AutoRob

Introduction to Autonomous Robotics
Michigan EECS 367

Robot Kinematics and Dynamics
Michigan ME 567 EECS 567 ROB 510

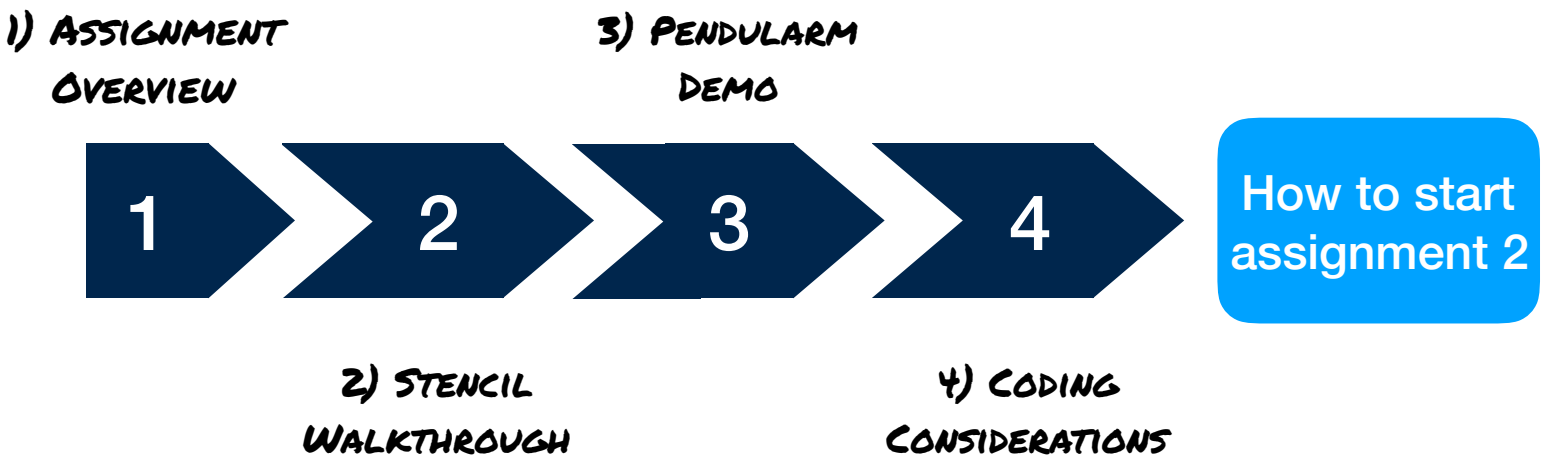
Fall 2019

EECS 367 Lab: Pendularm (Assignment 2) Code Overview

Administrative

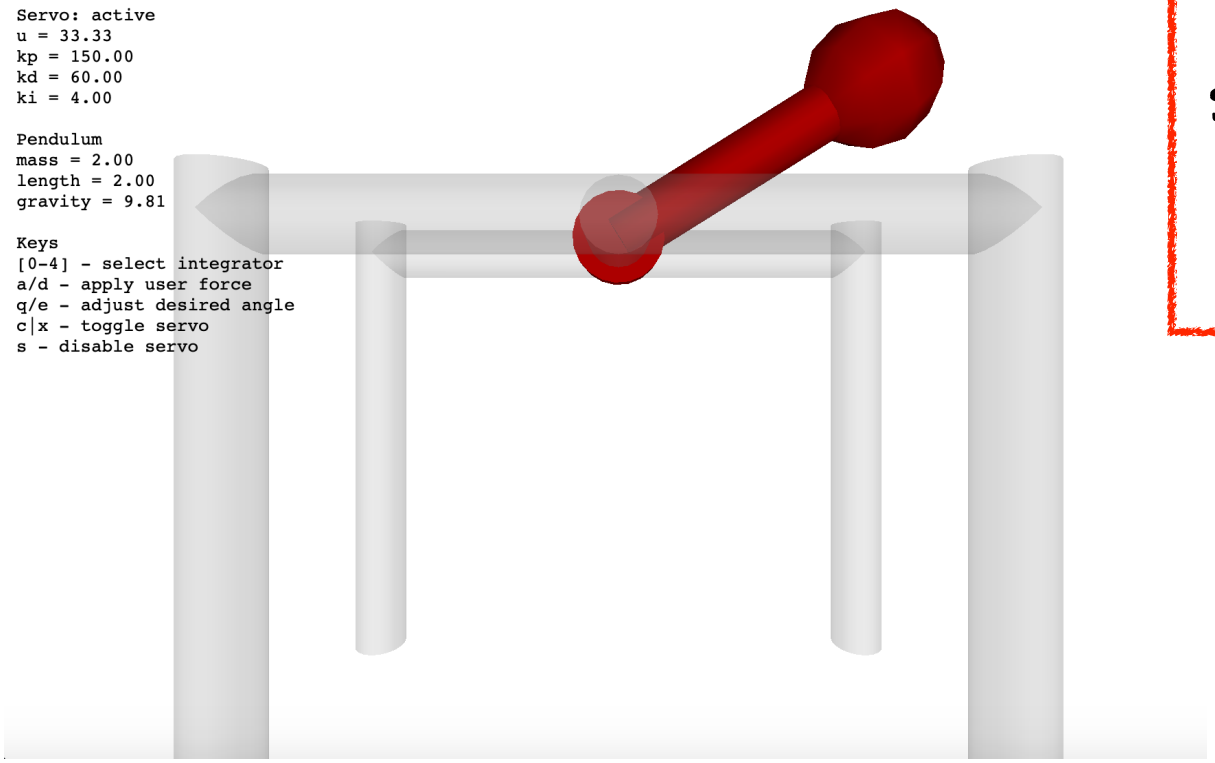
- Assignment #1: Path Planning
  - Regrade policy described on course website
  - Up to 80% credit can be earned after grading
- Assignment #2: Pendularm
  - Due 11:59pm, Wednesday, October 2

Lab Takeaways



Pendularm Overview

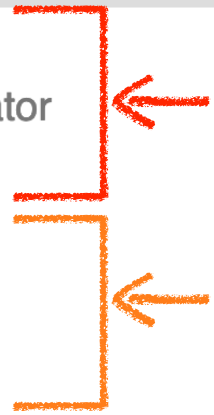
```
System
t = 69.60 dt = 0.05
integrator = velocity verlet
x = -4.16
x_dot = 0.00
x_desired = -4.16

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	All	Euler integrator
4	All	Velocity Verlet integrator
4	All	PID control
1	Grad	Verlet integrator
2	Grad	RK4 integrator
3	Grad	Double pendulum

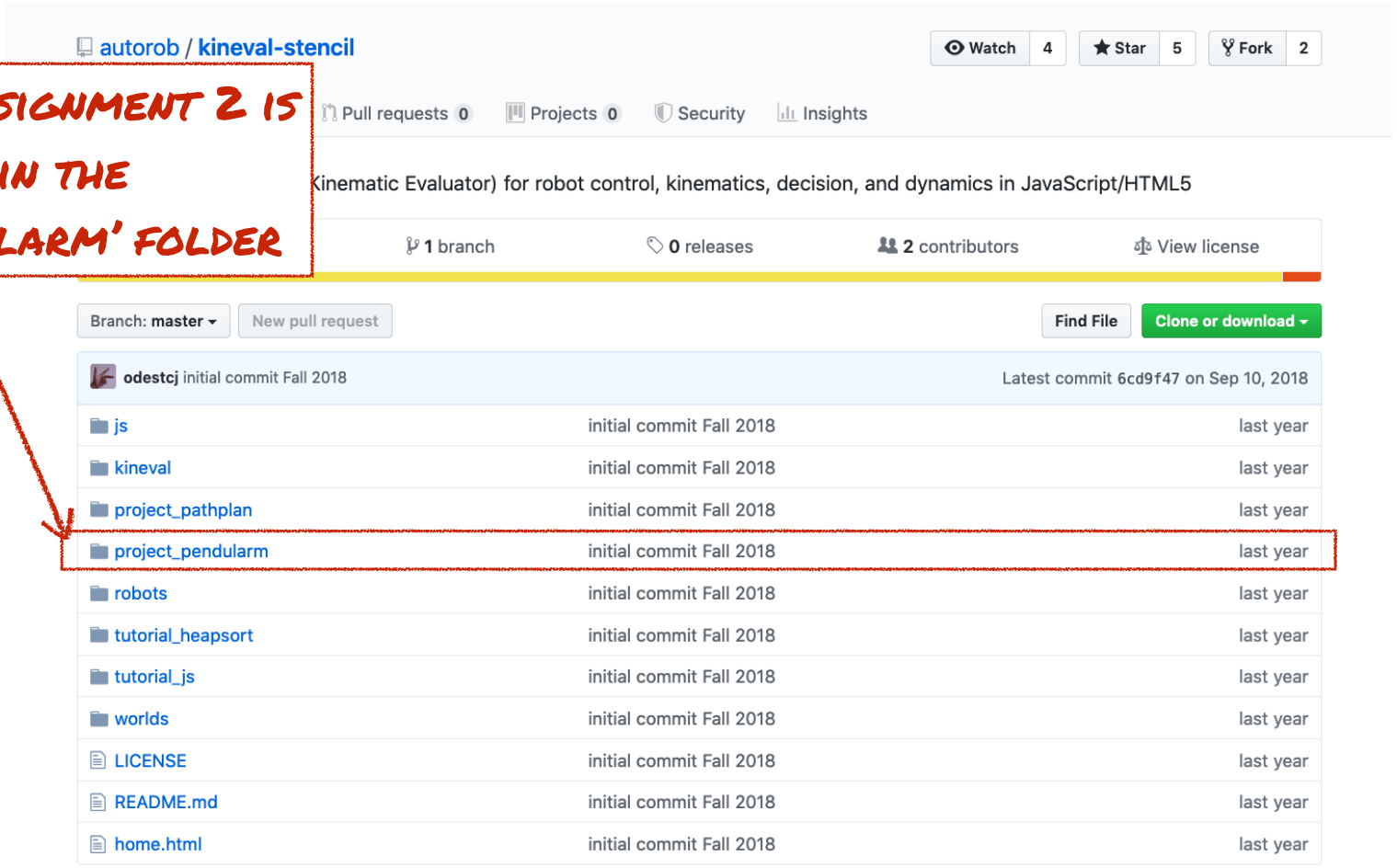


Features assigned to all sections

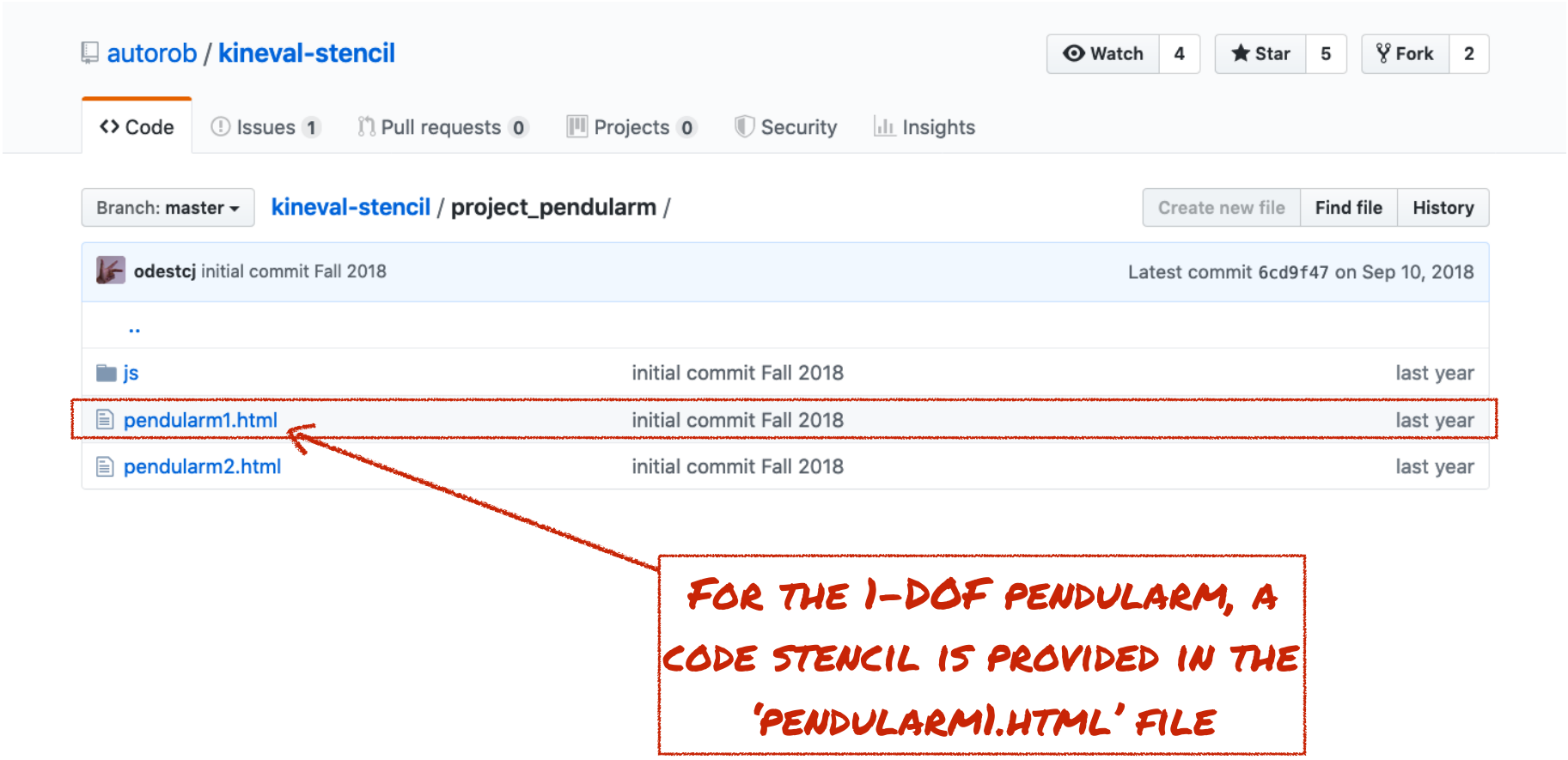
Features assigned to graduate sections

Stencil Walkthrough

ALL CODE FOR ASSIGNMENT 2 IS LOCATED IN THE 'PROJECT_PENDULARM' FOLDER



Stencil Walkthrough



Stencil Walkthrough

<html> OPEN TAG

<body> OPEN TAG

ANY CODE THAT FOLLOWS WILL BE DISPLAYED ON THE WEBPAGE

INCLUDE USEFUL JAVASCRIPT LIBRARIES FOR VISUALIZATION AND CONTROL

FOR THOSE INTERESTED IN WEB ANIMATION, CHECK OUT HTTPS://

CALL TO `init()` FUNCTION FOR ENVIRONMENT INITIALIZATION

CALL TO `animate()` FUNCTION FOR EXECUTION OF ALGORITHMS

Stencil Walkthrough

```
59  function init() {
60
61      // create pendulum object and its kinematic and dynamic parameters
62      pendulum = {length:2.0, mass:2.0, angle:Math.PI/2, angle_dot:0.0};
63
64      // initialize pendulum controls
65      pendulum.control = 0;
66      pendulum.desired = -Math.PI/2.5;
67      pendulum.desired_dot = 0;
68
69      // initialize integral term accumulated error to zero
70      accumulated_error = 0;
71
72      // set gravity
73      gravity = 9.81;  // Earth gravity
74
75      // initialize pendulum PID servo gains
76      pendulum.servo = {kp:0, kd:0, ki:0};  // no control
77
78      // initialize time and set timestep
79      t = 0;
80      dt = 0.05;  // default
81
82      // initialize method of numerical integration of dynamics
83      //numerical_integrator = "euler";
84      //numerical_integrator = "verlet";
85      //numerical_integrator = "velocity verlet";
86      //numerical_integrator = "runge-kutta";
```

GLOBAL VARIABLE INITIALIZATION

Stencil Walkthrough

```
111   function animate() {
112
113       // note: three.js includes requestAnimationFrame shim
114       // alternative to using setInterval for updating in-browser drawing
115       // this effectively request that the animate function be called again for next draw
116       // http://learningwebgl.com/blog/?p=3189
117       requestAnimationFrame( animate );
```

SETUP NEXT CALL TO `animate()`

•

•

•

USE THREE.JS TO RENDER THE SCENE

```
234
235       // threejs rendering update
236       renderer.render( scene, camera );
237
238   }
```

Stencil Walkthrough

```
function animate() {
```

```
119     // switch between numerical integrators based on user input
120     if (keyboard.pressed("0"))
121         numerical_integrator = "none";
122     if (keyboard.pressed("1"))
123         numerical_integrator = "euler";
124     if (keyboard.pressed("2"))
125         numerical_integrator = "verlet";
126     if (keyboard.pressed("3"))
127         numerical_integrator = "velocity verlet";
128     if (keyboard.pressed("4"))
129         numerical_integrator = "runge-kutta";
130
131     // update servo desired state from user interaction
132     if ( keyboard.pressed("e") )
133         pendulum.desired += 0.05; // move the desired angle for the servo
134     if ( keyboard.pressed("q") )
135         pendulum.desired += -0.05; // move the desired angle for the servo
136
137
138     // add user force from user interaction
139     if ( keyboard.pressed("d") )
140         pendulum.control += 50.0;  // add a motor force to the pendulum motor
141     else if ( keyboard.pressed("a") )
142         pendulum.control += -50.0; // add a motor force to the pendulum motor
```

AT EVERY CALL TO `animate()`
check for keyboard input

Update control variables using input

Stencil Walkthrough

```
function animate() {
```

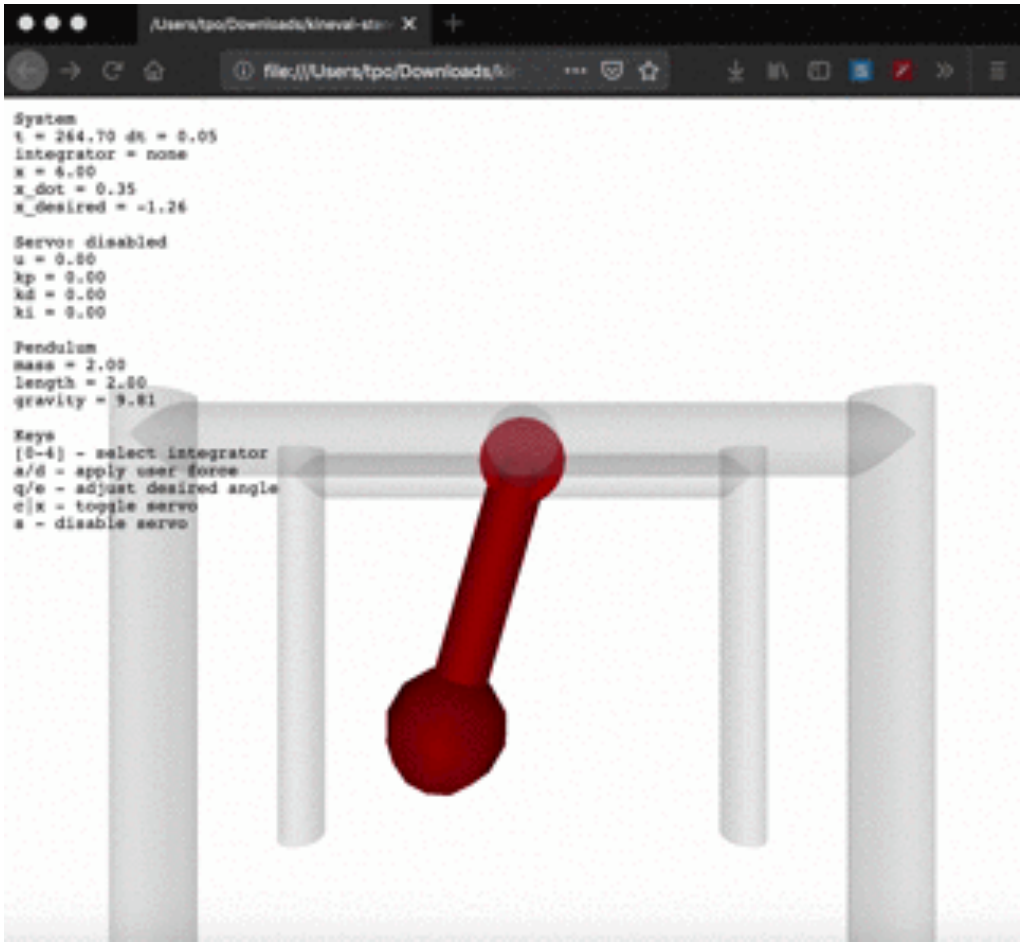
```
160     // integrate pendulum state forward in time by dt
161     if (typeof numerical_integrator === "undefined")
162         numerical_integrator = "none";
163
164     if (numerical_integrator === "euler") {
165
166     // STENCIL: a correct Euler integrator is REQUIRED for assignment
167
168     }
169     else if (numerical_integrator === "verlet") {
170
171     // STENCIL: basic Verlet integration
172
173     }
174     else if (numerical_integrator === "velocity verlet") {
175
176     // STENCIL: a correct velocity Verlet integrator is REQUIRED for assignment
177
178     }
179     else if (numerical_integrator === "runge-kutta") {
180
181     // STENCIL: Runge-Kutta 4 integrator
182     }
183     else {
184         pendulum.angle_previous = pendulum.angle;
185         pendulum.angle = (pendulum.angle+Math.PI/180)%(2*Math.PI);
186         pendulum.angle_dot = (pendulum.angle-pendulum.angle_previous)/dt;
187         numerical_integrator = "none";
188     }
189
190     // set the angle of the pendulum
191     pendulum.geom.rotation.y = pendulum.angle;  // threejs cylinders have their
```

← Feature stencils

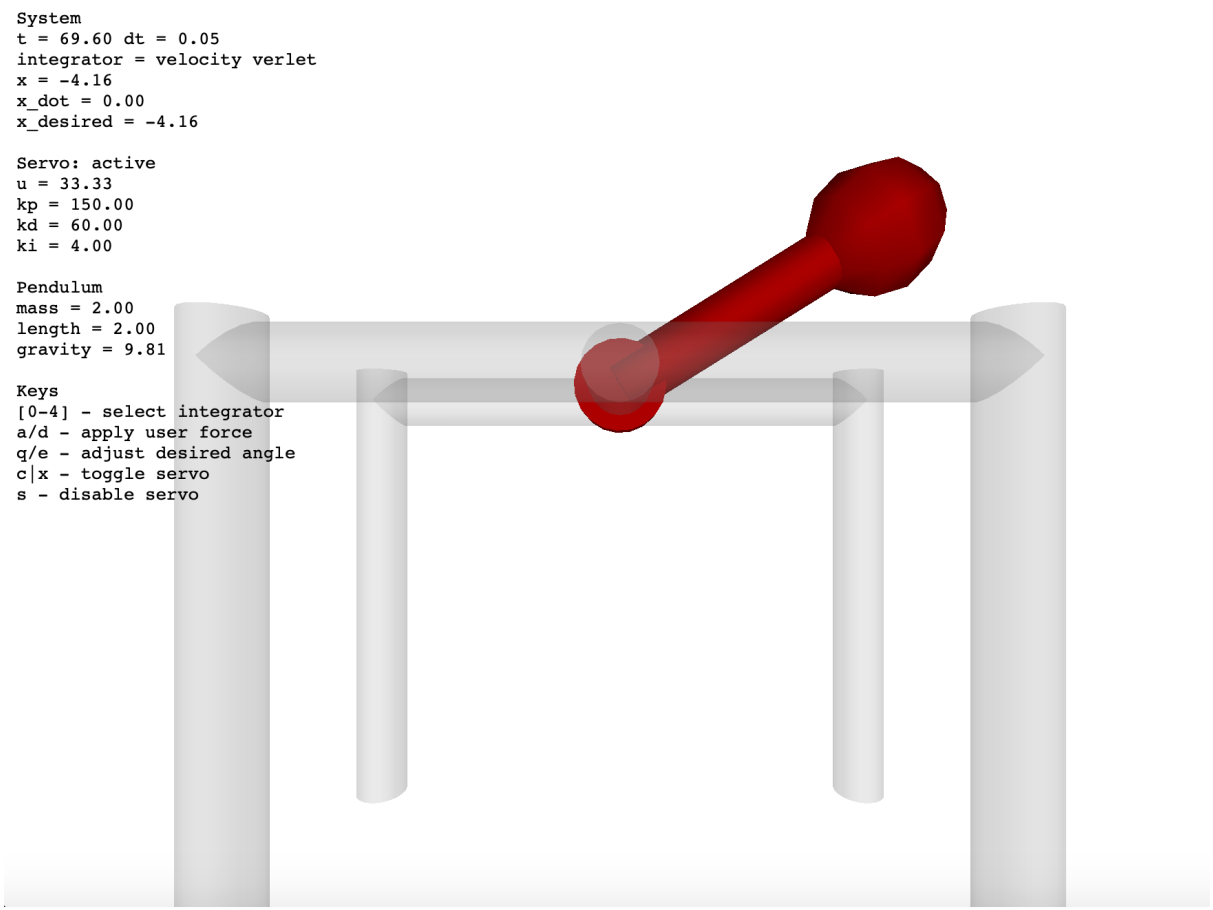
← Default rotation

← Update pendulum

Stencil Walkthrough



PendulArm Demo



Coding Considerations

- These concepts are optional, meant to help you on programming assignments
- Programming concepts to consider using for writing readable, easily debuggable(!) code
  - Use comments where complicated
  - Add whitespace for readability
  - Local variables to store indices/raw data
  - 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 = [1,2,3,4];
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(i=0;i<x.length;++i){
    y=doStuff(i,j,x);
    doMoreStuff(y);
  }
}
```

With Whitespace

```
// 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];

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+0][node.j-1]
nbr_r=G[node.i+1][node.j+0]
nbr_d=G[node.i+0][node.j+1]
nbr_l=G[node.i-1][node.j+0]
.
.
.
nbr_u=G[node.i+0][node.j-1]
nbr_r=G[node.i+1][node.j+0]
```

Single Function, Multiple Calls

```
Input: G, node

Function getNbrs(node){
  nb_idxs = [[node.i+0,node.j-1],
             [node.i+1,node.j+0],
             [node.i+0][node.j+1],
             [node.i-1][node.j+0]];
  return nb_idxs;
}
```