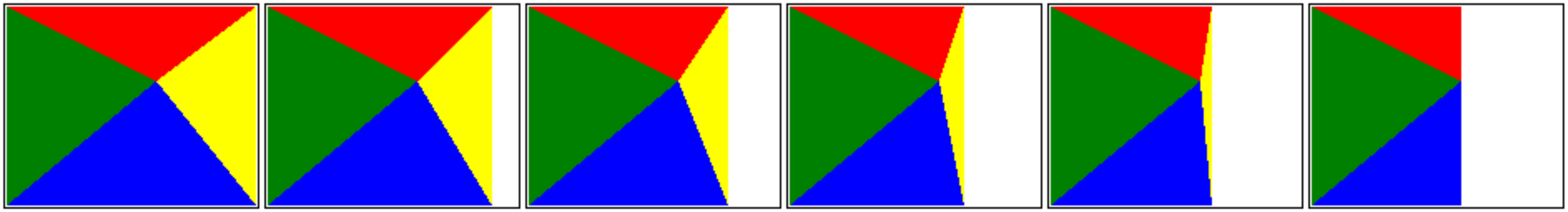
```
<div style="font-family:'Comic Sans
MS';
width:300;

border-left: 60 solid green;
border-right: 40 solid yellow;
border-top: 30 solid red;
border-bottom: 50 solid blue;">

My borders are bigger than yours
</div>
```



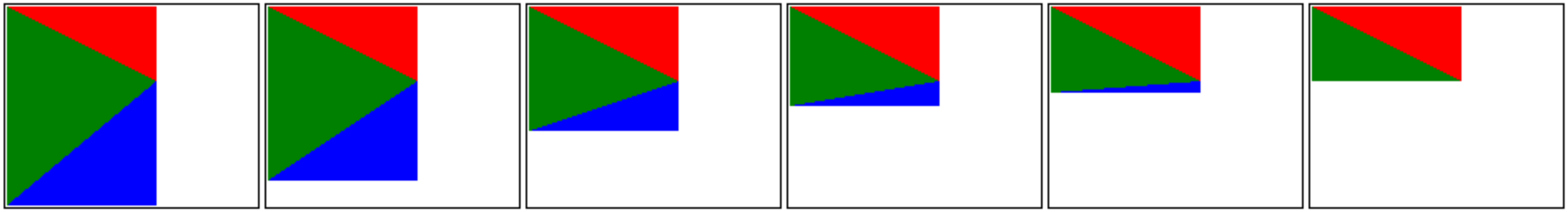
Could do any triangle with vertical or horizontal leg

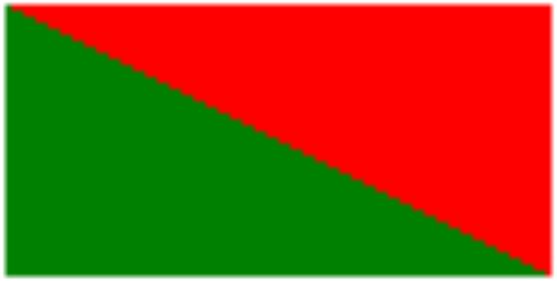
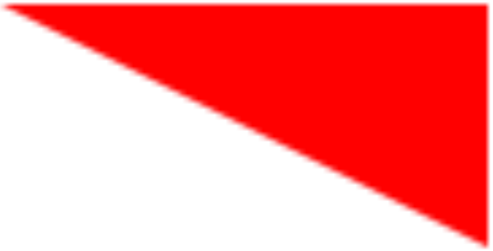
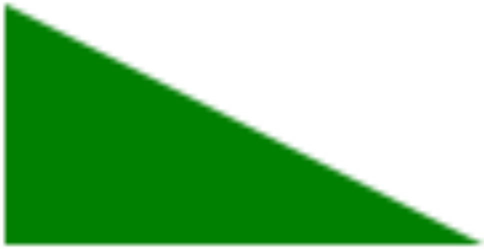
```
<div style="width:0;

border-left: 60 solid green;
border-right: 40 solid yellow;
border-top: 30 solid red;
border-bottom: 50 solid blue;">

</div>
```

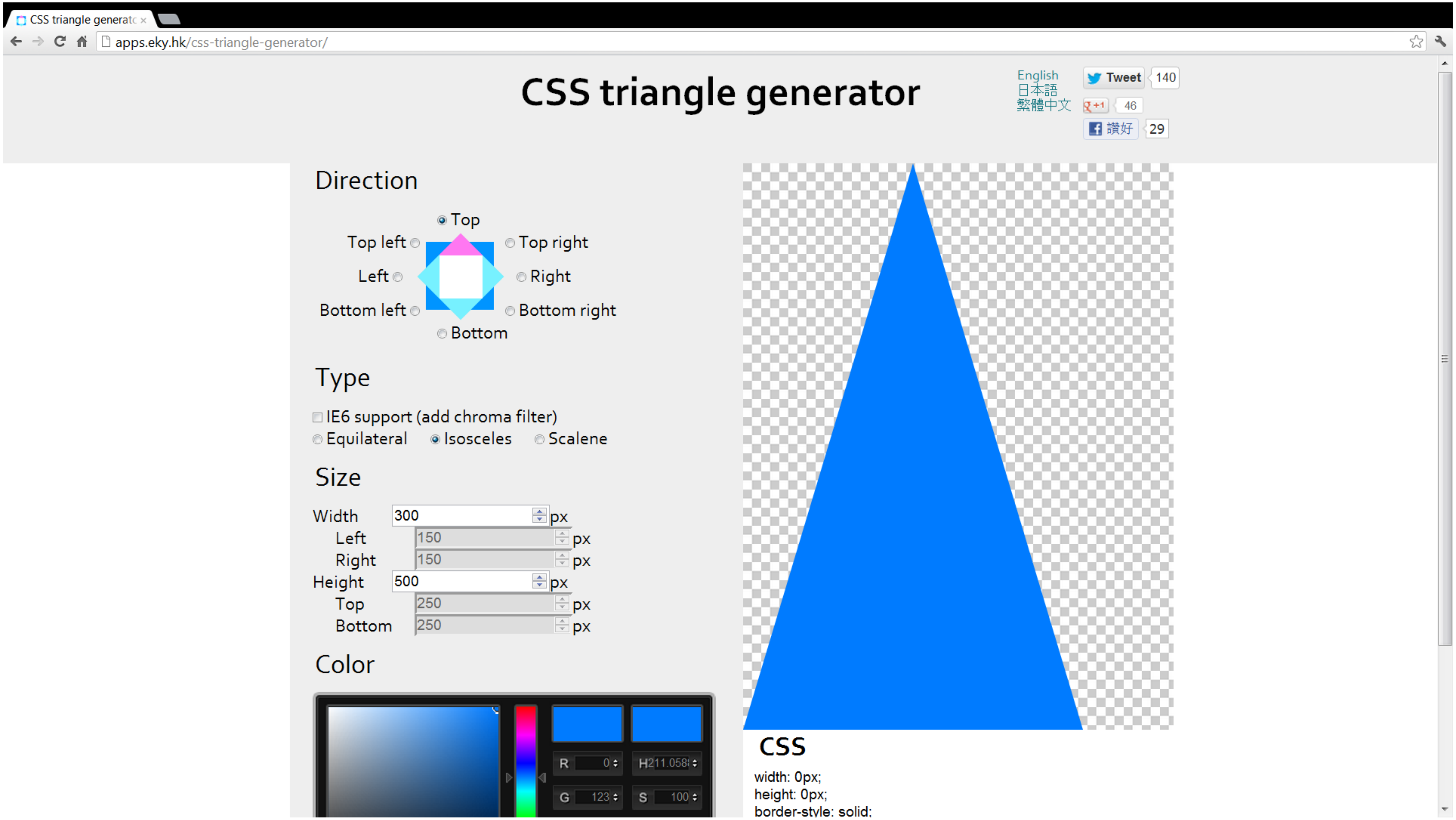
## Make right border zero width



## Make bottom border zero width



| | | |
|---|---|---|
|  |  |  |
| <div style="width:0; border-left: 60 solid green; border-right: 0 solid yellow; border-top: 30 solid red; border-bottom: 0 solid blue; "></div> | <div style="width:0; border-left: 60 transparent; border-right: 0 solid yellow; **border-top: 30 solid red;** border-bottom: 0 solid blue; "></div> | <div style="width:0; **border-left: 60 solid green;** border-right: 0 solid yellow; border-top: 30 transparent; border-bottom: 0 solid blue; "></div> |



CSS triangle generatc ×

apps.eky.hk/css-triangle-generator/

# CSS triangle generator

English
日本語
繁體中文

## Direction

● Top
Top left ○
○ Top right
Left ○
○ Right
Bottom left ○
○ Bottom right
○ Bottom

## Type

□ IE6 support (add chroma filter)

○ Equilateral ● Isosceles ○ Scalene

## Size

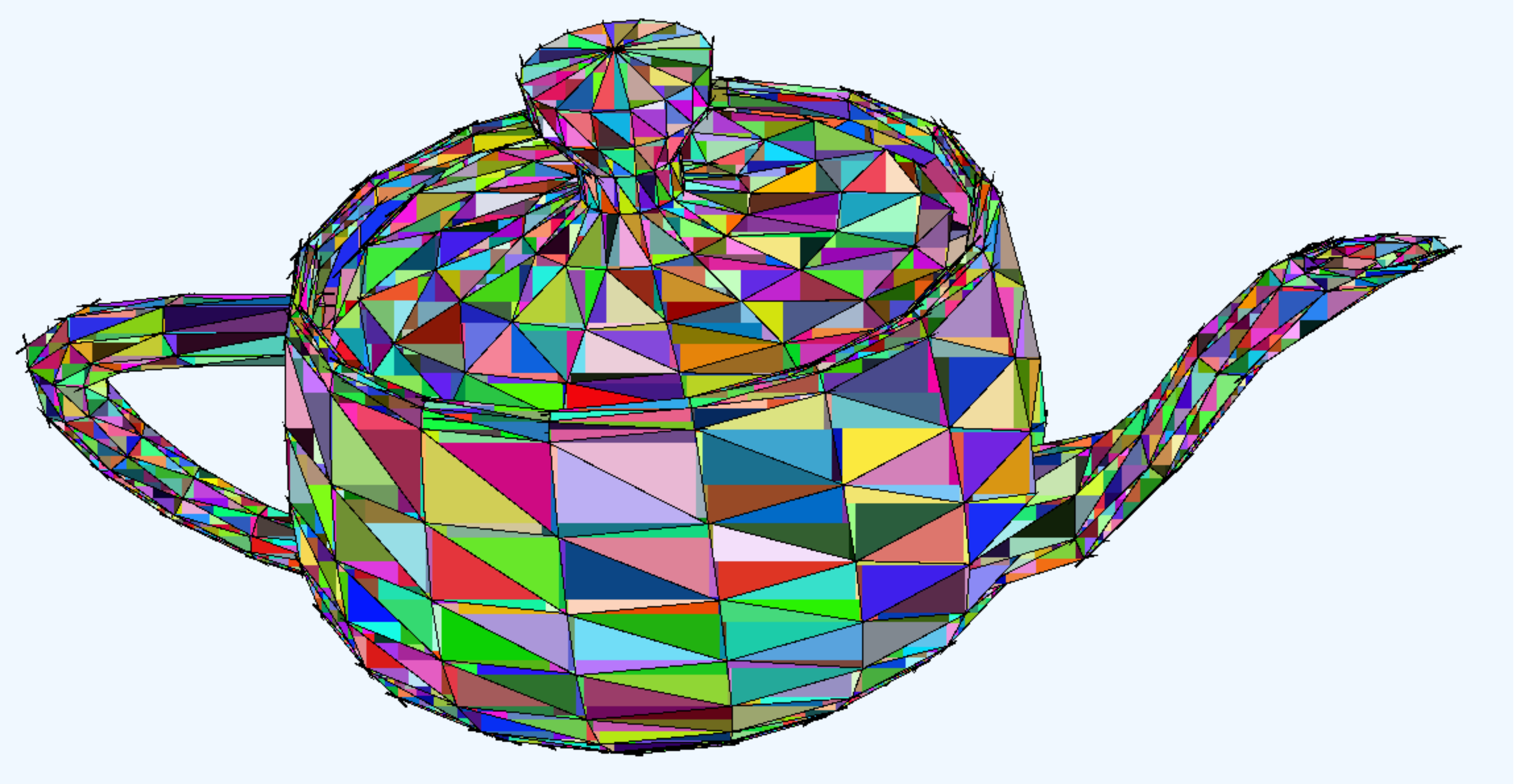
Width 300 px
Left 150 px
Right 150 px
Height 500 px
Top 250 px
Bottom 250 px

## Color

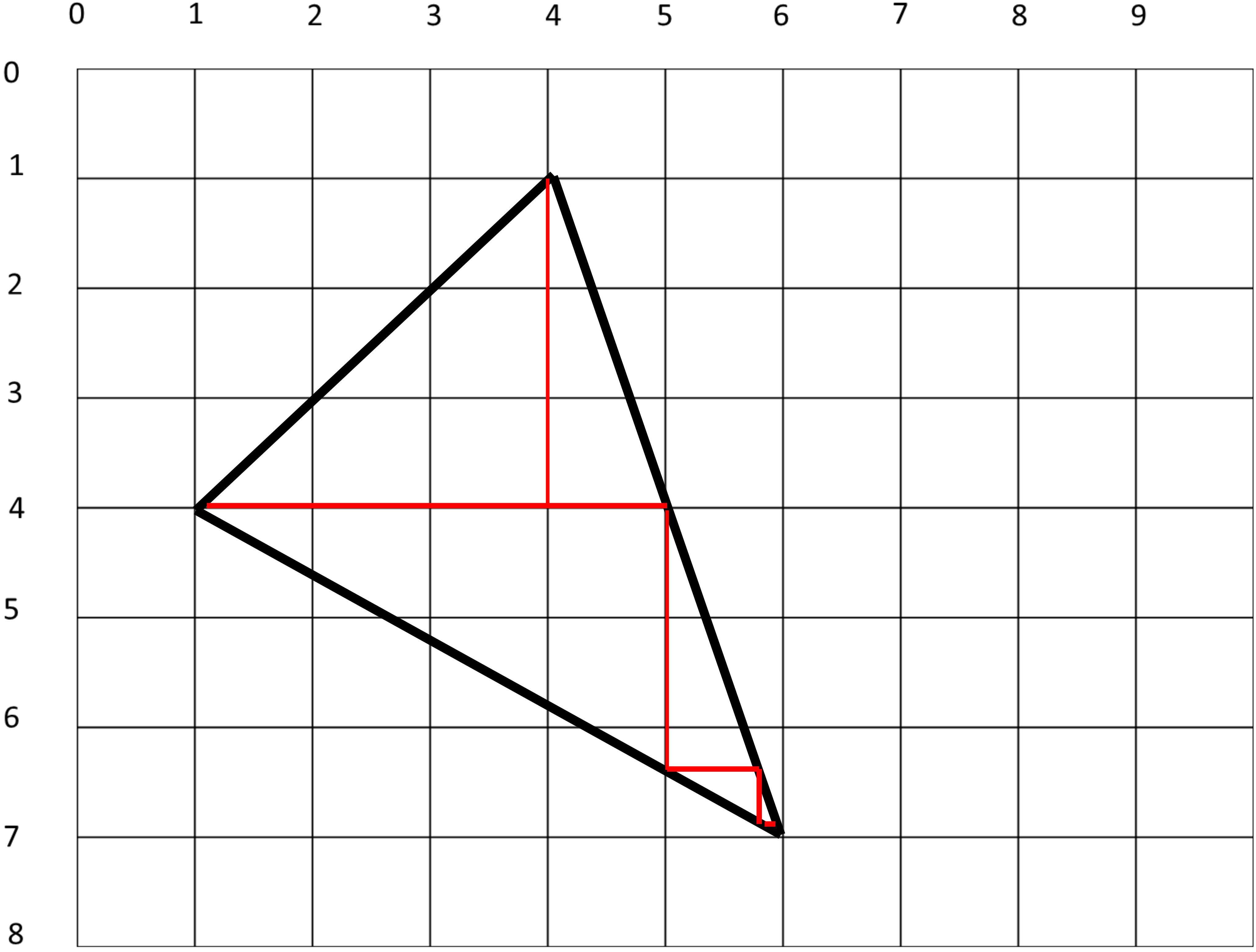
R 0
H 211.058
G 123
S 100

## CSS

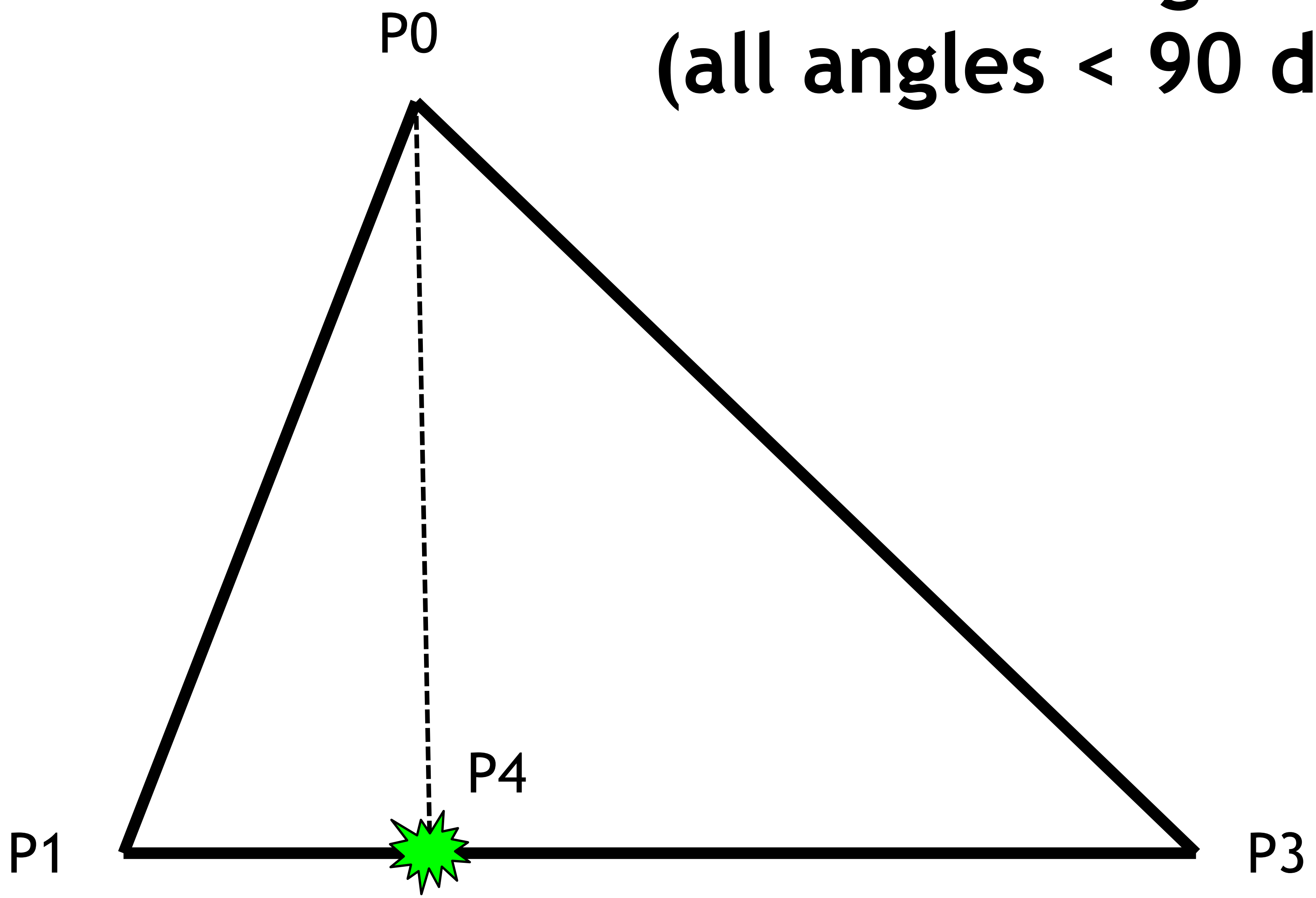
```
width: 0px;
height: 0px;
border-style: solid;
```



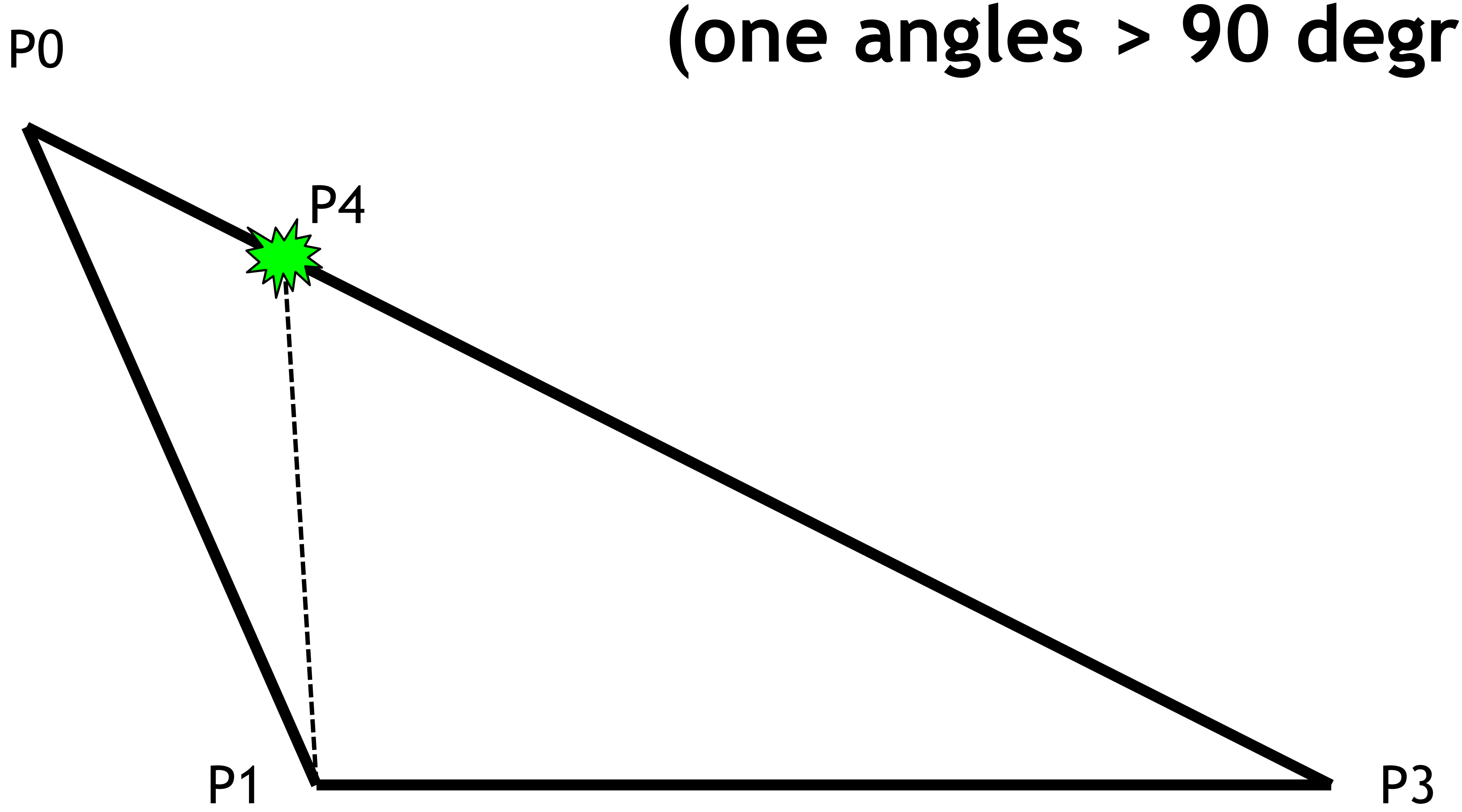
# How to decompose any triangle into right triangles


0
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
6
7
8

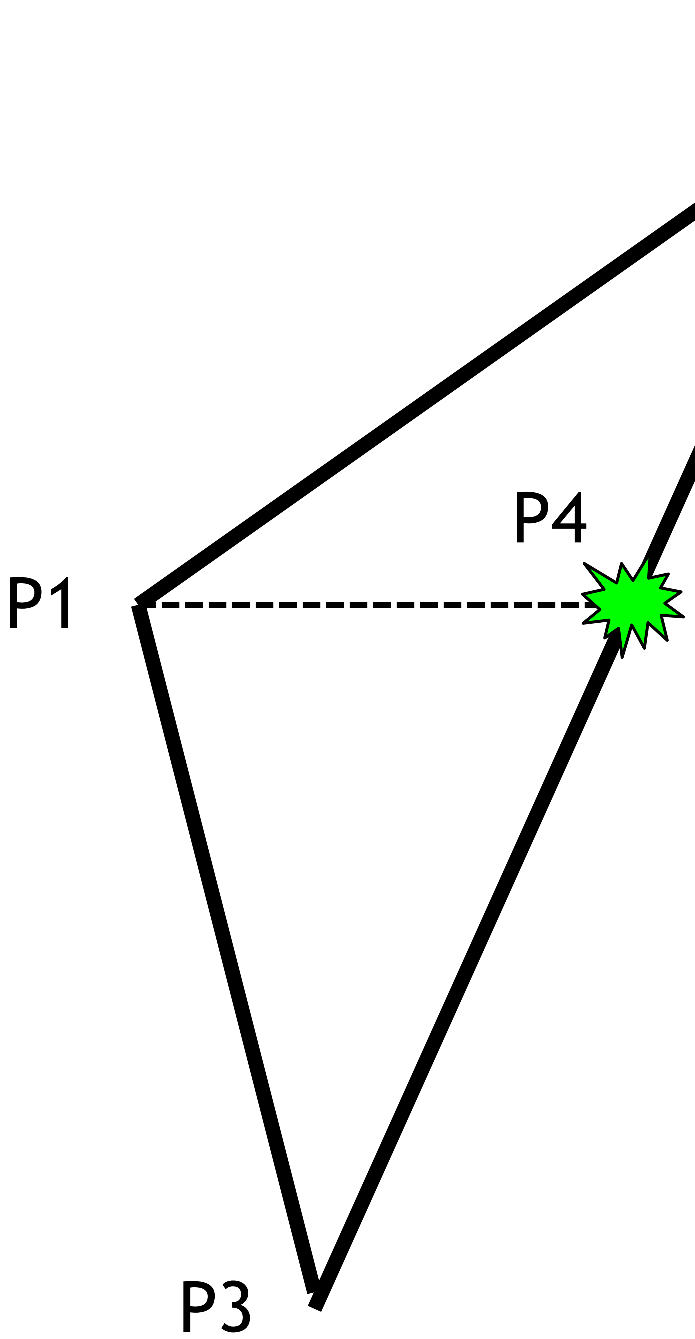
# Horizontal leg & Acute
# (all angles < 90 degree

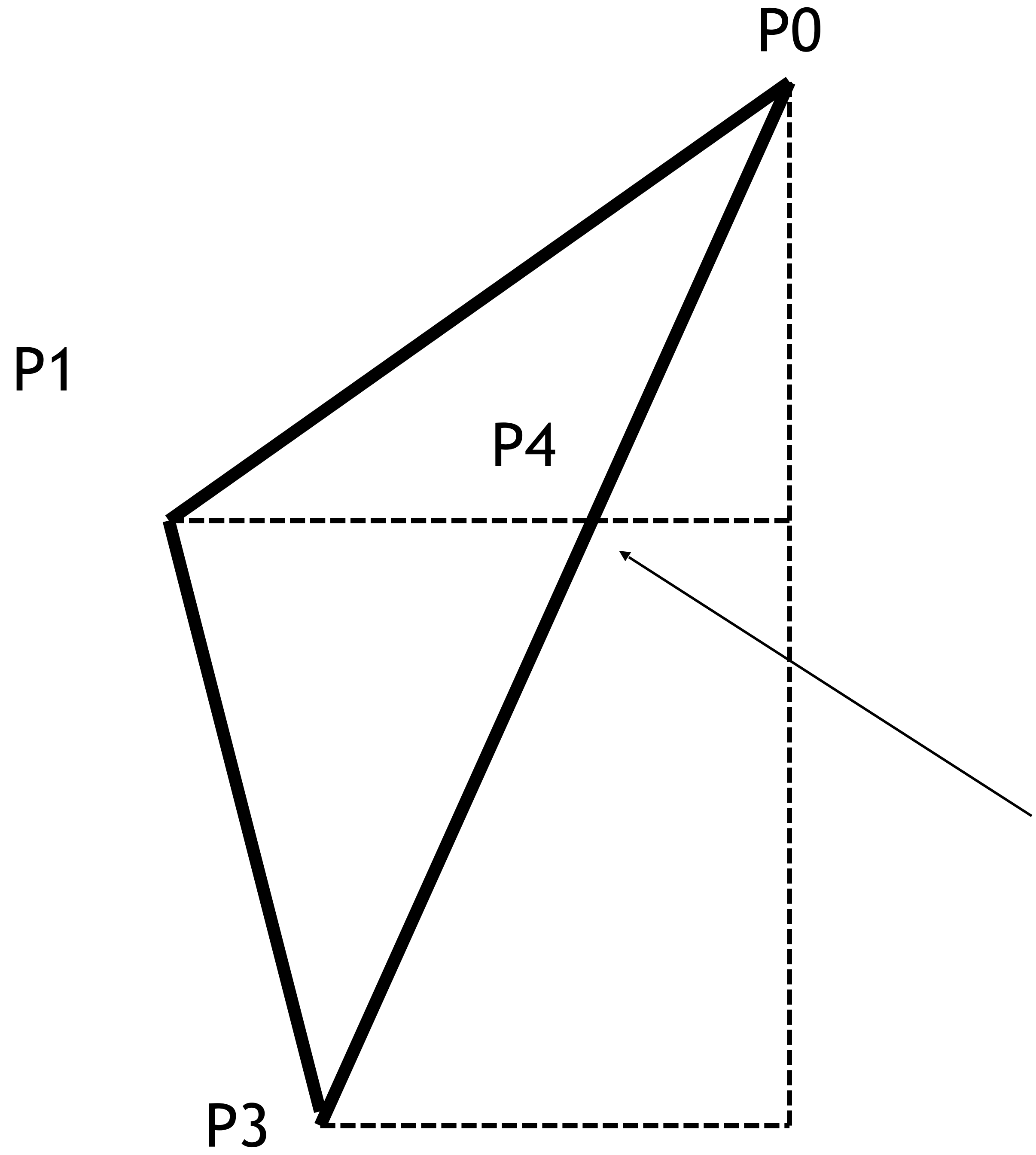


# Horizontal leg & Obtuse (one angles > 90 degree



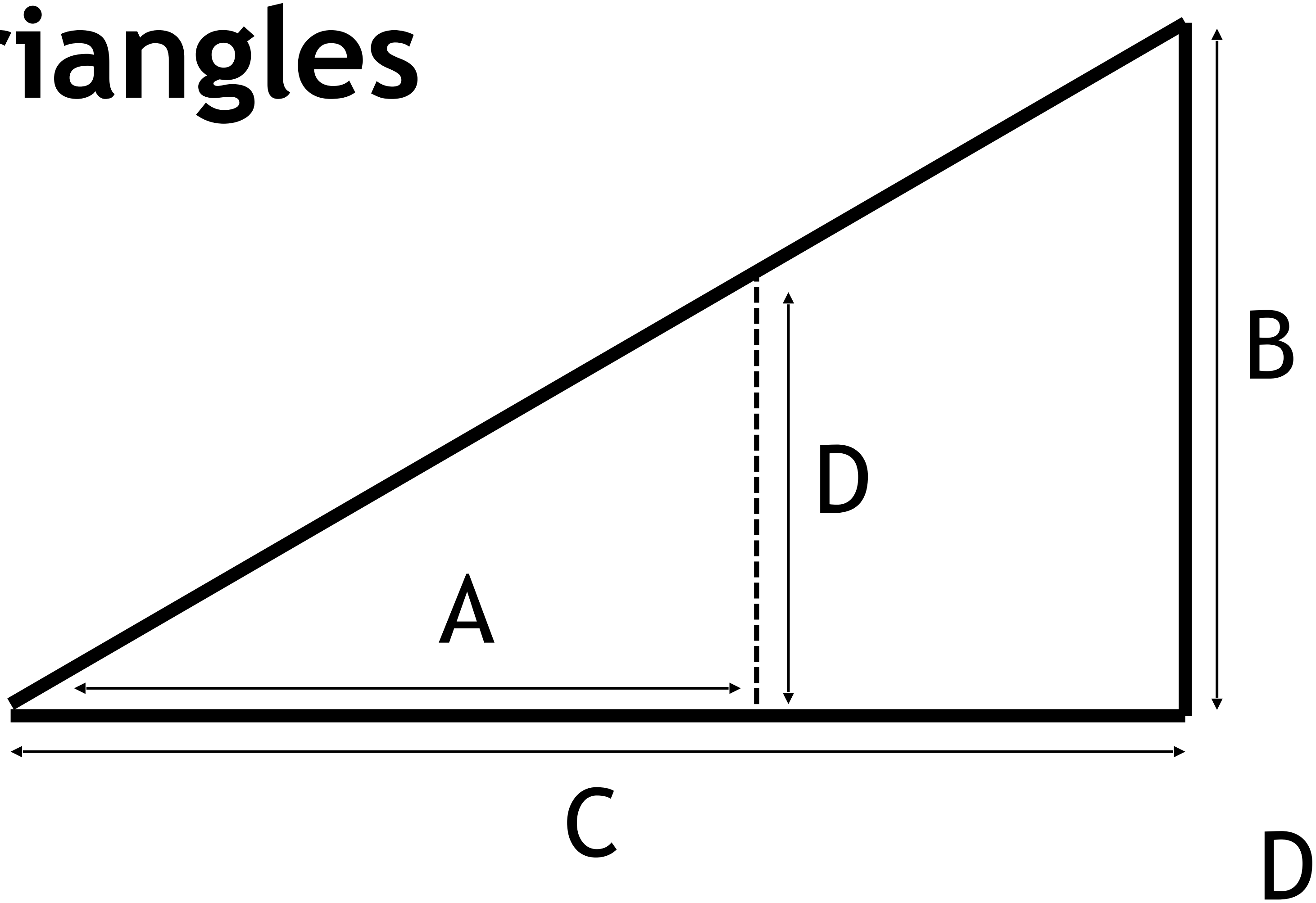
# No Horizontal le





How do we find this point?

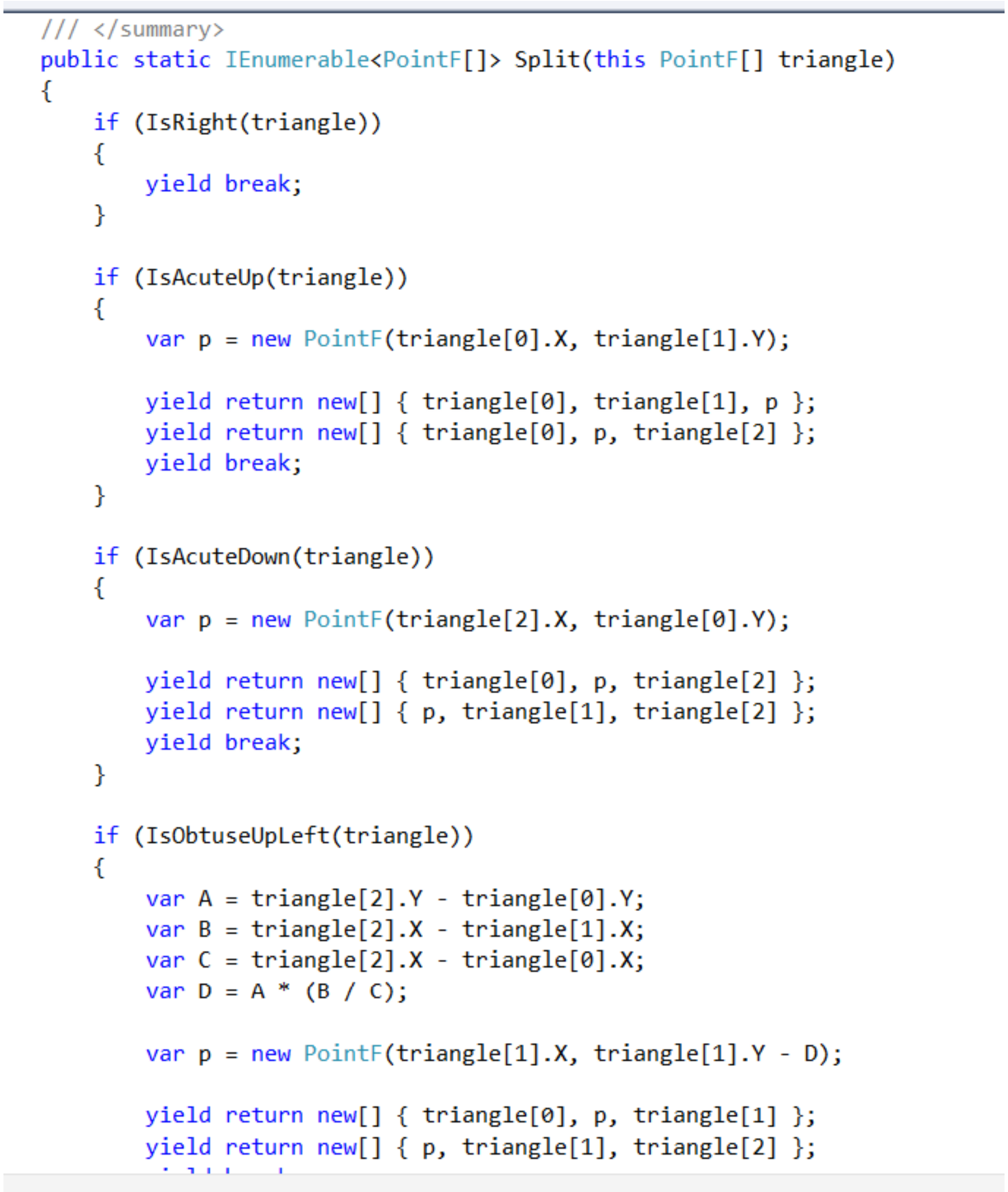
# Similar Triangles



$$D/A = B/C$$

$$\Longrightarrow$$

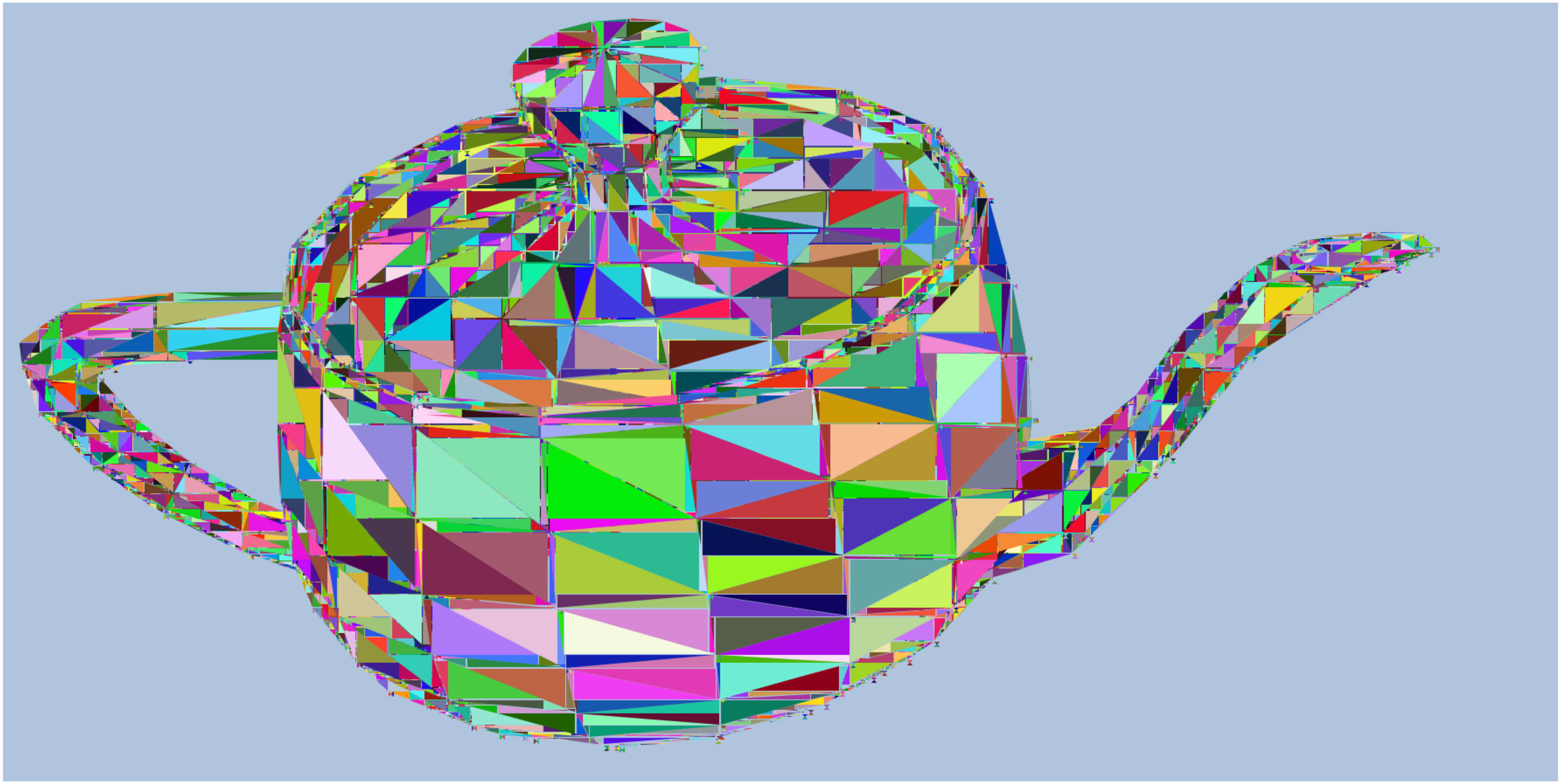
$$D = A^*(B/C)$$

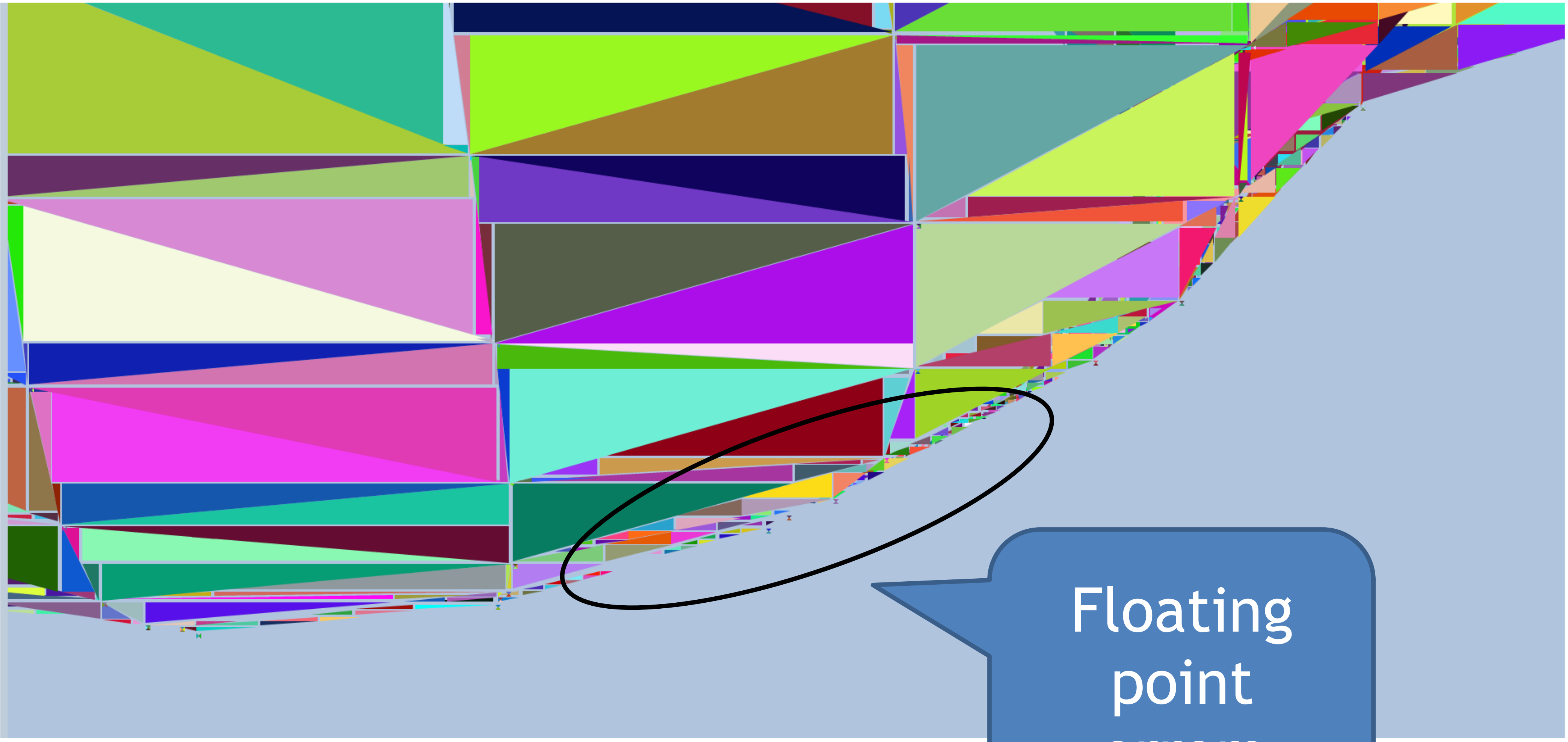


```
/// </summary>
public static IEnumerable<PointF[]> Split(this PointF[] triangle)
{
    if (IsRight(triangle))
    {
        yield break;
    }

    if (IsAcuteUp(triangle))
    {
        var p = new PointF(triangle[0].X, triangle[1].Y);

        yield return new[] { triangle[0], triangle[1], p };
        yield return new[] { triangle[0], p, triangle[2] };
        yield break;
    }

    if (IsAcuteDown(triangle))
    {
        var p = new PointF(triangle[2].X, triangle[0].Y);

        yield return new[] { triangle[0], p, triangle[2] };
        yield return new[] { p, triangle[1], triangle[2] };
        yield break;
    }

    if (IsObtuseUpLeft(triangle))
    {
        var A = triangle[2].Y - triangle[0].Y;
        var B = triangle[2].X - triangle[1].X;
        var C = triangle[2].X - triangle[0].X;
        var D = A * (B / C);

        var p = new PointF(triangle[1].X, triangle[1].Y - D);

        yield return new[] { triangle[0], p, triangle[1] };
        yield return new[] { p, triangle[1], triangle[2] };
```

```
panel.Paint += (s, e) =>
{
    var nrTriangles = 0;
    var graphics = panel.CreateGraphics();

    var divs = new List<string>();
    divs.Add(@"<div style=""position:relative;background-color:#b0c4de; width: 1200px; height: 600px;"">");

    foreach (var triangle in normalized)
    {
        // uncomment to show original triangles
        graphics.DrawPolygon(new Pen(Color.Black, 2), triangle);

        var rc = Triangle.RandomColor();

        var split = new[] { triangle }
                  .Expand(Triangle.Split)
                  .Where(Triangle.IsRight);

        nrTriangles += split.Count();
        foreach (var right in split)
        {
            graphics.FillPolygon(new SolidBrush(Triangle.RandomColor()), right);
            //graphics.DrawPolygon(new Pen(Color.Red, 1), right);

            // uncomment to paint slowly
            //System.Threading.Thread.Sleep(100);
            divs.Add(Div.FromTriangle(Triangle.RandomColor(), right).ToString());

        }
    }
```

Floating point “numbers” are Lawless creatures

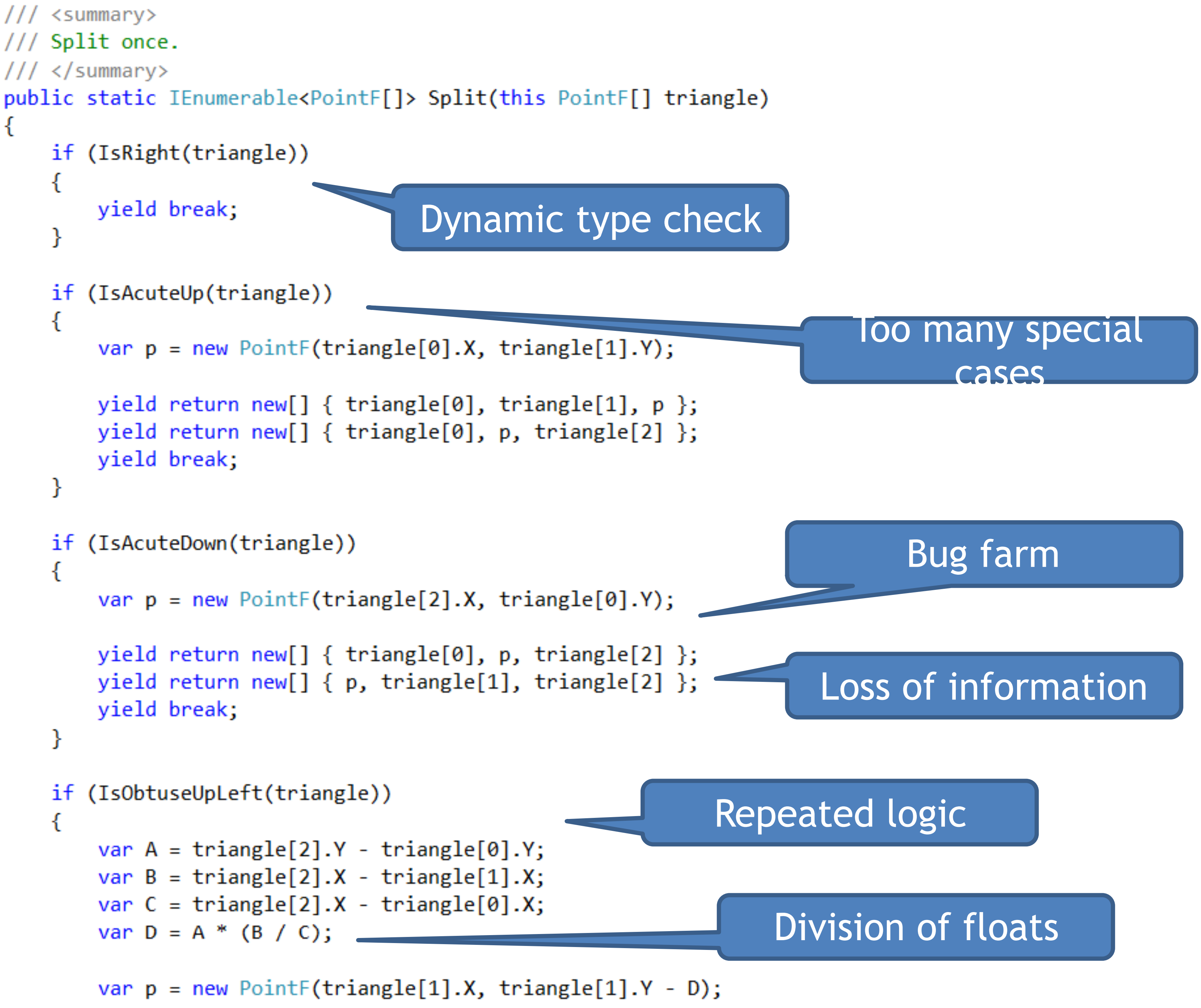




Floating point errors

# Take the Teapot Challenge!

# ~~Possible~~ Needed Improvements



```
/// <summary>
/// Split once.
/// </summary>
public static IEnumerable<PointF[]> Split(this PointF[] triangle)
{
    if (IsRight(triangle))
    {
        yield break;
    }

    if (IsAcuteUp(triangle))
    {
        var p = new PointF(triangle[0].X, triangle[1].Y);

        yield return new[] { triangle[0], triangle[1], p };
        yield return new[] { triangle[0], p, triangle[2] };
        yield break;
    }

    if (IsAcuteDown(triangle))
    {
        var p = new PointF(triangle[2].X, triangle[0].Y);

        yield return new[] { triangle[0], p, triangle[2] };
        yield return new[] { p, triangle[1], triangle[2] };
        yield break;
    }

    if (IsObtuseUpLeft(triangle))
    {
        var A = triangle[2].Y - triangle[0].Y;
        var B = triangle[2].X - triangle[1].X;
        var C = triangle[2].X - triangle[0].X;
        var D = A * (B / C);

        var p = new PointF(triangle[1].X, triangle[1].Y - D);
```

Dynamic type check

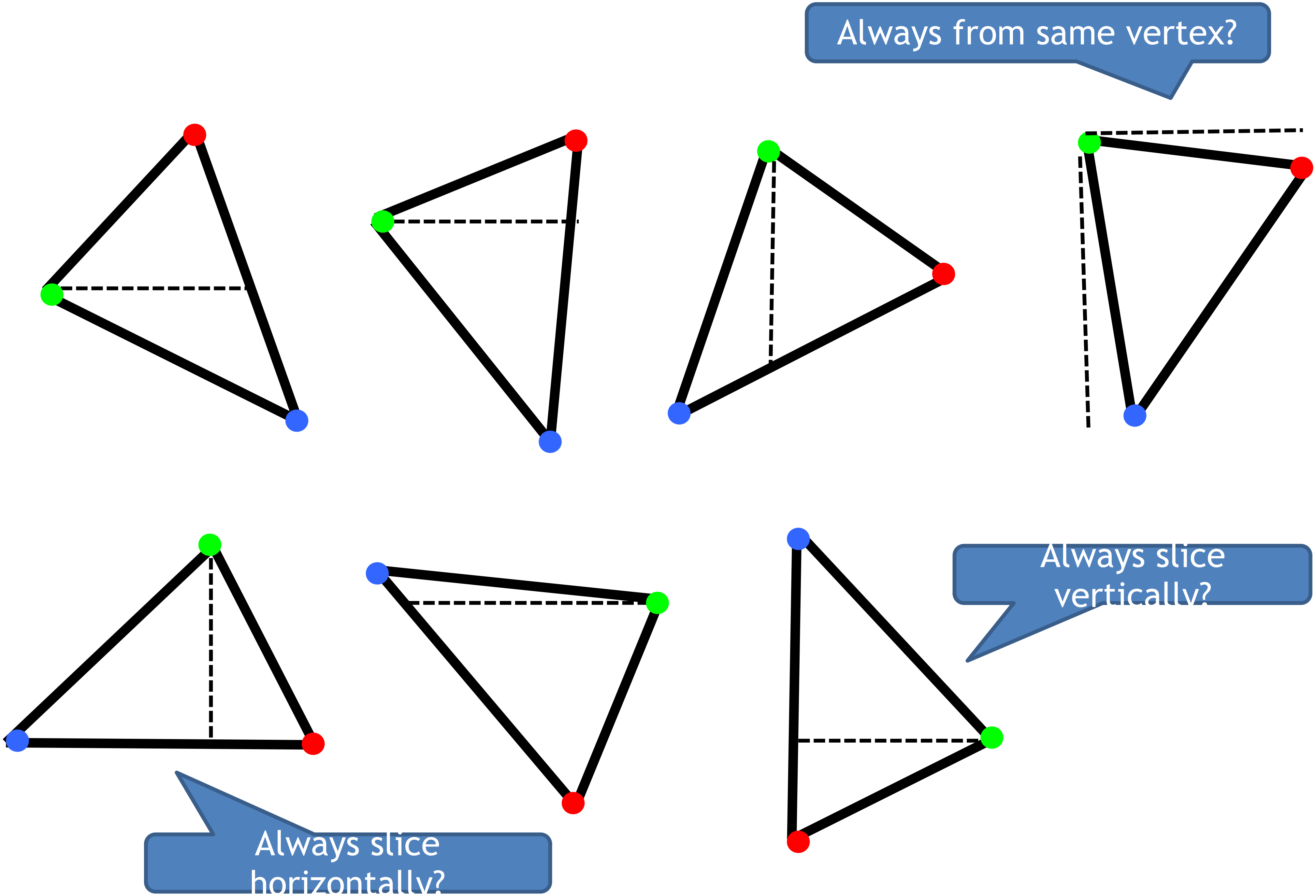
Too many special cases

Bug farm

Loss of information

Repeated logic

Division of floats

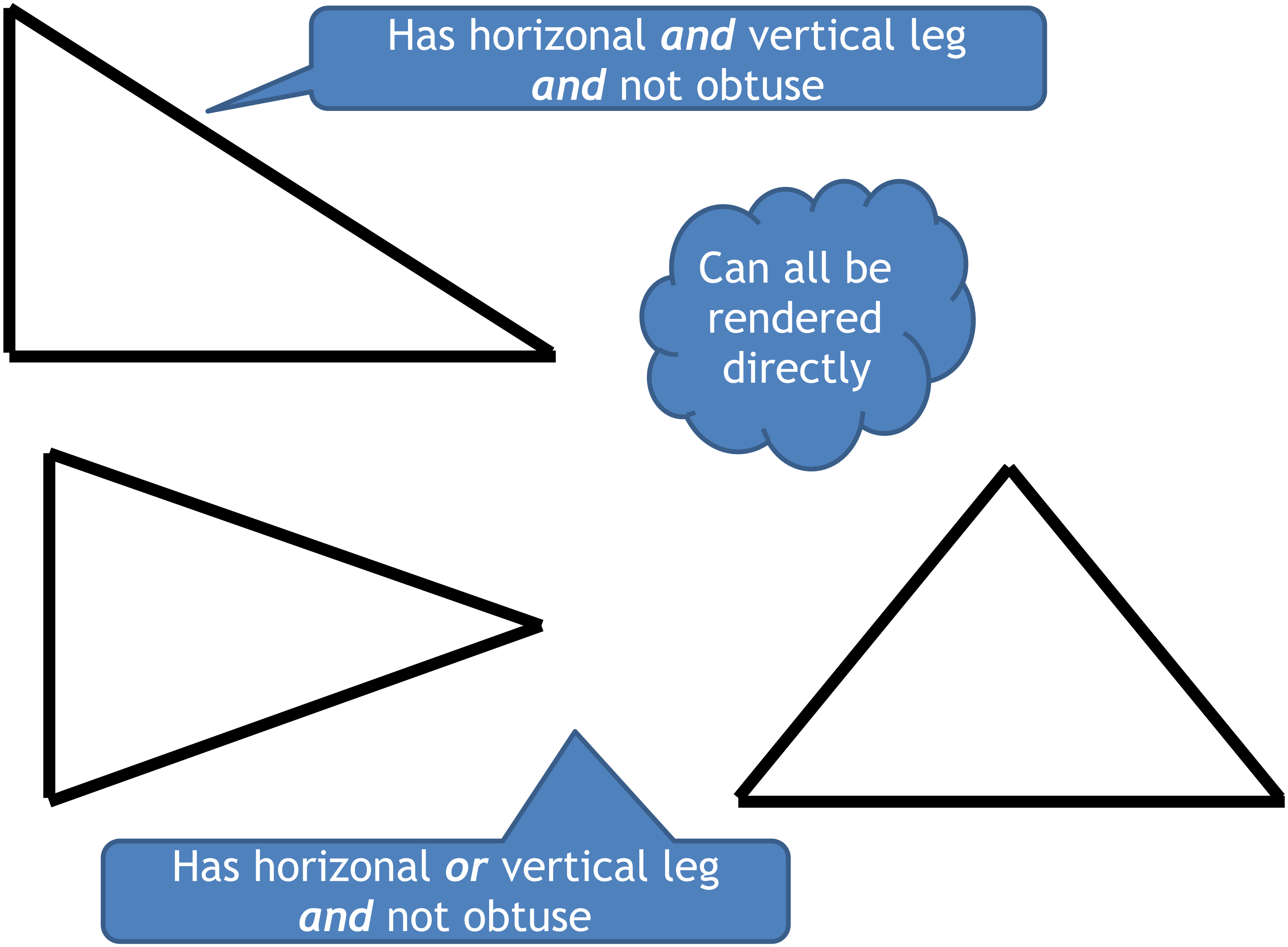


Always from same vertex?

Always slice vertically?

Always slice horizontally?

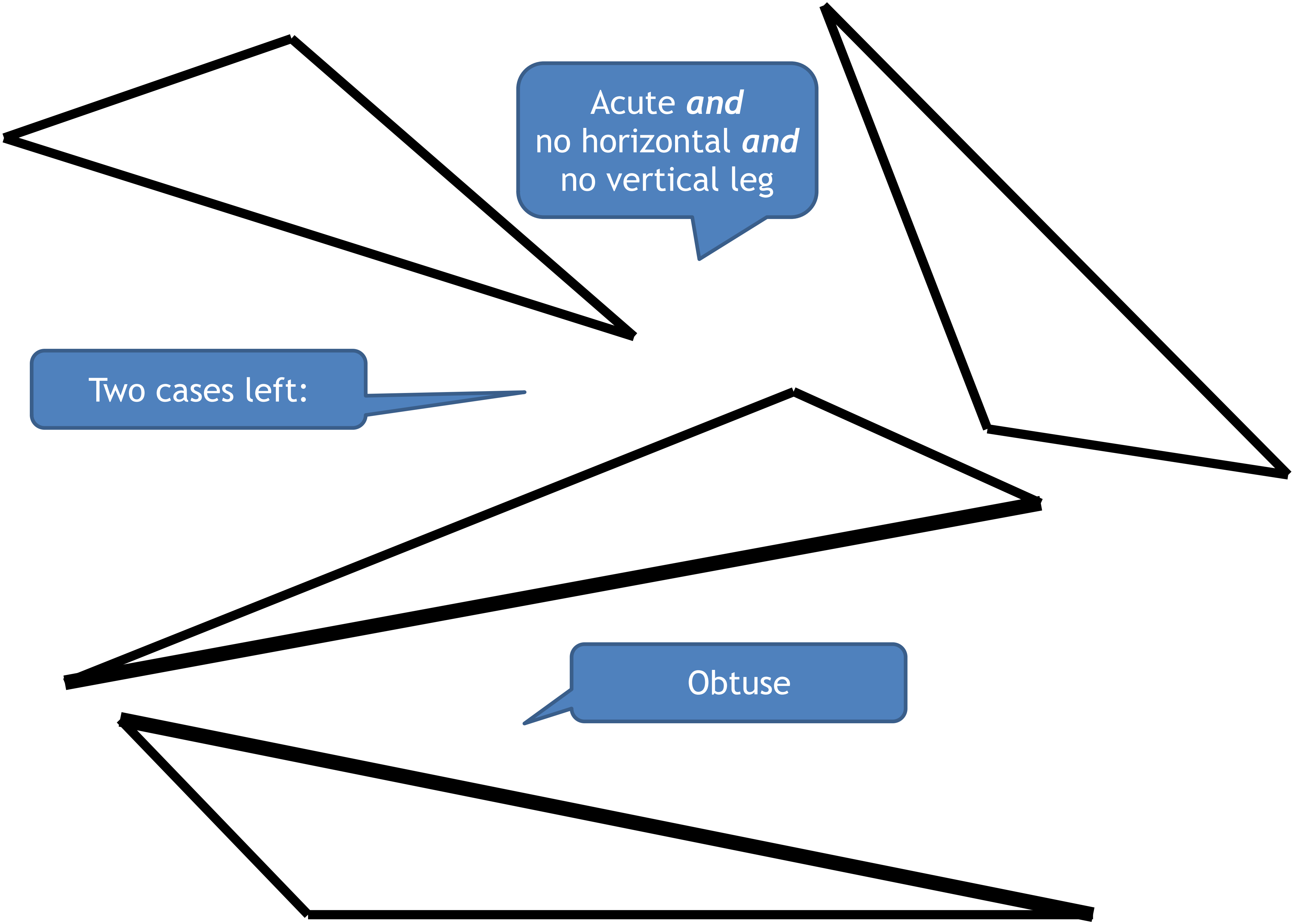
# Generalize (and simplify)



Has horizonal ***and*** vertical leg ***and*** not obtuse

Can all be rendered directly

Has horizonal ***or*** vertical leg ***and*** not obtuse



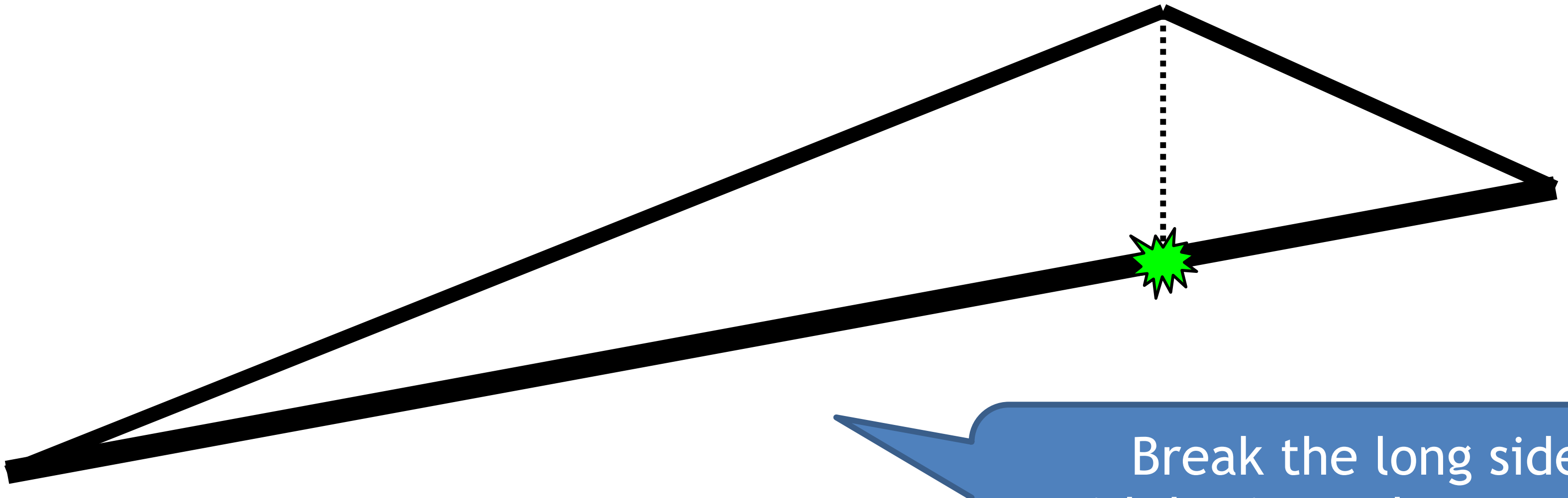
Two cases left:

Acute *and*
no horizontal *and*
no vertical leg

Obtuse

# Theorem

In a triangle, the longest side is across from the largest angle.



Break the long side with horizontal or vertical shot from short side

A triangle is obtuse when ***any*** angle is obtuse

```
public bool IsObtuse(out int i){}
```

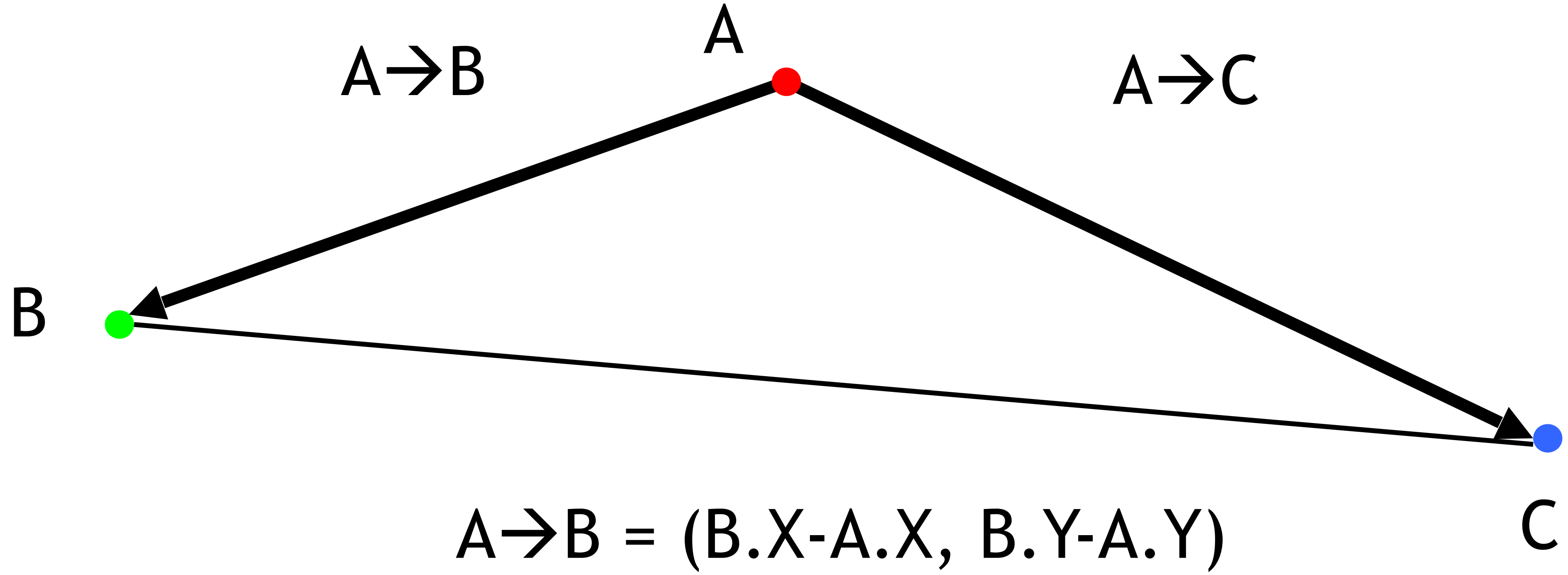
Return index of point with obtuse angle
as witness.

A triangle is acute when ***all*** angles are acute

```
public bool IsAcute(out int i){}
```

Return index of point with the largest angle
as witness.

# How to determine if angle is Right, Acute, or Obtuse



A→B = (B.X-A.X, B.Y-A.Y)

# Dot Product

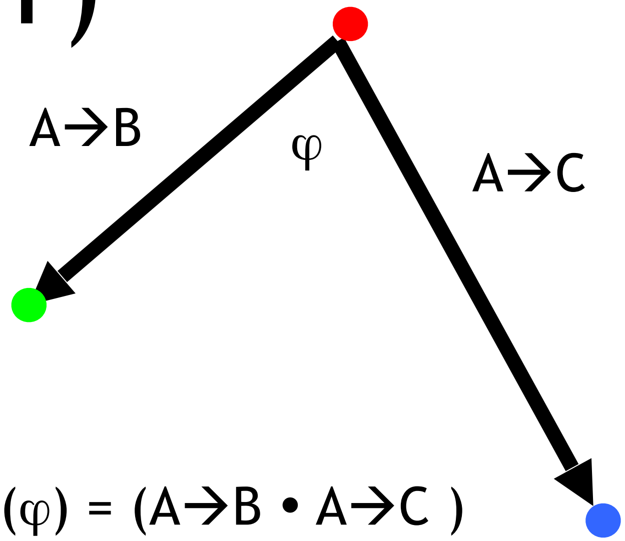
$$A\rightarrow B \bullet C\rightarrow D$$

$$=$$

$$(A\rightarrow B).X*(C\rightarrow D).X + (A\rightarrow B).Y*(C\rightarrow D).Y$$

$$=$$

$$(\ B.X-A.X)*(D.X-C.X) + (B.Y-A.Y)*(D.Y-C.Y)$$



$$A\rightarrow B \bullet A\rightarrow C$$

$= 0$ → Right

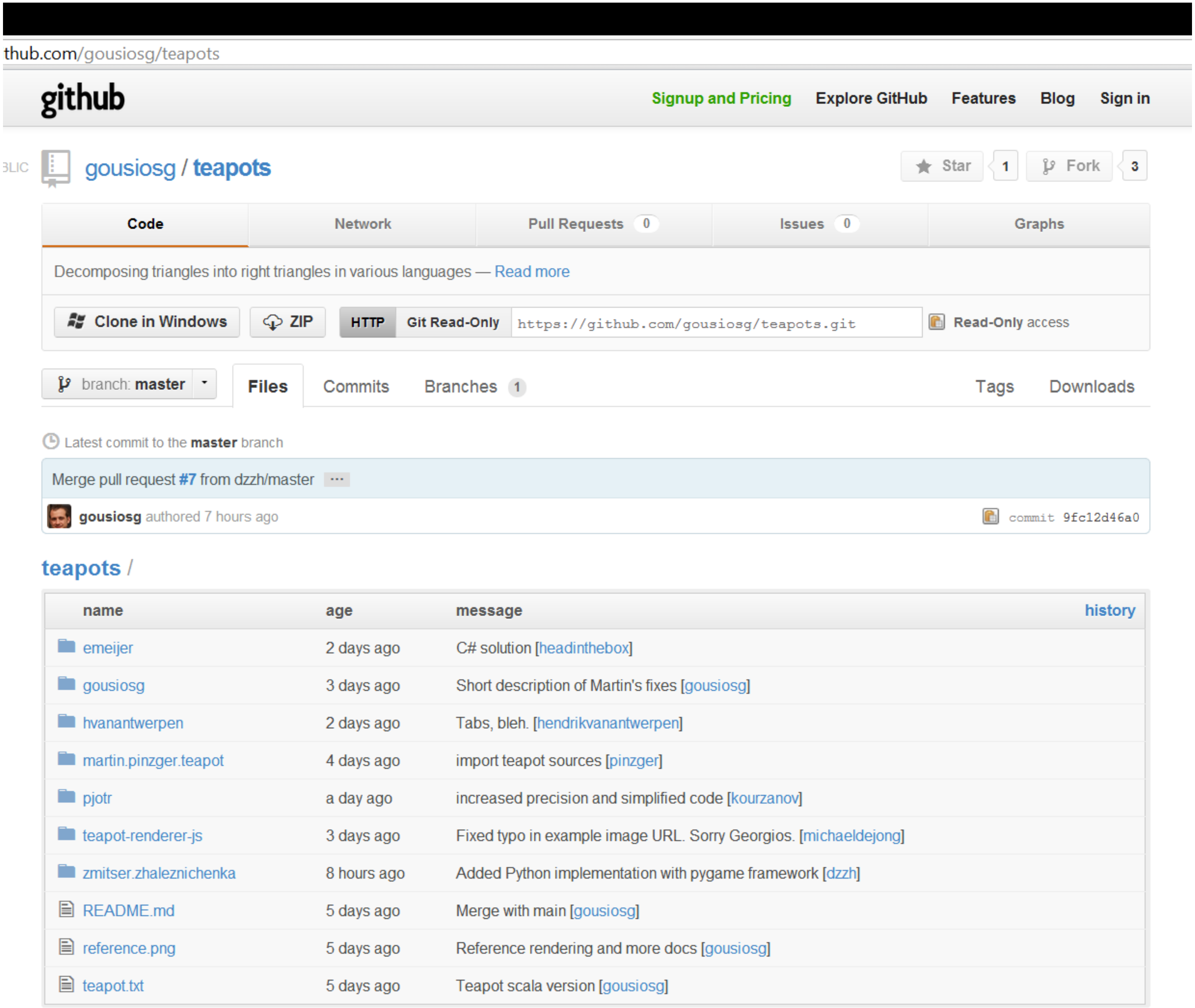
$> 0$ → Acute

$< 0$ → Obtuse

$$\text{Cosine}(\varphi) = (A\rightarrow B \bullet A\rightarrow C\ )\ /\ |A\rightarrow B|*|A\rightarrow C|$$

$$|A\rightarrow B| = \sqrt{}\ (A\rightarrow B \bullet A\rightarrow B)$$

# https://github.com/gousiosg/teapots



Decomposing triangles into right triangles in various languages — Read more

Latest commit to the **master** branch

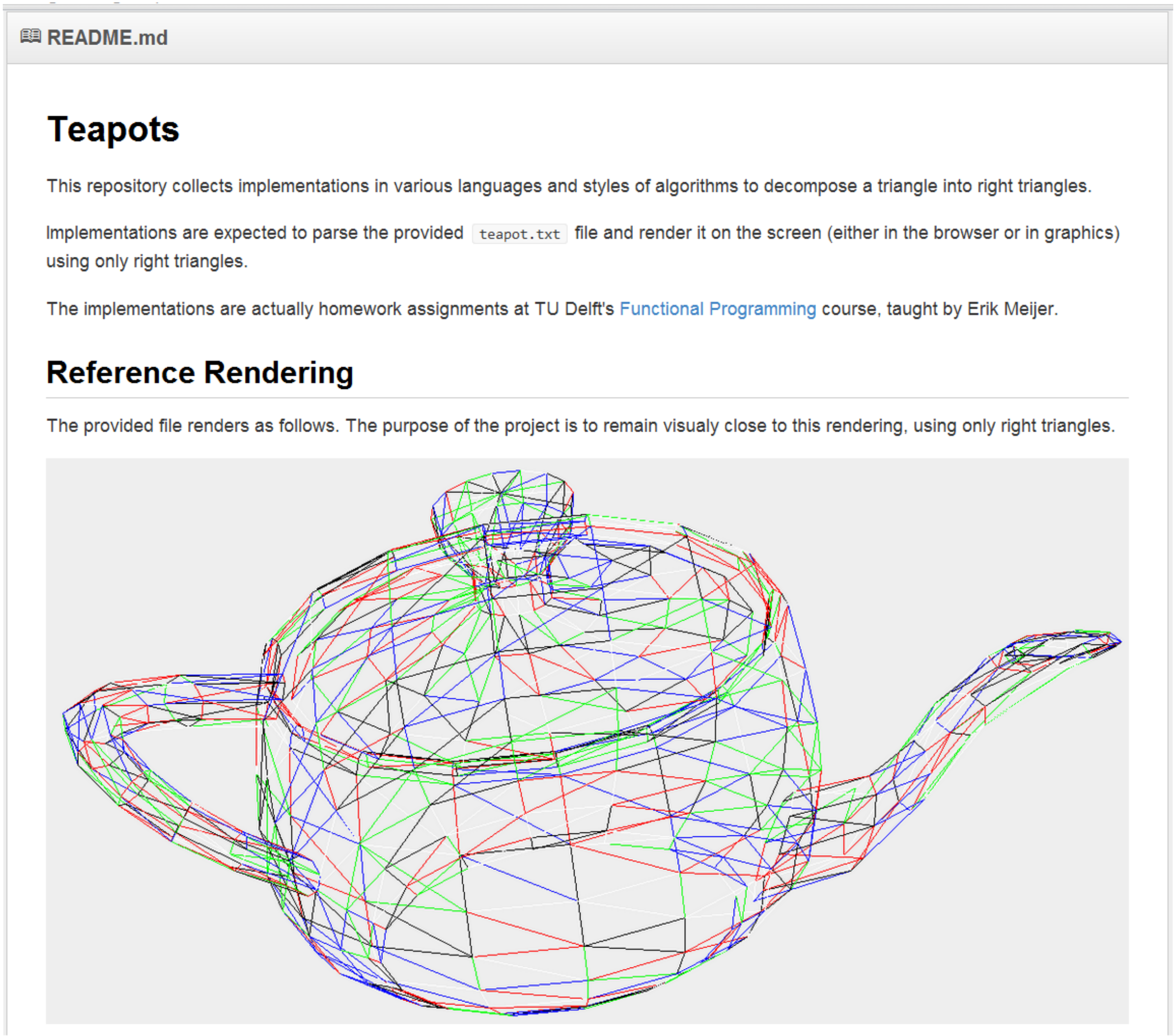
Merge pull request #7 from dzzh/master …

**gousiosg** authored 7 hours ago

commit 9fc12d46a0

teapots /

| name | age | message |
|---|---|---|
| emeijer | 2 days ago | C# solution [headinthebox] |
| gousiosg | 3 days ago | Short description of Martin's fixes [gousiosg] |
| hvanantwerpen | 2 days ago | Tabs, bleh. [hendrikvanantwerpen] |
| martin.pinzger.teapot | 4 days ago | import teapot sources [pinzger] |
| pjotr | a day ago | increased precision and simplified code [kourzanov] |
| teapot-renderer-js | 3 days ago | Fixed typo in example image URL. Sorry Georgios. [michaeldejong] |
| zmitser.zhaleznichenka | 8 hours ago | Added Python implementation with pygame framework [dzzh] |
| README.md | 5 days ago | Merge with main [gousiosg] |
| reference.png | 5 days ago | Reference rendering and more docs [gousiosg] |
| teapot.txt | 5 days ago | Teapot scala version [gousiosg] |



# Teapots

This repository collects implementations in various languages and styles of algorithms to decompose a triangle into right triangles.

Implementations are expected to parse the provided `teapot.txt` file and render it on the screen (either in the browser or in graphics) using only right triangles.

The implementations are actually homework assignments at TU Delft's Functional Programming course, taught by Erik Meijer.

## Reference Rendering

The provided file renders as follows. The purpose of the project is to remain visualy close to this rendering, using only right triangles.

# Do More
# With
# Less!