Three Dimensional Schedule Visualizer

Project Team Member Jacob Yoon Zeng Yew
Jacob.Yoon001@umb.edu
University of Massachusetts Boston

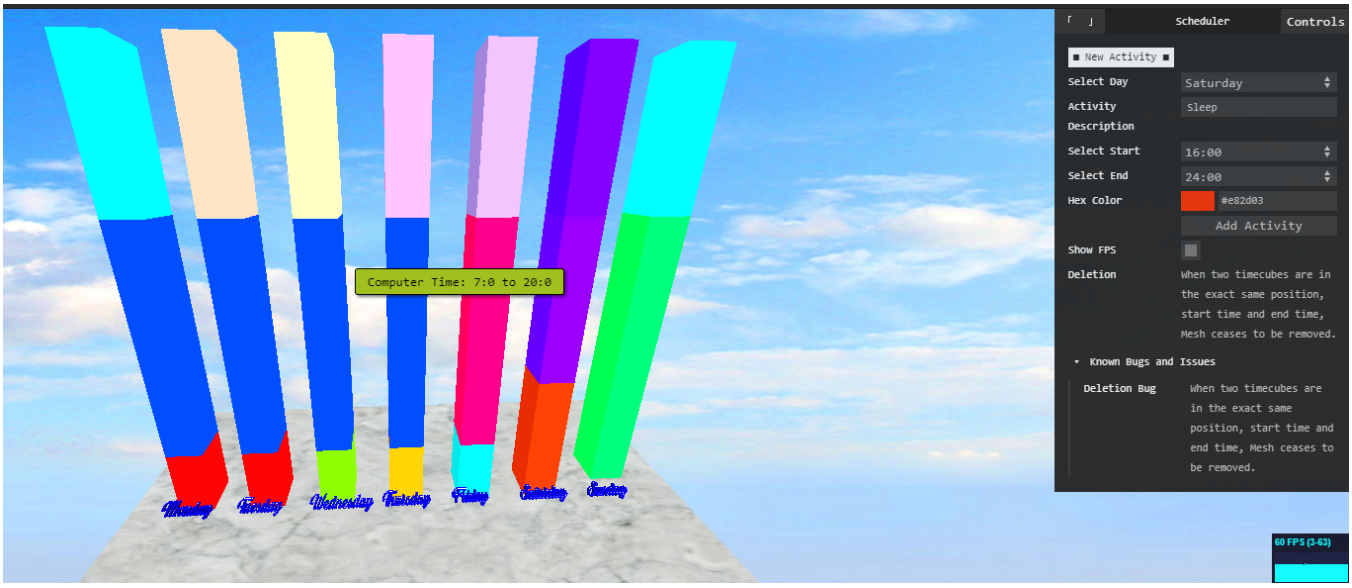

Figure 1: Example usage.

ABSTRACT

Visualization is a powerful illustration tool. Third Dimensional visualization of an individual's schedule provides scale, readability and overall comprehension of the time spent on any activity.

KEYWORDS

WebGL, Visualization, Tool, Self-help, Time, Illustration, 3D, Three.js, Schedule, Interactive

ACM Reference Format:
Project Team Member Jacob Yoon Zeng Yew. 2019. Three Dimensional Schedule Visualizer. In *CS460: Computer Graphics at UMass Boston, Fall 2019.* Boston, MA, USA, 4 pages. https://CS460.org

1 INTRODUCTION

Humans are inherently visual creatures; when something is large next to something tiny, we intuitively understand the scale of it.

When it comes to time management, traditional scheduling tools are often spreadsheets, written on paper or in timetable form.

CS460, Fall 2019, Boston, MA

ACM ISBN 1337.
https://CS460.org

While these methods may be effective for some, visual learners may be quick to disregard or bore easily.

While this project is not intended to replace these long-established practices, it seeks to provide an additional dimension to the visualization aspect, an alternative to conventional time organizational methods.

By utilizing WebGL, more specifically on the Three.js framework, this project is realized.

2 RELATED WORK

This project employs Three.js [1] as its main framework complimented by Guify [2], a graphical user interface for the schedulers main controls. Facetype [3] was used to convert fonts to a JavaScript.js file. Used a dated version of Stats.js by Three.js[1] provided by GitHub user stemkoski[4].

3 METHOD

By using Three.js, a floor plane was created and a sky-box implemented to introduce life to the visualizing space. Cuboids are used in place of total minutes in a day.

In order to add new timecuboids, a graphical user interface or GUI, was required for seamless user interfacing. Guify proved to be easy to use, customize visually and functionally. Users will have to use the drop down menu to select the Day they wish to add the activity, include a Activity description, select the start and end of said activity, customize the color of the timecuboid before

adding the activity by clicking on "Add Activity". To test for performance, a Frames per second monitor was added to the bottom right corner. The FPS counter can also be hidden through the GUI. A list of Known Bugs and Issues are also available through the GUI. Three.js's TextGeometry was placed on the now textured marble floor plane to provide visual indicate days.

3.1 Implementation

Two main files were used in the creation of this project: index.html and activity.js. index.html is where the scripts, both HTML and JavaScript code are implemented. activity.js is where the JavaScript object, Activity is located. The Activity Object requires seven arguments namely; DayName, DayPosition, ActivityDescription, Duration, StartAct, End$_Act$, $Color$.

```
Activity = function(DayName, DayPosition,
Activity_Description, Duration,
Start_Act, End_Act, Color)
```

Initially called Dayactivity, the Activity object required enough user generated arguments to create the appropriate size (Duration), customisation (Color), position (DayPosition) and user tooltips (Activitydescription, StartAct EndAct)

Within index.html, relevant scripts were sourced, variables declared, html <divs> created and gui implemented.

```
window.onload = function()
```

was used to initialize code and files necessary for this project. Here is where Three.js was used to create a

```
scene = new THREE.Scene();
camera = new THREE.PerspectiveCamera(fov, ratio, zNear, zFar);
```

Where fov,ratio, zNear and zFar are 80, window.innerWidth / window.innerHeight, 1,10000 respectively. with the camera set

```
camera.position.set( 0, 1000, 500);
```

and THREE.WebGLRenderer set, light was still required to shed light on the scene. This is where THREE.AmbientLight() and THREE.DirectionalLight was added.

THREE.js's TextureLoader loaded the marble floor and a THREE.PlaneBufferGeometry was adjusted to the appropriate size, then the THREE.MeshBasicMaterial. The floor is then created as a THREE.Mesh and then added to the scene.

THREE.OrbitControls were opted for this project for its usability and a consistent structure.

The GUI then needed to parse this information to the creation of a new Activity object.

Within activity.js was the creation of the timecuboids.

```
var cubeGeo = new THREE.BoxBufferGeometry(120, this.cubeSize, 120);
var cubeMat = new THREE.MeshPhongMaterial({color: '#FF0000'});
var mesh = new THREE.Mesh(cubeGeo, cubeMat);
```

where this.cubeSize, is the Duration of the activity, its measured in minutes. This comes to the max height of 1440,

```
24 hours in a day * 60 Minutes in an Hour
= 1440 Minutes
```

This would be needed to display relevant information in addition to setting the timecuboid's offset[Challenge 3].

Color set, and ActivityDescription was tied to the mesh as

```
mesh.userData.tooltipText = Activity_Description + ": "
  + timeConvert(Start_Act)
  + " to "
  + timeConvert(End_Act);
```

The function timeConvert(n) will be covered in Milestone 4 and challenge 4.

3.2 Milestones

The project's development was spread across 3 weeks.

3.2.1 Milestone 1. The framework, Activity, timecuboid and the GUI were conceptualized and written in pen and paper.

3.2.2 Milestone 2. The background and floor implemented, camera working and GUI framework added.

3.2.3 Milestone 3. timecuboids were added but not in the right positions [See Challenge 2]

3.2.4 Milestone 4. GUI was tied to the creation of Activity.

3.2.5 Milestone 5. Adding 3D text to the floor.

3.2.6 Milestone 6. Making sure tooltips displayed relevant information when user mouse hovered above it.

3.2.7 Milestone 7. Ensured edge cases such as intersections of timecuboids [see Challenge 1] cease.

3.3 Challenges

There were several challenges that proved difficult and often felt impossible until it was done. Detailed here are some of the challenges faced as a novice programmer. In no particular order, chronological or otherwise.

- Challenge 1: Time cuboids intersection Arguably biggest challenge I faced was ensuring user input does not intersect one another. As it appears there are several edge cases to consider, such as (but not limited to): if the bottom of a timecuboid intersecting with the beginning of a timecuboid, vise versa, a timecuboid within a timecuboid, a timecuboid on top of an existing timecuboid.
  A considerable amount of time was burned to ensure intersections were considered [see Figure 2].
  But alas, if the cubes were the same sizes, an error is not thrown at the user. After four days of contemplation, the project must continue.
  Errors have been added to be thrown at the User when one attempts to make an intersections between these cubes.
- Challenge 2: Positioning cuboids
  While there is a list to select the Days for the Activities, there isn't any indication on where they will be positioned.
  The quick and dirty method was to create two array, one for an array of name of days and the other the positions array. By using the built in indexOf() of the selected day on the list, then matching the index of the day positions array. The 3D text geometry also utilizes this array of daypositions.
- Challenge 3: Offset As it appears, the time cuboids required re-positioning as they seemed to be clipping through the floorplane. After some few days of burning the midnight

```
if (((Activity_Array[days_name.indexOf(
    day_Selection)][j].start_time <
    new_activity.start_time) && (new_activity.
    start_time < Activity_Array[days_name.
    indexOf(day_Selection)][j].end_time))
    ||
    ((Activity_Array[days_name.indexOf(
        day_Selection)][j].start_time <
        new_activity.end_time) && (
        new_activity.end_time <
        Activity_Array[days_name.indexOf(
        day_Selection)][j].end_time))
    ||
    ((Activity_Array[days_name.indexOf(
        day_Selection)][j].start_time >=
        new_activity.start_time) && (
        new_activity.end_time >
        Activity_Array[days_name.indexOf(
        day_Selection)][j].end_time))
    ||
    ((Activity_Array[days_name.indexOf(
        day_Selection)][j].start_time >
        new_activity.start_time) && (
        new_activity.end_time >=
        Activity_Array[days_name.indexOf(
        day_Selection)][j].end_time))
    ) {
```

Figure 2: If loops and logic to prevent intersections

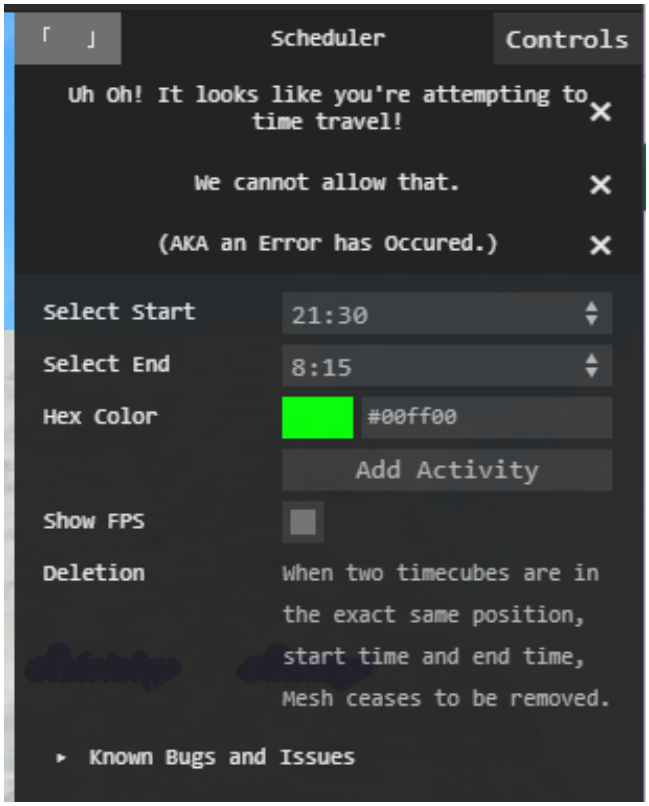

Figure 3: Error thrown at User if timecuboids intersect

oil, it turns out the origin of a BoxBufferGeometry is at the center. The offset then was simply the size of any particular cube divided by 2.

- Challenge 4: Time Conversion Needing to have the drop down list human readable, an array of time from 0:00 to 24:00 was needed. Fortunately this can be done in 7 lines of JavaScript

```
var duration = [], i, j;
for(i=0; i<24; i++) {
  for(j=0; j<4; j++) {
duration.push(i + ":" + (j===0 ? "00" : 15*j) );
  };
};

duration.push(24 + ":" + "00");
```

24:00 was manually added as the code itself starts at 0:00 but stops at 23:45.

- Challenge 5: fonts typefaces Three.js' fontloader turned out to be much more difficult to implement than originally thought. By way of gero3's Facetype.js converter, conversion of a font format to that of .js was only three clicks away.
- Challenge 6: Other Edge Cases
  There also exists another edge case possibility, What if Users decide to (or inadvertently) select the wrong time? That is when Select Start is the same as Select End, or when Select End Is greater than Select Start.
  In figure 3, displays the error thrown.

4 RESULTS

After some hardships, the prototype seen here in image 4, bore fruit. The result is an alternative to current methods.

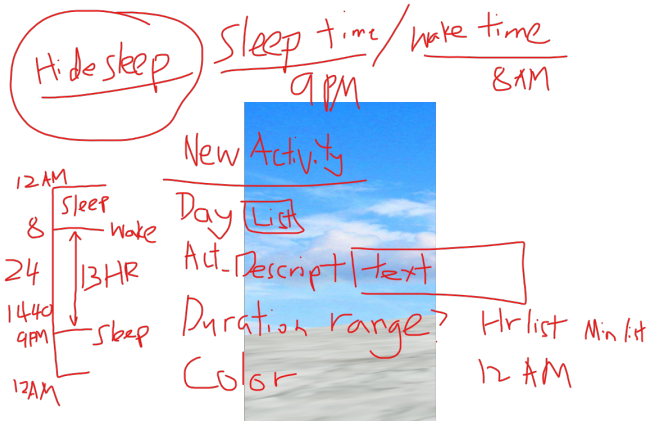

Figure 4: Protoype GUI and timecuboid sketch

Or you could add tables (see Table 1 - maybe with some timings?).

Table 1: Performance across multiple browsers and devices

Device	Performance
Android (Samsung S9+)	60 FPS
Android (Samsung S7)	60 FPS
PC (Browser Google Chrome Version 79.0.(64-bit))	60 FPS
PC (Browser Firefox Version 71.0 (64-bit))	60 FPS
PC (Browser Microsoft Edge 44.18362.449.0)	72 FPS

5 CONCLUSIONS

While there are some bugs left to squash, this project has provided some insight and much needed hands on experience. The trials and tribulations of the challenges were strenuous but still achievable.

REFERENCES

[1] Ricardo Cabello et al. 2010. Three.js. *URL: https://github. com/mrdoob/three.js* (2010).
[2] Jonathan Cole. 2017. Guify. *URL: https://github.com/colejd/guify* (2017).
[3] gero3. 2016. Facetype.js. *URL: https://gero3.github.io/facetype.js/* (2016).
[4] stemkoski. 2013. stemkoski. *URL: https://stemkoski.github.io/Three.js/index.html* (2013).