EECS 367 & ROB 320 Lab KinEval overview

Administrative

- Assignment #2: Pendularm
  - Due tonight, February 4 11:59pm
- Quiz #3: Next Monday, January 24th
  - Through gradescope, available 12:00am-11:59pm
  - Time limit of 30 minutes
  - Covers material from assignments #1,2
  - Don’t discuss quiz with other students; honor code

Administrative

- Pendularm Setpoint Competition!
  - Final results published over the weekend

Lab Takeaways

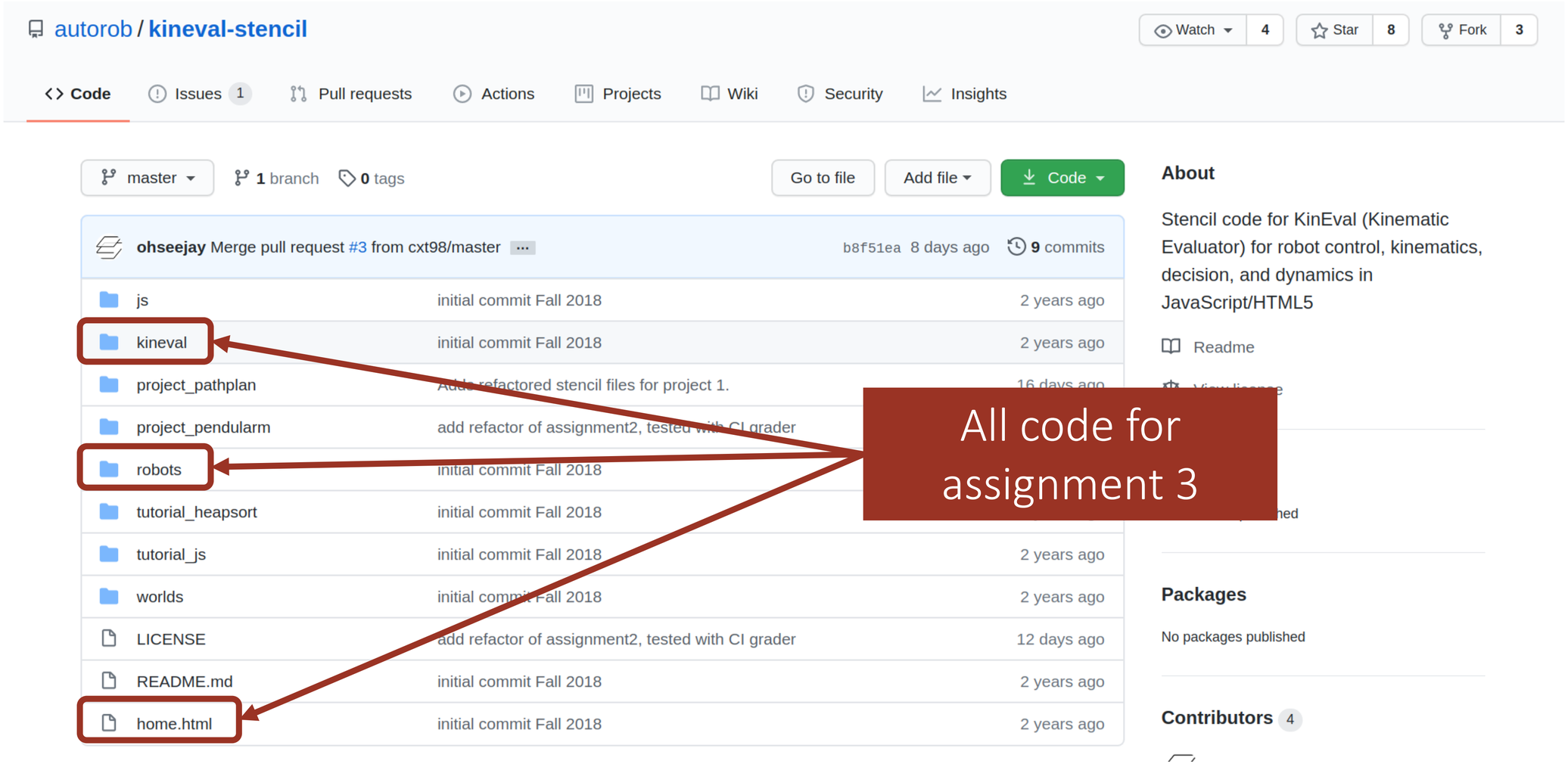
1. KinEval overview
2. KinEval walkthrough
3. Implementation advice

→ How to start Assignment 3

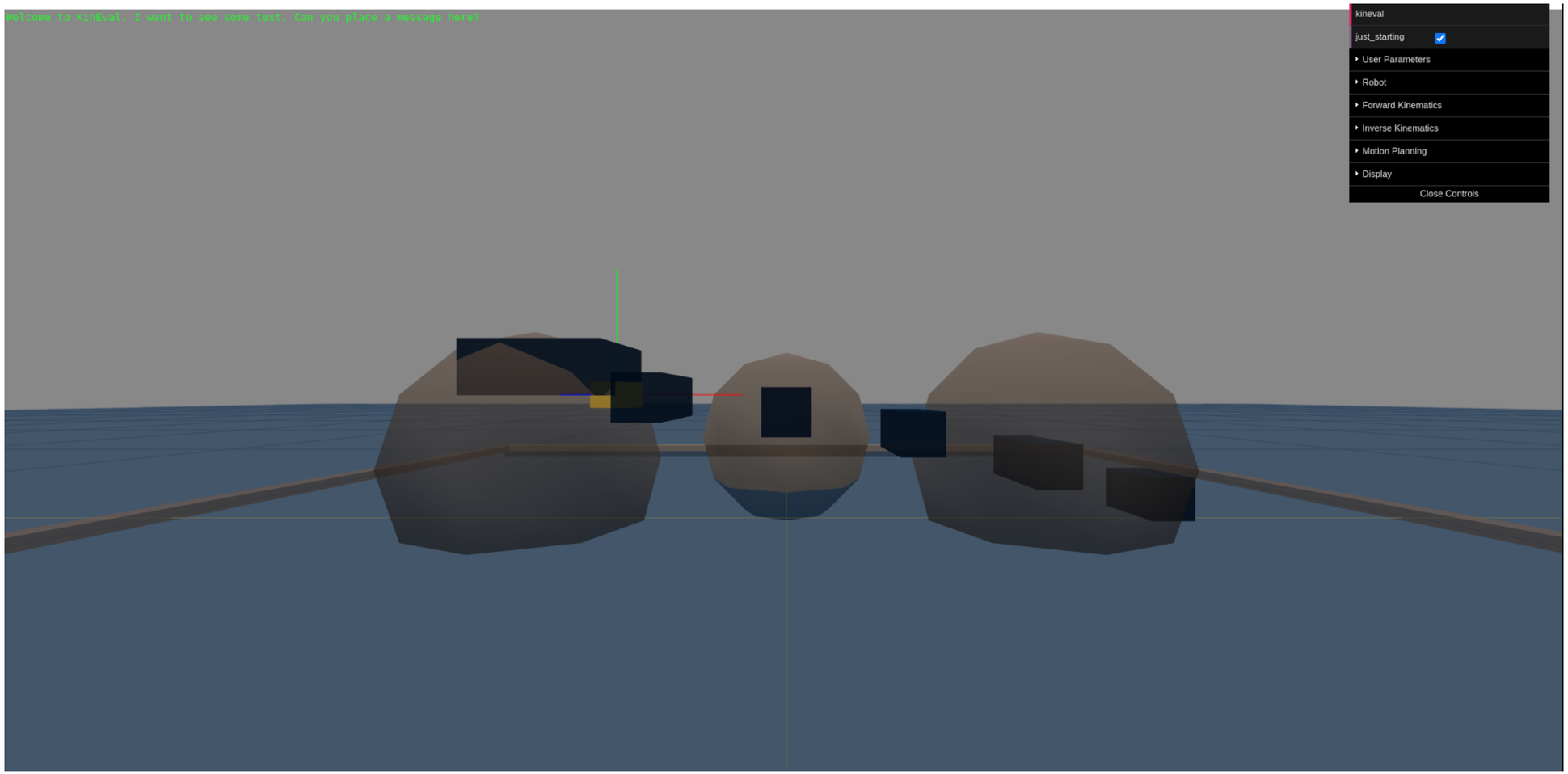
Forward Kinematics Overview

		Assignment 3: Forward Kinematics
2	All	Core matrix routines
8	All	FK transforms
2	All	Joint selection/rendering

KinEval Overview



home.html


welcome to KinEval. I want to see some text. Can you place a message here?
kineval
just_starting
User Parameters
Robot
Forward Kinematics
Inverse Kinematics
Motion Planning
Display
Close Controls

home.html

home.html

```
<!-- kineval includes -->
<script src="kineval/kineval.js"></script>
<script src="kineval/kineval_startingpoint.js"></script>
<script src="kineval/kineval_robot_init.js"></script>
<script src="kineval/kineval_robot_init_joints.js"></script>
<script src="kineval/kineval_threejs.js"></script>
<script src="kineval/kineval_userinput.js"></script>

<!-- kineval FK/drawing -->
<script src="kineval/kineval_forward_kinematics.js"></script>
<script src="kineval/kineval_matrix.js"></script>
<script src="kineval/kineval_quaternion.js"></script>

<!-- kineval FK/joint control -->
<script src="kineval/kineval_controls.js"></script>
<script src="kineval/kineval_servo_control.js"></script>

<!-- kineval IK -->
<script src="kineval/kineval_inverse_kinematics.js"></script>

<!-- kineval motion planning -->
<script src="kineval/kineval_rrt_connect.js"></script>
<script src="kineval/kineval_collision.js"></script>

<!-- kineval experimental rosbridge/ROS for connectivity to a real robot -->
<script type="text/javascript" src="js/eventemitter2.min.js"></script>
<script type="text/javascript" src="js/roslib.min.js"></script>
<script src="kineval/kineval_rosbridge.js"></script>
```

KinEval source files included here

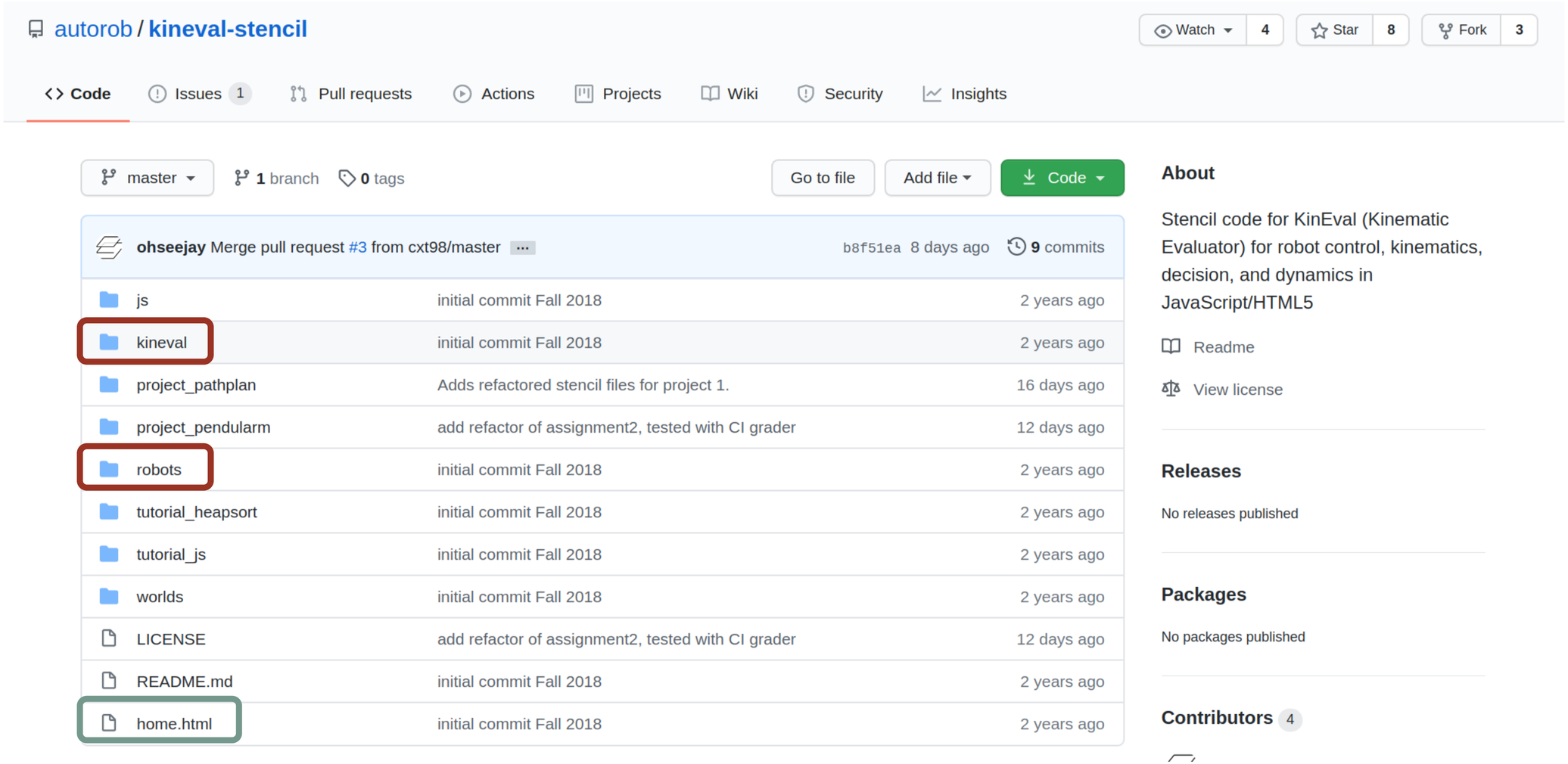
home.html

home.html

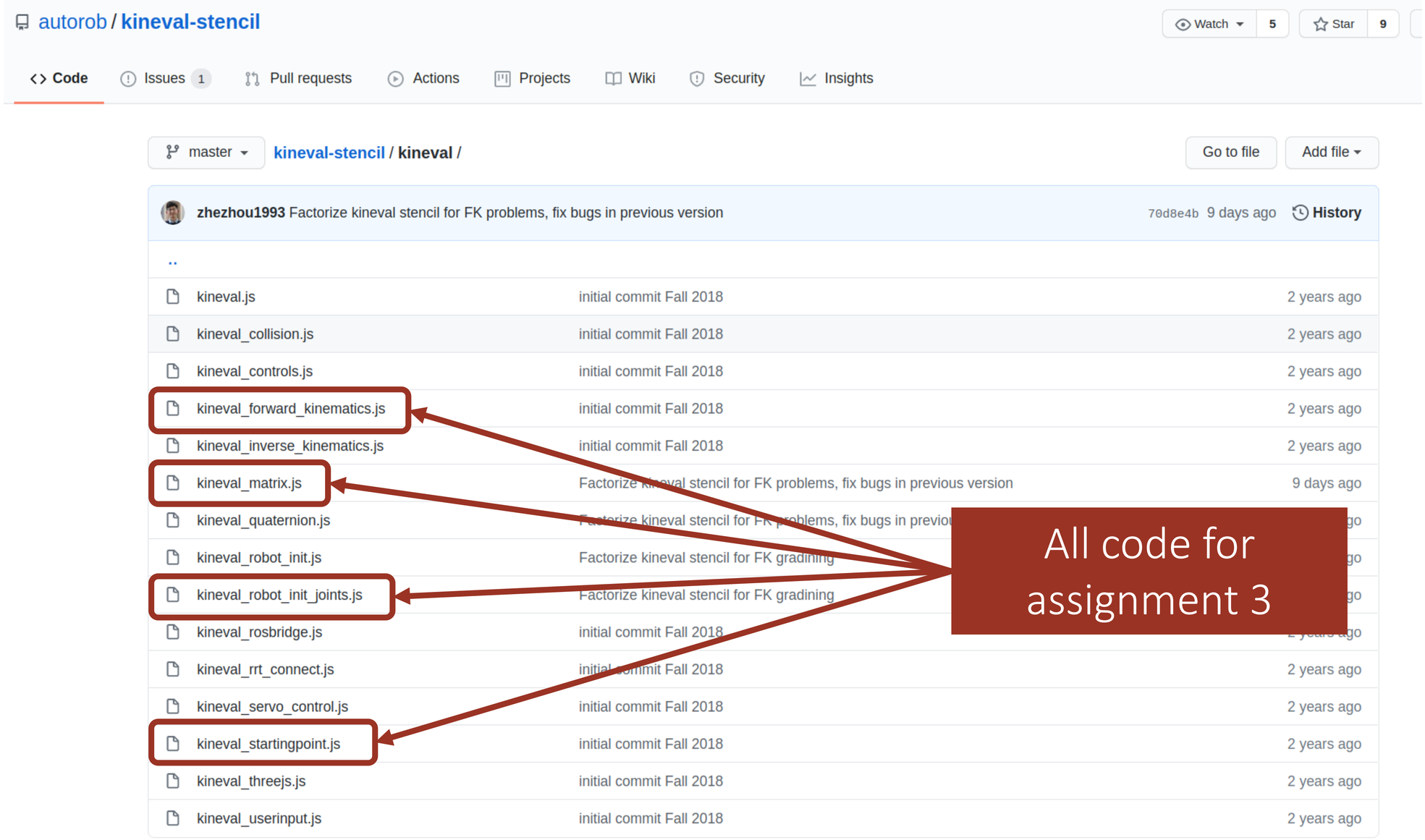
```
150  // STUDENT: my_animate is where your robot's controls and movement are updated over time
151  function my_animate() {
152
153      // set to starting point mode is true as default (initialized in kineval.js)
154      //   set to false once starting forward kinematics project
155      //kineval.params.just_starting = false;
156
157      if (kineval.params.just_starting == true) {
158          startingPlaceholderAnimate();
159          kineval.robotDraw();
160          return;
161      }
162
163      // ROBOT DYNAMICS
164
165      // update robot configuration from applied robot controls
166      //   (assuming pure kinematics for now)
167      kineval.applyControls(robot);
168
169      // HANDLE USER CONTROLS
170
171      // handle user input
172      kineval.handleUserInput();
173
174      // perform forward kinematics placing robot links in space wrt configuration
175      kineval.robotForwardKinematics();
```

`my_animate()` is called at every animation frame

KinEval Overview



KinEval Overview



kineval_startingpoint.js

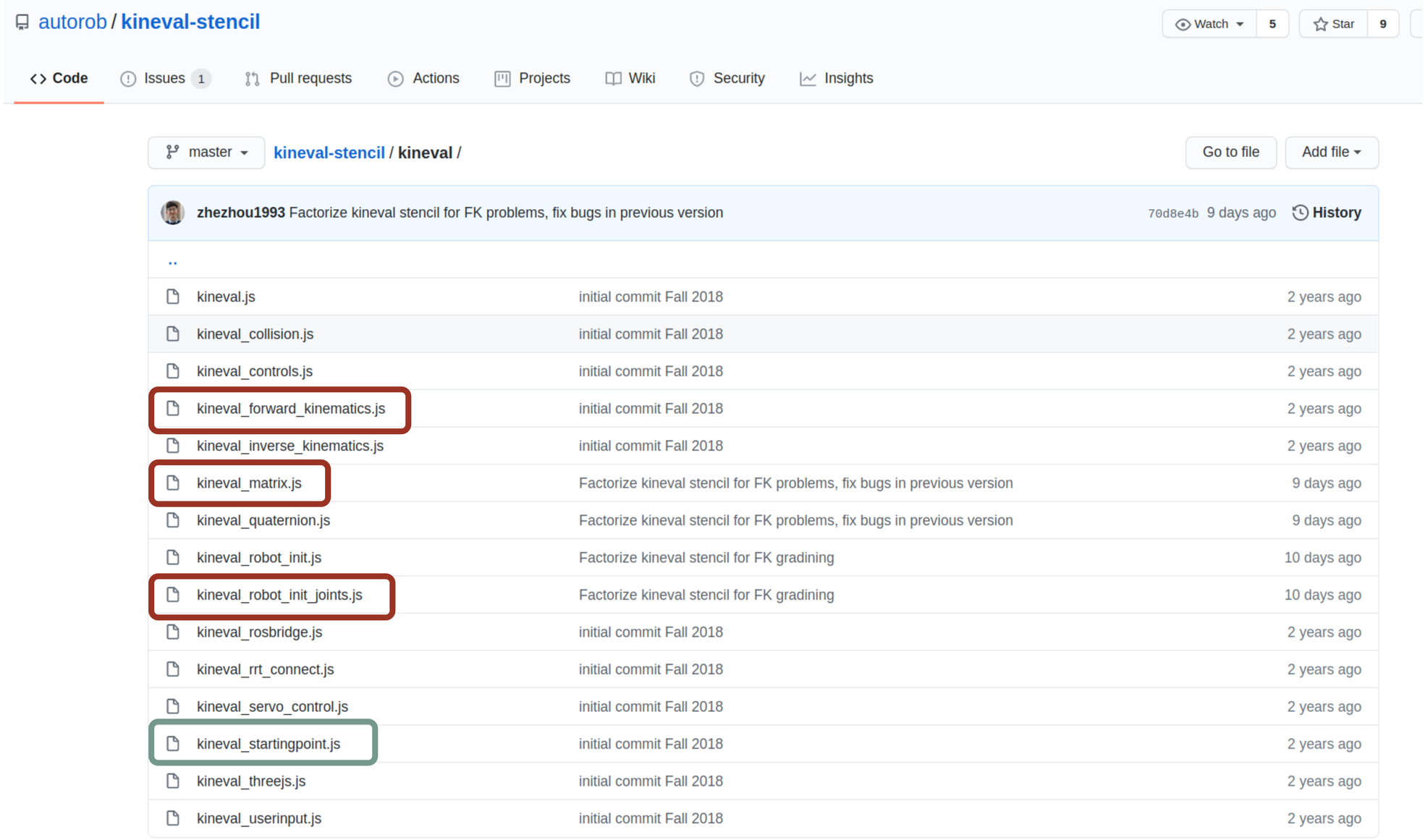
As the name suggests, this file is meant to build your comfort with the source code

kineval_startingpoint.js

```
365  kineval.startingPlaceholderUserInput = function startingPlaceholderUserInput() {
366
     /* keyboard is a threejs helper object for reading keyboard state.
          keyboard.pressed() will return true if a particular key is being
          pressed, without the need for a callback event handler
     */
     if (keyboard.pressed("shift+x")) {
         textbar.innerHTML = "come on down";  // make the objects move down
     // STENCIL: update the vertical offset variable
     }
     else if (keyboard.pressed("x")) {
376      textbar.innerHTML = "moving on up";  // make the objects move up
377  // STENCIL: update the vertical offset variable
378  }
379  else if (keyboard.pressed("shift+z")) {
380      // increase the jittering of the objects
381      textbar.innerHTML = "its time for the percolator";
382  // STENCIL: update the radius of the jittering
383  }
384  else if (keyboard.pressed("z")) {
385      // decrease the jittering of the objects
386      textbar.innerHTML = "relax your mind, let your conscience be free";
387  // STENCIL: update the radius of the jittering
388  }
389  else if (keyboard.pressed("shift+1")) {
390      // increase spacing along the x-axis between the objects
391      textbar.innerHTML = "sail away";
392  // STENCIL: update the global spacing variable
393  }
394  else if (keyboard.pressed("1")) {
```

Light implementation exercises for controlling webpage marked with ‘STENCIL’

KinEval Overview



kineval_matrix.js

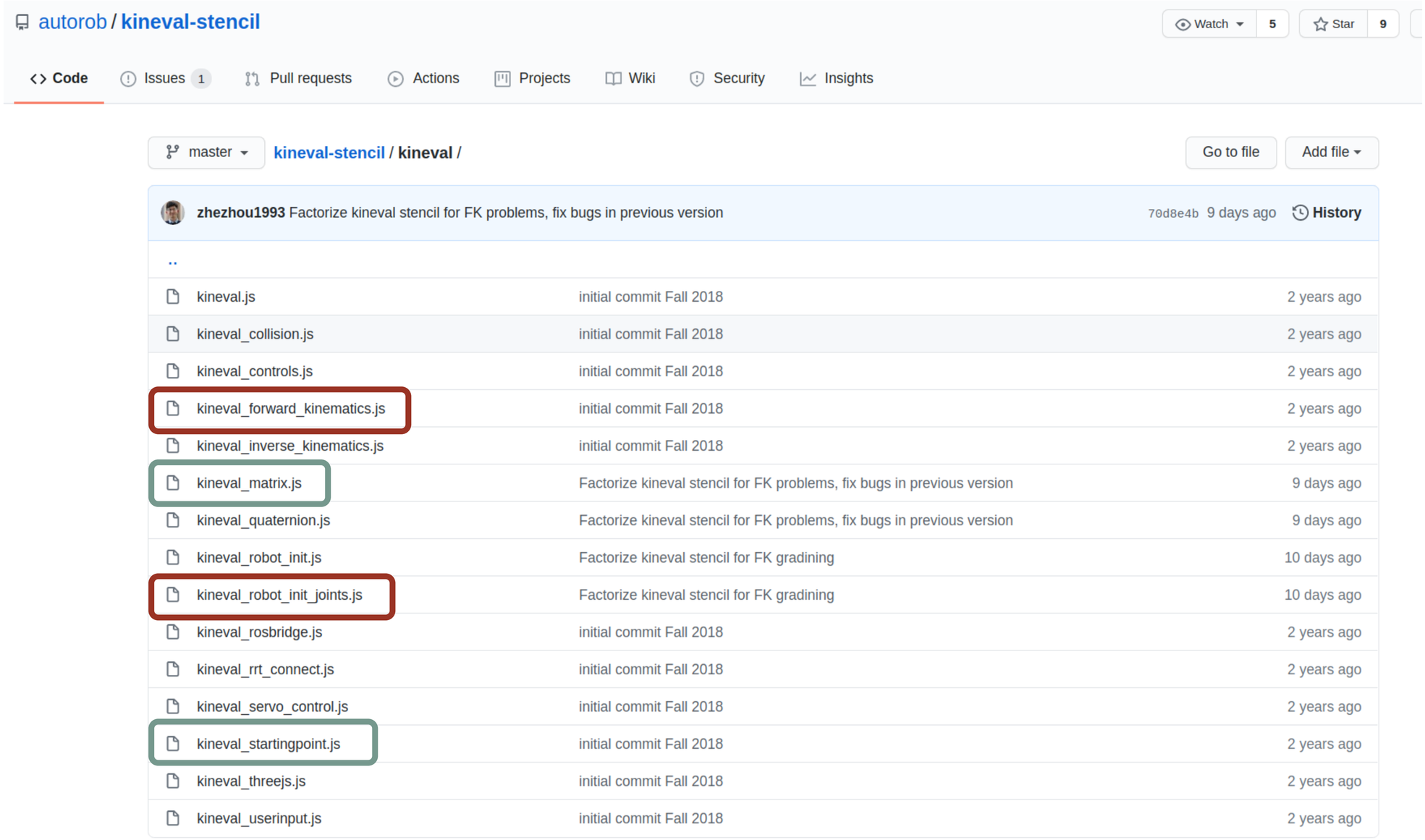
kineval_matrix.js

```
// STENCIL: reference matrix code has the following functions:
//   matrix_multiply
//   matrix_transpose
//   matrix_pseudoinverse
//   matrix_invert_affine
//   vector_normalize
//   vector_cross
//   generate_identity
//   generate_translation_matrix
//   generate_rotation_matrix_X
//   generate_rotation_matrix_Y
//   generate_rotation_matrix_Z



// **** Function stencils are provided below, please uncomment and implement them ****//



// function matrix_multiply(m1,m2) {
//     // returns 2D array that is the result of m1*m2

// }

// function matrix_transpose(m) {
//     // returns 2D array that is the result of m1*m2

```

Except for `matrix_pseudoinverse`, which is for a later assignment

Stencils for matrix operations that you need to implement

KinEval Overview



kineval_forward_kinematics.js

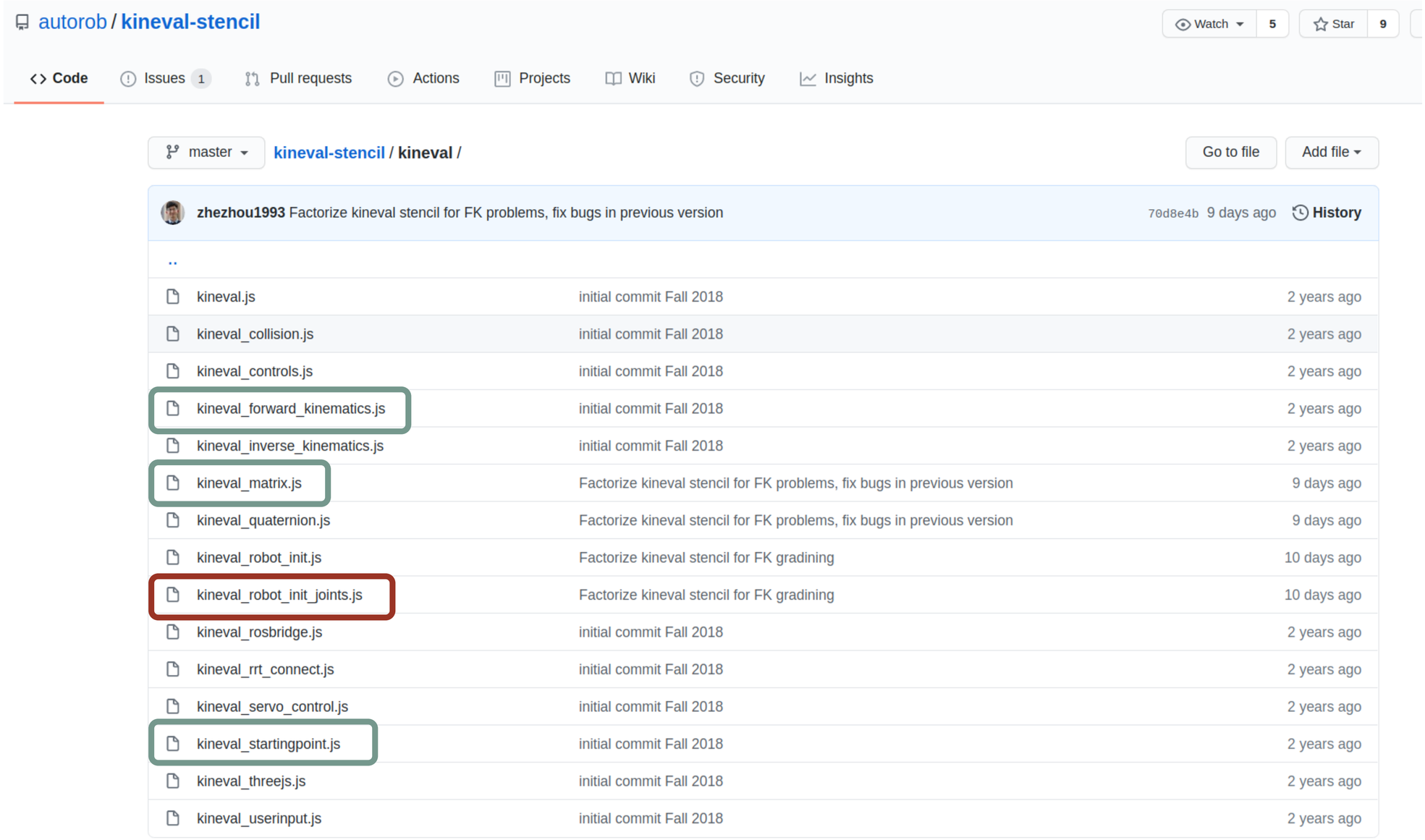
kineval_forward_kinematics.js

```
18
19  kineval.robotForwardKinematics = function robotForwardKinematics () {
20
21      if (typeof kineval.buildFKTransforms === 'undefined') {
22          textbar.innerHTML = "forward kinematics not implemented";
23          return;
24      }
25
26      // STENCIL: implement kineval.buildFKTransforms();
27
28  }
29
30      // STENCIL: reference code alternates recursive traversal over
31      //   links and joints starting from base, using following functions:
32      //     traverseFKBase
33      //     traverseFKLink
34      //     traverseFKJoint
35      //
```

Your recursive traversal of links and joints to build up matrix stack

Should result in updated `.xform` for each link and joint

KinEval Overview



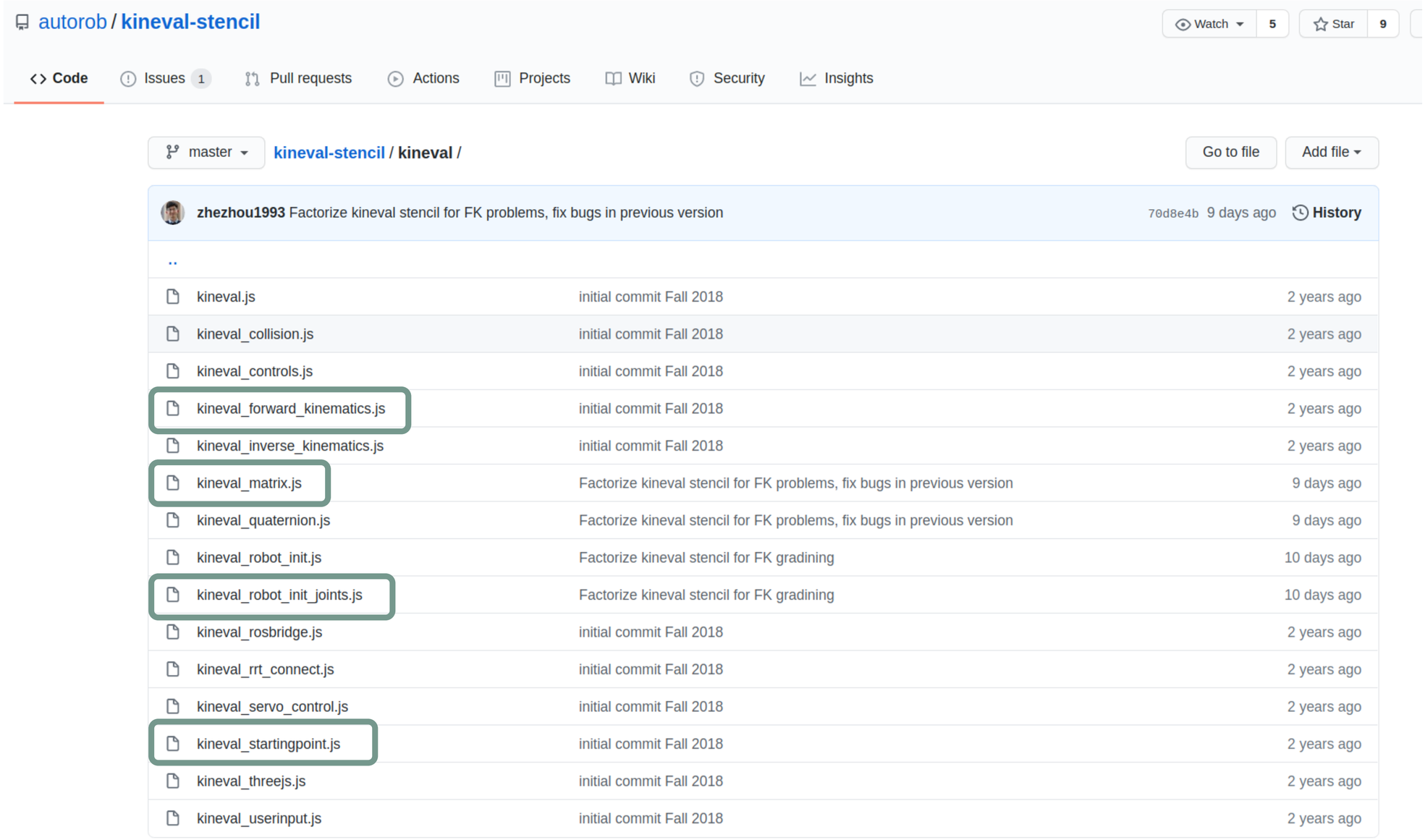
kineval_robot_init_joints.js

kineval_robot_init_joints.js

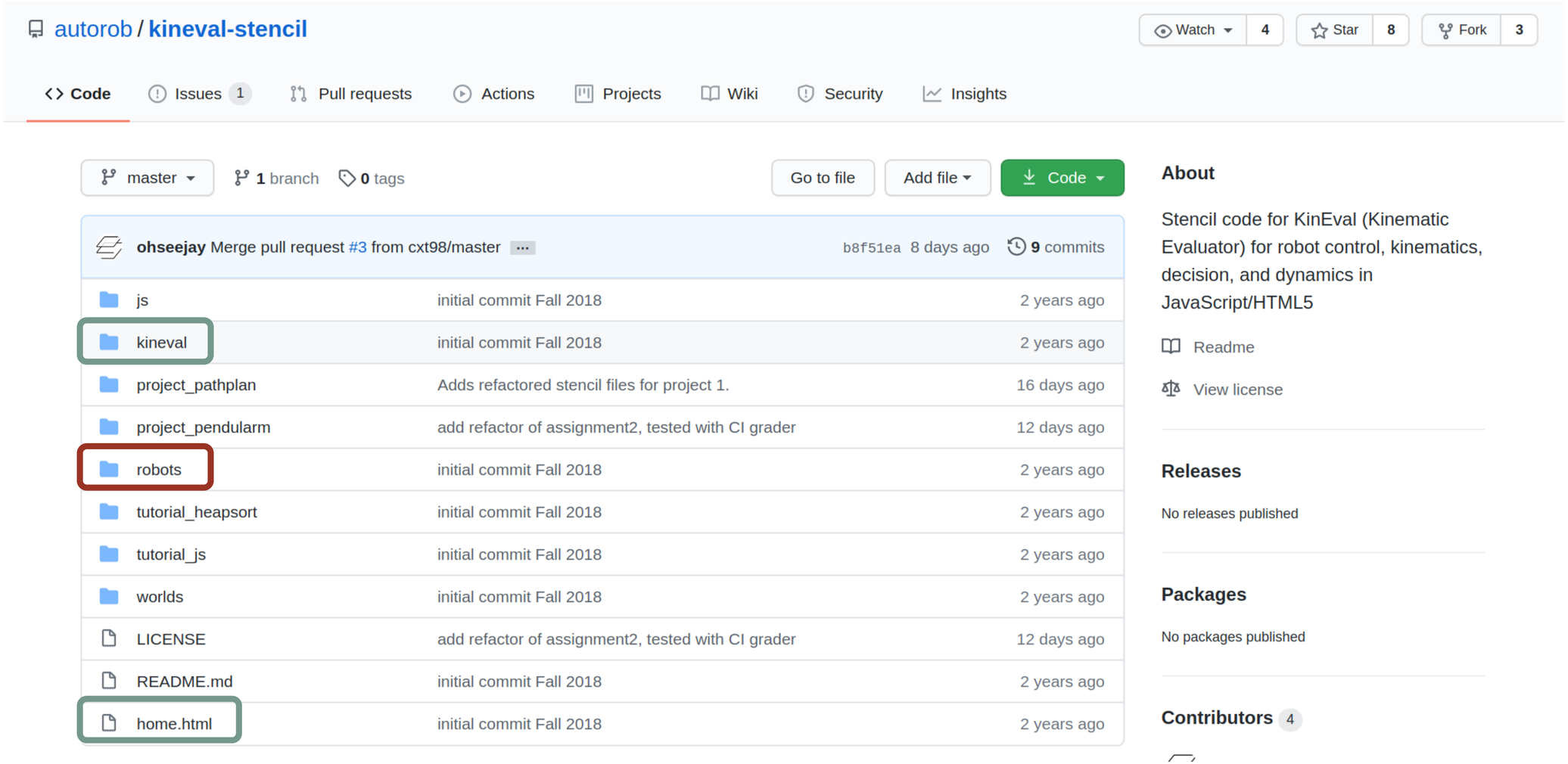
```
20      for (x in robot.joints) {
21
22          // give the joint its name as an id
23          robot.joints[x].name = x;
24
25          // initialize joint angle value and control input value
26          robot.joints[x].angle = 0;
27          robot.joints[x].control = 0;
28          robot.joints[x].servo = {};
29          //set appropriate servo gains for arm setpoint control
30          robot.joints[x].servo.p_gain = 0;
31          robot.joints[x].servo.p_desired = 0;
32          robot.joints[x].servo.d_gain = 0;
33  /* STENCIL START */
34      // STENCIL: complete kinematic hierarchy of robot for convenience.
35      //   robot description only specifies parent and child links for joints.
36      //   additionally specify parent and child joints for each link
```

Initialize robot's internal structure between links/joints

KinEval Overview

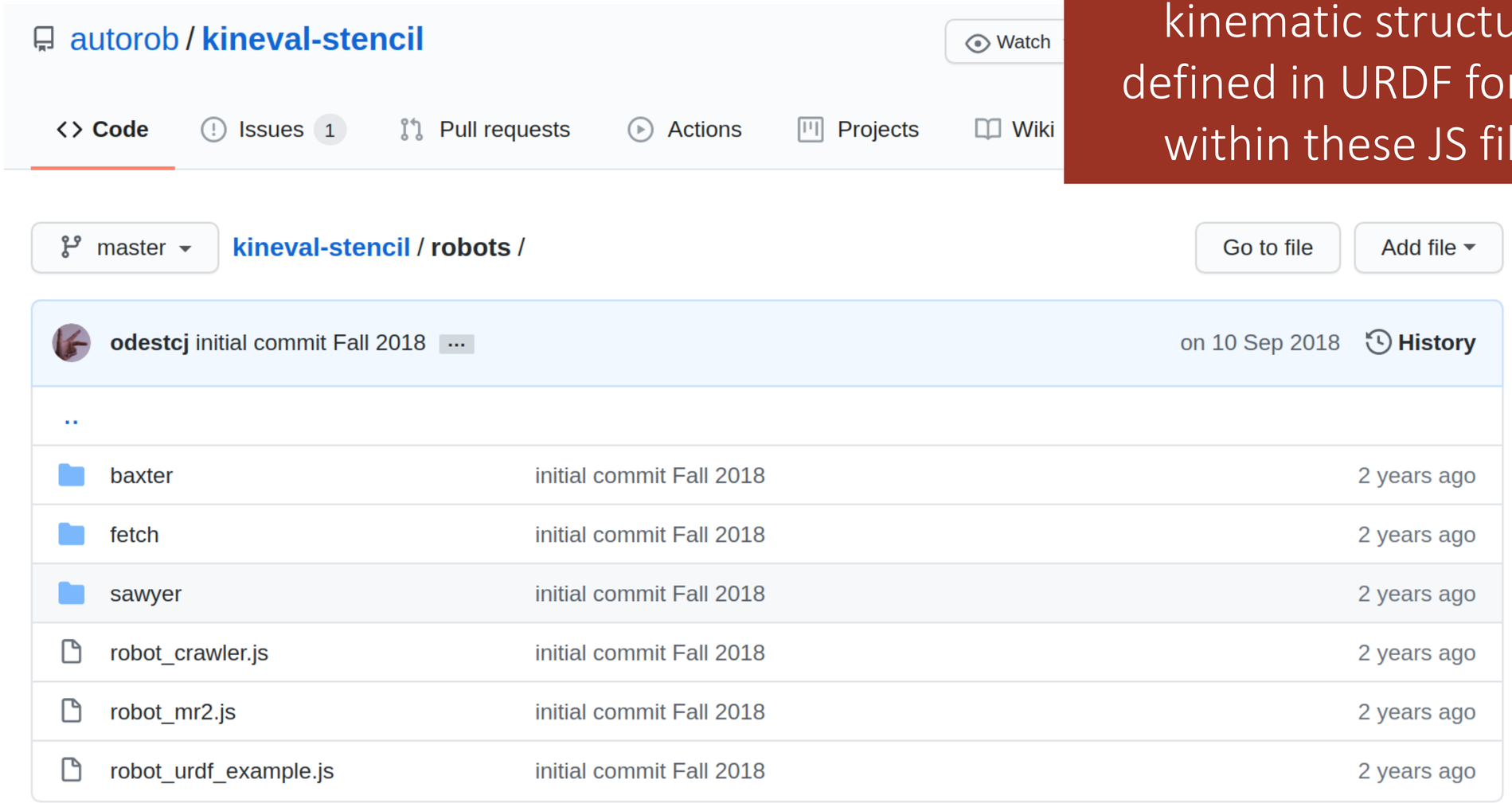


KinEval Overview



KinEval Overview

Each robot has its kinematic structure defined in URDF format within these JS files



robot_urdf_example.js

robot_urdf_example.js

```
1   //   CREATE ROBOT STRUCTURE
2
3   ////////////////////////////////////////////////
4   /////     DEFINE ROBOT AND LINKS
5   ////////////////////////////////////////////////
6
7   // create robot data object
8   robot = new Object(); // or just {} will create new object
9
10  // give the robot a name
11  robot.name = "urdf_example";
12
13  // initialize start pose of robot in the world
14  robot.origin = {xyz: [0,0.1,0], rpy:[0,0,0]};
15
16  // specify base link of the robot; robot.origin is transform of world to the robot base
17  robot.base = "link1";
18
19  // specify and create data obje
20  robot.links = {"link1": {}, "li
```

STENCIL in kineval_robot_init_joints.js for initializing this information

robot =

name: “urdf_example”

origin: xyz: [0,0.1,0]
rpy: [0,0,0]

base: “link1”

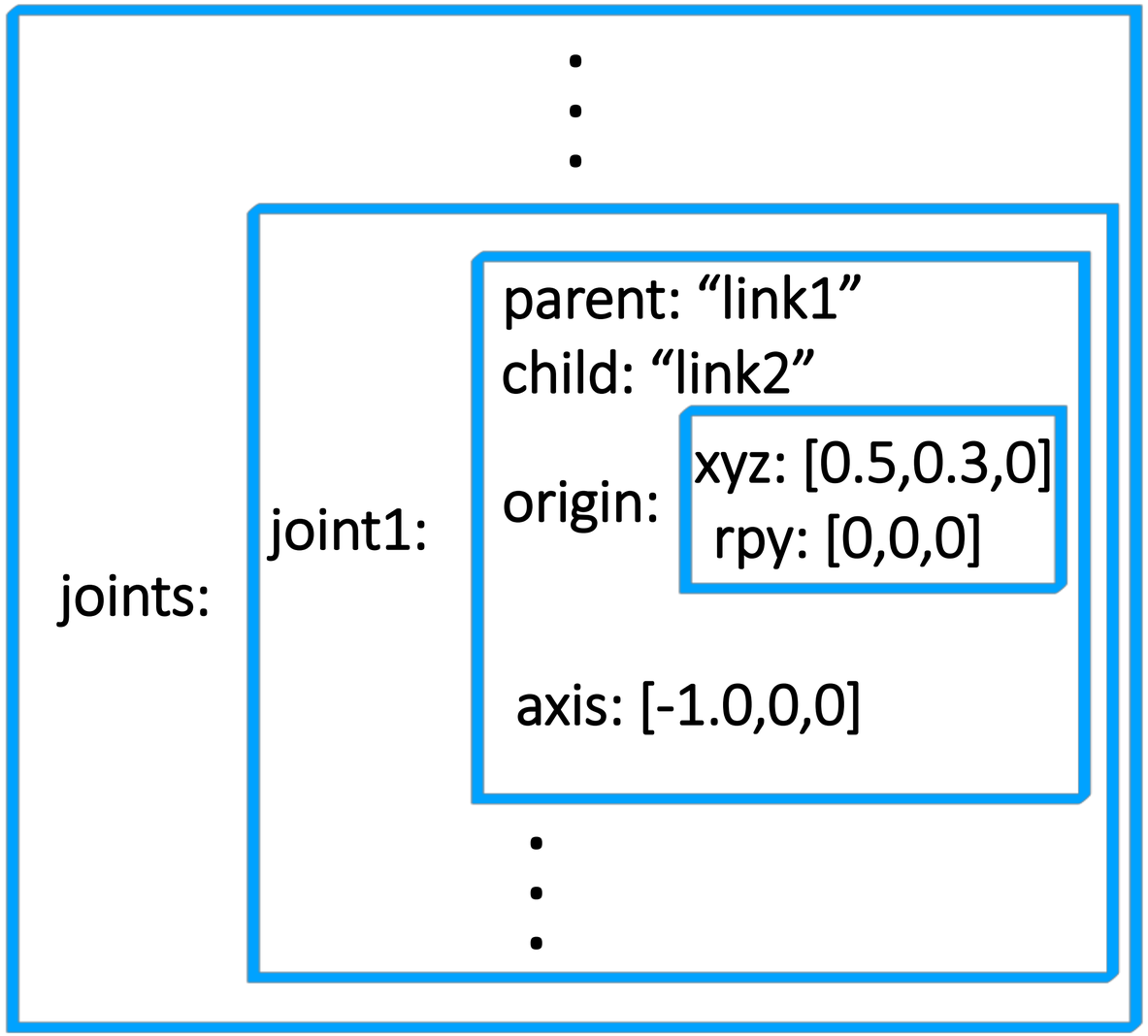
links:
- link1:
- link2:
- link3:
- link4:

robot_urdf_example.js

robot_urdf_example.js

```
22  ////////////////////////////////////////////////
23  /////     DEFINE JOINTS AND KINEMATIC HIERARCHY
24  ////////////////////////////////////////////////
25
26  /*      joint definition template
27          // specify parent/inboard link and child/outboard link
28          robot.joints.joint1 = {parent:"link1", child:"link2"};
29          // joint origin's offset transform from parent link origin
30          robot.joints.joint1.origin = {xyz: [5,3,0], rpy:[0,0,0]};
31          // joint rotation axis
32          robot.joints.joint1.axis = [0.0,0.0,1.0];
33  */
34
35
36  // roll-pitch-yaw defined by ROS as corresponding to x-y-z
37  //http://wiki.ros.org/urdf/Tutorials/Create%20your%20own%20urdf%20file
38
39  // specify and create data objects for the joints of the robot
40  robot.joints = {};
41
42  robot.joints.joint1 = {parent:"link1", child:"link2"};
43  robot.joints.joint1.origin = {xyz: [0.5,0.3,0.0], rpy:[0,0,0]};
44  robot.joints.joint1.axis = [-1.0,0.0,0];  // simpler axis
45
46  robot.joints.joint2 = {parent:"link1", child:"link3"};
47  //robot.joints.joint2.origin = {xyz: [-0.2,0.5,0], rpy:[0,0,1.57]};
48  robot.joints.joint2.origin = {xyz: [-0.2,0.5,0], rpy:[0,0,Math.PI/2]};
49  //robot.joints.joint2.axis = [-0.707,0.707,0];
50  robot.joints.joint2.axis = [-Math.cos(Math.PI/4),Math.cos(Math.PI/4),0];
51
52  robot.joints.joint3 = {parent:"link3", child:"link4"};
```

robot =

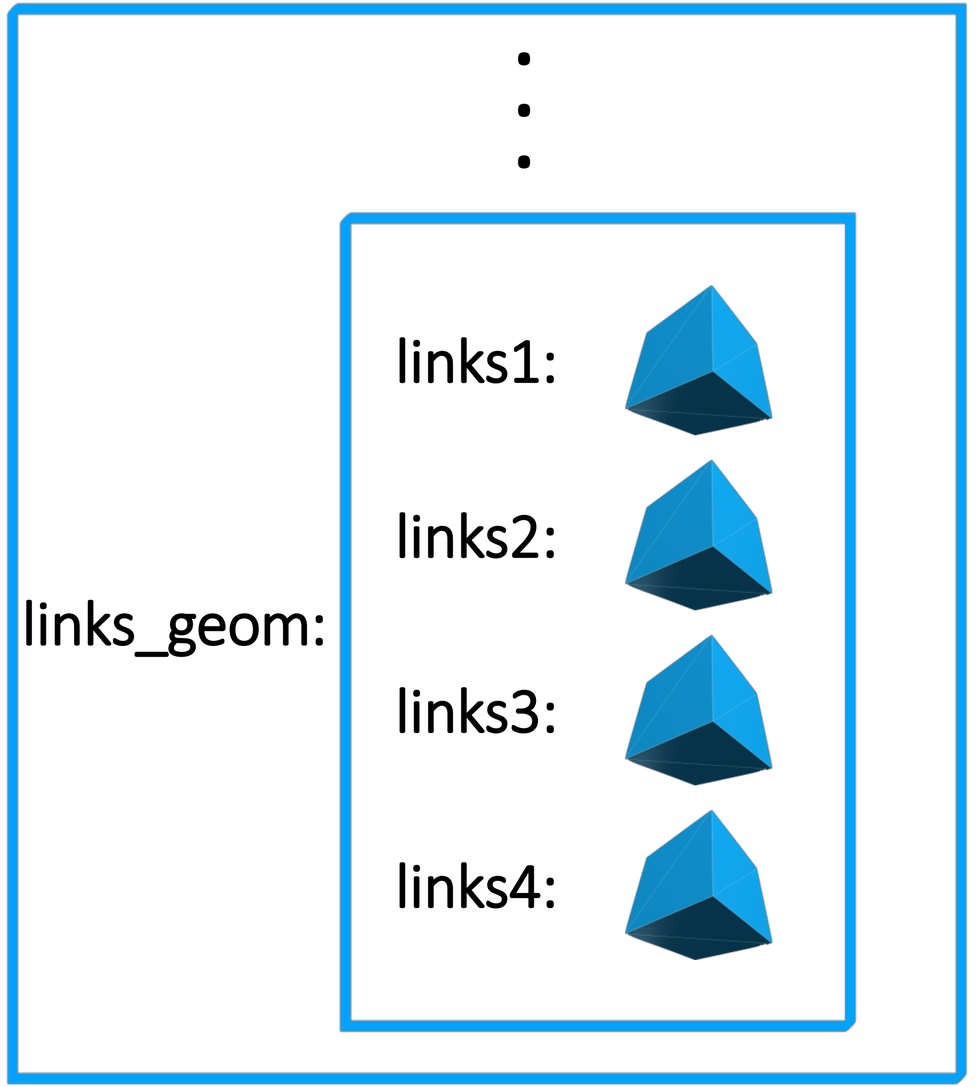


robot_urdf_example.js

robot_urdf_example.js

```
64  ////////////////////////////////////////////////
65  /////     DEFINE LINK threejs GEOMETRIES
66  ////////////////////////////////////////////////
67
68  /*  threejs geometry definition template, will be used by THREE.Mesh() to create threejs object
69      // create threejs geometry and insert into links_geom data object
70      links_geom["link1"] = new THREE.CubeGeometry( 5+2, 2, 2 );
71
72      // example of translating geometry (in object space)
73      links_geom["link1"].applyMatrix( new THREE.Matrix4().makeTranslation(5/2, 0, 0) );
74
75      // example of rotating geometry 45 degrees about y-axis (in object space)
76      var temp3axis = new THREE.Vector3(0,1,0);
77      links_geom["link1"].rotateOnAxis(temp3axis,Math.PI/4);
78  */
79
80  // define threejs geometries and associate with robot links
81  links_geom = {};
82
83  links_geom["link1"] = new THREE.CubeGeometry( 0.7+0.2, 0.5+0.2, 0.2 );
84  links_geom["link1"].applyMatrix( new THREE.Matrix4().makeTranslation((0.5-0.2)/2, 0.5/2, 0) );
85
86  links_geom["link2"] = new THREE.CubeGeometry( 0.5+0.2, 0.2, 0.2 );
87  links_geom["link2"].applyMatrix( new THREE.Matrix4().makeTranslation(0.5/2, 0, 0) );
88
89  links_geom["link3"] = new THREE.CubeGeometry( 0.5+0.2, 0.2, 0.2 );
90  links_geom["link3"].applyMatrix( new THREE.Matrix4().makeTranslation(0.5/2, 0, 0) );
91
92  links_geom["link4"] = new THREE.CubeGeometry( 0.5+0.2, 0.2, 0.2 );
93  links_geom["link4"].applyMatrix( new THREE.Matrix4().makeTranslation(0.5/2, 0, 0) );
```

robot =



Using URDF Data Structure

Get the base link object:

```
robot.links[robot.base]
```

Get `link`'s parent joint's transform:

```
robot.joints[link.parent].xform
```

Get `joint`'s child link:

```
robot.links[joint.child]
```

Get `joint`'s parent link's joint children:

```
robot.links[joint.parent].children
```

Implementation Advice

Be aware of **global variable scope**

- In scope across all included JavaScript files
- Change a global variable in one file, and that change will be reflected for all other files

Be aware of **direction of transform** in `.xform`

- `.xform` represents component frame to world frame transform
- Rotate then translate!

Motivation of Assignment

- Robots exist as a collection of parts within an environment
  - Each part has information like geometry, configuration state, control signal…
  - By definition, this information is independent from other parts and environment
- Collectively, the robot has information relating each part to all other parts
  - Independent of individual component information and environment
- Can acquire knowledge of the environment through sensing
  - Robot’s internal information is a source of prior knowledge about the environment
  - Has information that it exists in a known configuration within environment

Motivation of Assignment

Robots exist as a collection of parts within an environment

Collectively, the robot has information relating each part to all other parts

Can acquire knowledge of the environment through sensing

To accomplish some desired task, our robot should make use of all available knowledge; its actions should be as fully informed as possible

- We need to be able to relate each source of information
- Transform all information into a unified frame of reference = forward kinematics

Lab Takeaways

1. KinEval overview
2. KinEval walkthrough
3. Implementation advice

→ How to start Assignment 3