Zero to 2-d

(How to build a game from scratch)

```
<canvas id="game-view">
```


```
</canvas>
```

.io-like human vs zombies arcade game

Goals:

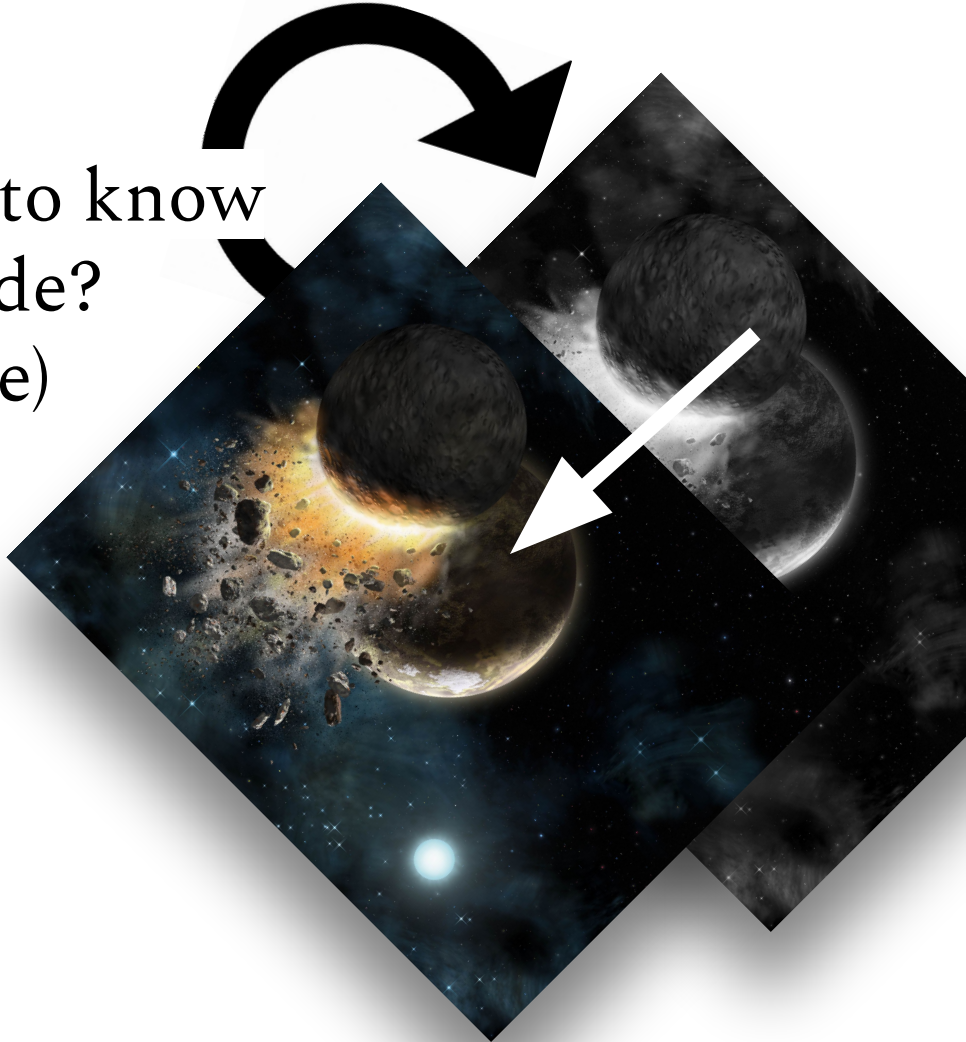
- Learn how to build a game with just your text editor and a compiler (or interpreter!)
- **Intuition > Optimization**
  - Visualizing a single approach can lead to understanding the more optimal ones

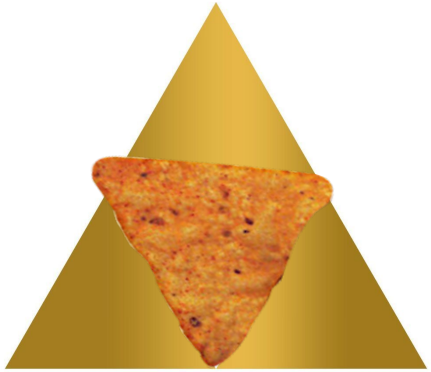
What we're not doing today:

- Rasterization, barycentric coordinates, a ton of linear algebra, shaders, concurrency, etc.
- `@see CU Graphics Club (6-7pm Tues, same room)`

Things we might want to know

- Do two objects collide?
- How do I turn (rotate)
- How do I move

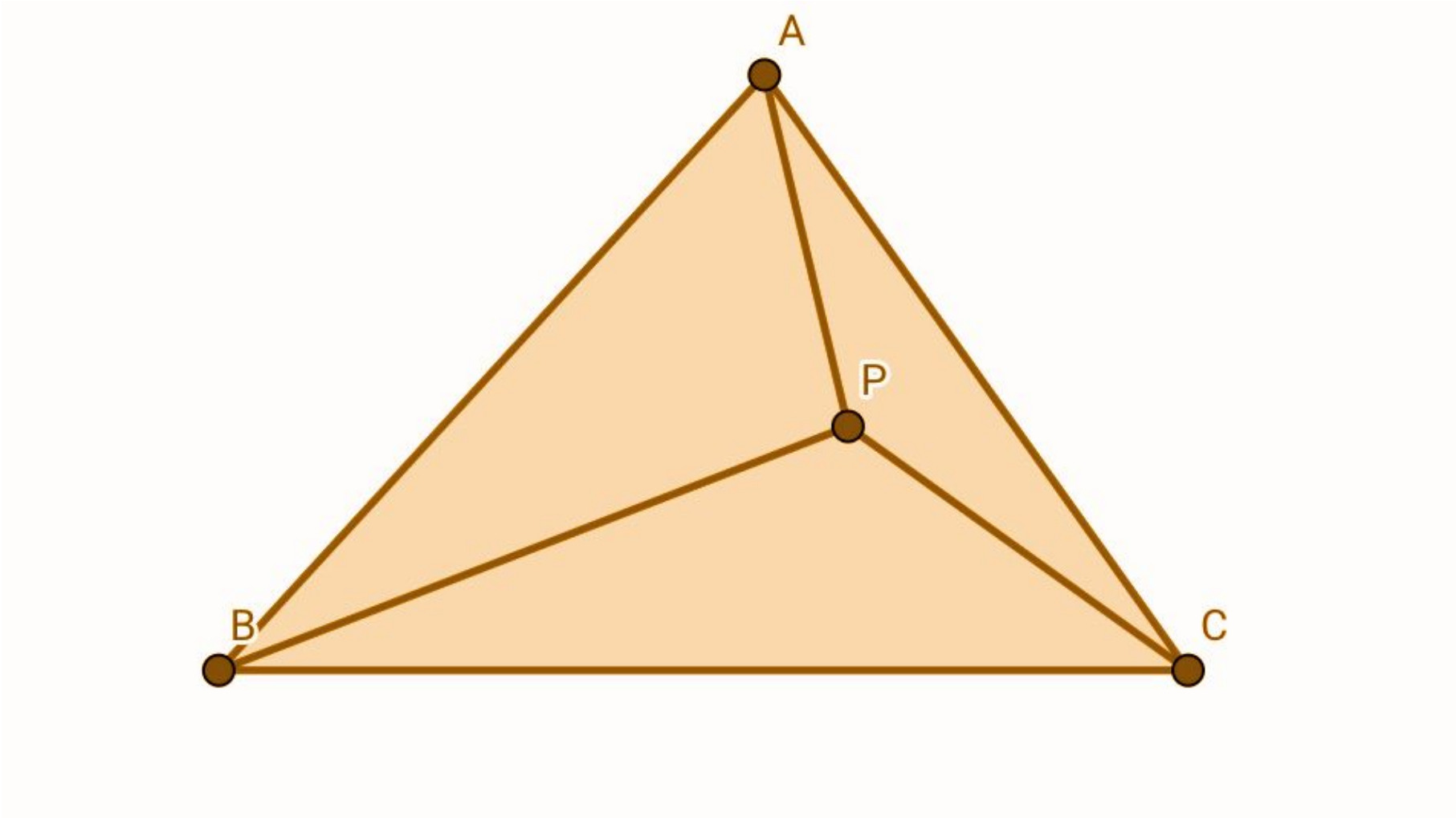


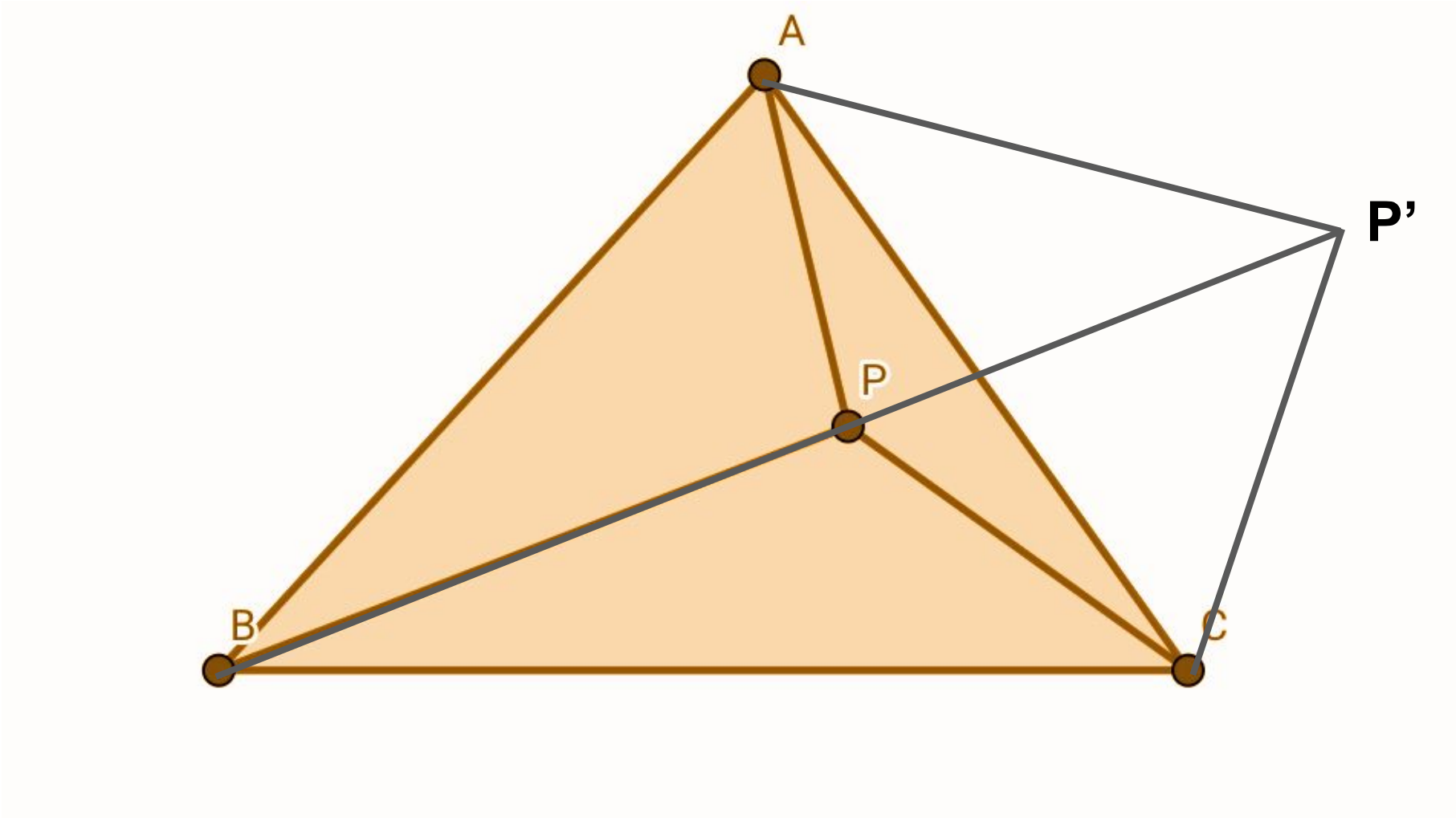


Triangles

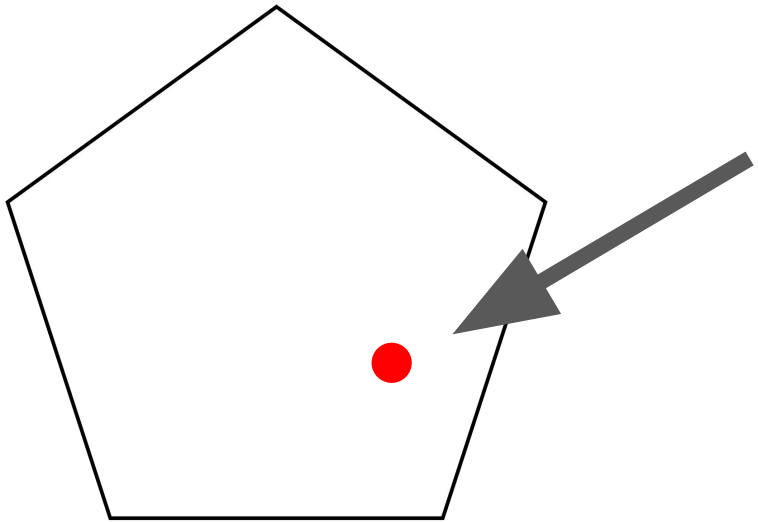
Fun facts

- Most basic polygon
- Can be represented as list of 3 points
- Triangulation means two things
- Defines a plane
- Inspired by Doritos™

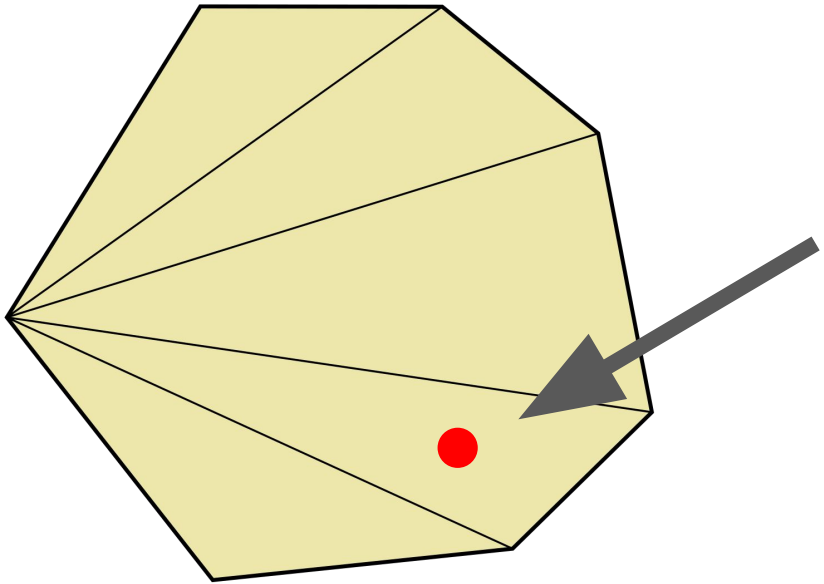

A
P
B
C


A
P'
P
B
C

```
const contains = (t, p, err) => {
  const sumArea = _area(t.a, t.b, p) +
                  _area(t.a, t.c, p) +
                  _area(t.b, t.c, p)

  const delta = Math.abs(area(t) - sumArea)

  return delta < err
}

const area = t => _area(t.a, t.b, t.c)

const _area = (a, b, c) => (
  Math.abs(
    a[0] * (b[1] - c[1]) +
    b[0] * (c[1] - a[1]) +
    c[0] * (a[1] - b[1])
  ) / 2
)
```

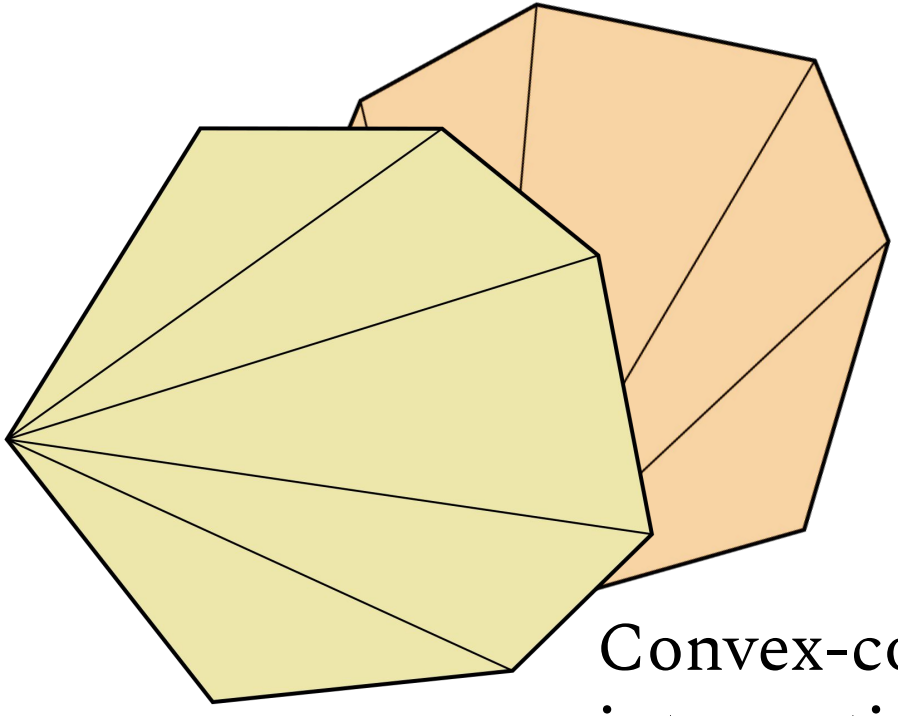


How do we know if the red point intersects this polygon?



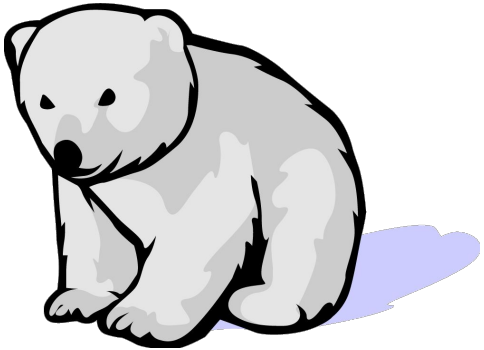
How do we know if the red point intersects this polygon?

Fan triangulation of convex polygon of V vertices can be done in O(V) time



Convex-convex polygon intersection?

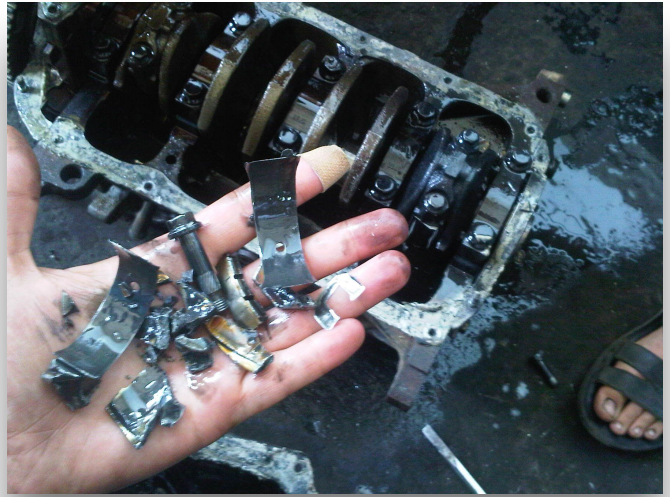
```
const _triangulate = poly => {
  const tris = []
  for (let i = 1; i < poly.length - 1; i++) {
    tris.push(Triangle.create(poly[0], poly[i], poly[i + 1]))
  }

  return tris
}

const contains = (poly, point) => (
  poly.tris.some(t => Triangle.contains(t, point, 0.05))
)
```



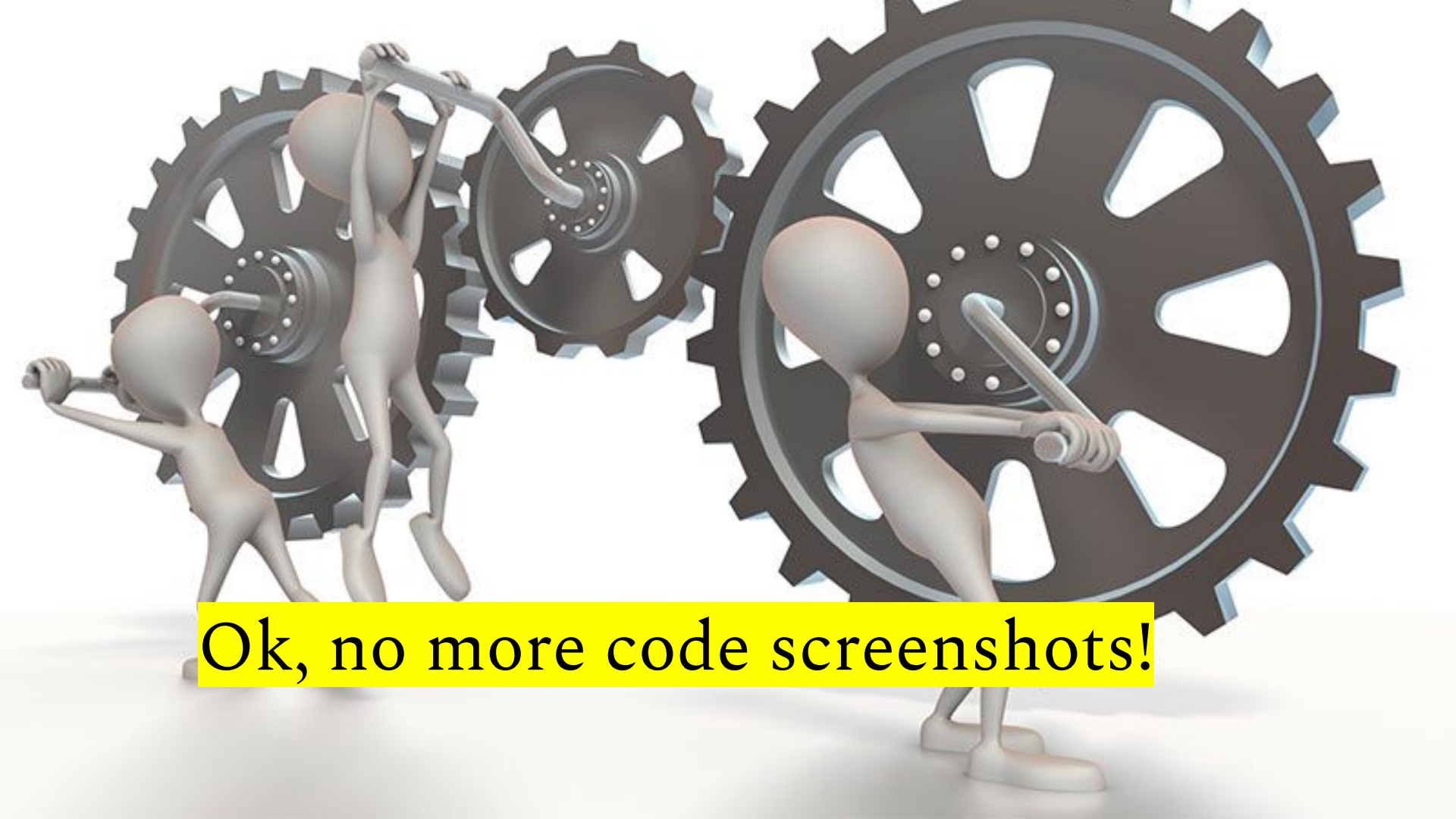
$$x = r \times \cos(\theta)$$

$$y = r \times \sin(\theta)$$

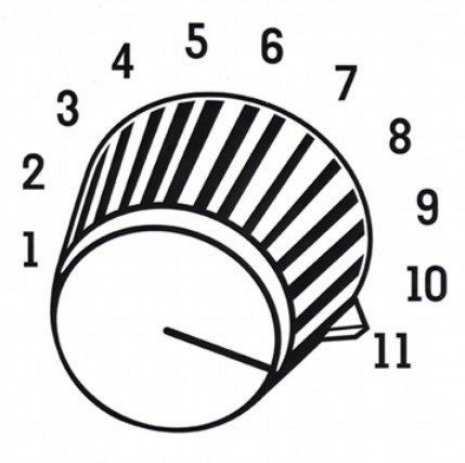
Engine Internals



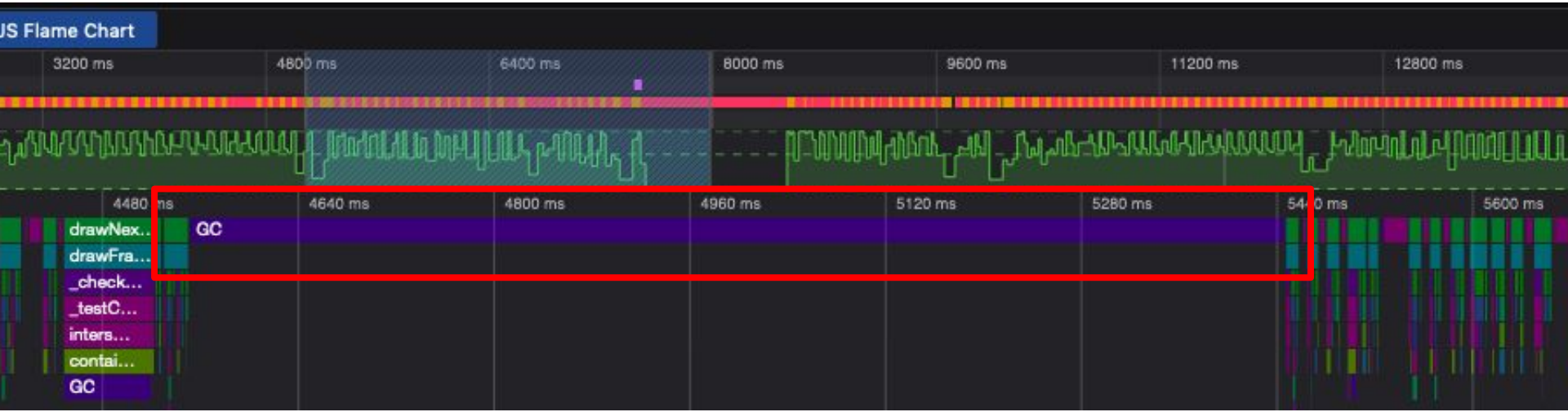
```
const create = (canvas, width, height) => ({
  suspended: false,
  ctx: canvas.getContext('2d', { alpha: false }),
  frame: { w: () => canvas.width, h: () => canvas.height },
  width,
  height,
  objs: [],
  listeners: {
    keyDown: [],
    keyUp: []
  },
  elapsed: 0,
  frames: 0,
  updates: 0
})
```


Ok, no more code screenshots!

Profiling...

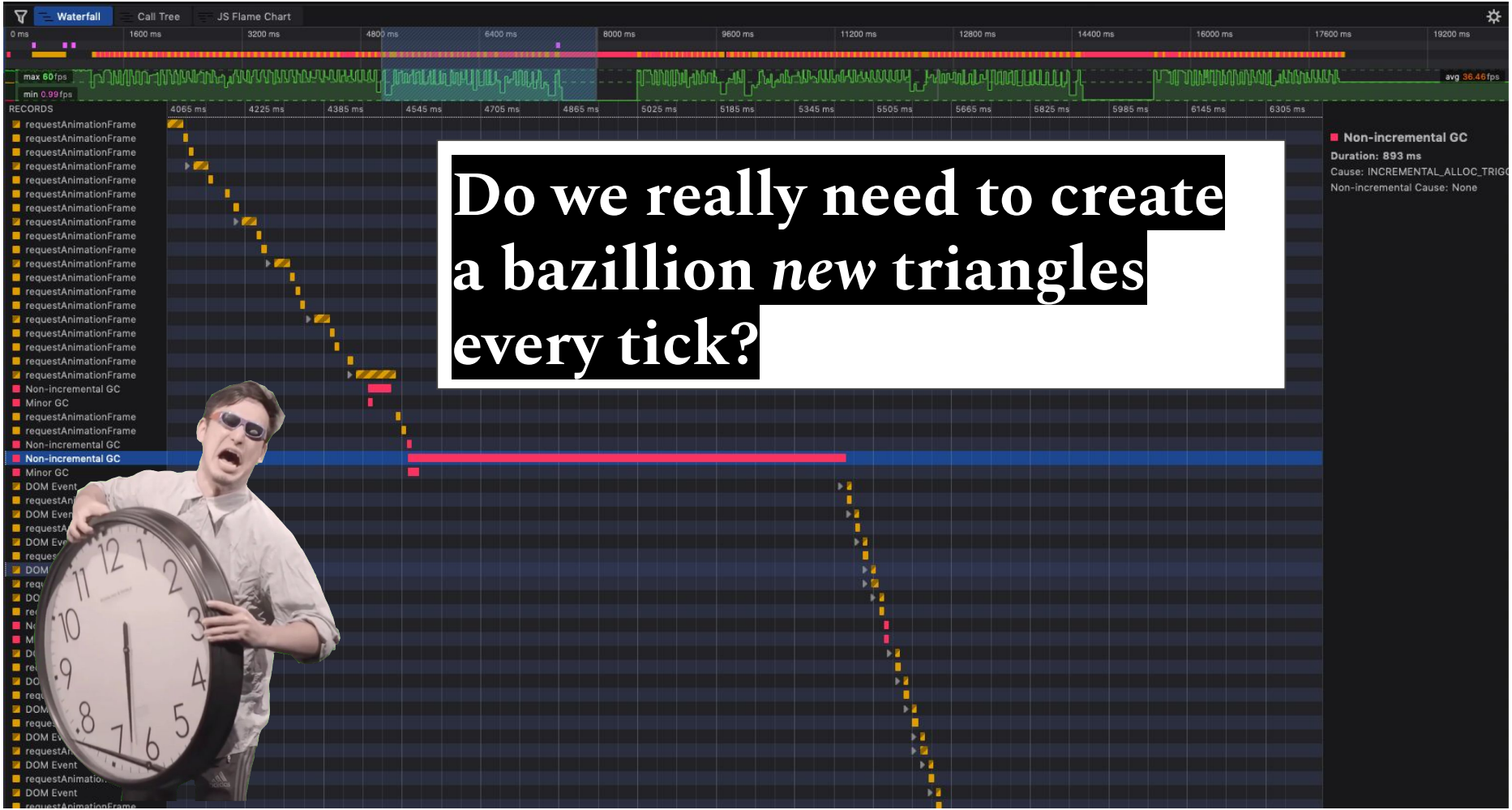


Getting super big lag spikes...
Time to move to C or C++?



Total Ti...	Total ...	Self Time	Self C...	Sam...	Function
153.7...	49.8...	1,153.7...	49.8...	1126	GC
08.83...	17.65%	408.83...	17.65%	399	Idle
53.08 ...	10.92%	253.08 ...	10.92%	247	JIT
84.43 ...	7.96%	184.43 ...	7.96%	180	DOM
2.99 ms	3.58%	82.99 ms	3.58%	81	contains ConvexPoly.js:56:17 localhost:8000
6.35 ms	2.43%	56.35 ms	2.43%	55	create Triangle.js:1:15 localhost:8000
1.76 ms	1.37%	31.76 ms	1.37%	31	createLookTri Zombie.js:24:22 localhost:8000
6.64 ms	1.15%	26.64 ms	1.15%	26	getObject Engine.js:19:18 localhost:8000
1.52 ms	0.93%	21.52 ms	0.93%	21	Graphics
6.39 ms	0.71%	16.39 ms	0.71%	16	update Zombie.js:48:23 localhost:8000
1.27 ms	0.49%				ost:8000
0.25 ms	0.44%				
9.22 ms	0.40%				55:36 localhost:8000
8.20 ms	0.35%	8.20 ms	0.35%	8	draw Zombie.js:8:21 localhost:8000
7.17 ms	0.31%	7.17 ms	0.31%	7	_checkCollisions Engine.js:86:25 localhost:8000
7.17 ms	0.31%	7.17 ms	0.31%	7	Layout
6.15 ms	0.27%	6.15 ms	0.27%	6	contains Triangle.js:3:17 localhost:8000
5.12 ms	0.22%	5.12 ms	0.22%	5	_triangulate ConvexPoly.js:47:21 localhost:8000
4.10 ms	0.18%	4.10 ms	0.18%	4	getObject/< Engine.js:19:46 localhost:8000
4.10 ms	0.18%	4.10 ms	0.18%	4	drawLife RenderUtils.js:1:24 localhost:8000

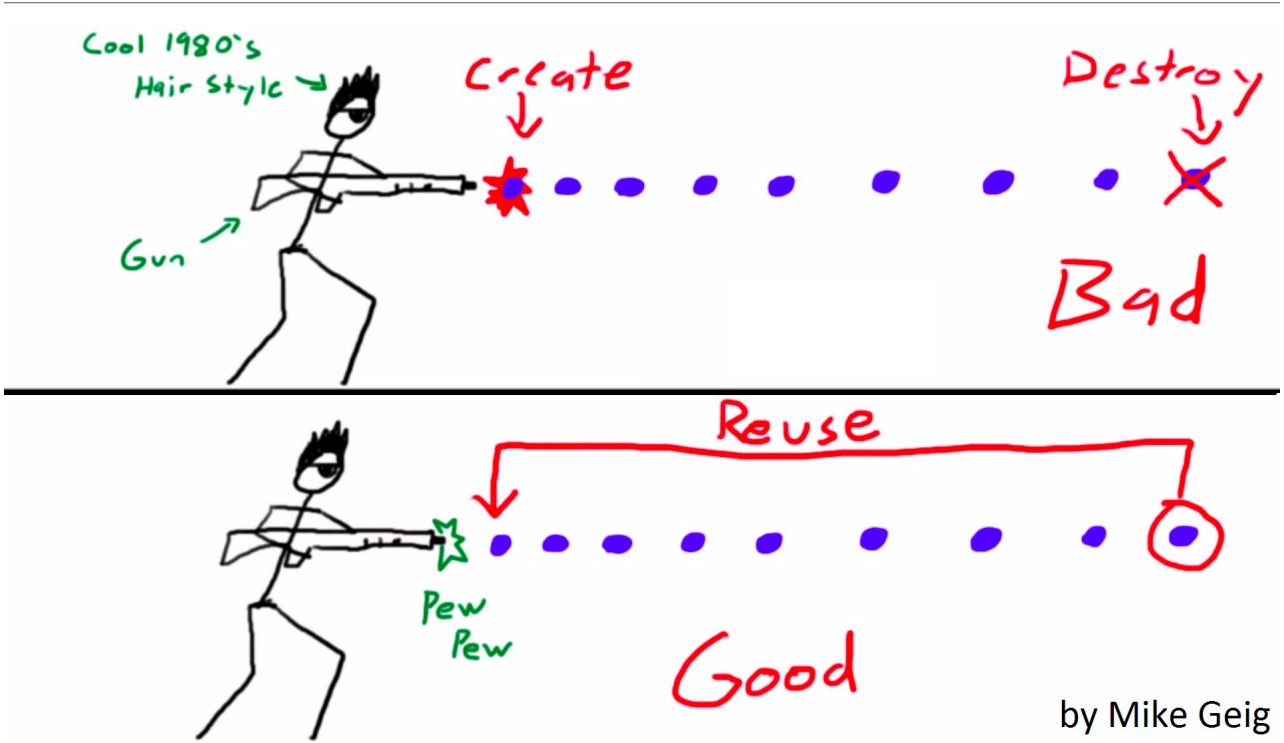
These are probably related...



Do we really need to create a bazillion *new* triangles every tick?

(well not really, but you get the point)

Visual Example of Object Pooling



https://gamedevn.wordpress.com/2015/09/27/optimizacion-rendimiento-videojuego-object-pool-pattern/

github.com/CU-Boulder-Game-Dev/zombruh