EECS 367 & ROB 320/511 Lab
Pendularm (assignment 2) code overview

Autorob Lab
Git-ting Started with Git

Michigan EECS 367 *Introduction to Autonomous Robotics* | ROB 320/511 *Robot Operating Systems*

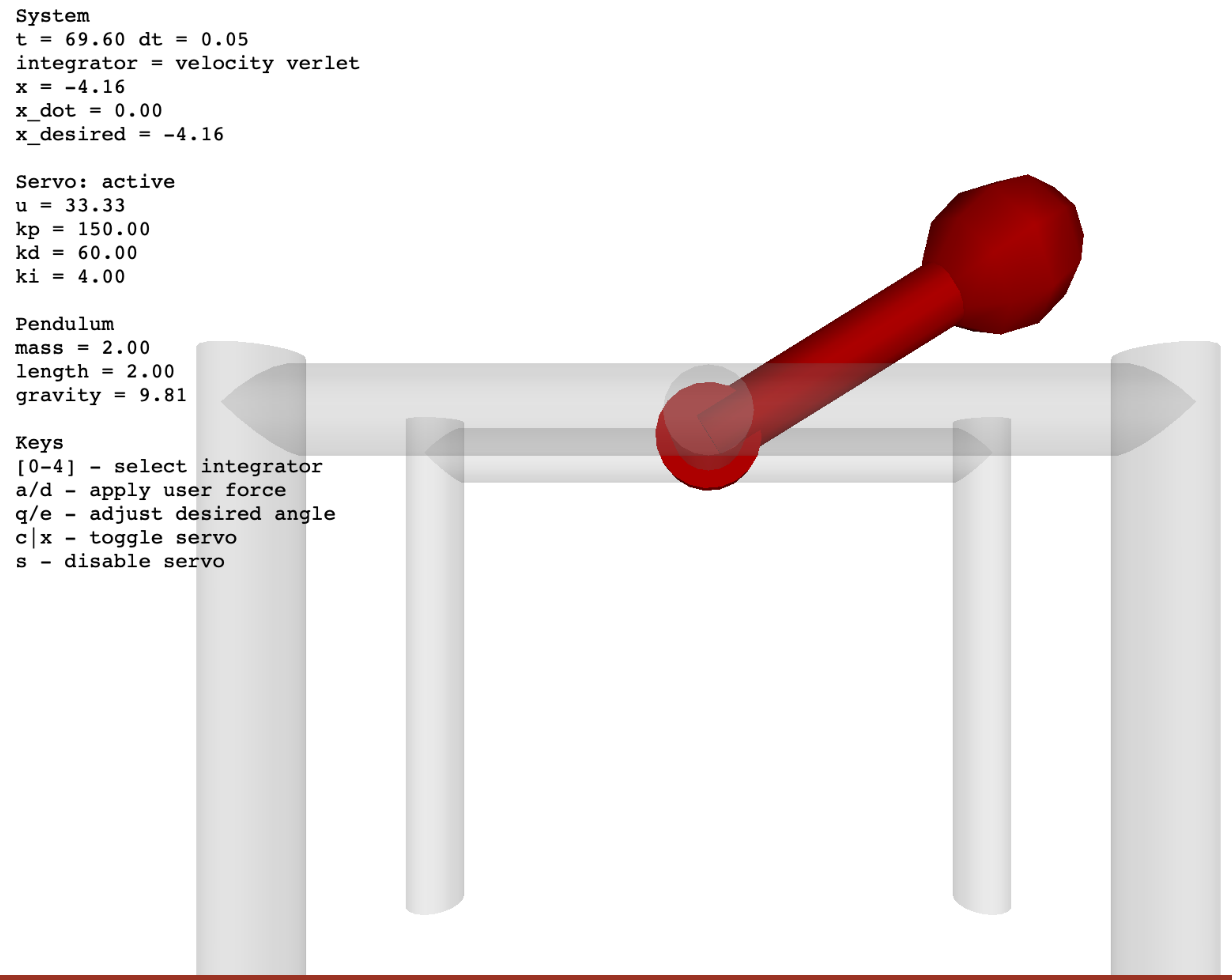
Administrative

- Assignment#0 is due
    - Grade.js: Check, Check - Partial Credit, Due, Broken
- Assignment #1 is due Monday, January 30th
- Jiayao (Vanessa) will become the CI Grader Queen

Lab Takeaways

1. Assignment overview
2. Stencil walkthrough
3. Pendularm demo
4. Coding considerations

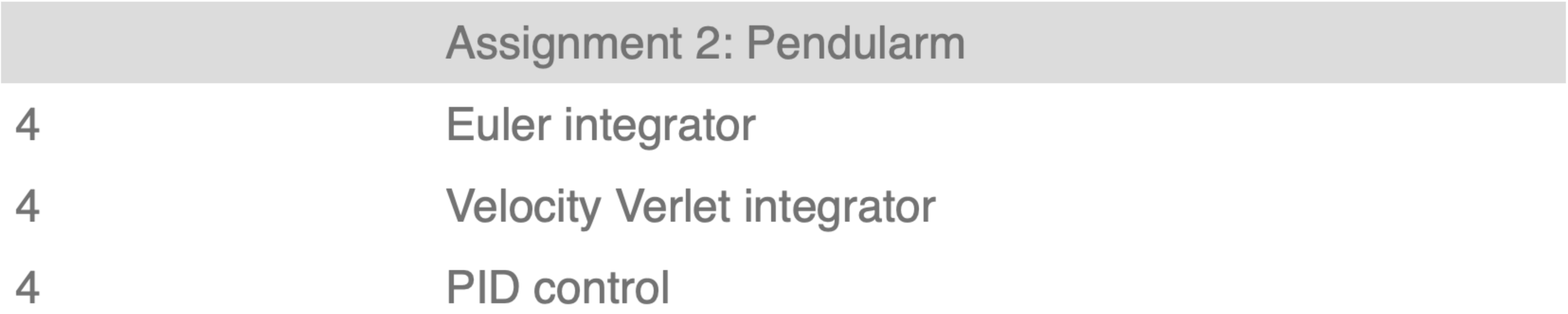
Pendularm Overview



```
System
t = 69.60 dt = 0.05
integrator = velocity verlet
x = -4.16
x_dot = 0.00
x_desired = -4.16

Servo: active
u = 33.33
kp = 150.00
kd = 60.00
ki = 4.00

Pendulum
mass = 2.00
length = 2.00
gravity = 9.81

Keys
[0-4] - select integrator
a/d - apply user force
q/e - adjust desired angle
c|x - toggle servo
s - disable servo
```

We will be implementing a servo controller for the pendularm!

Pendularm Overview

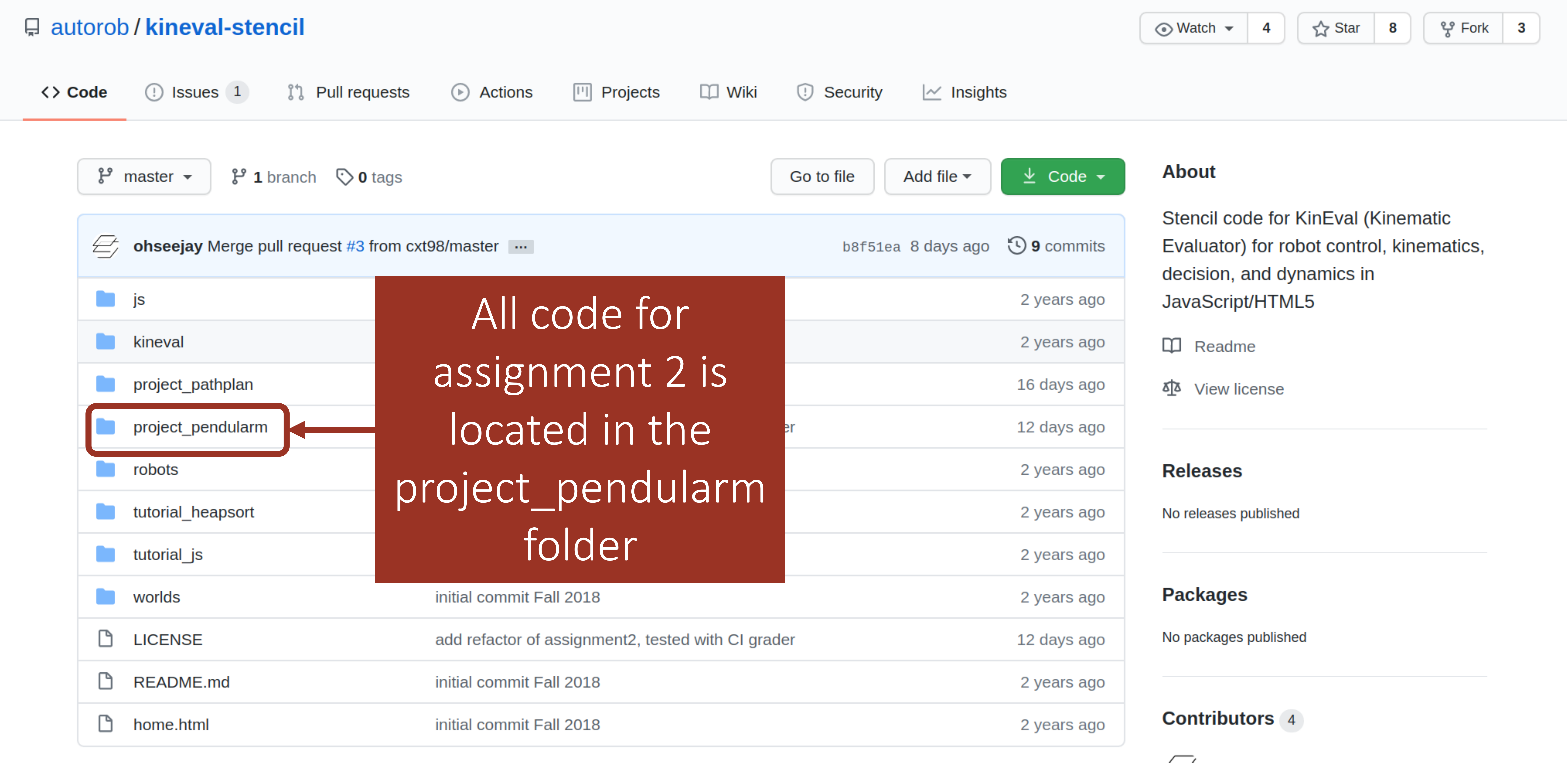


	Assignment 2: Pendularm
4	Euler integrator
4	Velocity Verlet integrator
4	PID control

Lab Takeaways

1. Assignment overview
2. Stencil walkthrough
3. Pendularm demo
4. Coding considerations

Stencil Walkthrough



autorob / kineval-stencil

ohseejay Merge pull request #3 from cxt98/master … b8f51ea 8 days ago 9 commits

js		2 years ago
kineval		2 years ago
project_pathplan		16 days ago
project_pendularm	…er	12 days ago
robots		2 years ago
tutorial_heapsort		2 years ago
tutorial_js		2 years ago
worlds	initial commit Fall 2018	2 years ago
LICENSE	add refactor of assignment2, tested with CI grader	12 days ago
README.md	initial commit Fall 2018	2 years ago
home.html	initial commit Fall 2018	2 years ago

About

Stencil code for KinEval (Kinematic Evaluator) for robot control, kinematics, decision, and dynamics in JavaScript/HTML5

Readme

View license

Releases

No releases published

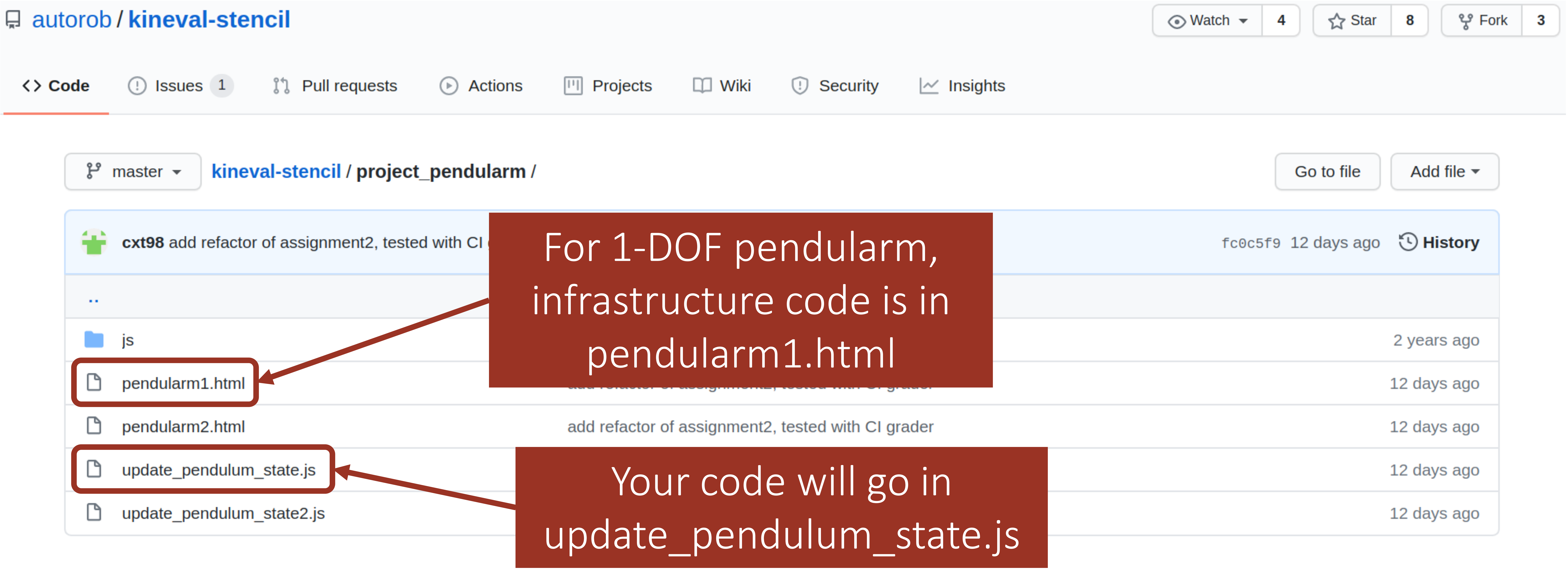
Packages

No packages published

Contributors 4

All code for assignment 2 is located in the project_pendularm folder

Stencil Walkthrough



autorob / **kineval-stencil**

kineval-stencil / **project_pendularm** /

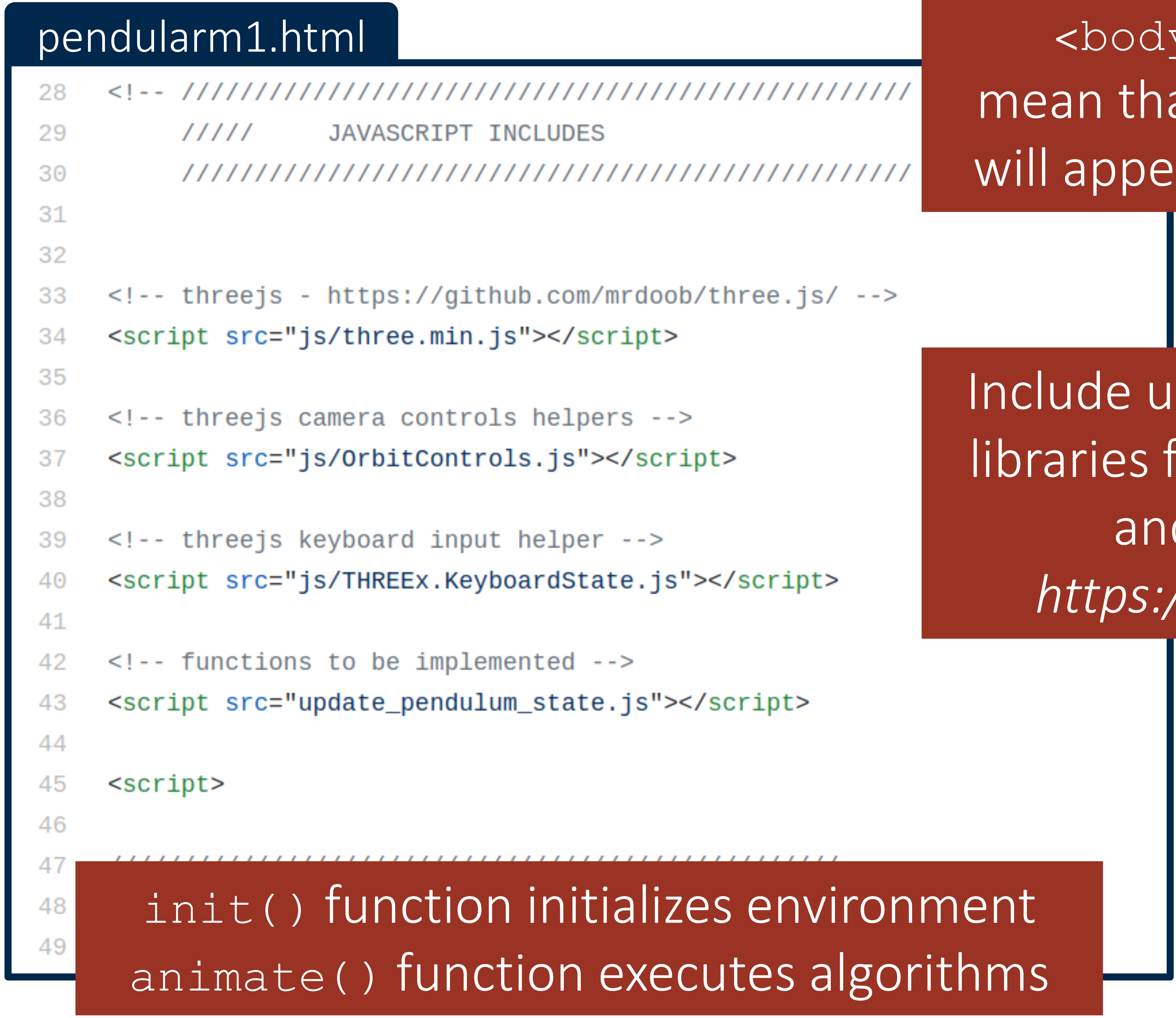
cxt98 add refactor of assignment2, tested with CI — fc0c5f9 12 days ago

..		
js		2 years ago
pendularm1.html	add refactor of assignment2, tested with CI grader	12 days ago
pendularm2.html	add refactor of assignment2, tested with CI grader	12 days ago
update_pendulum_state.js		12 days ago
update_pendulum_state2.js		12 days ago

For 1-DOF pendularm, infrastructure code is in pendularm1.html

Your code will go in update_pendulum_state.js

Stencil Walkthrough



pendularm1.html

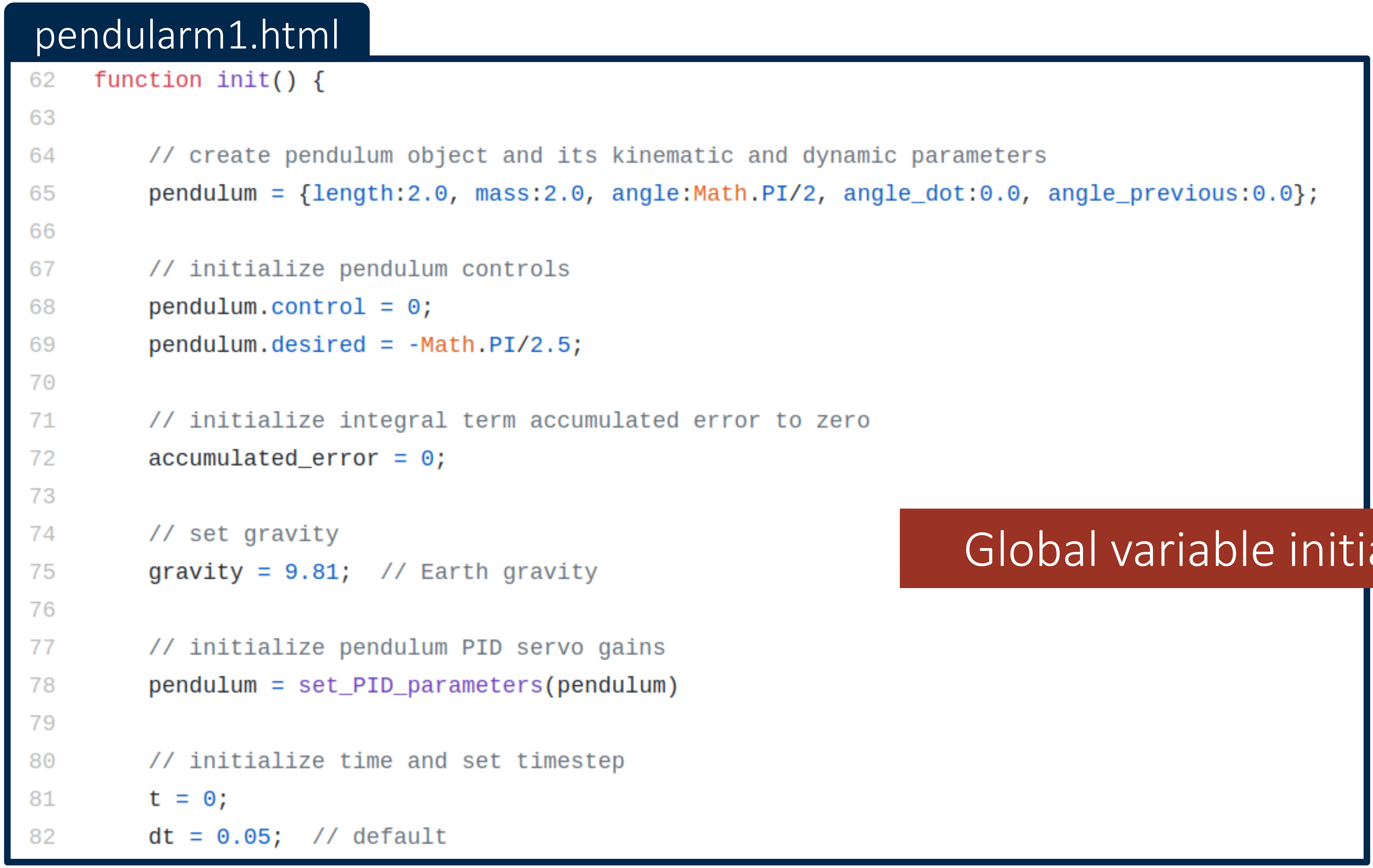
```
<!-- //////////////////////////////////////////////////
     /////     JAVASCRIPT INCLUDES
     //////////////////////////////////////////////////


<!-- threejs - https://github.com/mrdoob/three.js/ -->
<script src="js/three.min.js"></script>

<!-- threejs camera controls helpers -->
<script src="js/OrbitControls.js"></script>

<!-- threejs keyboard input helper -->
<script src="js/THREEx.KeyboardState.js"></script>

<!-- functions to be implemented -->
<script src="update_pendulum_state.js"></script>

<script>

```

`<html>` open tag
`<body>` open tag
mean that what follows
will appear on webpage

Include useful JavaScript libraries for visualization and control
https://threejs.org

`init()` function initializes environment
`animate()` function executes algorithms

Stencil Walkthrough



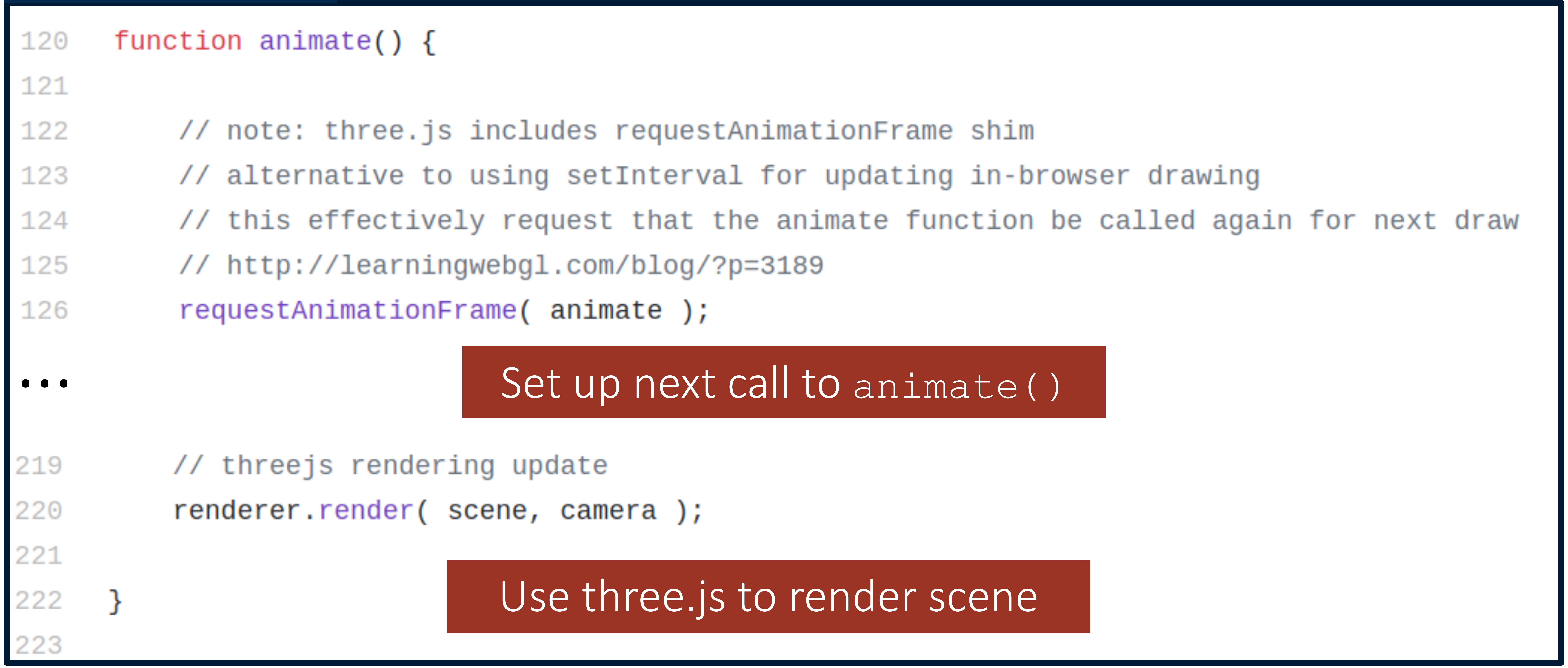
pendularm1.html

```
function init() {

    // create pendulum object and its kinematic and dynamic parameters
    pendulum = {length:2.0, mass:2.0, angle:Math.PI/2, angle_dot:0.0, angle_previous:0.0};

    // initialize pendulum controls
    pendulum.control = 0;
    pendulum.desired = -Math.PI/2.5;

    // initialize integral term accumulated error to zero
    accumulated_error = 0;

    // set gravity
    gravity = 9.81;  // Earth gravity

    // initialize pendulum PID servo gains
    pendulum = set_PID_parameters(pendulum)

    // initialize time and set timestep
    t = 0;
    dt = 0.05;  // default
```

Global variable initialization

Stencil Walkthrough

pendularm1.html



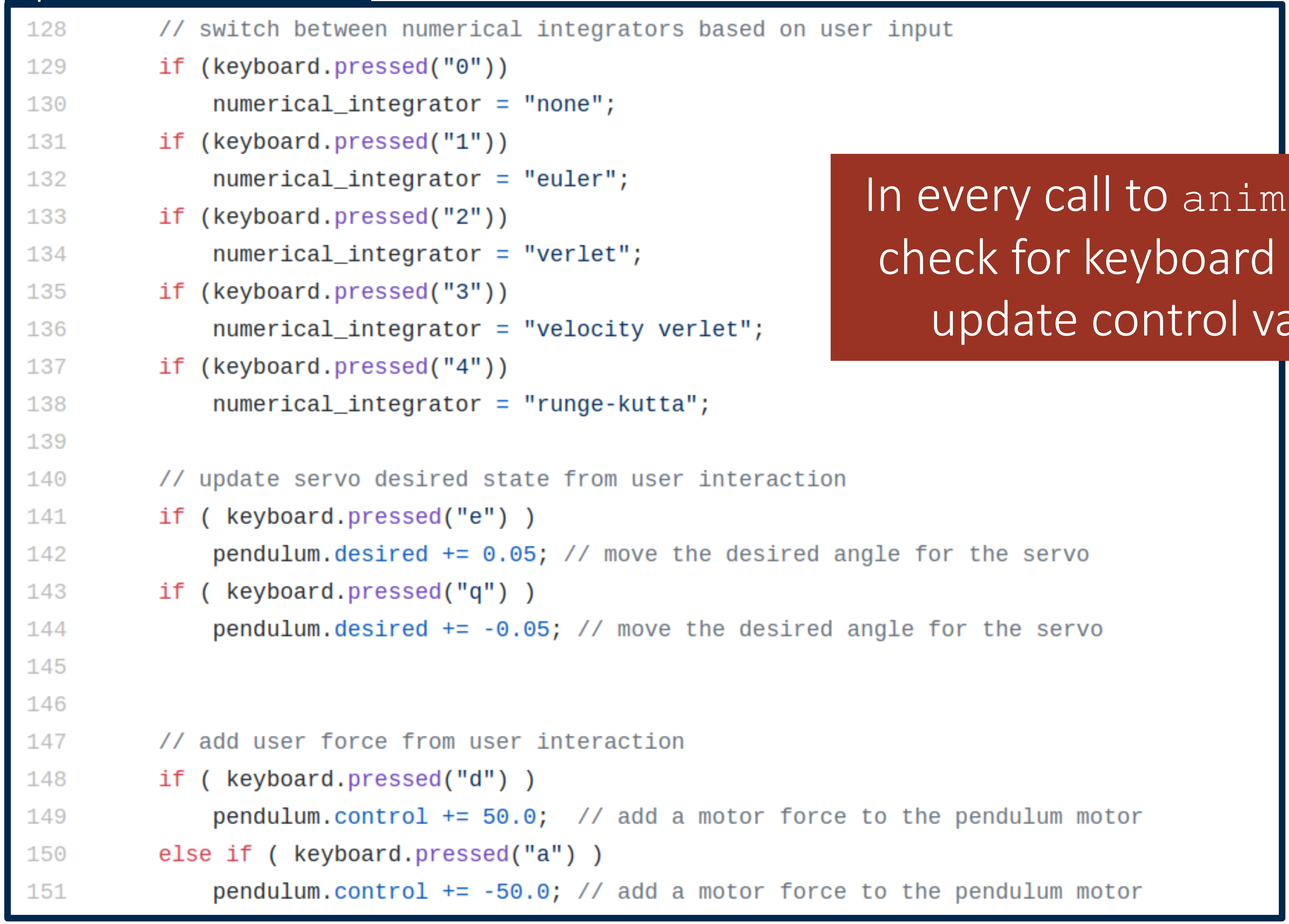
```
function animate() {

    // note: three.js includes requestAnimationFrame shim
    // alternative to using setInterval for updating in-browser drawing
    // this effectively request that the animate function be called again for next draw
    // http://learningwebgl.com/blog/?p=3189
    requestAnimationFrame( animate );
...
    // threejs rendering update
    renderer.render( scene, camera );

}
```

Set up next call to `animate()`

Use three.js to render scene

Stencil Walkthrough

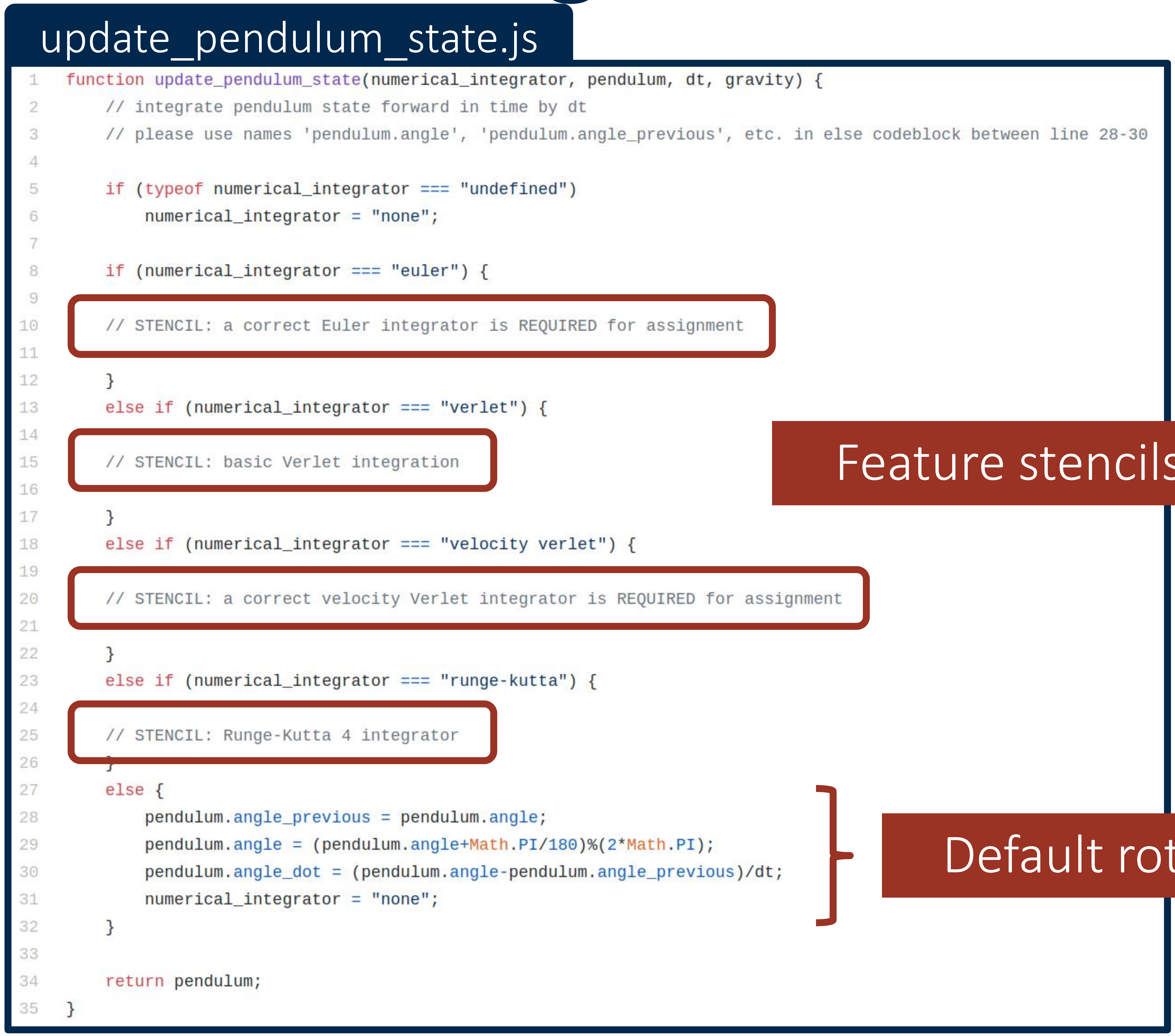
pendularm1.html



```
    // switch between numerical integrators based on user input
    if (keyboard.pressed("0"))
        numerical_integrator = "none";
    if (keyboard.pressed("1"))
        numerical_integrator = "euler";
    if (keyboard.pressed("2"))
        numerical_integrator = "verlet";
    if (keyboard.pressed("3"))
        numerical_integrator = "velocity verlet";
    if (keyboard.pressed("4"))
        numerical_integrator = "runge-kutta";

    // update servo desired state from user interaction
    if ( keyboard.pressed("e") )
        pendulum.desired += 0.05; // move the desired angle for the servo
    if ( keyboard.pressed("q") )
        pendulum.desired += -0.05; // move the desired angle for the servo


    // add user force from user interaction
    if ( keyboard.pressed("d") )
        pendulum.control += 50.0;  // add a motor force to the pendulum motor
    else if ( keyboard.pressed("a") )
        pendulum.control += -50.0; // add a motor force to the pendulum motor
```

In every call to `animate()`, we check for keyboard input and update control variables

Stencil Walkthrough



update_pendulum_state.js

```
1  function update_pendulum_state(numerical_integrator, pendulum, dt, gravity) {
2      // integrate pendulum state forward in time by dt
3      // please use names 'pendulum.angle', 'pendulum.angle_previous', etc. in else codeblock between line 28-30
4
5      if (typeof numerical_integrator === "undefined")
6          numerical_integrator = "none";
7
8      if (numerical_integrator === "euler") {
9
10     // STENCIL: a correct Euler integrator is REQUIRED for assignment
11
12     }
13     else if (numerical_integrator === "verlet") {
14
15     // STENCIL: basic Verlet integration
16
17     }
18     else if (numerical_integrator === "velocity verlet") {
19
20     // STENCIL: a correct velocity Verlet integrator is REQUIRED for assignment
21
22     }
23     else if (numerical_integrator === "runge-kutta") {
24
25     // STENCIL: Runge-Kutta 4 integrator
26     }
27     else {
28         pendulum.angle_previous = pendulum.angle;
29         pendulum.angle = (pendulum.angle+Math.PI/180)%(2*Math.PI);
30         pendulum.angle_dot = (pendulum.angle-pendulum.angle_previous)/dt;
31         numerical_integrator = "none";
32     }
33
34     return pendulum;
35 }
```

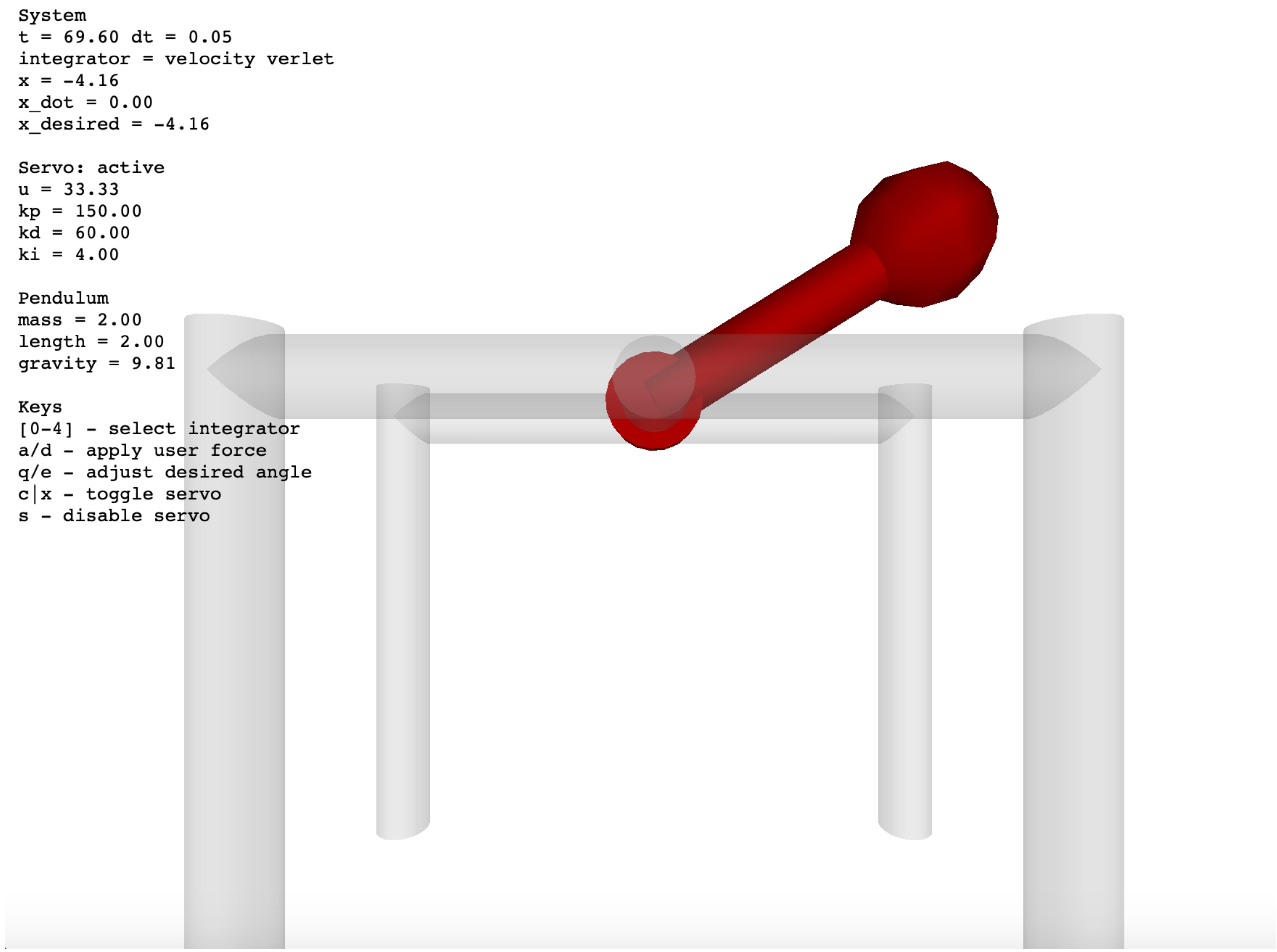
Feature stencils

Default rotation

Lab Takeaways

1. Assignment overview
2. Stencil walkthrough
3. Pendularm demo
4. Coding considerations

Pendularm Demo



```
System
t = 69.60 dt = 0.05
integrator = velocity verlet
x = -4.16
x_dot = 0.00
x_desired = -4.16

Servo: active
u = 33.33
kp = 150.00
kd = 60.00
ki = 4.00

Pendulum
mass = 2.00
length = 2.00
gravity = 9.81

Keys
[0-4] - select integrator
a/d - apply user force
q/e - adjust desired angle
c|x - toggle servo
s - disable servo
```

Lab Takeaways

1. Assignment overview
2. Stencil walkthrough
3. Pendularm demo
4. Coding considerations

Coding Considerations

These concepts are optional, meant to help you on programming assignments

Concepts to consider for writing readable, easily debug-able code:

1. Use comments where complicated
2. Add whitespace for readability
3. Local variables to store indices/raw data
4. Helper functions that reduce code duplication

Using Comments

WITHOUT COMMENTS

```
var x = data;
var y = -1;
for (i=0; i<x.length; ++i){
   if (y<x[i]){
      y = x[i];
   }
}
```

WITH COMMENTS

```
// initialize data and min value so far
var x = data;
var y = -1;
// iterate over items in array x
for (i=0; i<x.length; ++i){
   // if current item in array is less than
   // min value so far
   if (y<x[i]){
      // update min value
      y = x[i];
   }
}
```

Using Whitespace

WITHOUT WHITESPACE

```
for (i=0; i<x.length; ++i){
   for (j=0; j<x[i].length; ++j){
      y = doStuff(i,j, x);
      doMoreStuff(y);
   }
}
```

WITH WHITESPACE AND COMMENTS

```
// iterate over every element in array x
for (i=0; i<x.length; ++i){
   for (j=0; j<x[i].length; ++j){

      // perform computation with current
      // position in x
      y = doStuff(i,j, x);

      // use result to do more stuff
      doMoreStuff(y);
   }
}
```

Local Variables for Temp Storage

COMPLICATED INDEX

Input: `G, node`

```
// index offset of neighbor
var offset = [0, 1];

// index into G at neighbor
G[node.i+offset[0]][node.j+offset[1]]
```

READABLE INDEX

Input: `G, node`

```
// index offset of neighbor
var offset = [0, 1];

// calculate indices and store in local var
var nbr_i = node.i+offset[0];
var nbr_j = node.j+offset[1];

// index into G at neighbor
G[nbr_i][nbr_j]
```

Helper Functions

DUPLICATED CODE

Input: `G, node`

```
//index into neighbors
nbr_u = G[node.i][node.j-1];
nbr_r = G[node.i+1][node.j];
nbr_d = G[node.i][node.j+1];
nbr_l = G[node.i-1][node.j];
…
//index into neighbors again
nbr_u = G[node.i][node.j-1];
nbr_r = G[node.i+1][node.j];
nbr_d = G[node.i][node.j+1];
nbr_l = G[node.i-1][node.j];
```

SINGLE FUNCTION, MULTIPLE CALLS

Input: `G, node`

```
function getNeighbors(node){
   nbr_u = G[node.i][node.j-1];
   nbr_r = G[node.i+1][node.j];
   nbr_d = G[node.i][node.j+1];
   nbr_l = G[node.i-1][node.j];

   return [nbr_u, nbr_r, nbr_d, nbr_l];
}
```

Lab Takeaways

1. Assignment overview
2. Stencil walkthrough
3. Pendularm demo
4. Coding considerations