A4. GPU / GLSL assignment

$\rightarrow$ make small changes $\nearrow$ debug with colors

$\rightarrow$ requires ;

20 – picking and interaction

Input for Interaction

```
var clickStart: MousePosition | null = null;
var mousePosition: MousePosition | undefined = undefined;

canvas.onmousedown = (ev: MouseEvent) => { }
canvas.onmouseup = (ev: MouseEvent) => { }
canvas.onmousemove = (ev: MouseEvent) => { }
canvas.onmouseout = (ev: MouseEvent) => { }
```

Touch events

are events

Polling vs Asynchronous Events

some native APIs call function to get input

callbacks → most interative apps

get some input (each frame)

→ if input " deal with it "

→ while rendering continuously

Javascript is single threaded

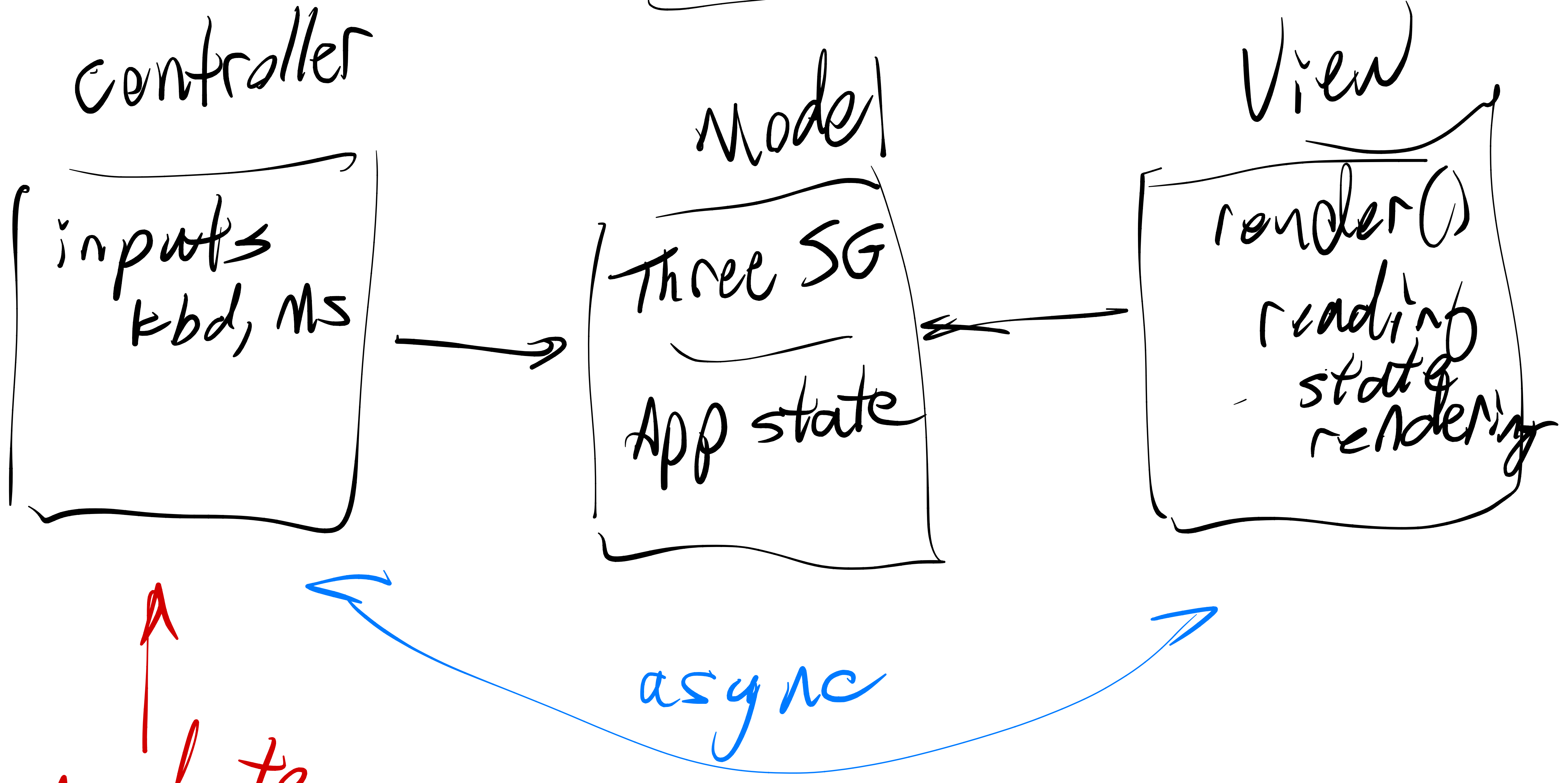
initialize() ← setup things → constructor of our main object

start rAF → $\overset{\text{call}}{\text{"render()"}}$

```
render() {
    do something with mousePosition
        or with click
        or keyboard
    render stuff
    rAF(render) → schedules another call to render
}
```

↳ calls on next possible render time

MVC



need to leave model in consistent

← easy with single threading

in contrast Java, C#, ...

single threading in a pain

raf → triggers events

timeOut (functionclosure, delay in ms);

Promises

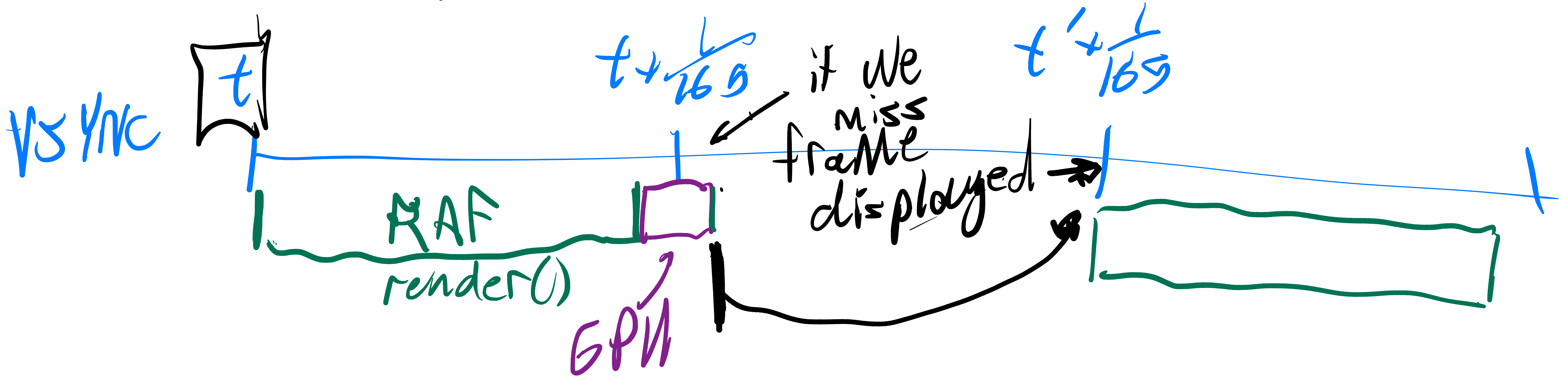
```
p = new Promise( (res, rej) => {
        do some stuff

        res( result );   or   rej(error)
}).then ((res) => { console.log(res)})
```

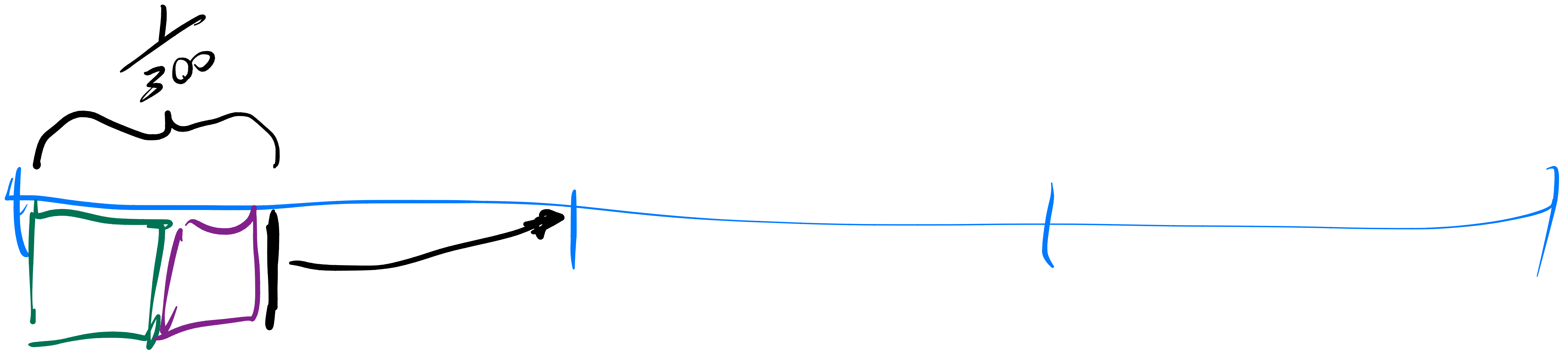
```
async function foo() {
    r = await(p)
}

    let f = foo().then(. - ~ )
```

Render seq:



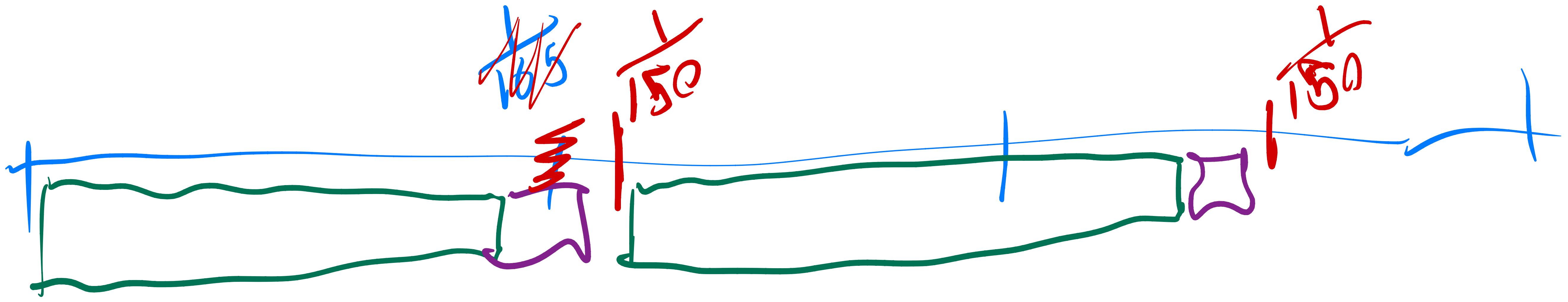
result doesn't appear till $\frac{2}{165}$'s later



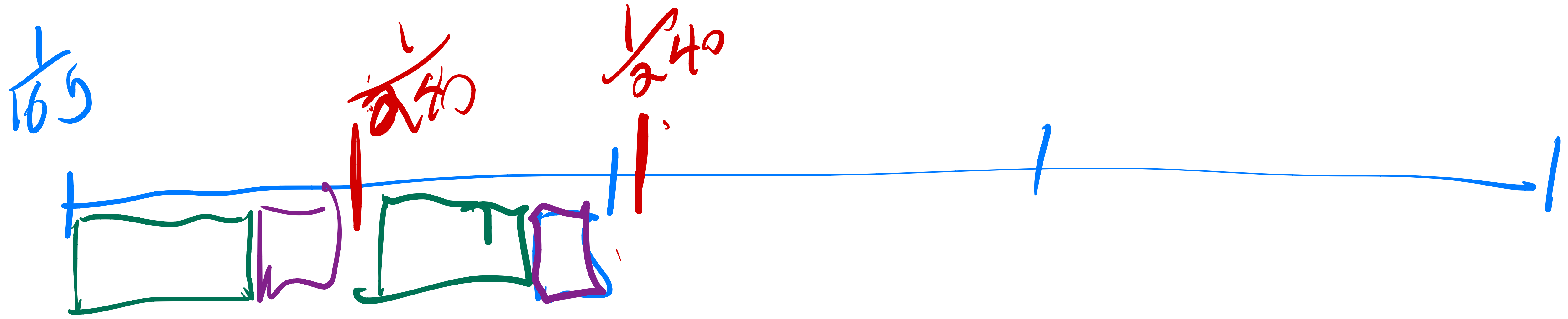
Two implications

- prediction $\rightarrow$ ahead to next frame
  (can't deal with input you don't have)

- GSYNC / freesynch
  NVIDIA AMD
  variable refresh



GSYNC/FreeSynch



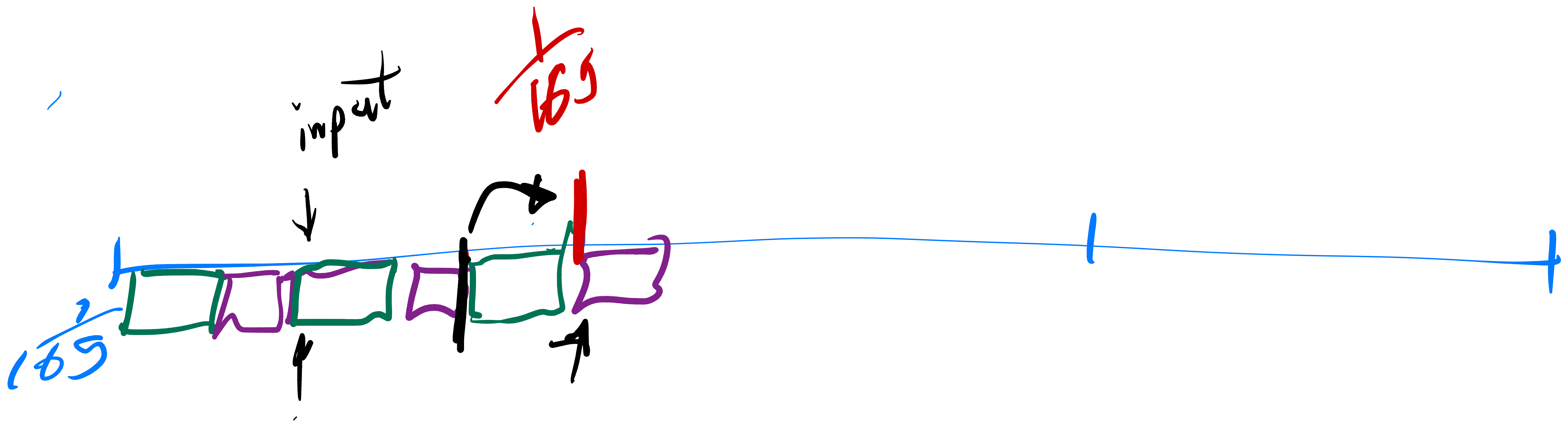
Why talking about this?

→ latency of input → display

→ graphics display "sometime after render finishes)

Not framelock ← Not in JS

OpenGL → swapBuffers()



Basic Code Structure

```
render(t) {
    update app data based
      on simulation (move things, NPCs, ...)
    physics / collision
    render
}
```

$\frac{1}{16.6}$

callbacks
- kbd, mouse
- network
  ↳ stuff per frame

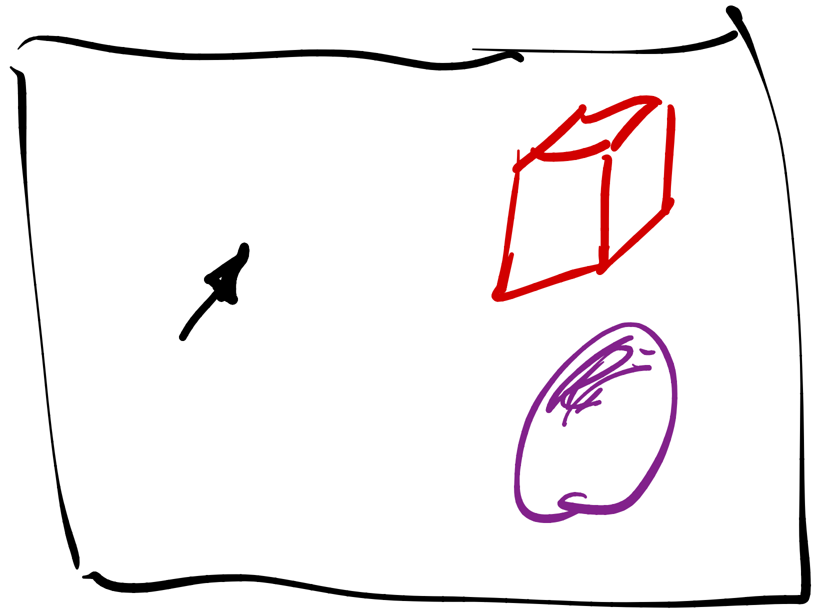
multithreding?

web → "workers"

↳ doesn't buy you much

How to Select

- Rays and Pixels: CPU vs GPU
- https://threejsfundamentals.org/threejs/lessons/threejs-picking.html



how to tell where I am clicking?

Ray-based (CPU)

→ create a Ray
    → utility on Projection camera to create ray throught
→ intersect Ray w/ all objects

Pixel-based (GPU)

→ zoom camera into 1 pixel and see what actually appears

Pixel Based

http://voxelent.com/html/beginners-guide/chapter_8/ch8_Picking.html

http://learnwebgl.brown37.net/11_advanced_rendering/selecting_objects.html

https://www.sixhat.net/webgl-3d-picking-p5js-color-buffer.html

https://bl.ocks.org/duhaime/1eafa293e7ce16b074a6d55cac67badc

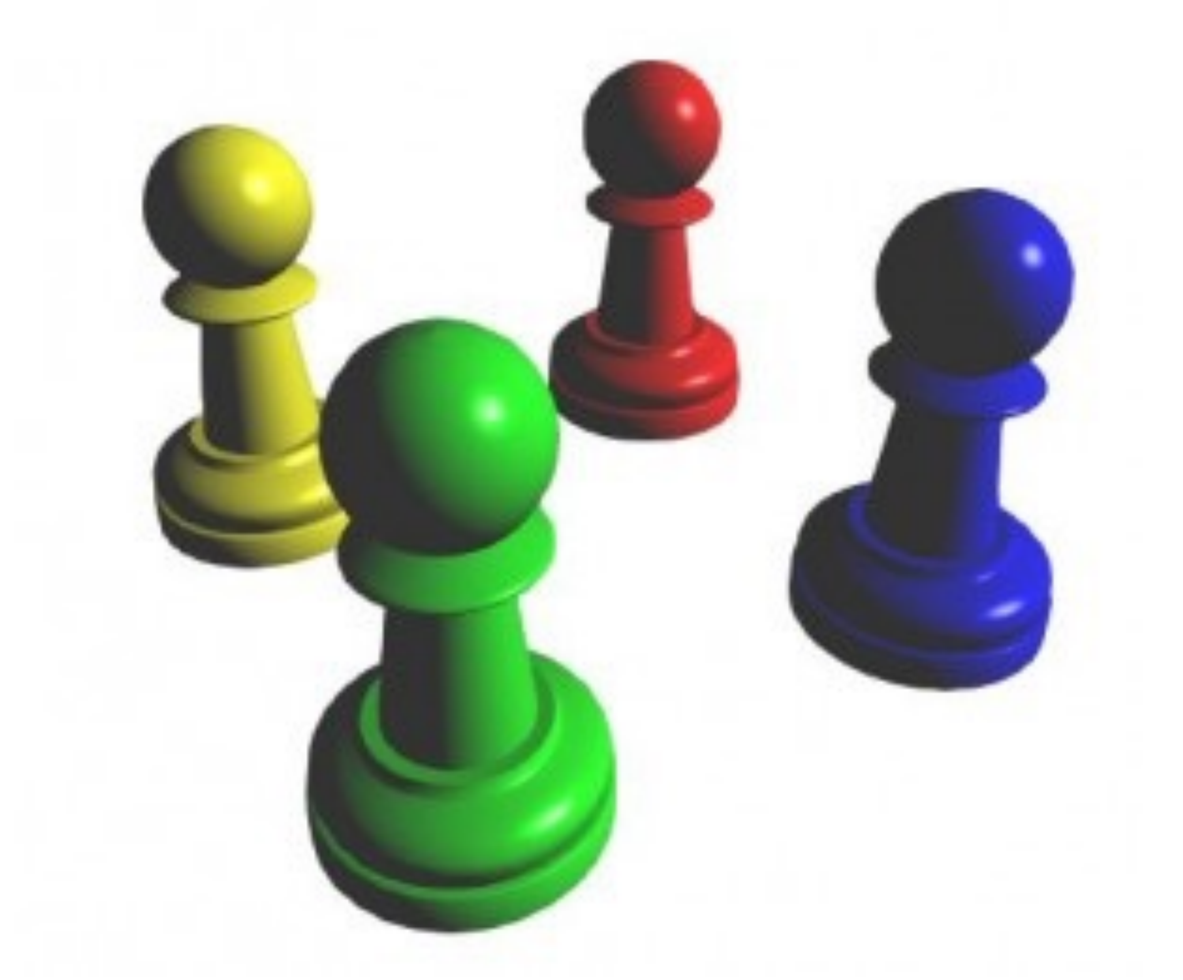
→ deals with transparency

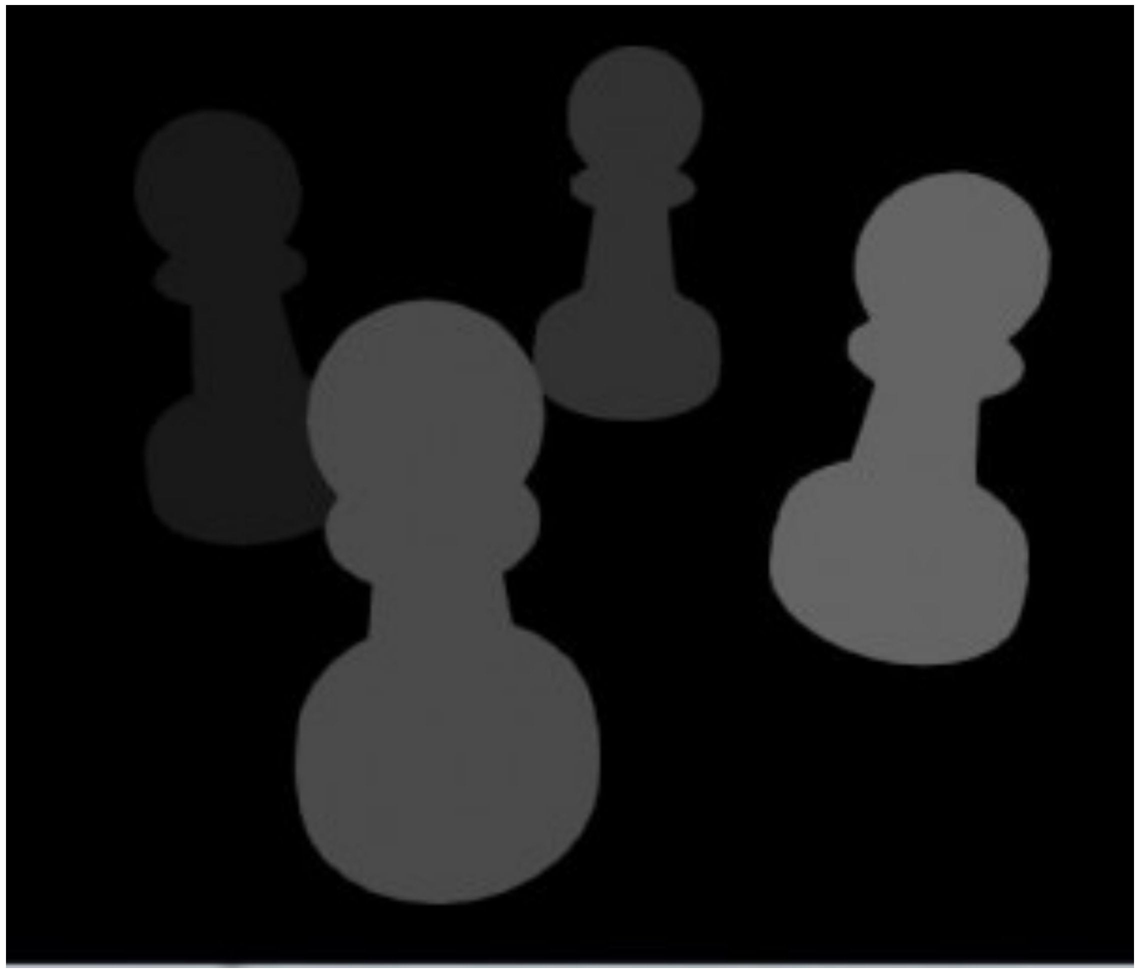
→ deals with shaders

→ vertex shaders move geom!

4 objects $\downarrow$

assign diff color to each (constant)





http://www.lighthouse3d.com/tutorials/opengl-selection-tutorial/

Pixel-based picking: three.js

```
pickingTexture = new THREE.WebGLRenderTarget(w, h);
canvas.addEventListener('mousemove', function(e) {
        renderer.render(pickingScene, camera, pickingTexture);
        var pixelBuffer = new Uint8Array(4);
        renderer.readRenderTargetPixels( pickingTexture, e.clientX,
                pickingTexture.height - e.clientY, 1, 1, pixelBuffer );
        var id = (pixelBuffer[0]<<16)|(pixelBuffer[1]<<8)|(pixelBuffer[2]);
}
// better:  make target 1,1 and use setViewOffset
```

aside: synch's GPU & CPU

Raycasting: three.js

-1...1 x,y

perspective

```
raycaster = new THREE.Raycaster();
raycaster.setFromCamera(~~nor~~malizedScreenPosition, camera);
intersectedObjects = raycaster.intersectObjects(scene.children);
```

↳ array of hit objects
↳ object
face
u/v within face
3D position

Basic Code Structure

State Machines

e.g., https://github.com/eonarheim/TypeState