EECS 367 & ROB 320 Lab

RRT-Connect in Search Canvas and KinEval

Administrative

- Next Wednesday, March 23
  - Due: Assignment #5 Inverse Kinematics
  - Released: Assignment #6 Motion Planning
  - Released: Assignment #7 Best use of robotics
- Lab section Friday, April 1
  - Connect to Fetch via rosbridge (optional)
  - In robotics building

Lab Takeaways

1. Revisit the search canvas
2. KinEval overview
3. KinEval walkthrough

→ How to start Assignment 6

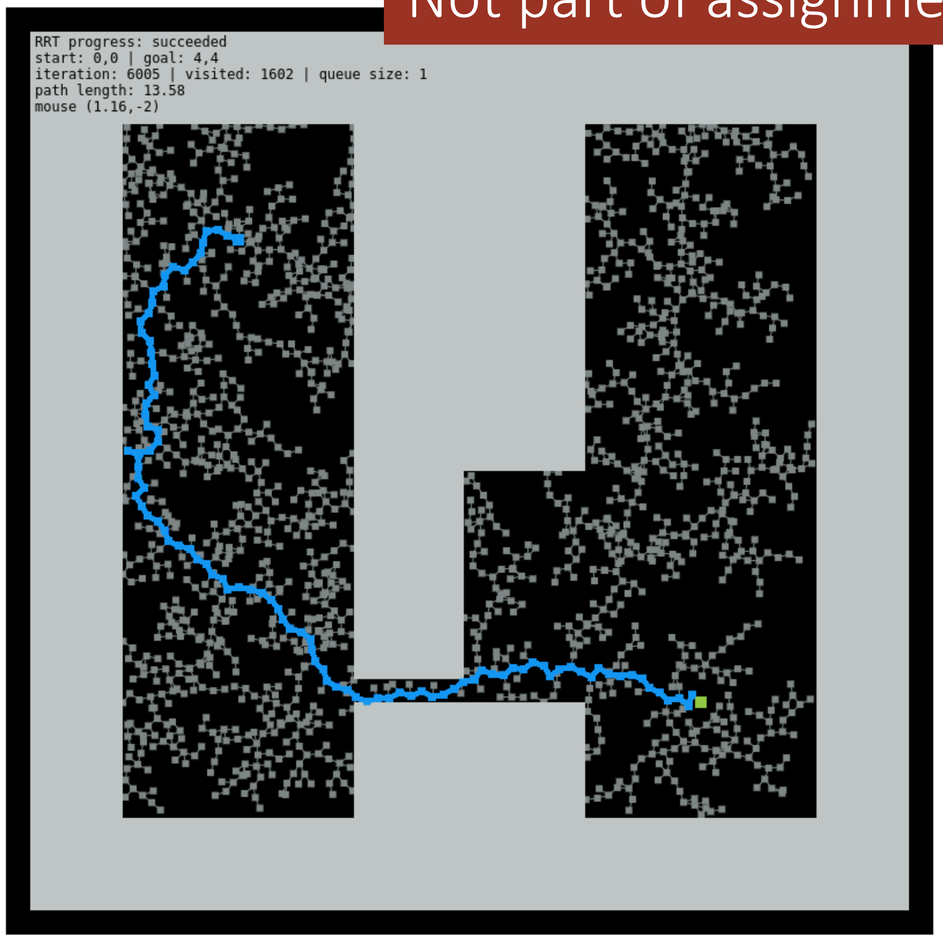
Motion Planning Overview

	Assignment 6: Motion Planning
4	Collision detection
6	2D RRT-Connect
2	Configuration space RRT-Connect

Rapidly-Exploring Random Trees (RRT)

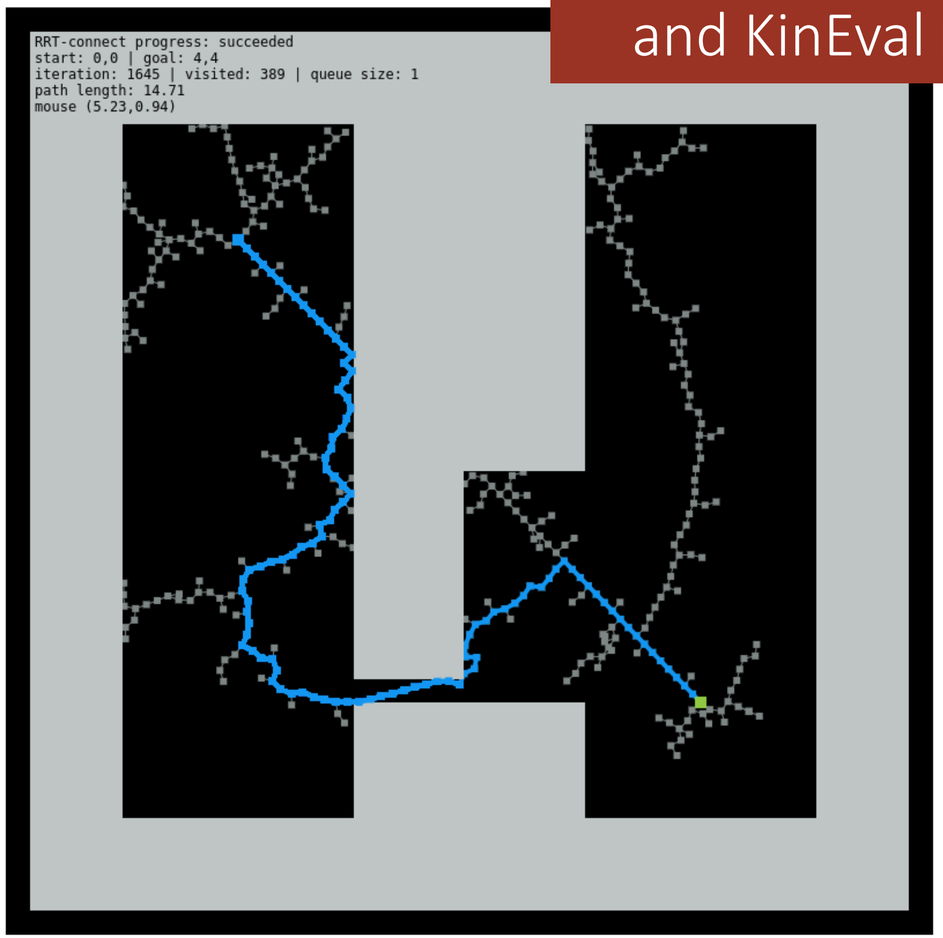
RRT

Not part of assignment

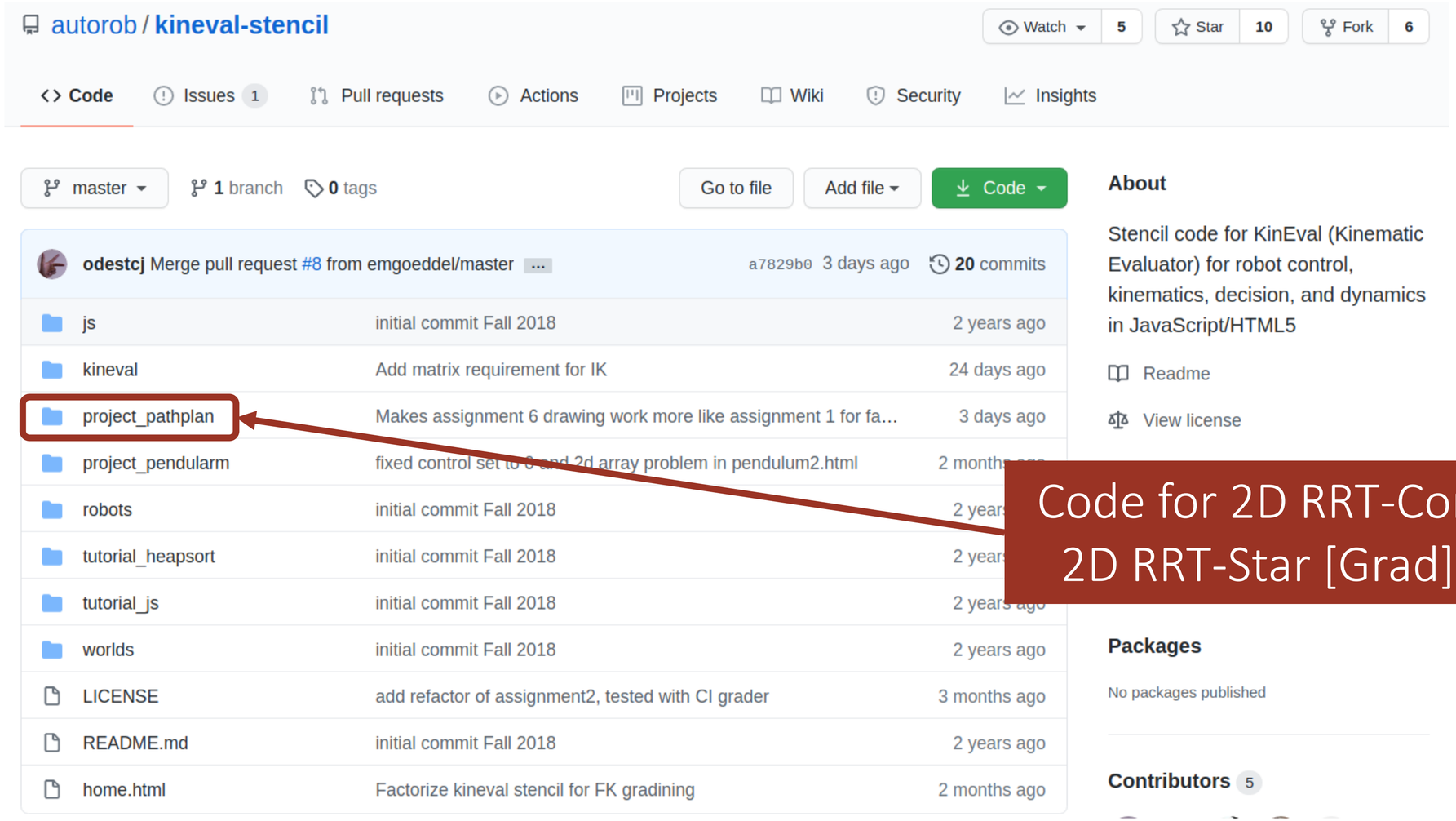


RRT-CONNECT

Will implement for 2D and KinEval robots

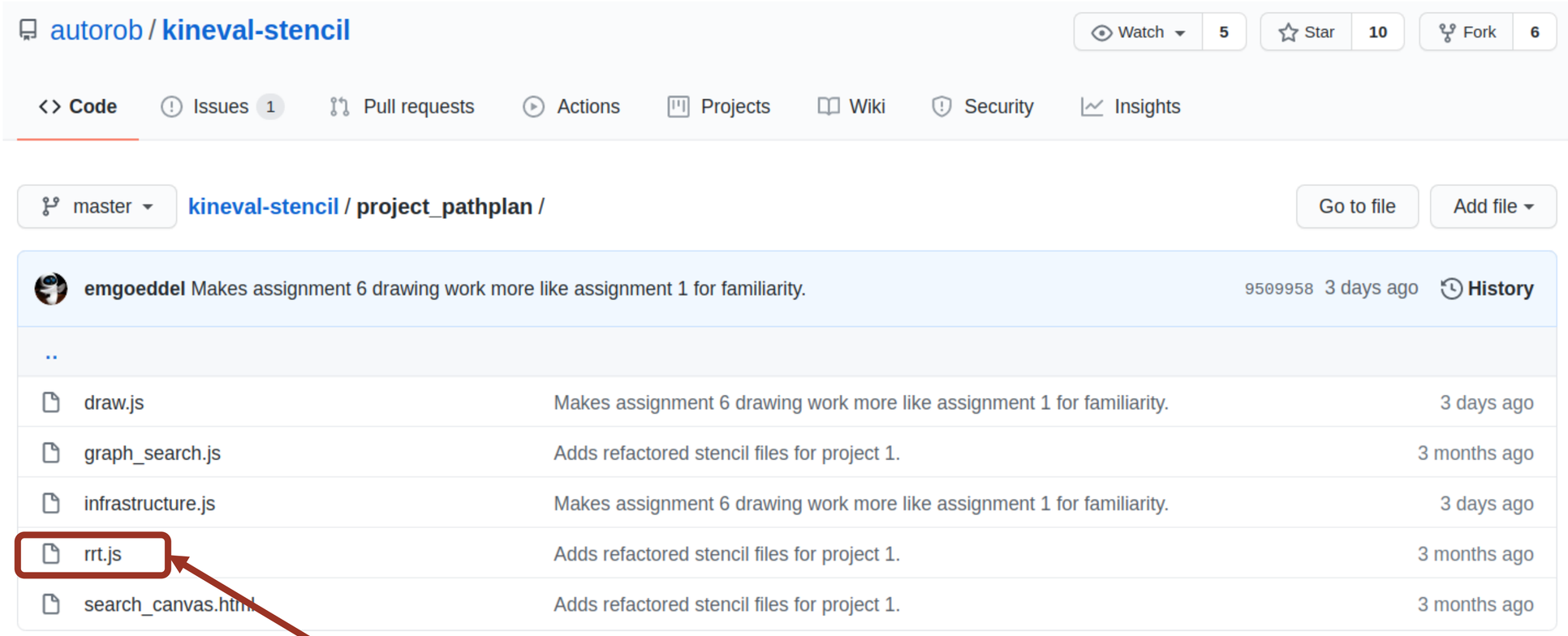


Revisiting the Search Canvas



Code for 2D RRT-Connect and 2D RRT-Star [Grad] features

Revisiting the Search Canvas



Code for 2D RRT-Connect and 2D RRT-Star [Grad] features

2D RRT-Connect

Recommended: Start project with 2D RRT-Connect in project_pathplan/rrt.js

rrt.js

```
45  function iterateRRTConnect() {
46
47
48      // STENCIL: implement a single iteration of an RRT-Connect algorithm.
49      //   An asynch timing mechanism is used instead of a for loop to avoid
50      //   blocking and non-responsiveness in the browser.
51      //
52      //   Return "failed" if the search fails on this iteration.
53      //   Return "succeeded" if the search succeeds on this iteration.
54      //   Return "extended" otherwise.
55      //
56      //   Provided support functions:
57      //
58      //   testCollision - returns whether a given configuration is in collision
59      //   insertTreeVertex - adds and displays new configuration vertex for a tree
60      //   insertTreeEdge - adds and displays new tree edge between configurations
61      //   drawHighlightedPath - renders a highlighted path in a tree
62  }
```

2D RRT-Connect

Similar to Assignment 1 search algorithms, implement as a single step within the iterative algorithm

rrt.js

```
45  function iterateRRTConnect() {
46
47
48      // STENCIL: implement a single iteration of an RRT-Connect algorithm.
49      //   An asynch timing mechanism is used instead of a for loop to avoid
50      //   blocking and non-responsiveness in the browser.
51      //
52      //   Return "failed" if the search fails on this iteration.
53      //   Return "succeeded" if the search succeeds on this iteration.
54      //   Return "extended" otherwise.
55      //
56      //   Provided support functions:
57      //
58      //   testCollision - returns whether a given configuration is in collision
59      //   insertTreeVertex - adds and displays new configuration vertex for a tree
60      //   insertTreeEdge - adds and displays new tree edge between configurations
61      //   drawHighlightedPath - renders a highlighted path in a tree
62  }
```

RRT Data Structure

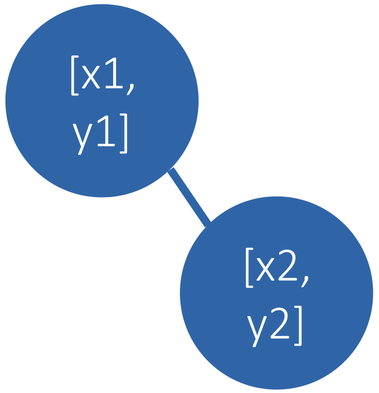
infrastructure.js

```
64  function initRRT(q) {
65
66      // create tree object
67      var tree = {};
68
69      // initialize with vertex for given configuration
70      tree.vertices = [];
71      tree.vertices[0] = {};
72      tree.vertices[0].vertex = q;
73      tree.vertices[0].edges = [];
74
75      // maintain index of newest vertex added to tree
76      tree.newest = 0;
77
78      return tree;
79  }
80
81  function insertTreeVertex(tree,q) {
82
```

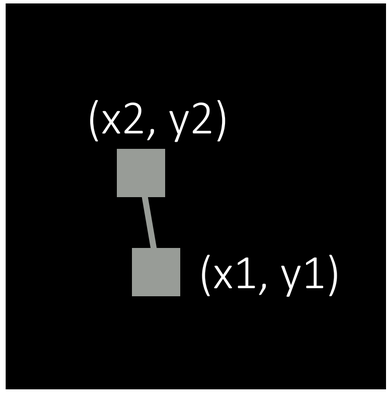
Tree implemented as a JavaScript object with array of vertices

```
tree = {"vertices":
            [
             {"vertex":[x1,y1],
              "edges":[tree.vertices[1]]},
             {"vertex":[x2,y2],
              "edges":[tree.vertices[0]]}
            ],
        "newest":1}
```

Data structure view



Search canvas view



RRT Functions

infrastructure.js

```
64  function initRRT(q) {
65
66      // create tree object
67      var tree = {};
68
69      // initialize with vertex for given configuration
70      tree.vertices = [];
71      tree.vertices[0] = {};
72      tree.vertices[0].vertex = q;
73      tree.vertices[0].edges = [];
74
75      // maintain index of newest vertex added to tree
76      tree.newest = 0;
77
78      return tree;
79  }
80
81  function insertTreeVertex(tree,q) {
82
```

Helper functions available for basic tree operations

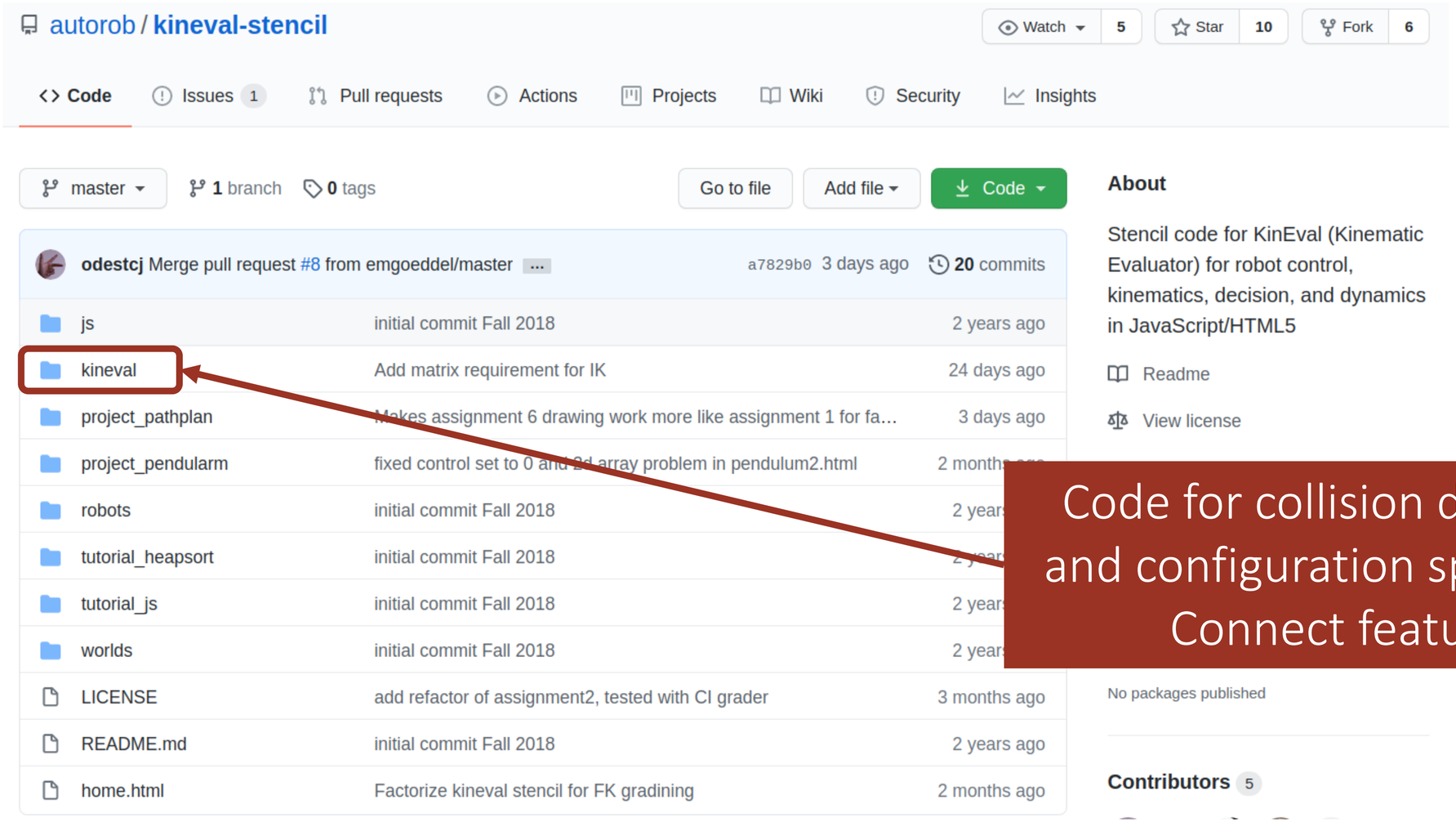
rrt.js

```
68  ////////////////////////////////////////////////
69  /////     RRT IMPLEMENTATION FUNCTIONS
70  ////////////////////////////////////////////////
71
72      // STENCIL: implement RRT-Connect functions here, such as:
73      //   extendRRT
74      //   connectRRT
75      //   randomConfig
76      //   newConfig
77      //   findNearestNeighbor
78      //   dfsPath
```

Suggested functions for you to add for RRT-Connect implementation

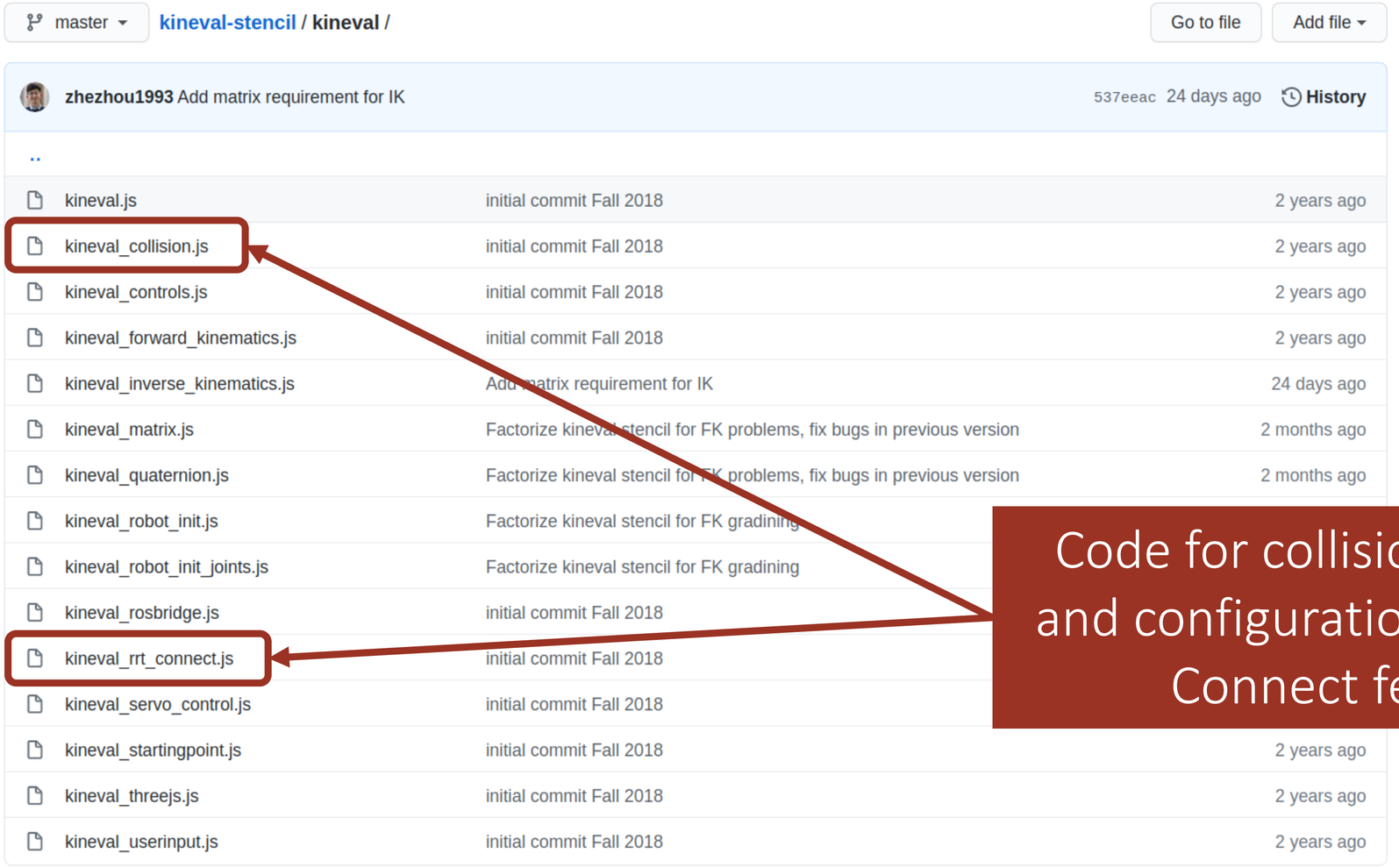
Demo

KinEval Overview



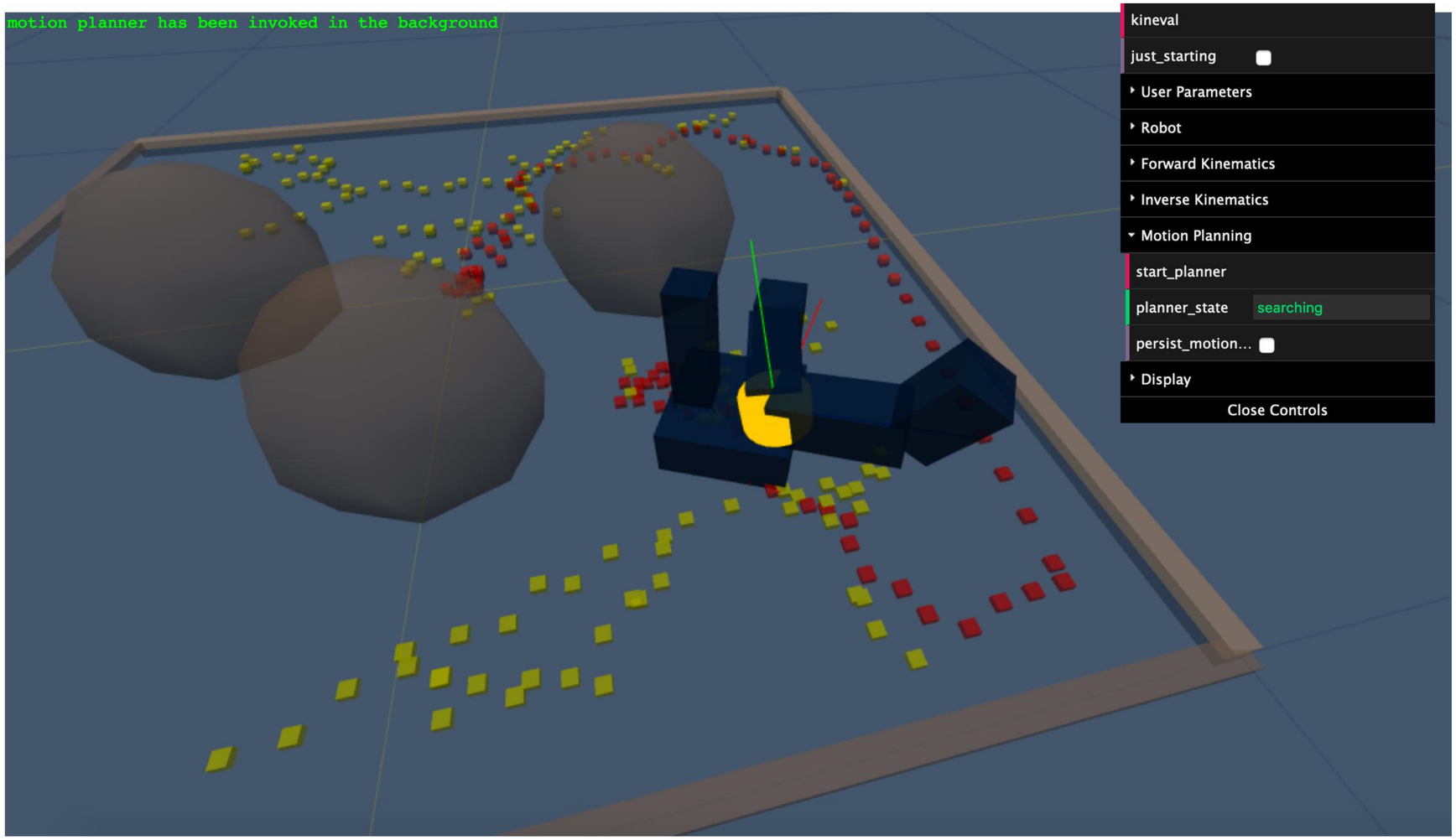
Code for collision detection and configuration space RRT-Connect features

KinEval Overview



Code for collision detection and configuration space RRT-Connect features

Configuration Space RRT



kineval_collision.js

kineval_collision.js

```
22  kineval.robotIsCollision = function robot_iscollision() {
23      // test whether geometry of current configuration of robot is in collision with planning world
24
25      // form configuration from base location and joint angles
26      var q_robot_config = [
27          robot.origin.xyz[0],
28          robot.origin.xyz[1],
29          robot.origin.xyz[2],
30          robot.origin.rpy[0],
31          robot.origin.rpy[1],
32          robot.origin.rpy[2]
33      ];
34
35      q_names = {};  // store mapping between joint names and q DOFs
36
37      for (x in robot.joints) {
38          q_names[x] = q_robot_config.length;
39          q_robot_config = q_robot_config.concat(robot.joints[x].angle);
40      }
41
42      // test for collision and change base color based on the result
43      collision_result = kineval.poseIsCollision(q_robot_config);
44
45      robot.collision = collision_result;
46  }
```

`q_robot_config` is an array representing current pose as point within configuration space

Dimension of configuration space is a function of the specific robot!

kineval_collision.js

kineval_collision.js

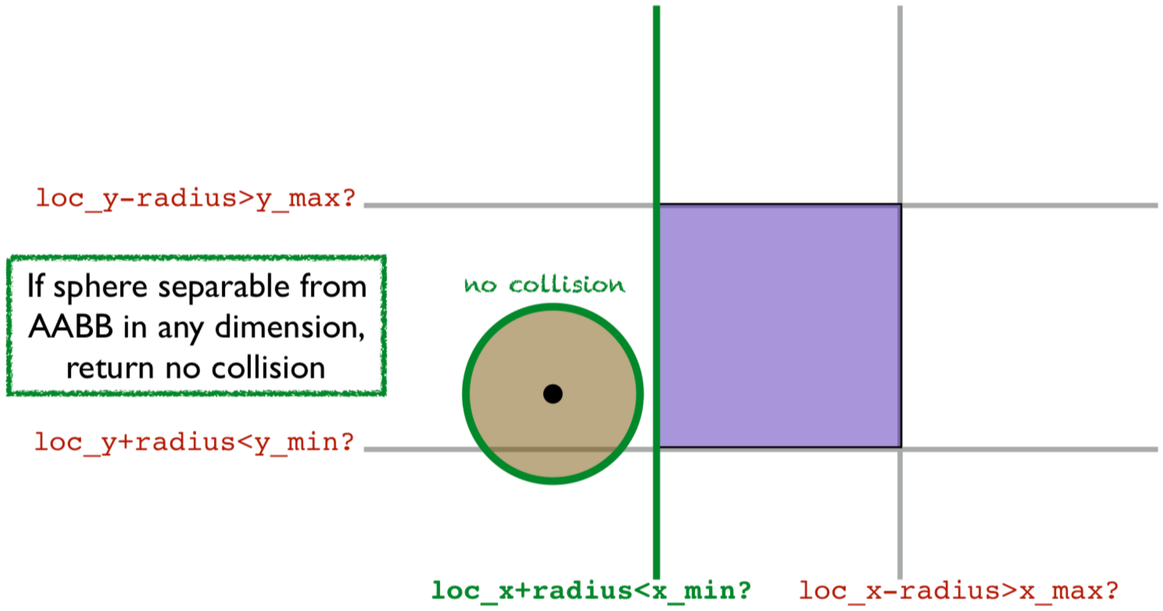
```
49  kineval.poseIsCollision = function robot_collision_test(q) {
50      // perform collision test of robot geometry against planning world
51
52      // test base origin (not extents) against world boundary extents
53      if ((q[0]<robot_boundary[0][0])||(q[0]>robot_boundary[1][0])||(q[2]<robot_boundary[0][2])||(q[2]>robot_boundary[1][2]))
54          return robot.base;
55
56      // traverse robot kinematics to test each body for collision
57      // STENCIL: implement forward kinematics for collision detection
58      //return robot_collision_forward_kinematics(q);
59
60  }
```

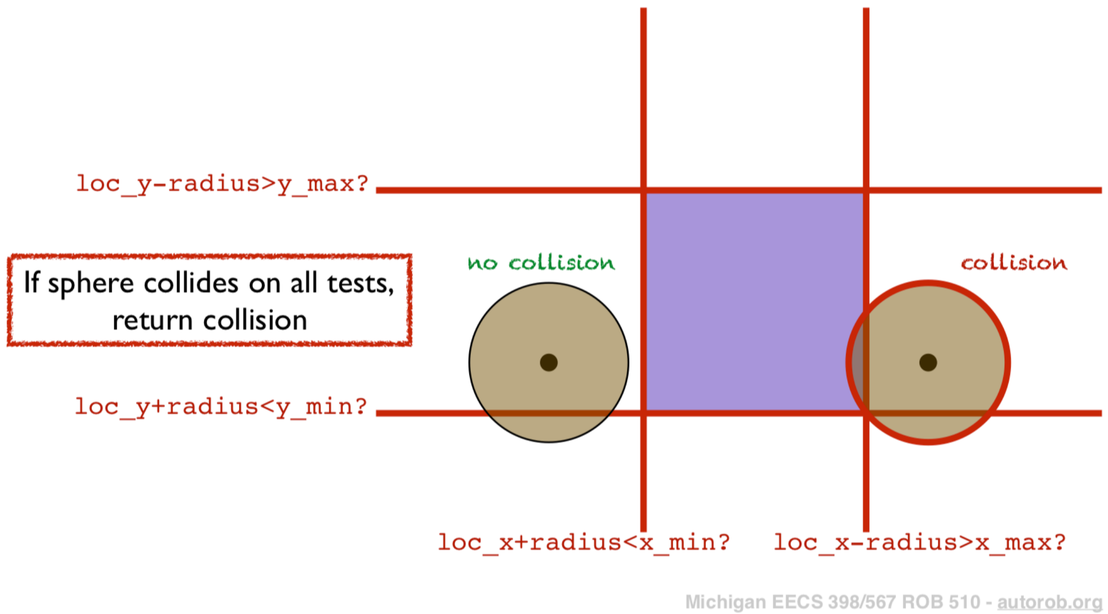
STENCIL: Check each link for collision with spherical obstacles

Collision detection pseudocode:

For each link in robot
    For each obstacle in world
        If intersection(link, obstacle)
            Return link is in collision
Return no collision

AABB Link Collision Detection





kineval_collision.js

kineval_collision.js

```
64  function traverse_collision_forward_kinematics_link(link,mstack,q) {
65
66      /* test collision FK
67      console.log(link);
68      */
69      if (typeof link.visual !== 'undefined') {
70          var local_link_xform = matrix_multiply(mstack,generate_translation_matrix(link.visual.origin.xyz[0],link
71      }
72      else {
73          var local_link_xform = matrix_multiply(mstack,generate_identity());
74      }
75
76      // test collision by transforming obstacles in world to link space
77  /*
78      mstack_inv = matrix_invert_affine(mstack);
79  */
80      mstack_inv = numeric.inv(mstack);
81
82      var i;
83      var j;
84
85      // test each obstacle against link bbox geometry by transforming obstacle into link frame and testing against
86      //for (j=0;j<robot_obstacles.length;j++) {
87      for (j in robot_obstacles) {
88
89          var obstacle_local = matrix_multiply(mstack_inv,robot_obstacles[j].location);
90
91          // assume link is in collision as default
92          var in_collision = true;
```

Need to compute `mstack` for use here

AABB collision check for a link provided for you in this function, but you need to add the rest of FK traversal

kineval_rrt.js

Implement `robot_rrt_planner_iterate()` as a single iteration of the RRT-Connect planning algorithm

kineval_rrt.js

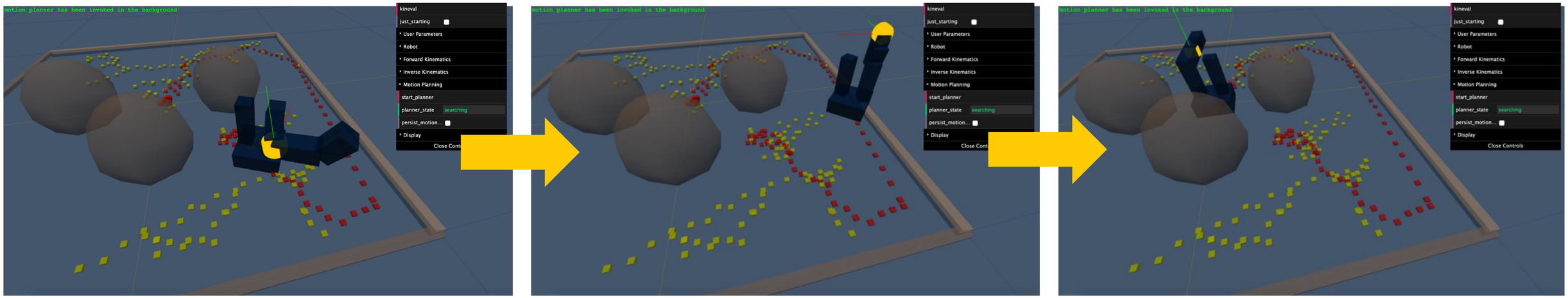
```
132  function robot_rrt_planner_iterate() {
133
134      var i;
135      rrt_alg = 1;  // 0: basic rrt (OPTIONAL), 1: rrt_connect (REQUIRED)
136
137      if (rrt_iterate && (Date.now()-cur_time > 10)) {
138          cur_time = Date.now();
139
140      // STENCIL: implement single rrt iteration here. an asynch timing mechanism
141      //   is used instead of a for loop to avoid blocking and non-responsiveness
142      //   in the browser.
143      //
144      //   once plan is found, highlight vertices of found path by:
145      //     tree.vertices[i].vertex[j].geom.material.color = {r:1,g:0,b:0};
146      //
147      //   provided support functions:
148      //
149      //   kineval.poseIsCollision - returns if a configuration is in collision
150      //   tree_init - creates a tree of configurations
151      //   tree_add_vertex - adds and displays new configuration vertex for a tree
152      //   tree_add_edge - adds and displays new tree edge between configurations
153      }
154
155
```

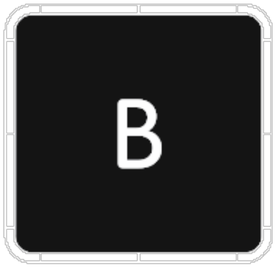
Include any helper functions in this file

Desired Result

Arbitrary initial configuration

Collision free path to home





Backward step along motion plan

Forward step along motion plan

Demo

Final Tips

CI grader is a rough guideline, so you must do your own testing to verify correct implementation

Make sure you use the robot's **full configuration space**, including all joint DOFs, and not just base movement in the ground plane

Do not move the robot outside of the X-Z plane

- No translation along Y axis
- No rotation around X or Z axes