**Progress:**

Implemented all TO-DO items in the milestone report and additional features.

**Features:**

1. Accept user specified initial values and physical constants.
2. Two predefined settings: 1. special solution: circular motion 2. Sun, Jupiter, and Earth.
3. Toggling frame counter.
4. Start and pause simulation.
5. Camera position can be changed by dragging mouse on the scene.

**Knowledge learned:**

1. Learned the basic structure of a Webgl program, including vertex shader, fragment shader, attribute, uniform, and varying. Learned preparation work before running a shader program.
2. Honed the skill of model-view-perspective matrix transformation and lighting.
3. Learned how to create animation with Webgl with requestAnimationFrame function.
4. Practiced the skill of Euler’s midpoint method and applied it to a real program.
5. Learned basic HTML UI, including input/output, buttons, and interaction with JavaScript.

**Problems encountered and solved:**

1. The scale, translate, and rotate function in glMatrix applies the transformation matrix to the right of the original matrix. Therefore, we need to write the transformations in reversed order.
2. glMatrix does not support operands on matrices and vectors. Should use built-in functions add, subtract, multiply, scaleAndAdd, mat4.multiplyVec3 instead of +, -, \*.
3. When the camera position is changed, the light position in the camera space will also change. When implementing view changing, I forgot to update the camera space light position, as a result the lighting had weird behavior. Updating the light position in camera space each frame solves this problem.
4. When implementing hiding/showing the frame counter, the frame counter will be displayed on a new line. Setting the style attribute to “display: inline-block” instead of “display: block” can make it on the same line as the last element.
5. When trying to use Sun, Jupiter and Earth as the three bodies, the real position values are too large, Webgl can’t display or shows unexpected behavior. Therefore, we need to scale the values to make the simulation displayable and maintain the same kinetic process. To achieve this, position, velocity and acceleration should be scaled by the same value. By the gravity formula, F=GmM/R^2, the acceleration is proportional to GM/R^2. If R is scaled by a factor of n while other variables staying the same, acceleration will be scaled by a factor of n^(-2). However, we want acceleration also scaled by a factor of n. Thus, we need to scale M by n^3.

For instance, if initial position is divided by 10, initial velocity should also be divided by 10 and mass should be divided by 1000.

**Third-party code used:**

1. gl-matrix-min.js from glMatrix
2. webgl-utils.js from Webgl Fundamentals
3. trackball-rotator.js
4. Sphere object from http://web.mit.edu/djwendel/www/weblogo/shapes/basic-shapes/sphere/sphere.obj