CS375 - Sean Sponsler

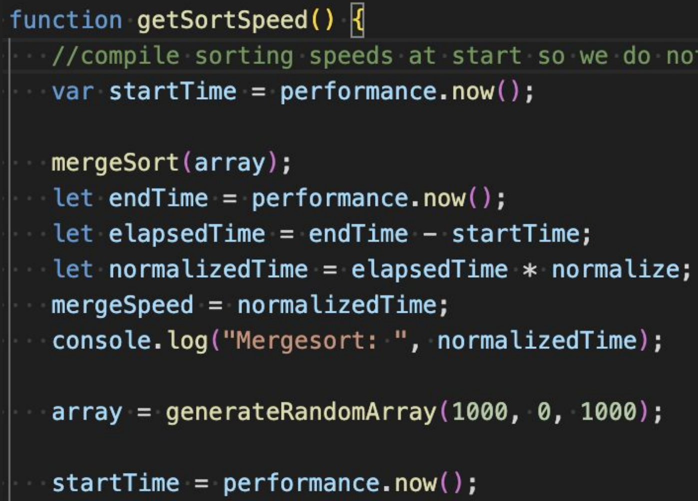
https://github.com/ssponsler/CS375/tree/main/Final-Project

Original Proposal / Modification

- Original: Algorithm speed visualization using a simple animation
- Modified: Algorithm speed “visualization” using a cartoon plane GLB object with ThreeJS

What was achieved against the original design

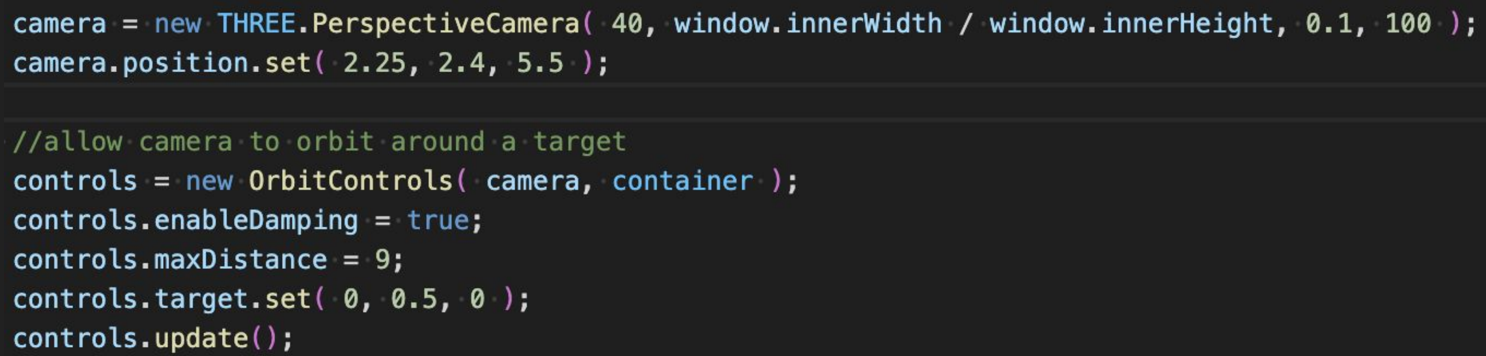
- Generalization of the unstable nature of sorting algorithms against relatively small input size
- Created functions to time the computation required for different sorting algorithms using a randomly generated array of n=100, 1000, 10000
    - Found that the timing values vary widely in this specific environment and certain coincidences seem common
    - Sometimes selection sort (O(n^2)) would perform better than Mergesort (O(nlogn)) for a random set of data with n=10000, etc



```
function getSortSpeed() {
    //compile sorting speeds at start so we do no
    var startTime = performance.now();

    mergeSort(array);
    let endTime = performance.now();
    let elapsedTime = endTime - startTime;
    let normalizedTime = elapsedTime * normalize;
    mergeSpeed = normalizedTime;
    console.log("Mergesort: ", normalizedTime);

    array = generateRandomArray(1000, 0, 1000);

    startTime = performance.now();
```

What I learned

- ThreeJS
  - It is extremely simple to implement orbital camera controls and a perspective camera in comparison to WebGL.
  - Far easier to code with little knowledge of JavaScript/HTML/CSS in comparison to WebGL
  - Works very well with…
- GTLF / GLB objects
  - Most online 3D models for free use are available to be exported into the GLB / GTLF format which works very well with ThreeJS as well as containing certain metadata about what geometric dimensions determine stated objects as well as entire animation cycles which can be loaded on demand. Can be easily imported into Blender to learn about how the object is created

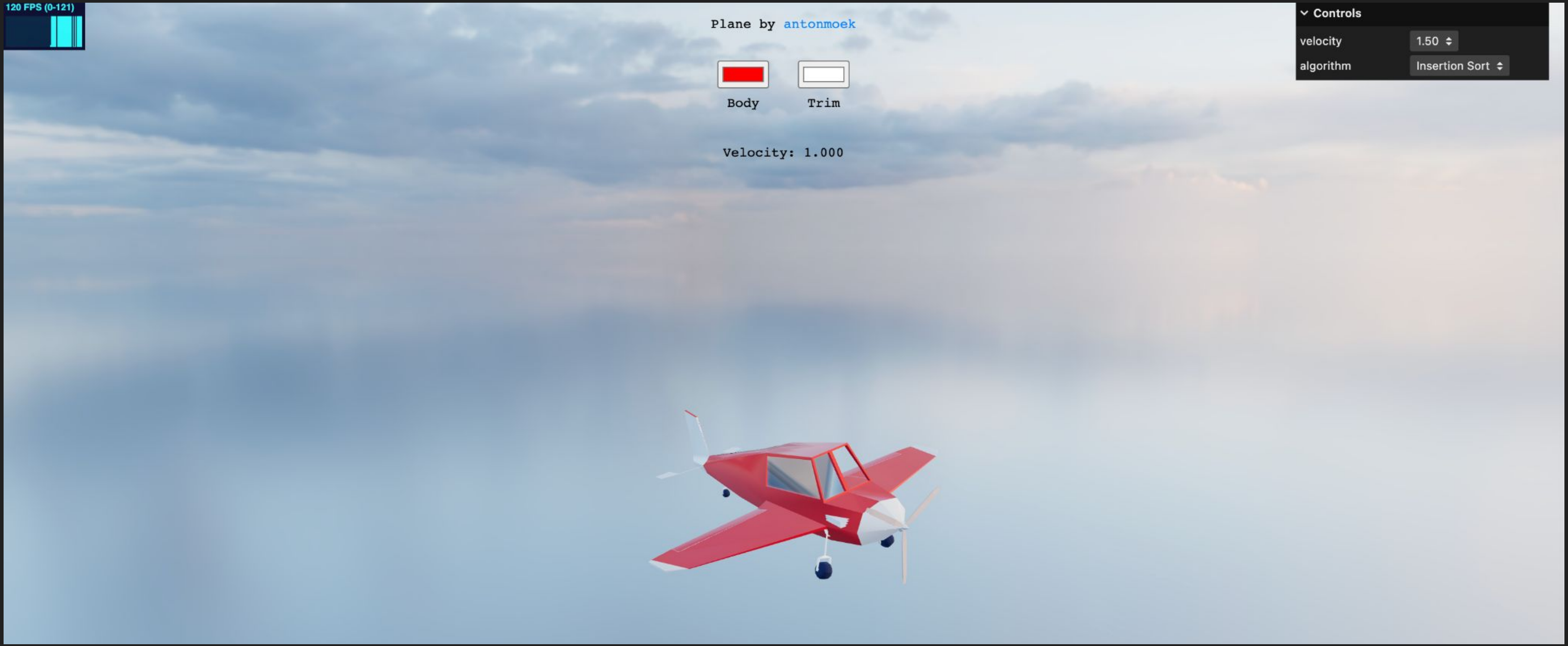


```
camera = new THREE.PerspectiveCamera( 40, window.innerWidth / window.innerHeight, 0.1, 100 );
camera.position.set( 2.25, 2.4, 5.5 );

//allow camera to orbit around a target
controls = new OrbitControls( camera, container );
controls.enableDamping = true;
controls.maxDistance = 9;
controls.target.set( 0, 0.5, 0 );
controls.update();
```

What I learned

- Backgrounds in ThreeJS
  - Can be CubeMaps, sphere meshes, etc
  - Crap moment: realized most of the background textures I wanted to import were heavily watermarked or compressed to the point where I felt stuck in trying to represent an atmosphere outside of the ThreeJS default background.
  - Equirectangular background textures via https://polyhaven.com/hdris are very simple to implement into ThreeJS but does not demonstrate a moving object velocity very well.

Final version


120 FPS (0-121)

Plane by antonmoek

Body

Trim

Velocity: 1.000

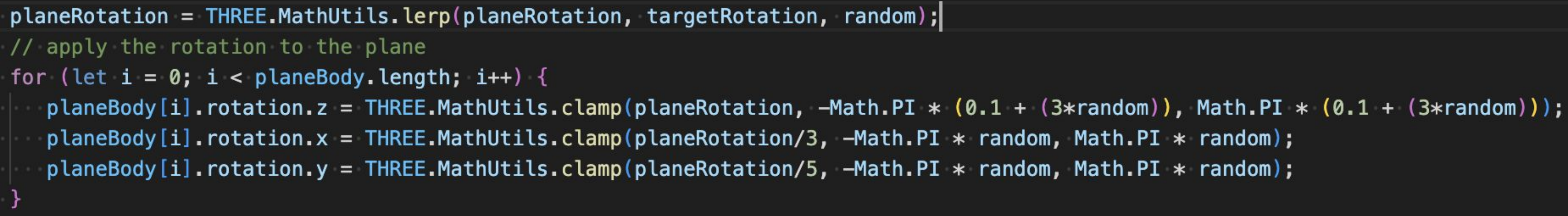
Controls

velocity 1.50

algorithm Insertion Sort

What was missing

- CubeMap or other background implementation, pannable to accurately show plane velocity outside of “simulated” turbulence animation
- Plane unfortunately broke its window miles back in the page.
- Turbulence “noise”, currently resembles the same animation repeating, modified by the speed of the program and a random linear interpolation value for smoothing.
- Central light/sun to demonstrate shadows
    - Fortunately not extremely relevant with the background texture chosen and scenario



```
planeRotation = THREE.MathUtils.lerp(planeRotation, targetRotation, random);
// apply the rotation to the plane
for (let i = 0; i < planeBody.length; i++) {
    planeBody[i].rotation.z = THREE.MathUtils.clamp(planeRotation, -Math.PI * (0.1 + (3*random)), Math.PI * (0.1 + (3*random)));
    planeBody[i].rotation.x = THREE.MathUtils.clamp(planeRotation/3, -Math.PI * random, Math.PI * random);
    planeBody[i].rotation.y = THREE.MathUtils.clamp(planeRotation/5, -Math.PI * random, Math.PI * random);
}
```

References / Examples used

- https://threejs.org/examples/?q=car#webgl_materials_car
  - Color-picker interface, relating to GLB models as well as CSS reference
- https://observablehq.com/@sdl60660/threejs-sphere-background-set-as-scene-background
  - Equirectangular background implementation
- https://lil-gui.georgealways.com/
  - ThreeJS simple GUI interface