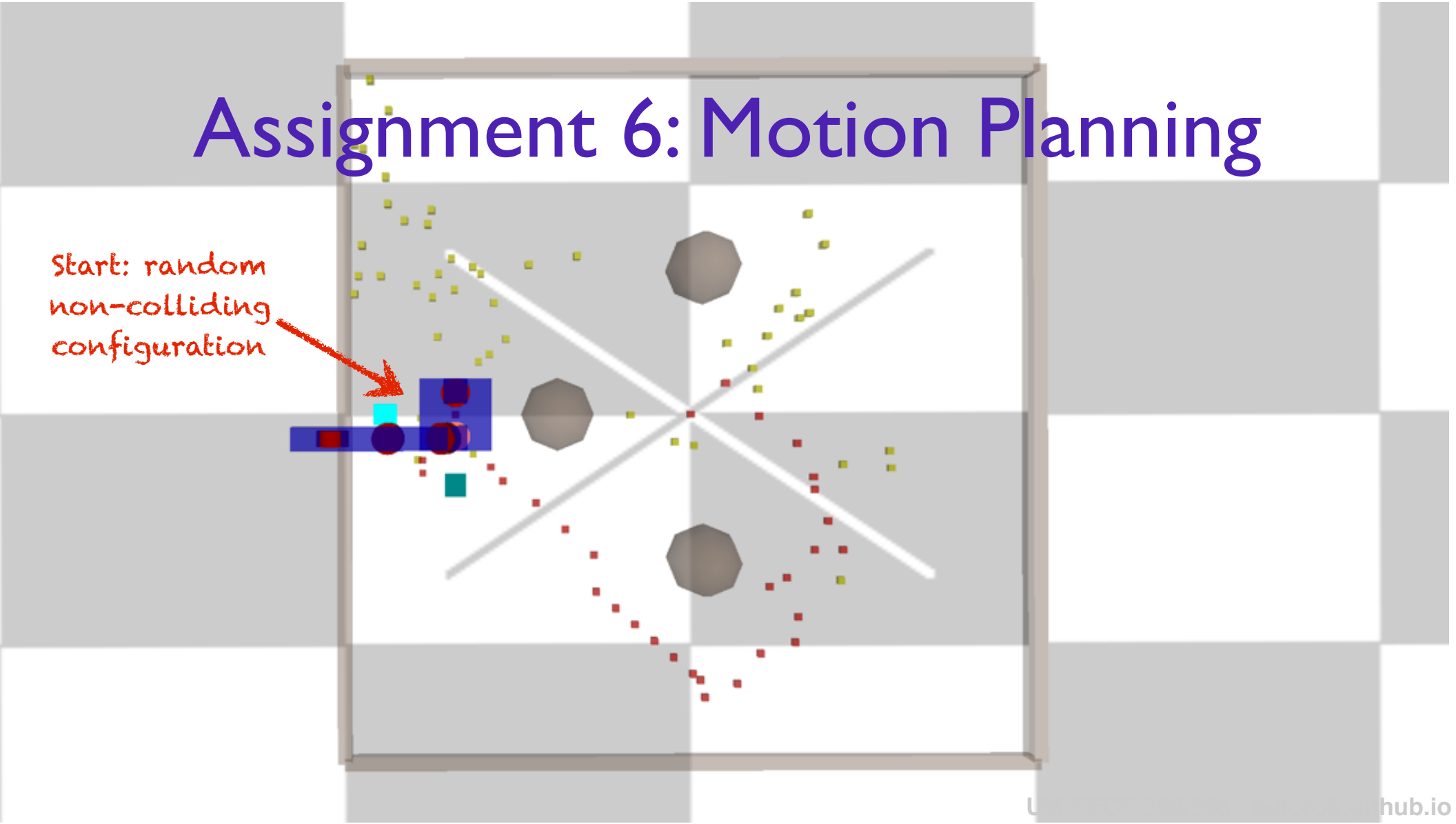

U.S. Department of Transportation
https://www.youtube.com/watch?v=B8Ct5hjs0Jw
Collision Detection
autorob.github.io

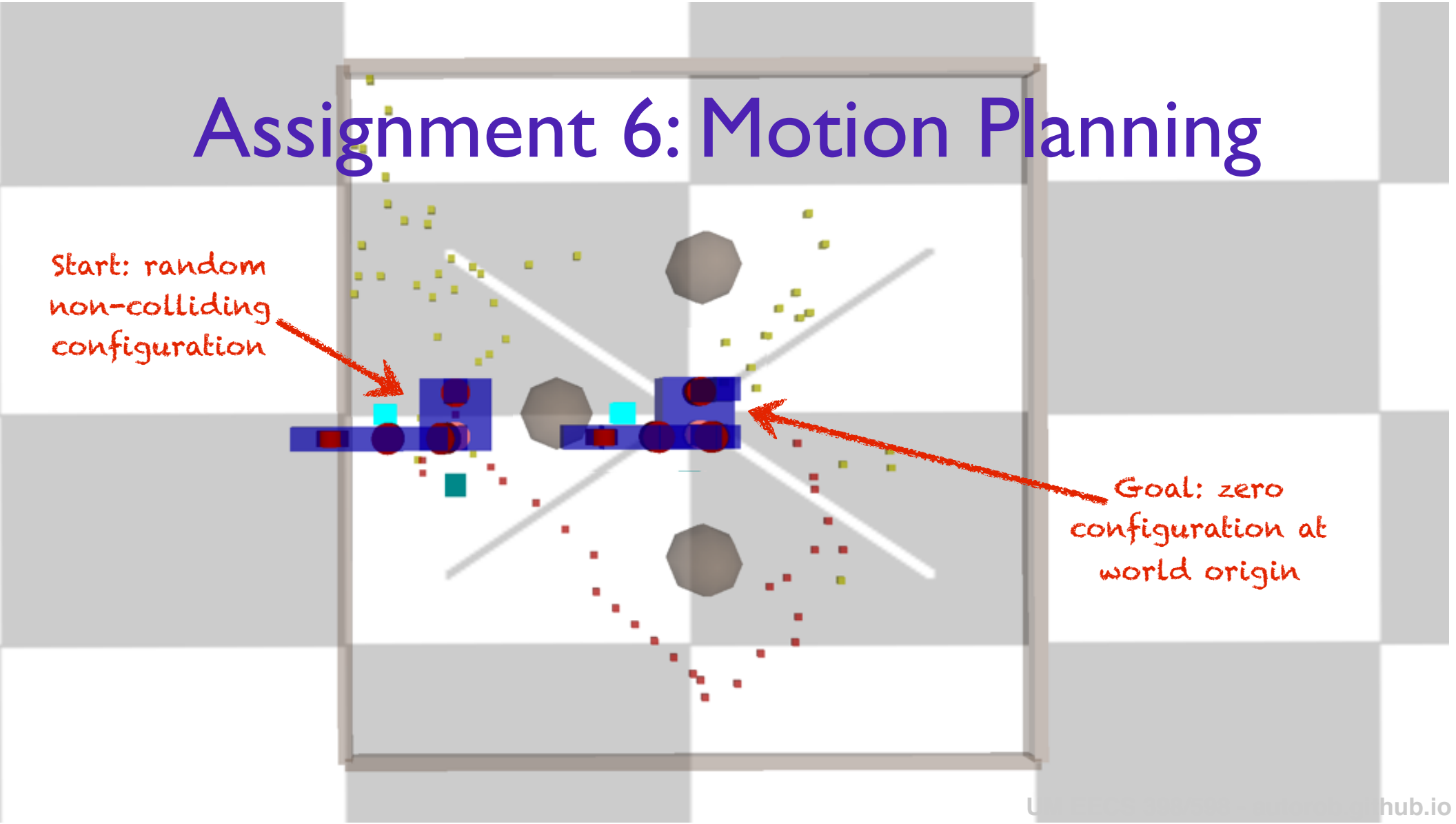
Assignment 6: Motion Planning

- Generate a collision free motion plan to the world origin and zero joint angle configuration

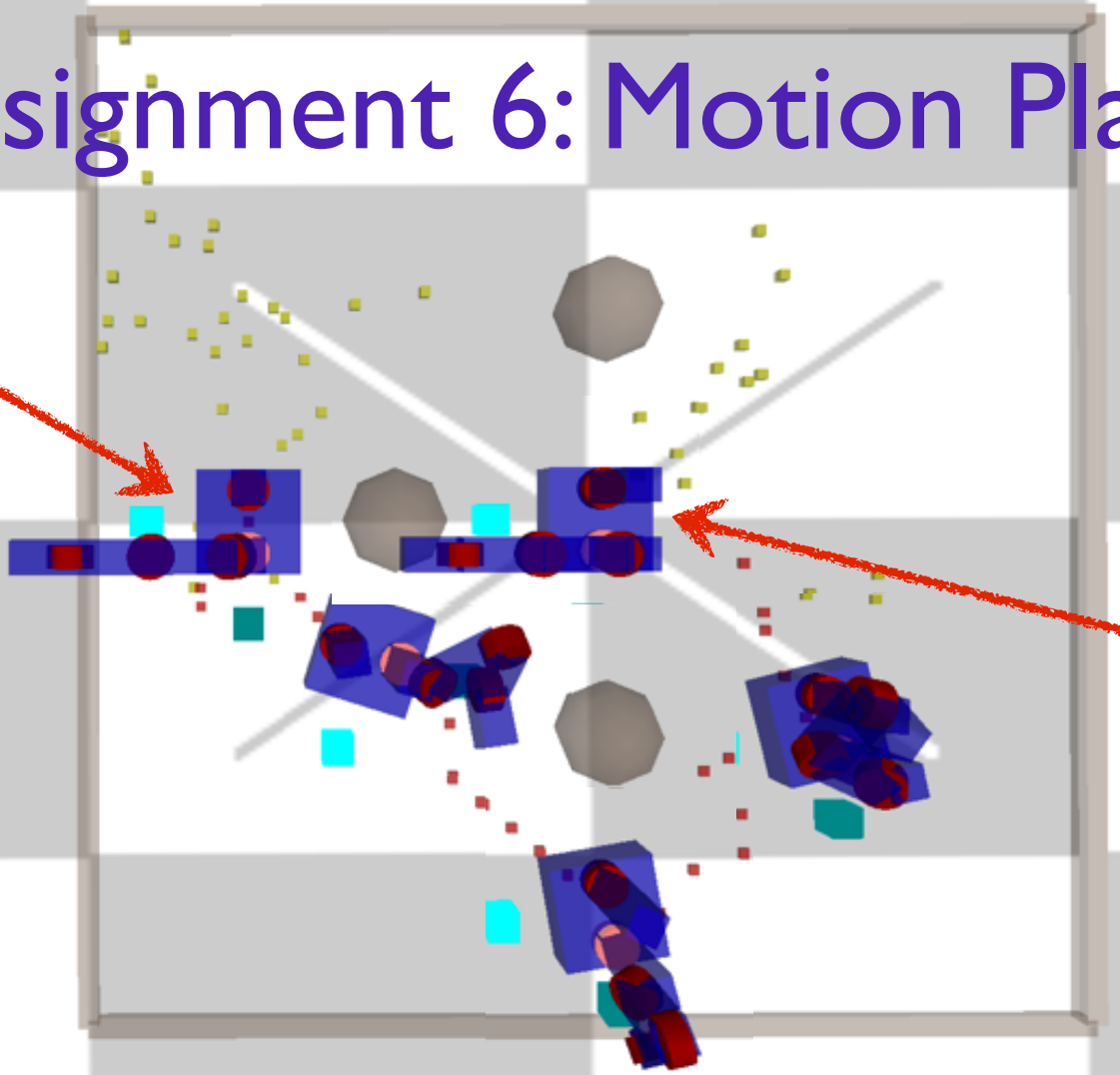
Assignment 6: Motion Planning



Assignment 6: Motion Planning



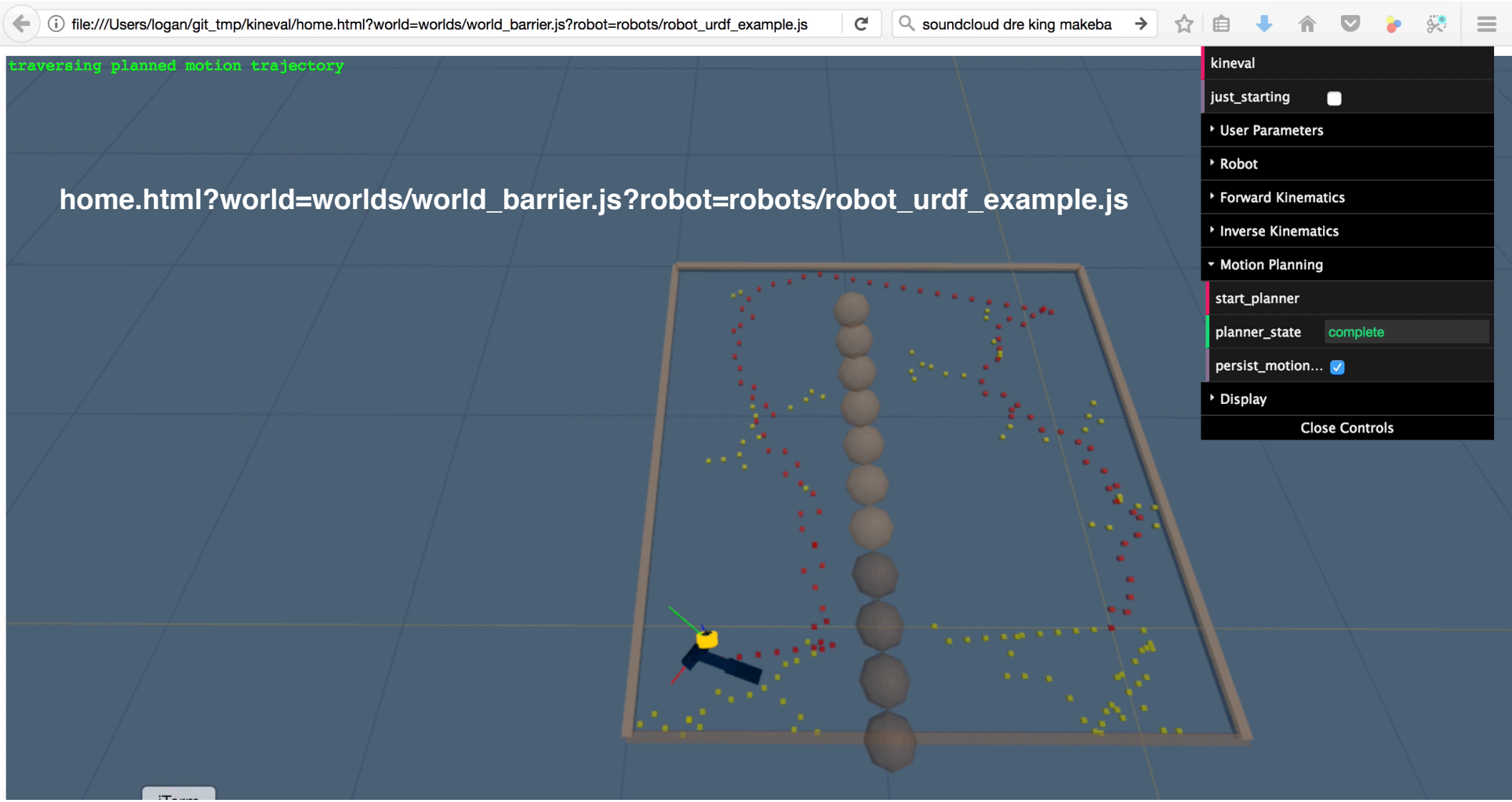
Assignment 6: Motion Planning

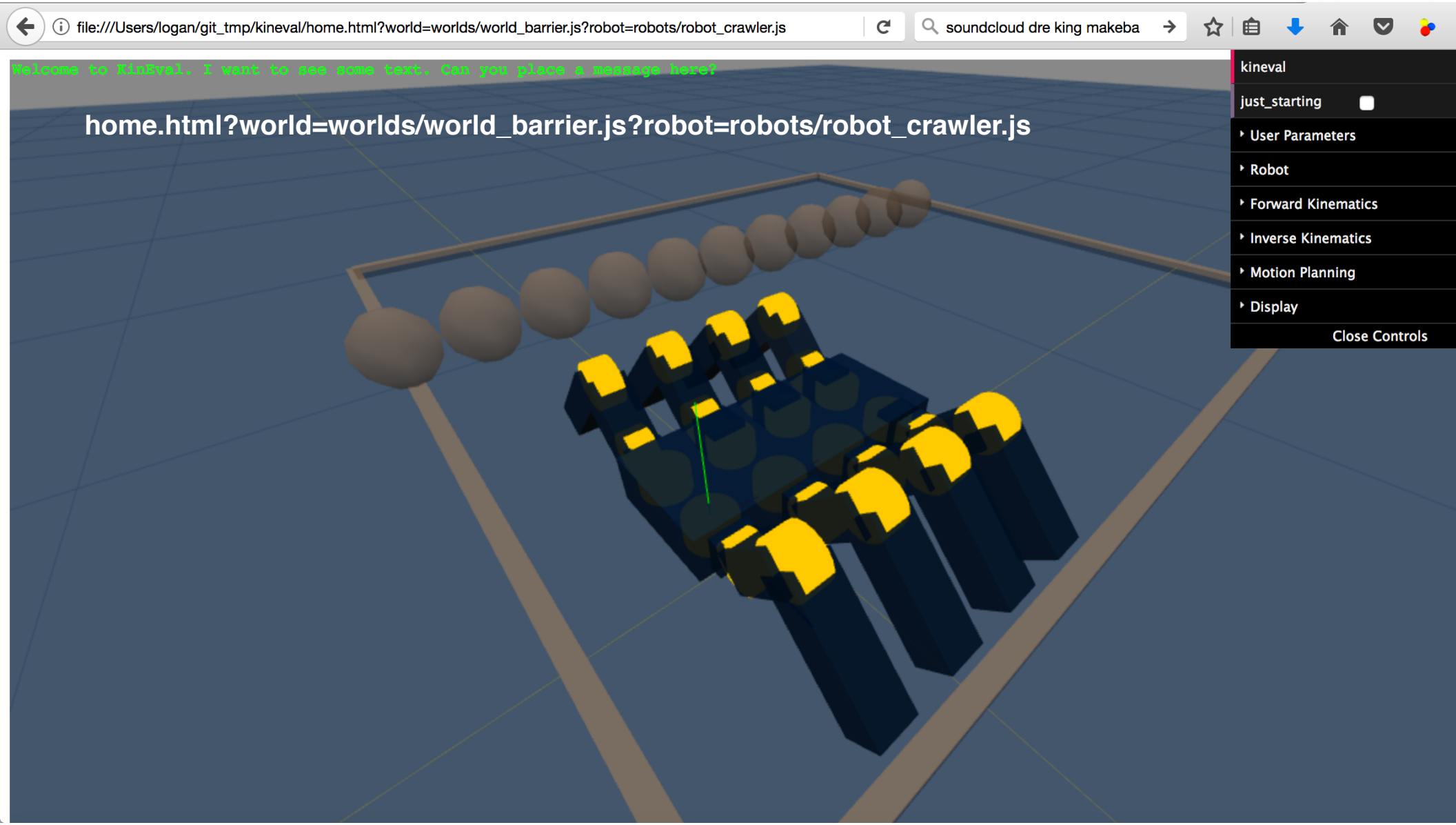


Start: random non-colliding configuration

Goal: zero configuration at world origin

Generate collision-free motion plan


file:///Users/logan/git_tmp/kineval/home.html?world=worlds/world_barrier.js?robot=robots/robot_urdf_example.js
soundcloud dre king makeba
traversing planned motion trajectory
home.html?world=worlds/world_barrier.js?robot=robots/robot_urdf_example.js
kineval
just_starting
User Parameters
Robot
Forward Kinematics
Inverse Kinematics
Motion Planning
start_planner
planner_state
complete
persist_motion...
Display
Close Controls


file:///Users/logan/git_tmp/kineval/home.html?world=worlds/world_barrier.js?robot=robots/robot_crawler.js
soundcloud dre king makeba
Welcome to KinEval. I want to see some text. Can you place a message here?
home.html?world=worlds/world_barrier.js?robot=robots/robot_crawler.js
kineval
just_starting
User Parameters
Robot
Forward Kinematics
Inverse Kinematics
Motion Planning
Display
Close Controls

GitHub, Inc. (US) https://github.com/ohseejay/kineval-stencil-fall16 rrt connect

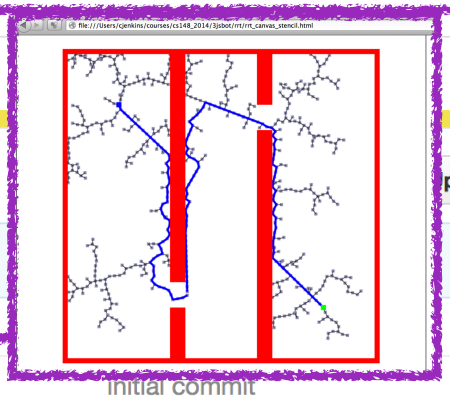
ohseejay / **kineval-stencil-fall16**

Watch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0

Stencil code for KinEval (Kinematic Evaluator) for robo

Recommended starting point:
Update code stencil from Path Planning project



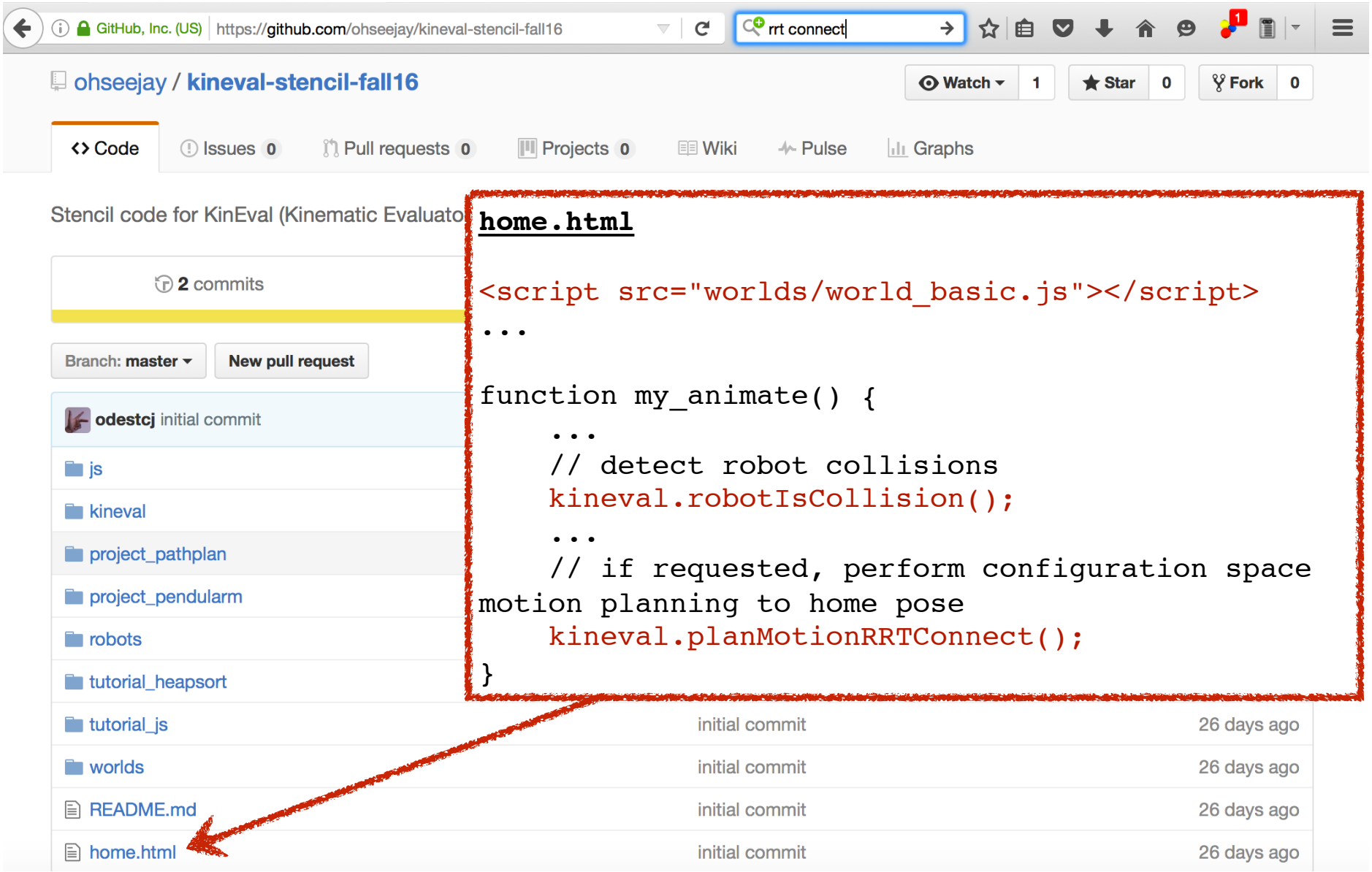
2 commits 1 branch 2 contributors

Branch: master New pull request pload files Find file Clone or download

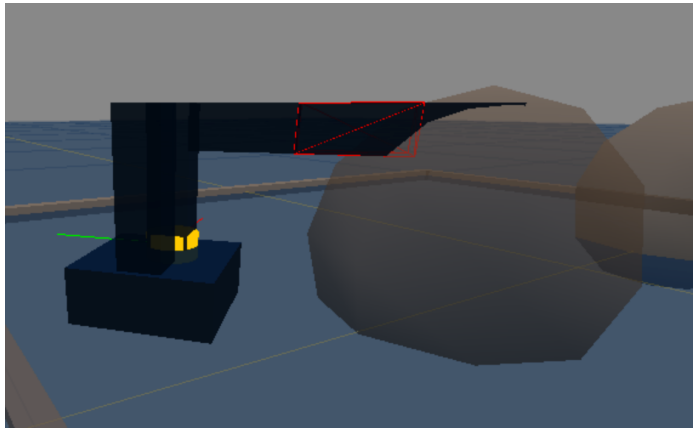
odestcj initial commit Latest commit 5fd521e 26 days ago

js		26 days ago
kineval	initial commit	26 days ago
project_pathplan	initial commit	26 days ago
project_pendularm		
robots		
tutorial_heapsort		
tutorial_js	initial commit	26 days ago
worlds	initial commit	26 days ago
README.md	initial commit	26 days ago
home.html	initial commit	26 days ago

various 3D worlds for testing robots included by:
home.html?world=worlds/world_local_minima.js


GitHub, Inc. (US) https://github.com/ohseejay/kineval-stencil-fall16
rrt connect
ohseejay / kineval-stencil-fall16
Watch 1
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Pulse
Graphs
Stencil code for KinEval (Kinematic Evaluato
2 commits
Branch: master
New pull request
odestcj initial commit
js
kineval
project_pathplan
project_pendularm
robots
tutorial_heapsort
tutorial_js initial commit 26 days ago
worlds initial commit 26 days ago
README.md initial commit 26 days ago
home.html initial commit 26 days ago
home.html
<script src="worlds/world_basic.js"></script>
...
function my_animate() {
...
// detect robot collisions
kineval.robotIsCollision();
...
// if requested, perform configuration space motion planning to home pose
kineval.planMotionRRTConnect();
}

world file can be alternatively loaded by a script tag (avoid doing this)

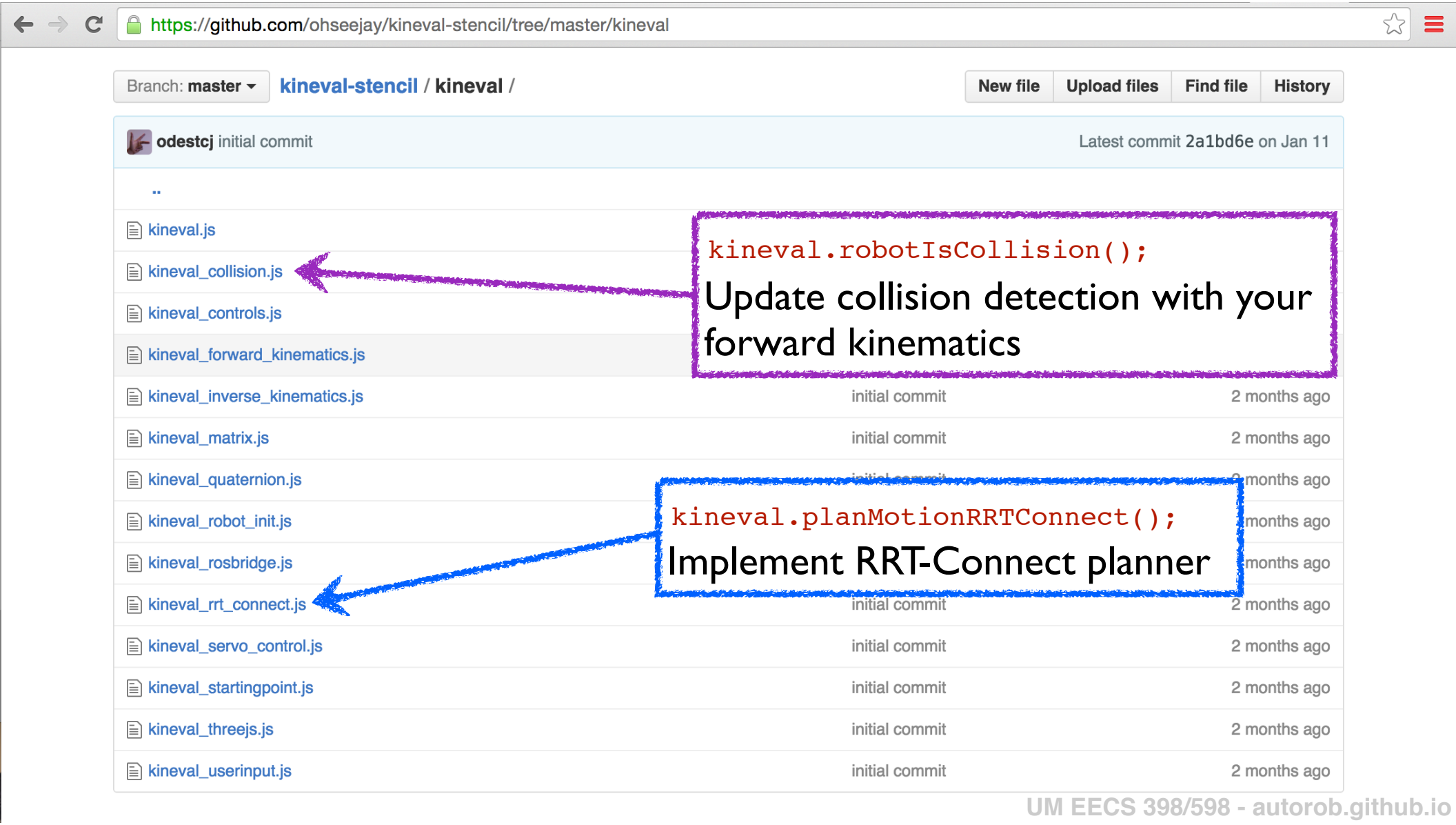


home.html

```
<script src="worlds/world_basic.js"></script>
...

function my_animate() {
    ...
    // detect robot collisions
    kineval.robotIsCollision();
    ...
    // if requested, perform configuration space
motion planning to home pose
    kineval.planMotionRRTConnect();

}
```

detect if current configuration is in collision (colliding link turns red)

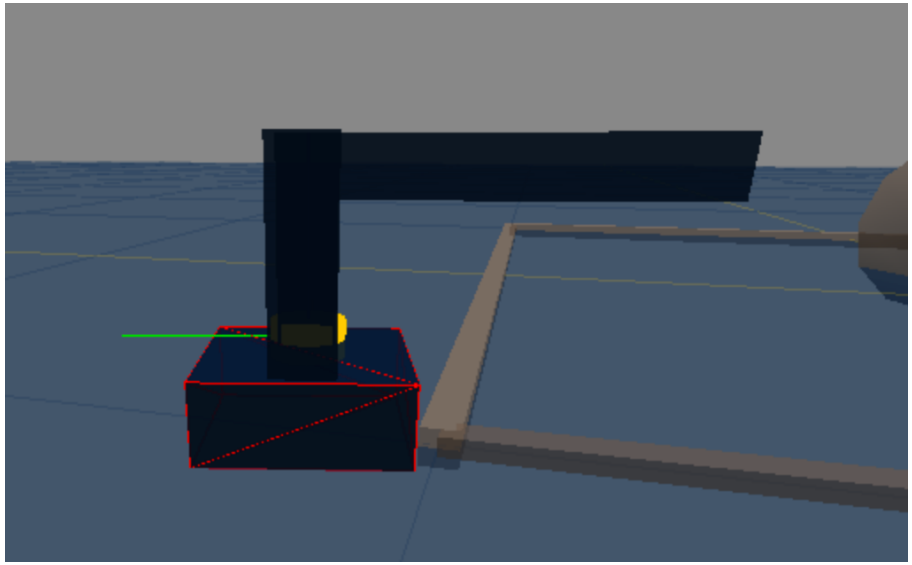
iterate motion planner


https://github.com/ohseejay/kineval-stencil/tree/master/kineval
Branch: master
kineval-stencil / kineval /
New file
Upload files
Find file
History
odestcj initial commit
Latest commit 2a1bd6e on Jan 11
..
kineval.js
kineval_collision.js
kineval_controls.js
kineval_forward_kinematics.js
kineval_inverse_kinematics.js initial commit 2 months ago
kineval_matrix.js initial commit 2 months ago
kineval_quaternion.js
kineval_robot_init.js
kineval_rosbridge.js
kineval_rrt_connect.js initial commit 2 months ago
kineval_servo_control.js initial commit 2 months ago
kineval_startingpoint.js initial commit 2 months ago
kineval_threejs.js initial commit 2 months ago
kineval_userinput.js initial commit 2 months ago
kineval.robotIsCollision();
Update collision detection with your forward kinematics
kineval.planMotionRRTConnect();
Implement RRT-Connect planner

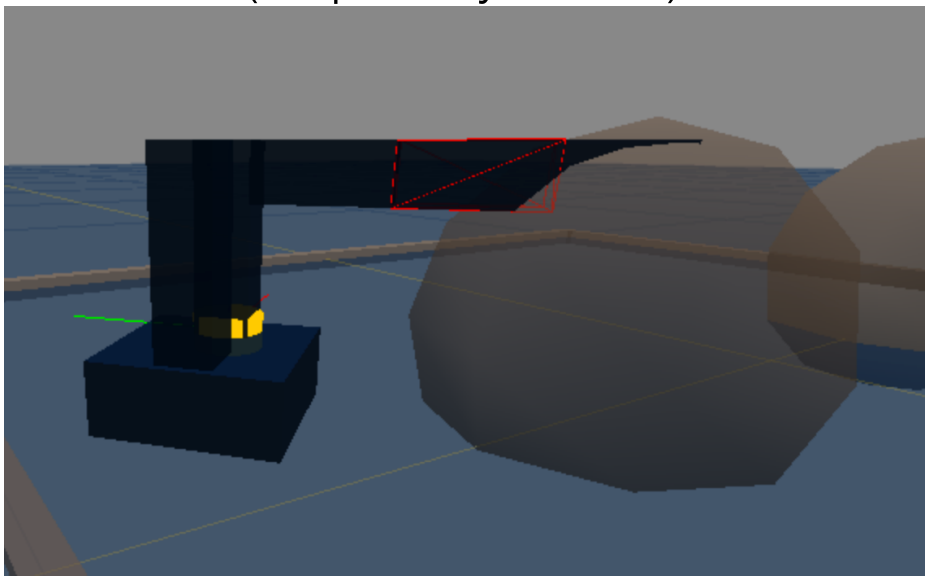
```
kineval.robotIsCollision();
```

Assignment 6 collision detection

Boundary Collision
(provided by default)



Link Collision
(requires your FK)



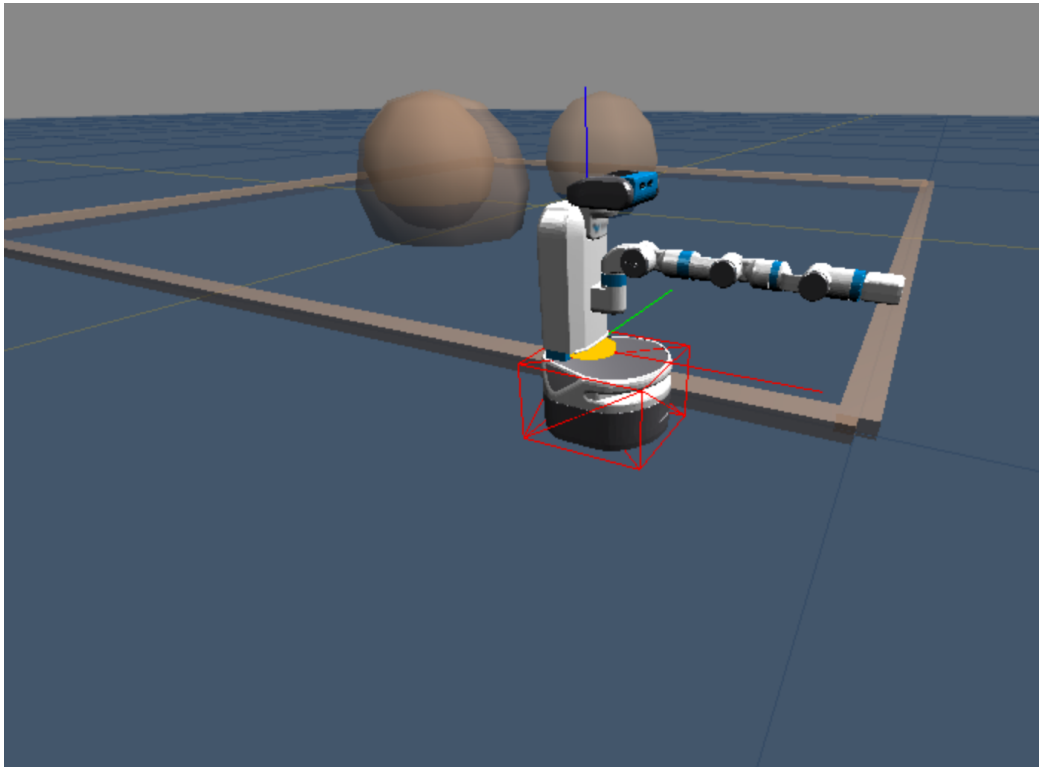
input: q (robot configuration)
output: false (for no collision) or name of link in collision

kineval_collision.js

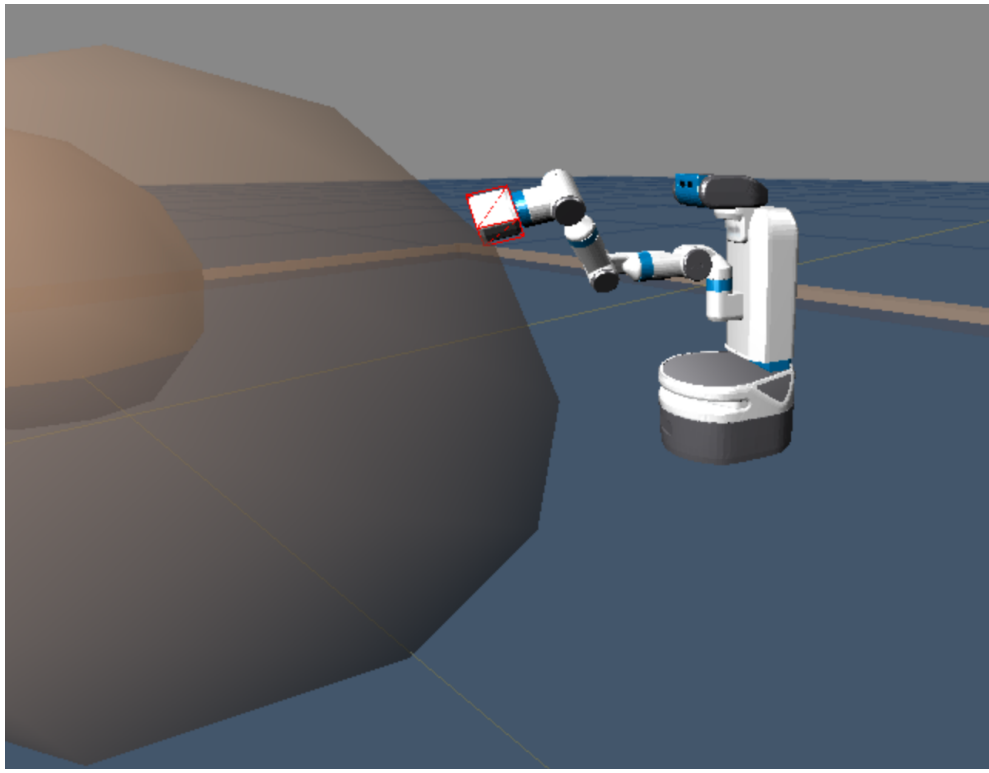
```
kineval.poseIsCollision = function robot_collision_test(q) {
    // perform collision test of robot geometry against planning world

    // test base origin (not extents) against world boundary extents
    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]))
        return robot.base;

    // traverse robot kinematics to test each body for collision
    // STENCIL: implement forward kinematics for collision detection
    return robot_collision_forward_kinematics(q);

}
```

world boundary detection is provided

Uncomment this call;
Implement this function with FK transforms to test for collisions;
Use provided link collision function to test bounding box of each link



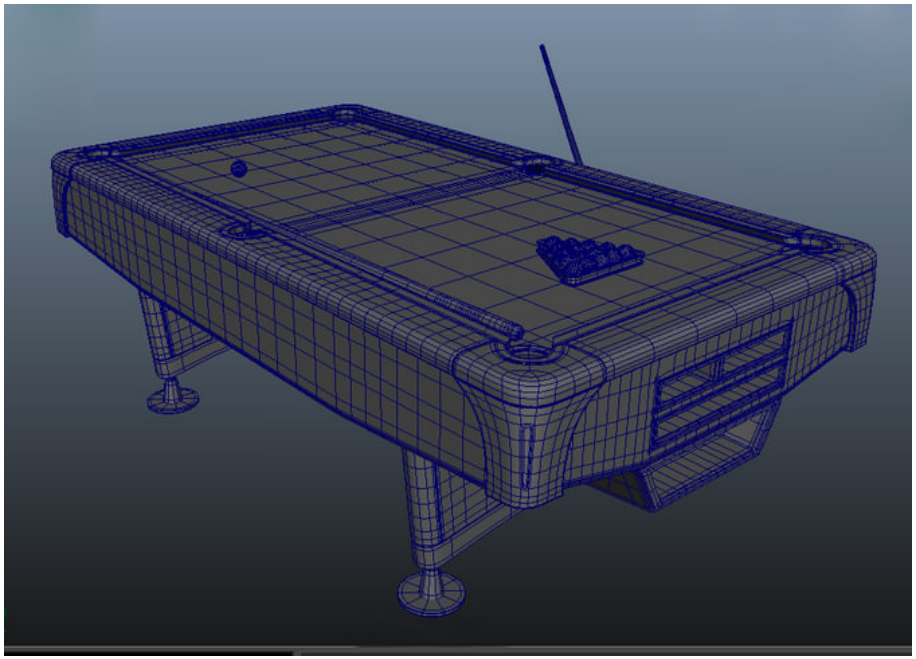
```
// test base origin (not extents) against world boundary extents
if ((q[0]<robot_boundary[0][0])||(q[0]>robot_boundary[1][0])||
    (q[2]<robot_boundary[0][2])||(q[2]>robot_boundary[1][2]))
    return robot.base;
```



```
// traverse robot kinematics to test each body for collision
// STENCIL: implement forward kinematics for collision detection
return robot_collision_forward_kinematics(q);
```

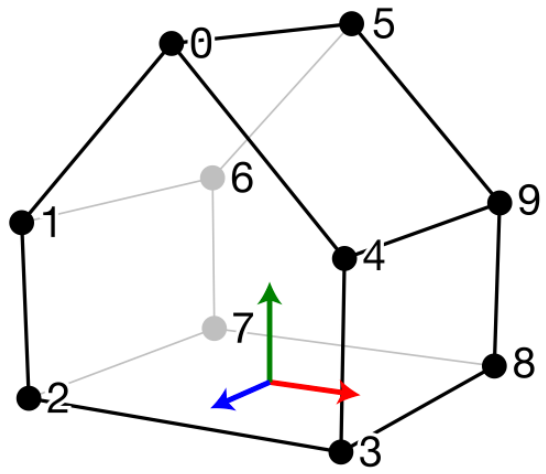
What item is not real
in this picture?

Link geometries are represented as triangular geometric meshes



Remember:

Link Geometry



i	x	y	z
0	0.0	1.0	0.5
1	–0.5	0.5	0.5
2	–0.5	0.0	0.5
3	0.5	0.0	0.5
4	0.5	0.5	0.5
5	0.0	1.0	–0.5
6	–0.5	0.5	–0.5
7	–0.5	0.0	–0.5
8	0.5	0.0	–0.5
9	0.5	0.5	–0.5

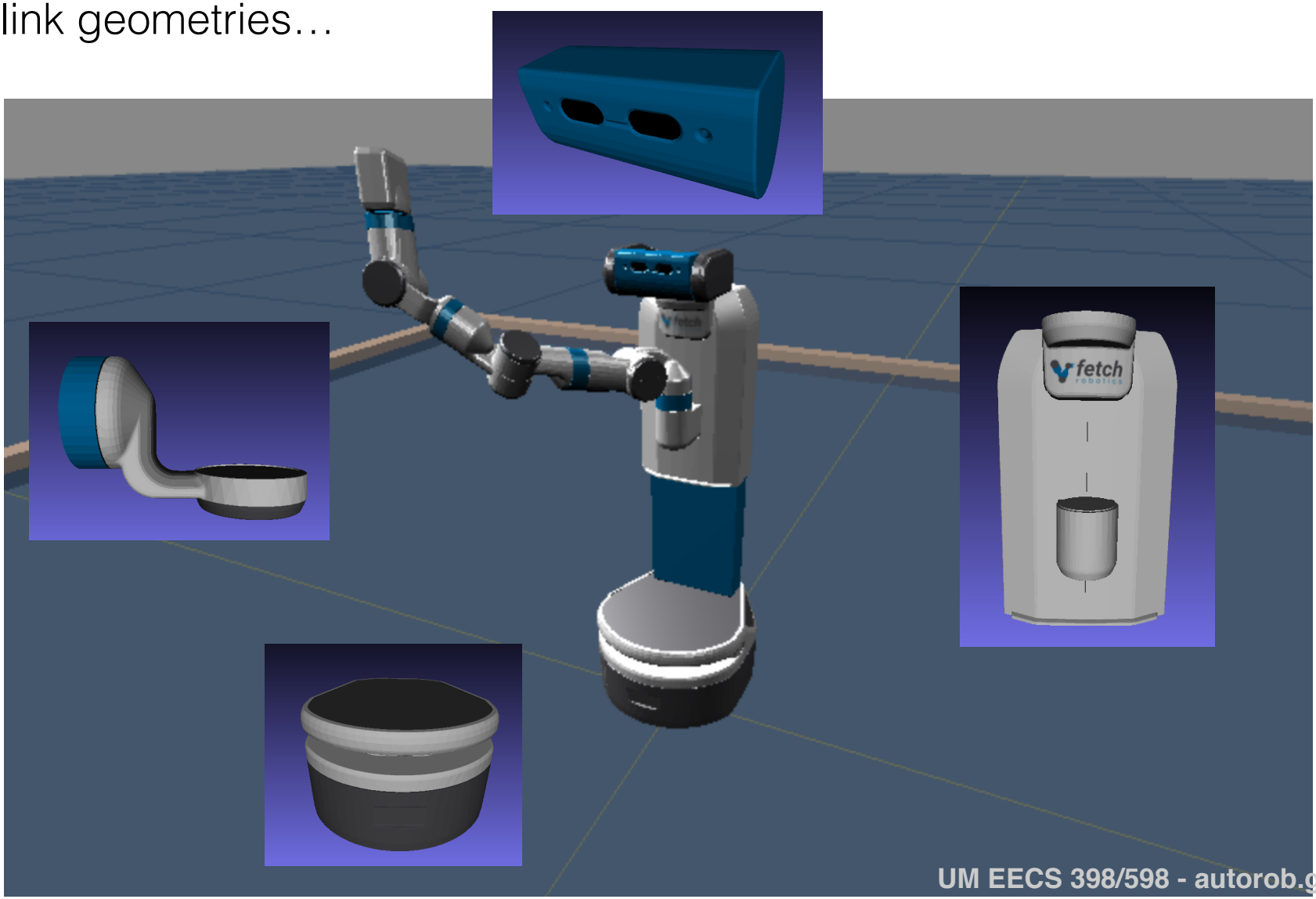
Each link has a geometry specified as 3D vertices in the frame of the link connected into faces of its surface

http://csc.lsu.edu/~kooima/courses/csc4356/

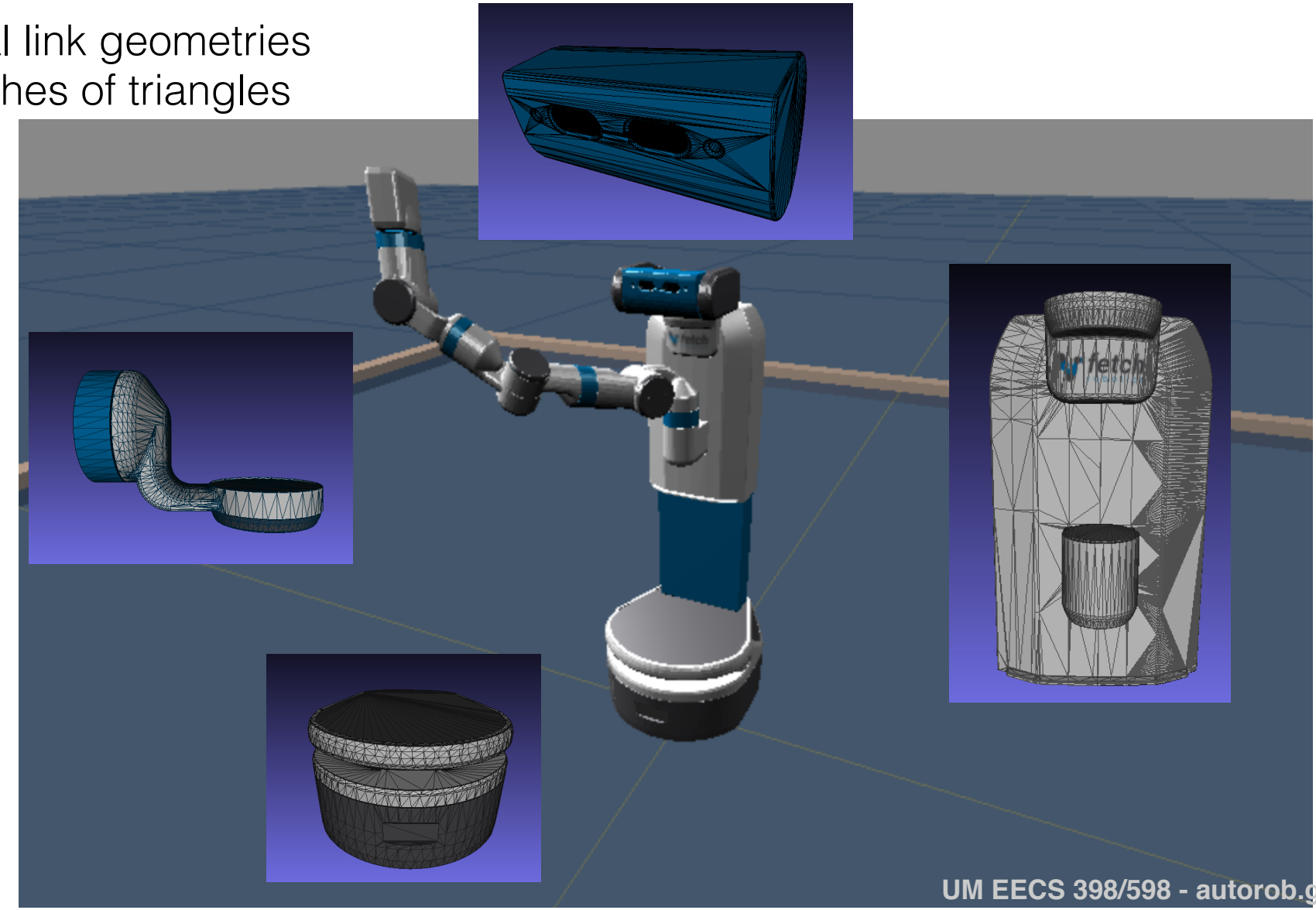
Individual link geometries…

Individual link geometries…

Individual link geometries…



Individual link geometries are meshes of triangles



GitHub, Inc. (US) https://github.com/fetchrobotics/fetch_ros/tree/indigo-devel/fetch_description/meshes

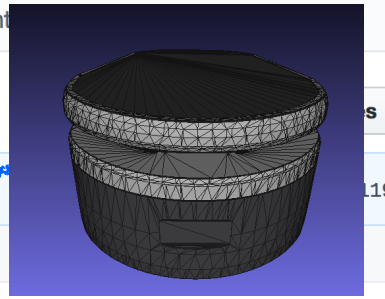
marketing vs accomplishment

This repository Search Pull requests Issues Marketplace Explore

fetchrobotics / fetch_ros

Watch 18 Star 35 Fork 57

Code Issues 3 Pull requests 1 Projects 0 Wiki Insight



Branch: indigo-devel fetch_ros / fetch_description / meshes / Find file History

mikeferguson update gripper model 119857 on Oct 10, 2015

..

base_link.dae	add fetch description package	3 years ago
base_link_collision.STL	remove laser opening from collision mesh	3 years ago
base_link_uv.png	add fetch description package	3 years ago
bellows_link.STL	add fetch description package	3 years ago
bellows_link_collision.STL	add fetch description package	3 years ago
elbow_flex_link.dae	add fetch description package	3 years ago
elbow_flex_link_collision.STL	add fetch description package	3 years ago
elbow_flex_uv.png	add fetch description package	3 years ago
estop_link.STL	add fetch description package	3 years ago
forearm_roll_link.dae	add fetch description package	3 years ago
forearm_roll_link_collision.STL	add fetch description package	3 years ago

GitHub, Inc. (US) https://github.com/fetchrobotics/fetch_ros/tree/indigo-devel/fetch_description/meshes

marketing vs accomplishment

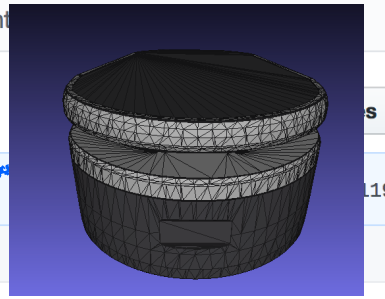
This repository Search Pull requests Issues Marketplace Explore

fetchrobotics / fetch_ros

Watch 18 Star 35 Fork 57

Code Issues 3 Pull requests 1 Projects 0 Wiki Insight

Branch: indigo-devel fetch_ros / fetch_description / meshes / Find file History



mikeferguson update gripper model 119857 on Oct 10, 2015

..

base_link.dae	add fetch description package	3 years ago
base_link_collision.STL	remove laser opening from collision mesh	3 years ago
base_li		
bellows		
bellows		
elbow_		
elbow_		
elbow_		
estop_l		
forearm		
forearm_roll_link_collision.STL	add fetch description package	3 years ago

COLLADA

COLLADA (***COLLA**borative **D**esign **A**ctivity*) is an interchange file format for interactive 3D applications. It is managed by the nonprofit technology consortium, the Khronos Group, and has been adopted by ISO as a publicly available specification, ISO/PAS 17506.[1]

COLLADA defines an open standard XML schema for exchanging digital assets among various graphics software applications that might otherwise store their assets in incompatible file formats. COLLADA documents that describe digital assets are XML files, usually identified with a **.dae** (digital asset exchange) filename extension.

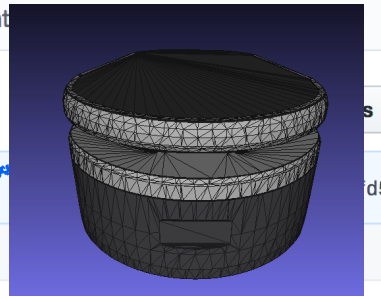
GitHub, Inc. (US) https://github.com/ohseejay/kineval-stencil-fall16/tree/master/robots/fetch kineval-fall16

This repository Search Pull requests Issues Marketplace Explore

ohseejay / **kineval-stencil-fall16**

Watch 2 Star 1 Fork 2

Code Issues 0 Pull requests 0 Projects 0 Wiki Insight



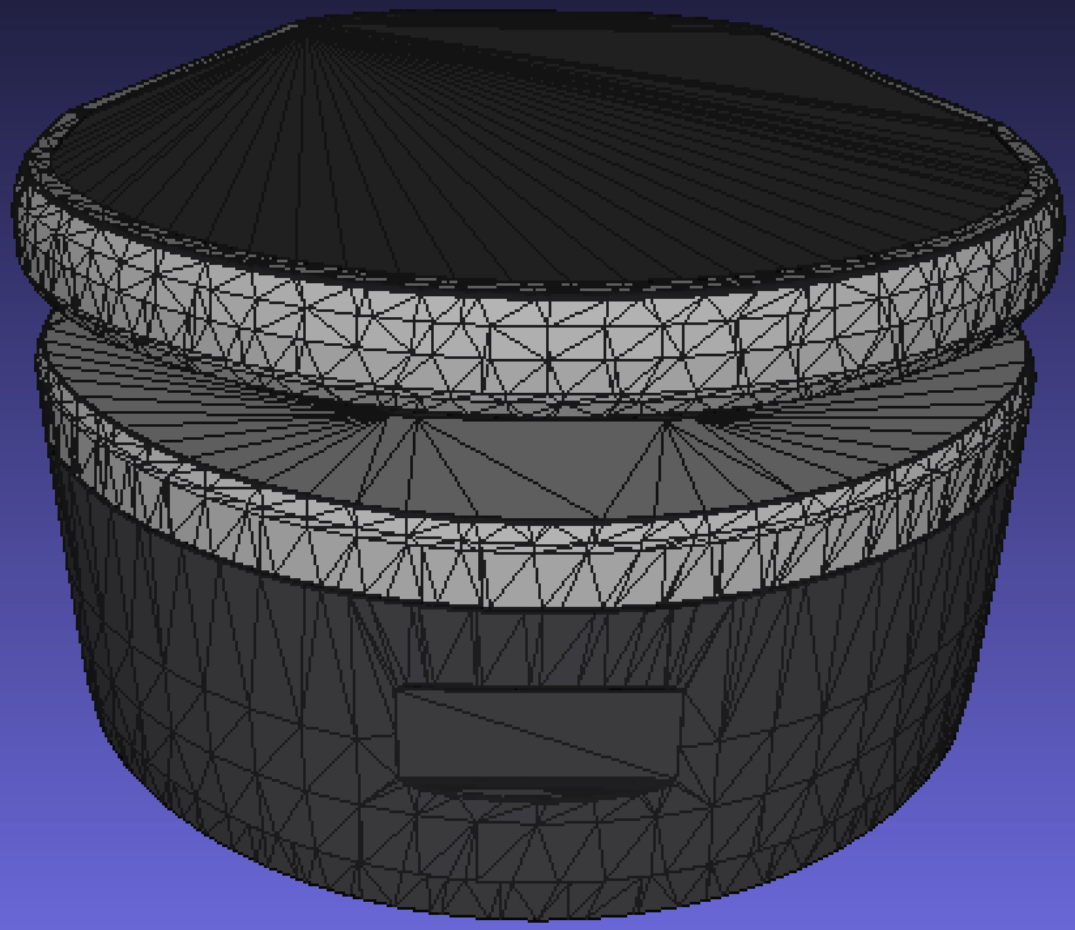
Branch: master **kineval-stencil-fall16** / robots / **fetch** / Find file History

odestcj initial commit d521e on Oct 19, 2016

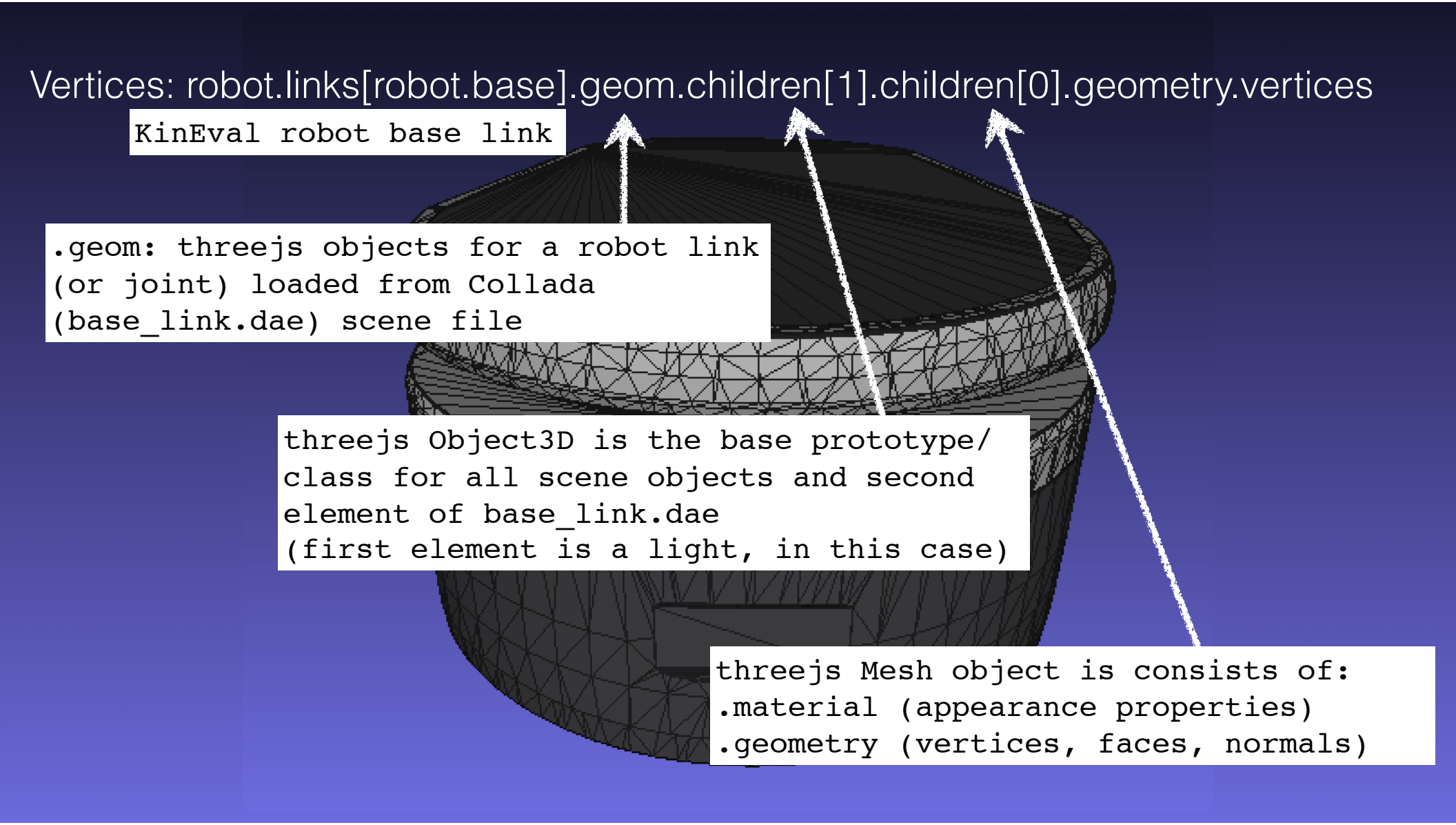
..

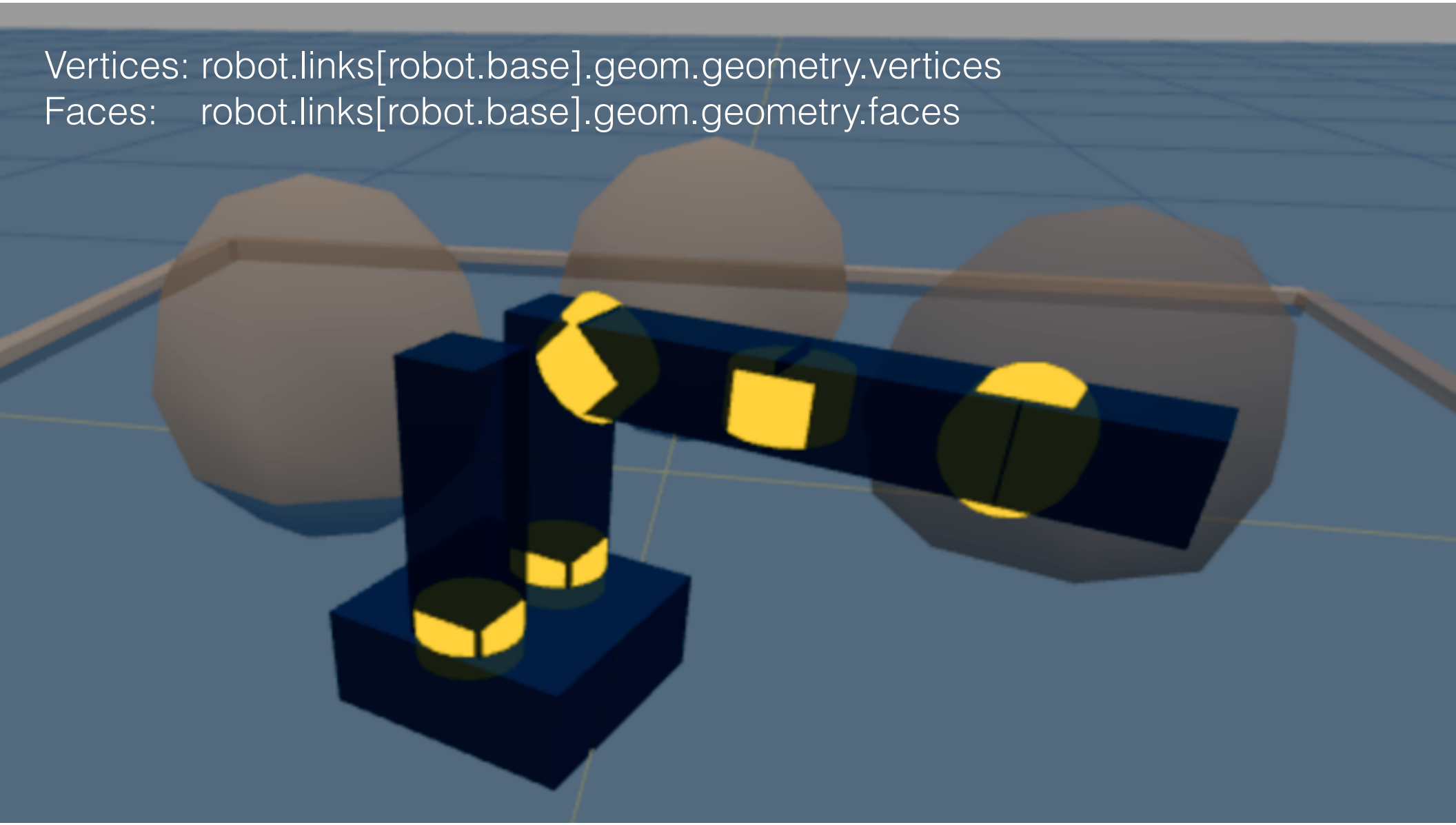
base_link.dae	initial commit	a year ago
base_link_adjusted.dae	initial commit	a year ago
base_link_collision.STL	initial commit	a year ago
base_link_uv.png	initial commit	a year ago
bellows_link.STL	initial commit	a year ago
bellows_link_collision.STL	initial commit	a year ago
elbow_flex_link.dae	initial commit	a year ago
elbow_flex_link_adjusted.dae	initial commit	a year ago
elbow_flex_link_collision.STL	initial commit	a year ago
elbow_flex_uv.png	initial commit	a year ago
estop_link.STL	initial commit	a year ago

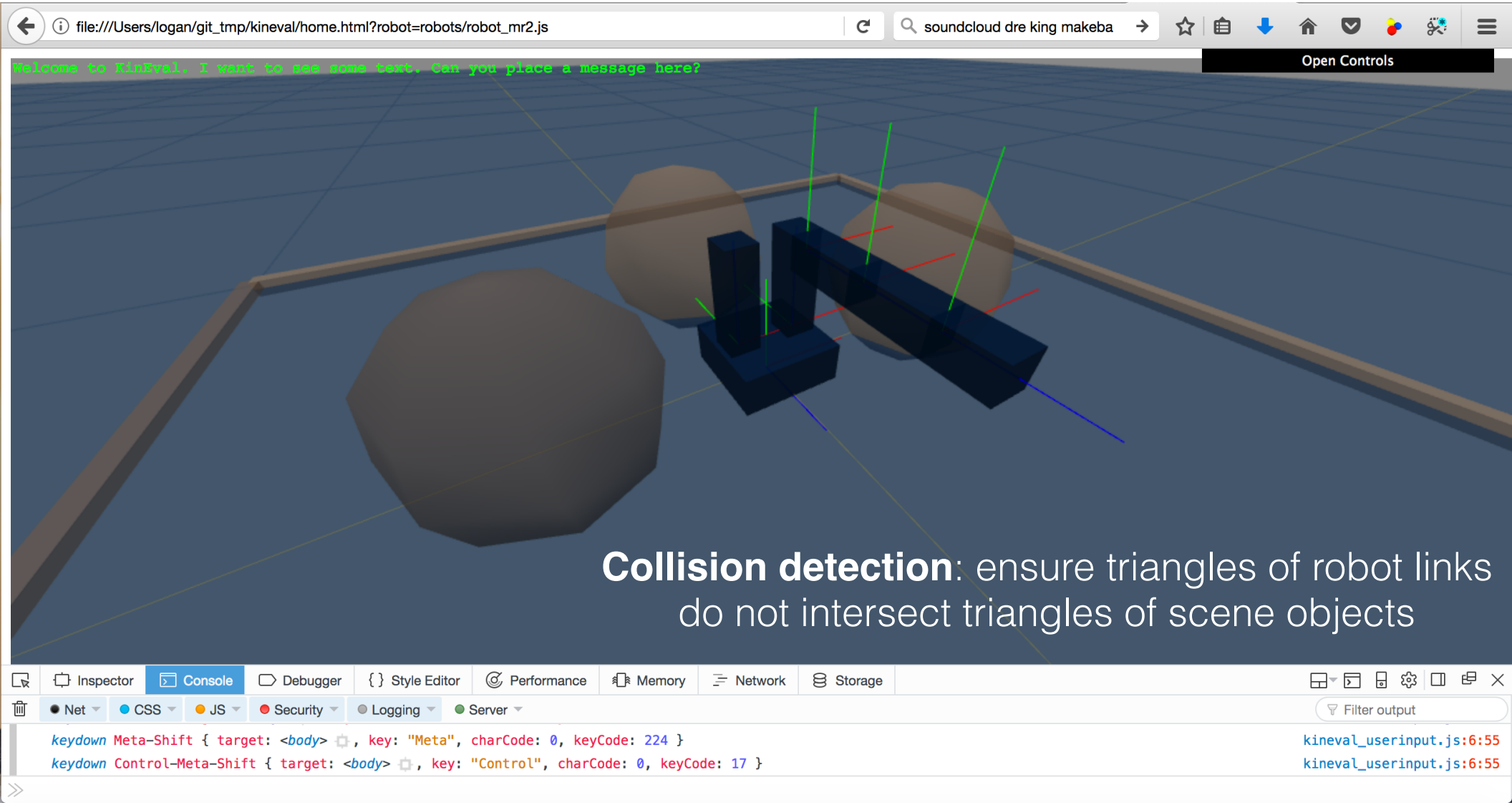
Vertices: robot.links[robot.base].geom.children[1].children[0].geometry.vertices
Faces: robot.links[robot.base].geom.children[1].children[0].geometry.faces

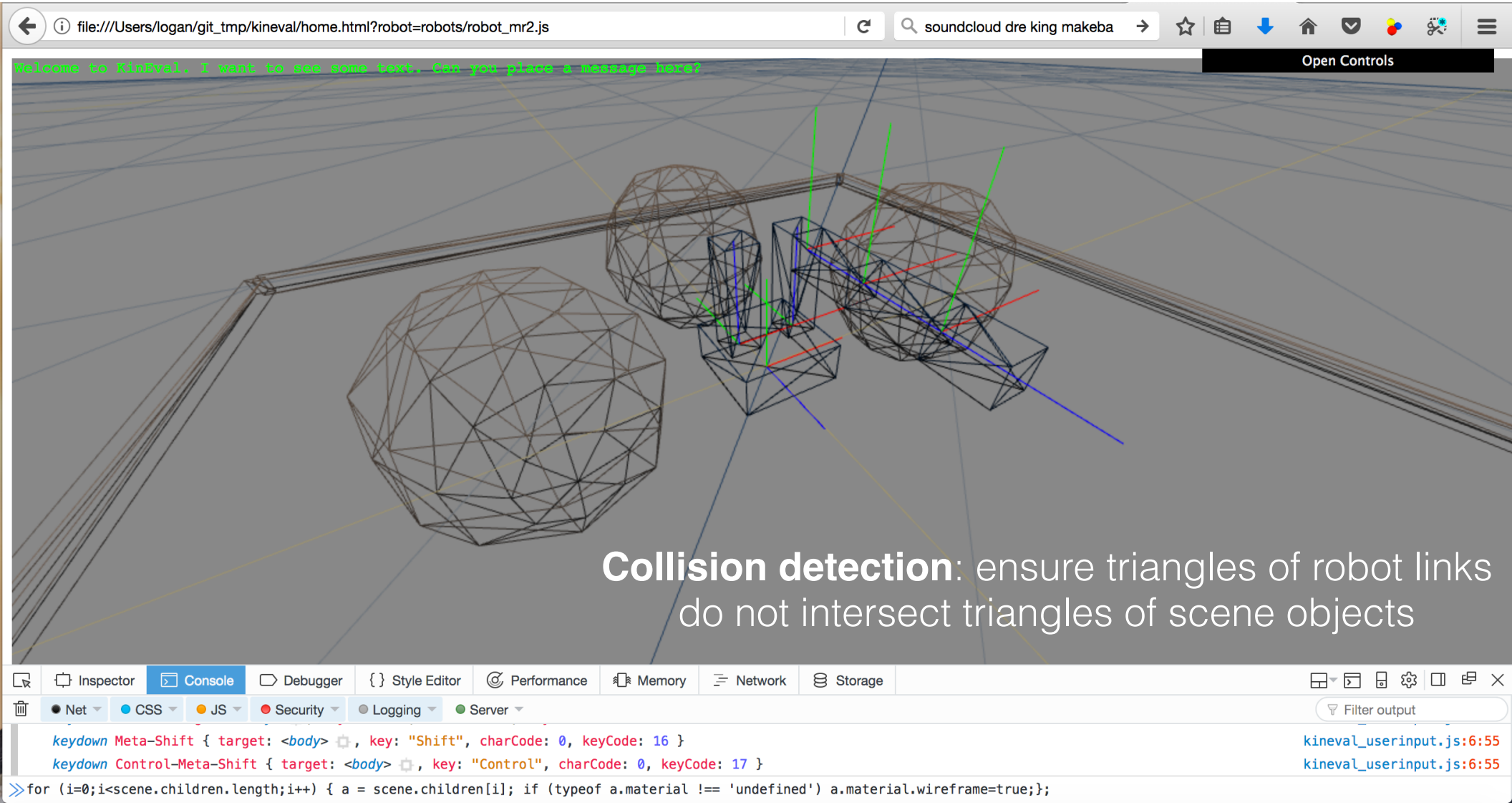


Vertices: robot.links[robot.base].geom.children[1].children[0].geometry.vertices




Vertices: robot.links[robot.base].geom.geometry.vertices
Faces: robot.links[robot.base].geom.geometry.faces

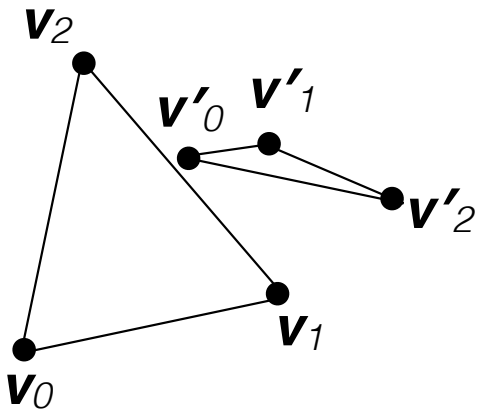

file:///Users/logan/git_tmp/kineval/home.html?robot=robots/robot_mr2.js
soundcloud dre king makeba
Welcome to KinEval. I want to see some text. Can you place a message here?
Open Controls
Collision detection: ensure triangles of robot links do not intersect triangles of scene objects
Inspector
Console
Debugger
Style Editor
Performance
Memory
Network
Storage
Net
CSS
JS
Security
Logging
Server
Filter output
keydown Meta-Shift { target: <body> , key: "Meta", charCode: 0, keyCode: 224 }
kineval_userinput.js:6:55
keydown Control-Meta-Shift { target: <body> , key: "Control", charCode: 0, keyCode: 17 }
kineval_userinput.js:6:55


file:///Users/logan/git_tmp/kineval/home.html?robot=robots/robot_mr2.js
soundcloud dre king makeba
Welcome to KinEval. I want to see some text. Can you place a message here?
Open Controls
Collision detection: ensure triangles of robot links do not intersect triangles of scene objects
Inspector
Console
Debugger
Style Editor
Performance
Memory
Network
Storage
Net
CSS
JS
Security
Logging
Server
Filter output
keydown Meta-Shift { target: <body> , key: "Shift", charCode: 0, keyCode: 16 }
kineval_userinput.js:6:55
keydown Control-Meta-Shift { target: <body> , key: "Control", charCode: 0, keyCode: 17 }
kineval_userinput.js:6:55
for (i=0;i<scene.children.length;i++) { a = scene.children[i]; if (typeof a.material !== 'undefined') a.material.wireframe=true;};

How do we test whether two triangles intersect?

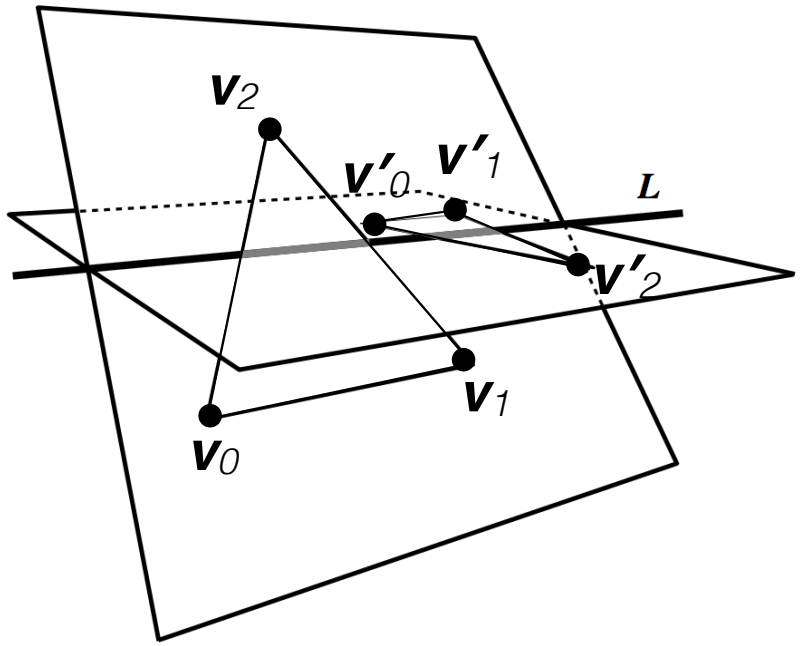
3D Triangle-Triangle Test

- Given two triangles each with three vertices
  - $T = \{\boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_2\}$
  - $T' = \{\boldsymbol{v}'_0, \boldsymbol{v}'_1, \boldsymbol{v}'_2\}$
- Return true if T and T' intersect



3D Triangle-Triangle Test

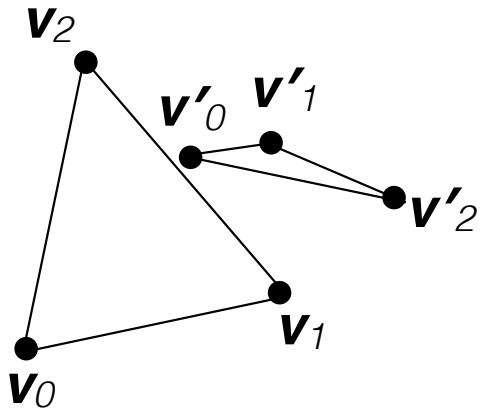
1. Compute plane equation of triangle 2.
2. Reject as trivial if all points of triangle 1 are on same side.
3. Compute plane equation of triangle 1.
4. Reject as trivial if all points of triangle 2 are on same side.
5. Compute intersection line and project onto largest axis.
6. Compute the intervals for each triangle.
7. Intersect the intervals.



Möller 1997

3D Triangle-Triangle Test

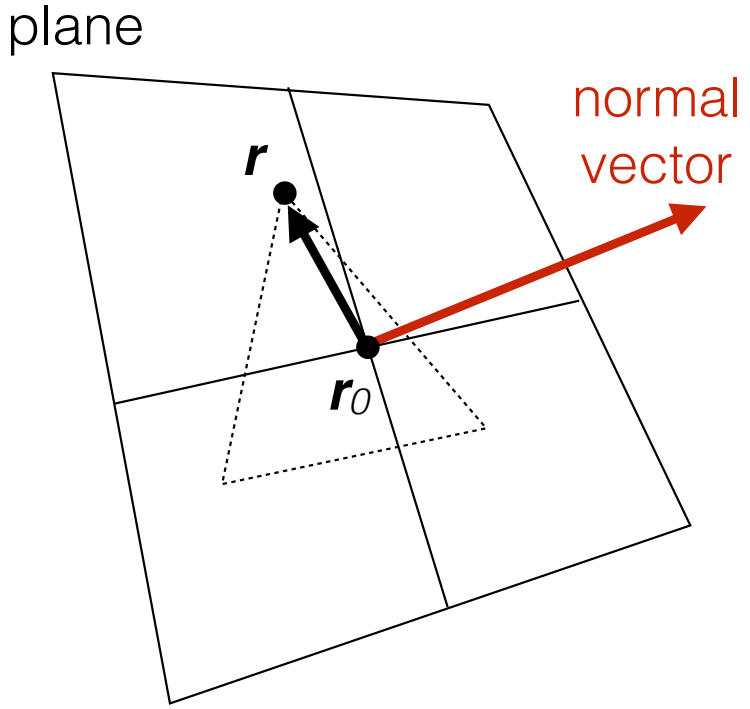
1. Compute plane equation of triangle 2.
2. Reject as trivial if all points of triangle 1 are on same side.
3. Compute plane equation of triangle 1.
4. Reject as trivial if all points of triangle 2 are on same side.
5. Compute intersection line and project onto largest axis.
6. Compute the intervals for each triangle.
7. Intersect the intervals.



Möller 1997

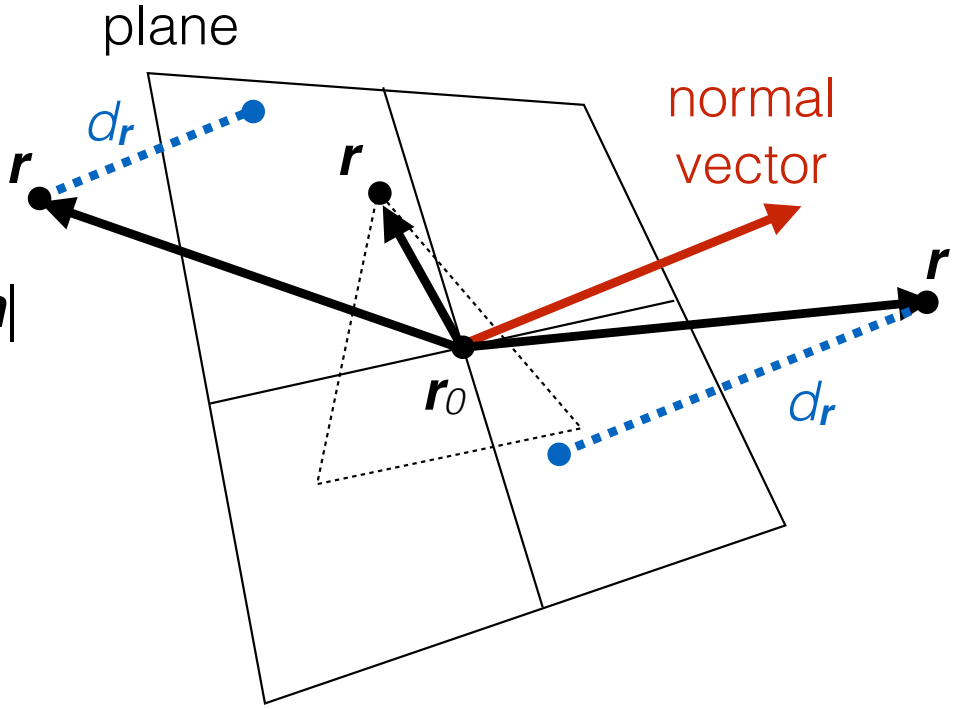
3D Plane Definition

- General form
  - $ax + by + cz + d = 0$
  - $d = -(ax_0 + by_0 + cz_0)$
- Point-normal form (equivalently)
  - $\boldsymbol{n} \cdot (\boldsymbol{r} - \boldsymbol{r}_0) = 0$
- Normal vector $\boldsymbol{n} = [a,b,c]$ orthogonal to plane rooted at location $\boldsymbol{r}_0 = [x_0,y_0,z_0]$
- Any point $\boldsymbol{r} = [x,y,z]$ lying within this plane will evaluate to zero



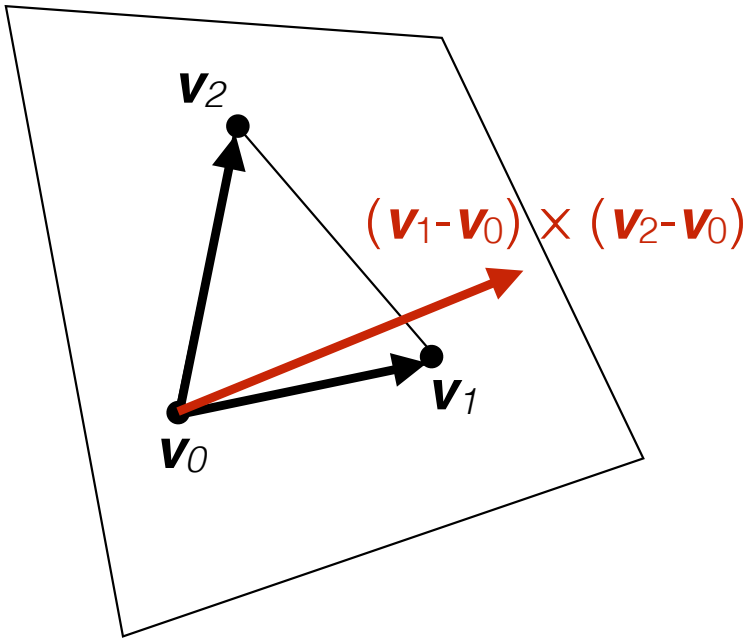
3D Plane Definition

- Scalar projection with normal gives signed distance of point from plane
  - $d_{\boldsymbol{r}} = (\boldsymbol{r} - \boldsymbol{r}_0) \cdot \boldsymbol{n} \,/\, |\boldsymbol{n}| = (\boldsymbol{n} \cdot \boldsymbol{r} - d) \,/\, |\boldsymbol{n}|$
- Any point $\boldsymbol{r} = [x,y,z]$ lying within this plane will have distance $d_{\boldsymbol{r}} = 0$
- Any point below plane: $d_{\boldsymbol{r}} < 0$
- Any point above plane: $d_{\boldsymbol{r}} > 0$



3D Plane Definition

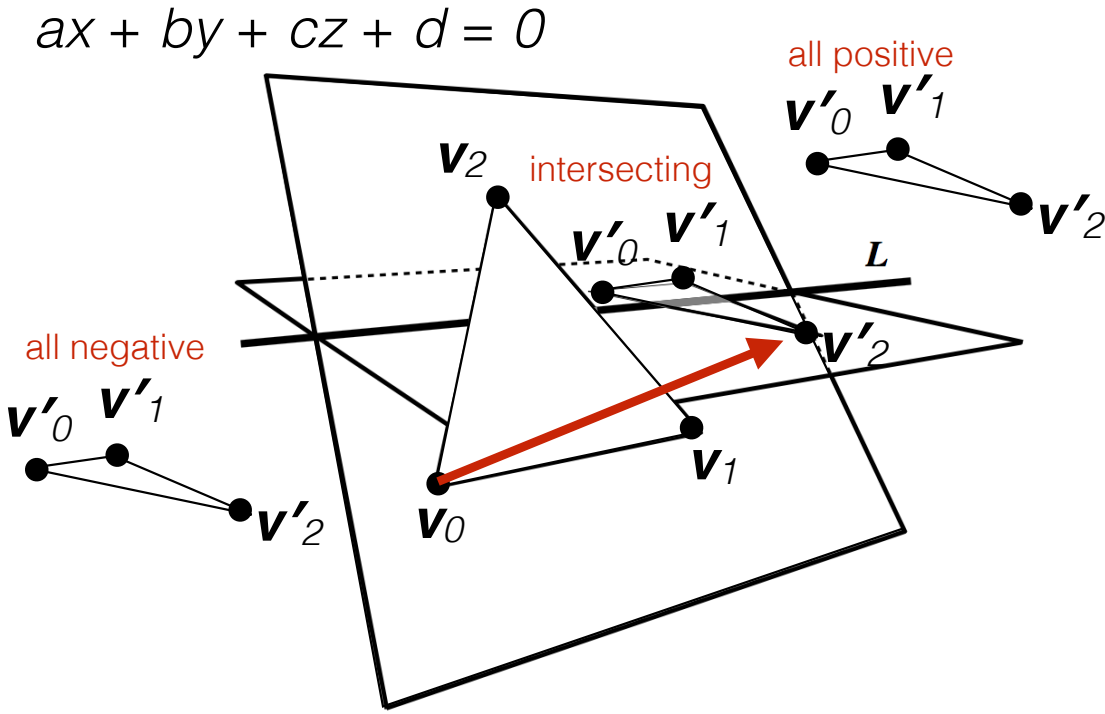
$ax + by + cz + d = 0$



- Plane coefficients can be computed from points of triangle
  - $\boldsymbol{n} = [a,b,c] = (\boldsymbol{v}_1-\boldsymbol{v}_0) \times (\boldsymbol{v}_2-\boldsymbol{v}_0)$
  - $d = -\,\boldsymbol{v}_2 \cdot \boldsymbol{n}$

3D Triangle-Triangle Test

1. Compute plane equation of triangle 2.
2. Reject as trivial if all points of triangle 1 are on same side.
3. Compute plane equation of triangle 1.
4. Reject as trivial if all points of triangle 2 are on same side.
5. Compute intersection line and project onto largest axis.
6. Compute the intervals for each triangle.
7. Intersect the intervals.



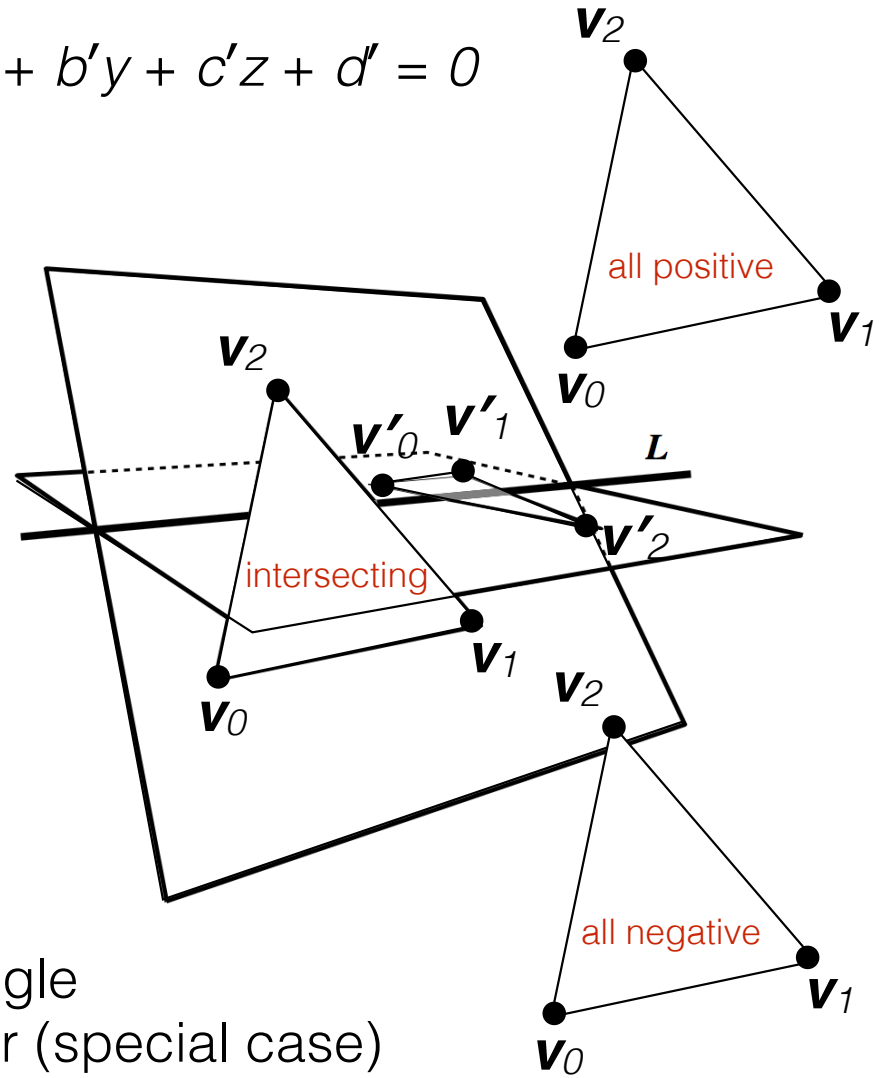
Input points into plane equation.
If all have the same sign, planes of triangles do not intersect

Möller 1997

3D Triangle-Triangle Test

$a'x + b'y + c'z + d' = 0$

1. Compute plane equation of triangle 2.
2. Reject as trivial if all points of triangle 1 are on same side.
3. Compute plane equation of triangle 1.
4. Reject as trivial if all points of triangle 2 are on same side.
5. Compute intersection line and project onto largest axis.
6. Compute the intervals for each triangle.
7. Intersect the intervals.



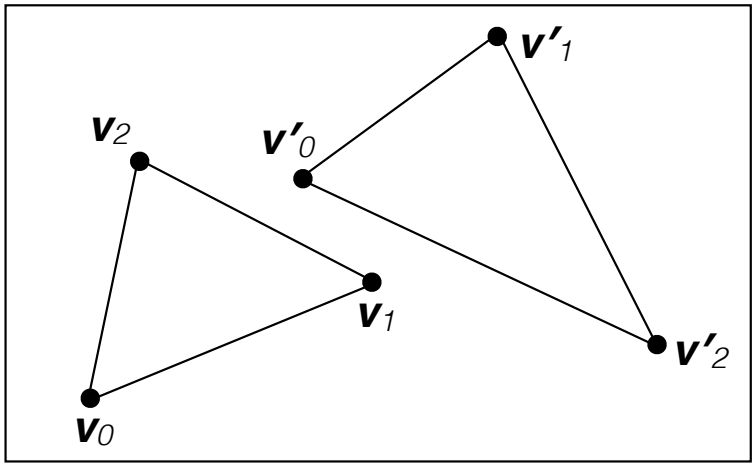
Repeat for other plane of other triangle

If all evaluations are zero, triangles are co-planar (special case)

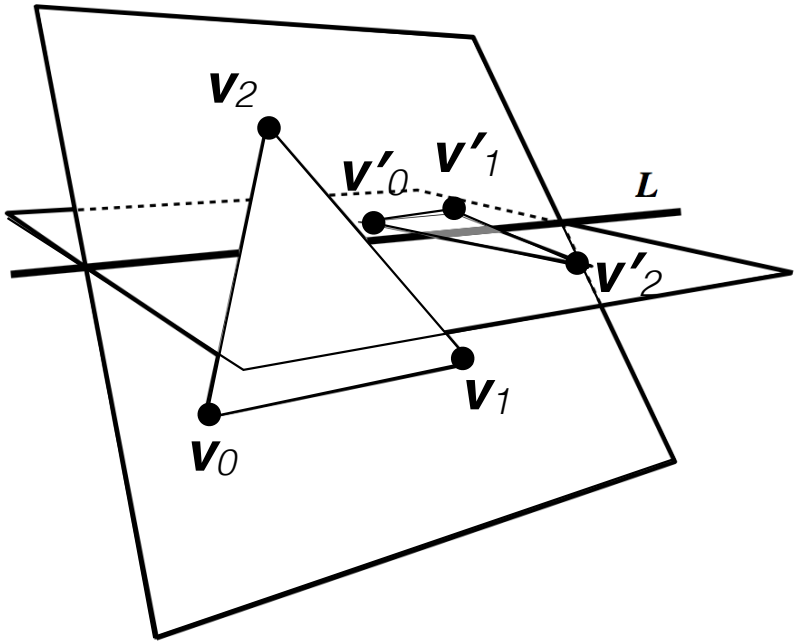
Three possible cases can occur based on
evaluation of vertices of one triangle against the plane of the other triangle

1. Triangle does not intersect plane
(all positive or all negative evaluations)
return non-collision

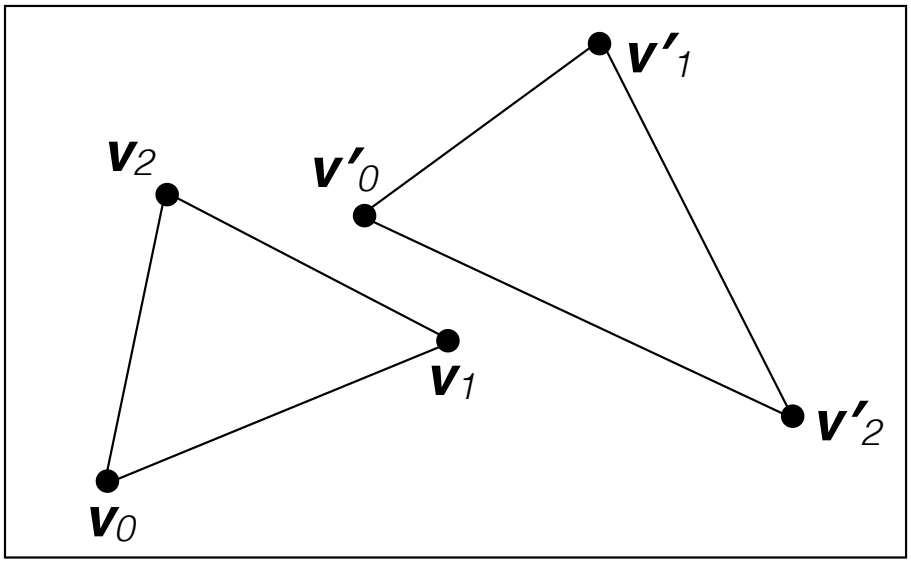
2. Triangles are coplanar
(all evaluations are zero)

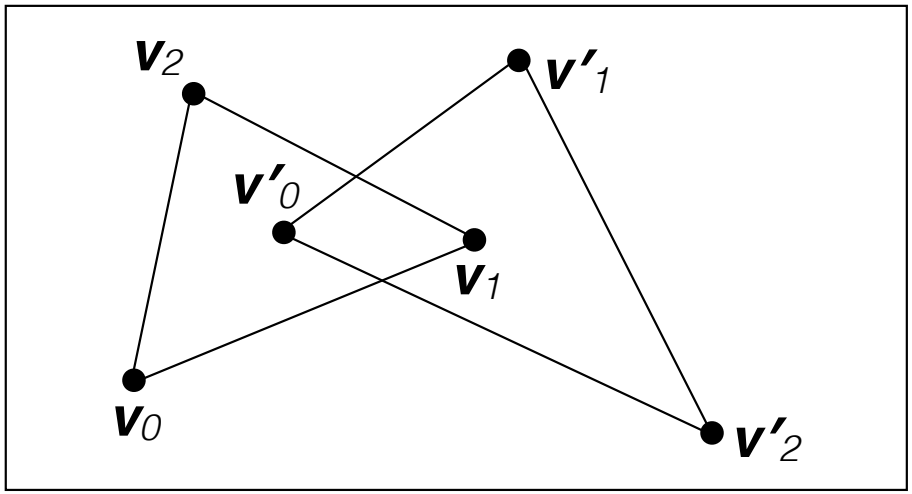


3. Triangles are not coplanar
(positive and negative evaluations)

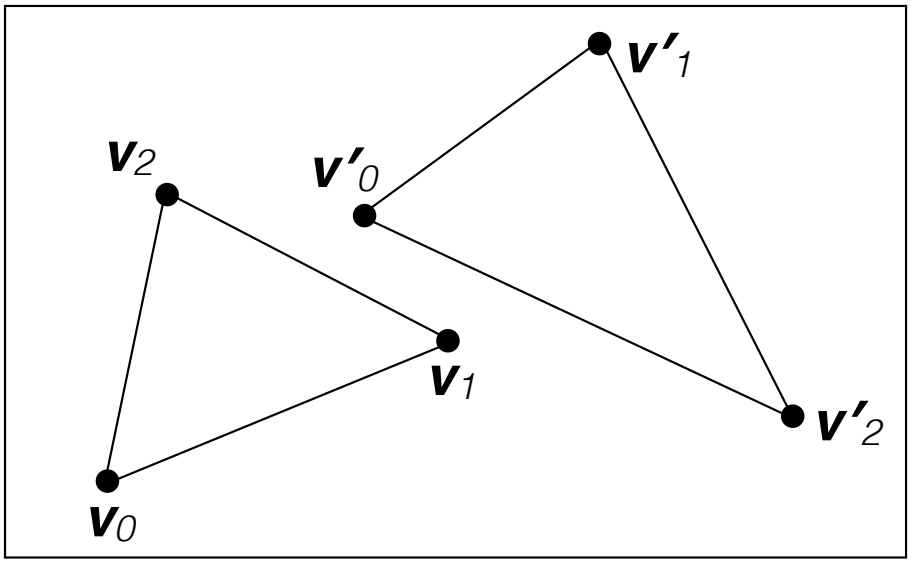


Suppose triangles are coplanar

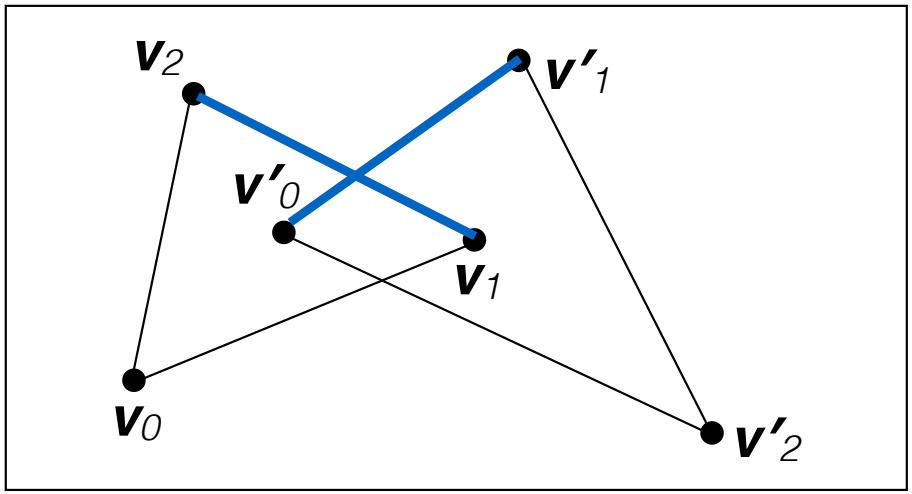




Suppose triangles are coplanar



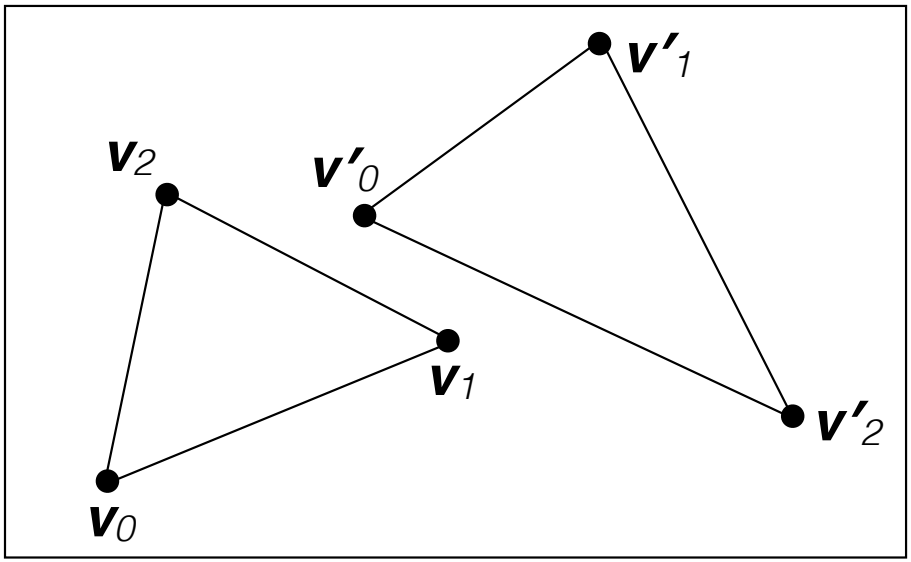
Compare each pair of line segments

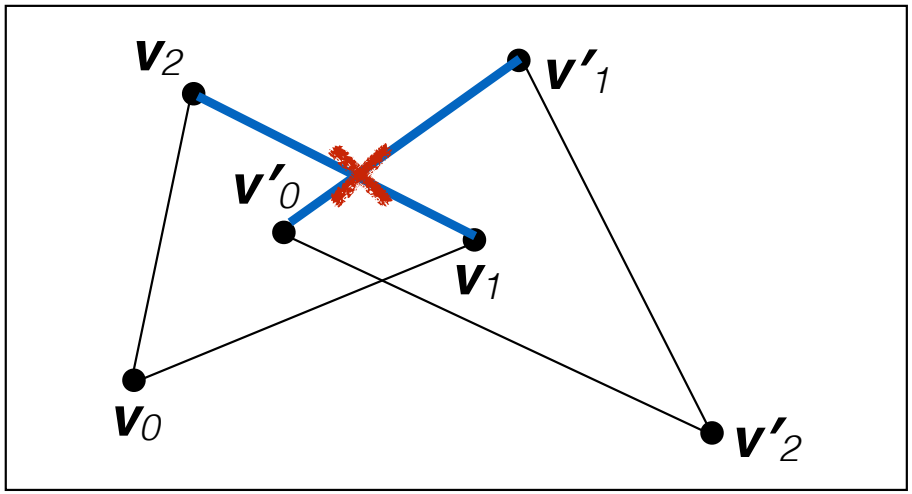


Suppose triangles are coplanar

Compare each pair of line segments

Find intersection point as solution to linear system

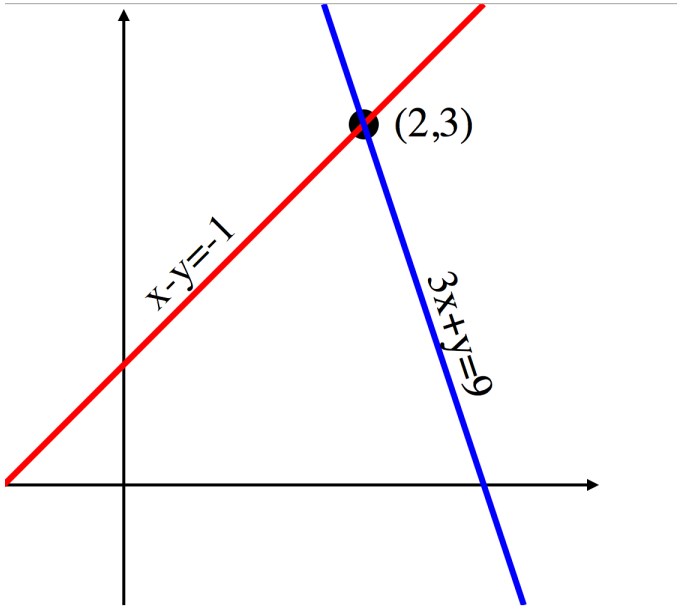




Remember:

2D Example

only single point satisfies both lines

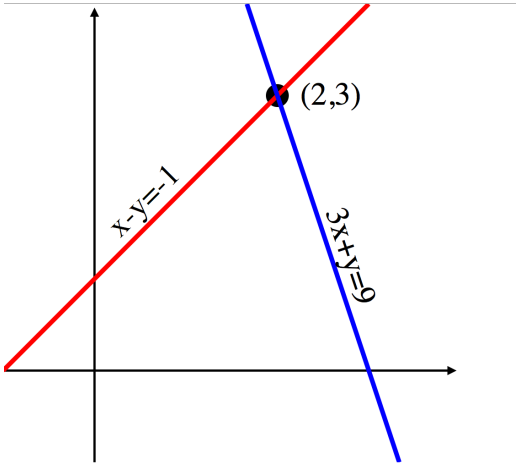


Cramer's rule

From Wikipedia, the free encyclopedia

In linear algebra, **Cramer's rule** is an explicit formula for the solution of a system of linear equations with as many equations as unknowns, valid whenever the system has a unique solution. It expresses the solution in terms of the determinants of the (square) coefficient matrix and of matrices obtained from it by replacing one column by the vector of right-hand-sides of the equations. It is named after Gabriel Cramer (1704–1752), who published the rule for an arbitrary number of unknowns in 1750,[1][2] although Colin Maclaurin also published special cases of the rule in 1748[3] (and possibly knew of it as early as 1729).[4][5][6]

Cramer's rule is computationally very inefficient for systems of more than two or three equations;[7] its asymptotic complexity is O(n·n!) compared to elimination methods that have polynomial time complexity.[8][9] Cramer's rule is also numerically unstable even for 2×2 systems.[10]

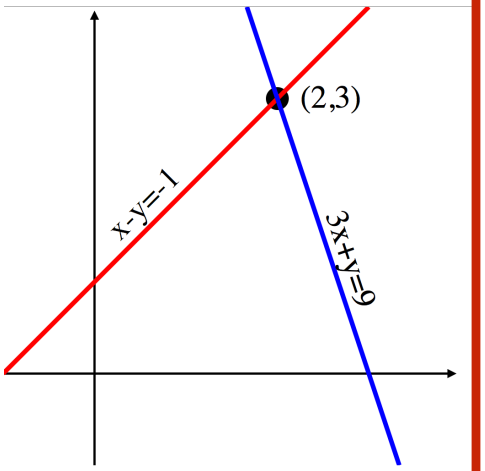


Cramer's rule

From Wikipedia, the free encyclopedia

In linear algebra, **Cramer's rule** is an explicit formula for the solution of a system of linear equations with as many equations as unknowns, valid whenever the system has a unique solution. It expresses the solution in terms of the determinants of the (square) coefficient matrix and of matrices obtained from it by replacing one column by the vector of right-hand-sides of the equations. It is named after Gabriel Cramer (1704–1752), who published the rule for an arbitrary number of unknowns in 1750,[1][2] although Colin Maclaurin also published special cases of the rule in 1748[3] (and possibly knew

Cramer's rule is computationally very i
to elimination methods that have polyn



Explicit formulas for small systems [edit]

Consider the linear system

$$\begin{cases} a_1 x + b_1 y &= c_1 \\ a_2 x + b_2 y &= c_2 \end{cases}$$

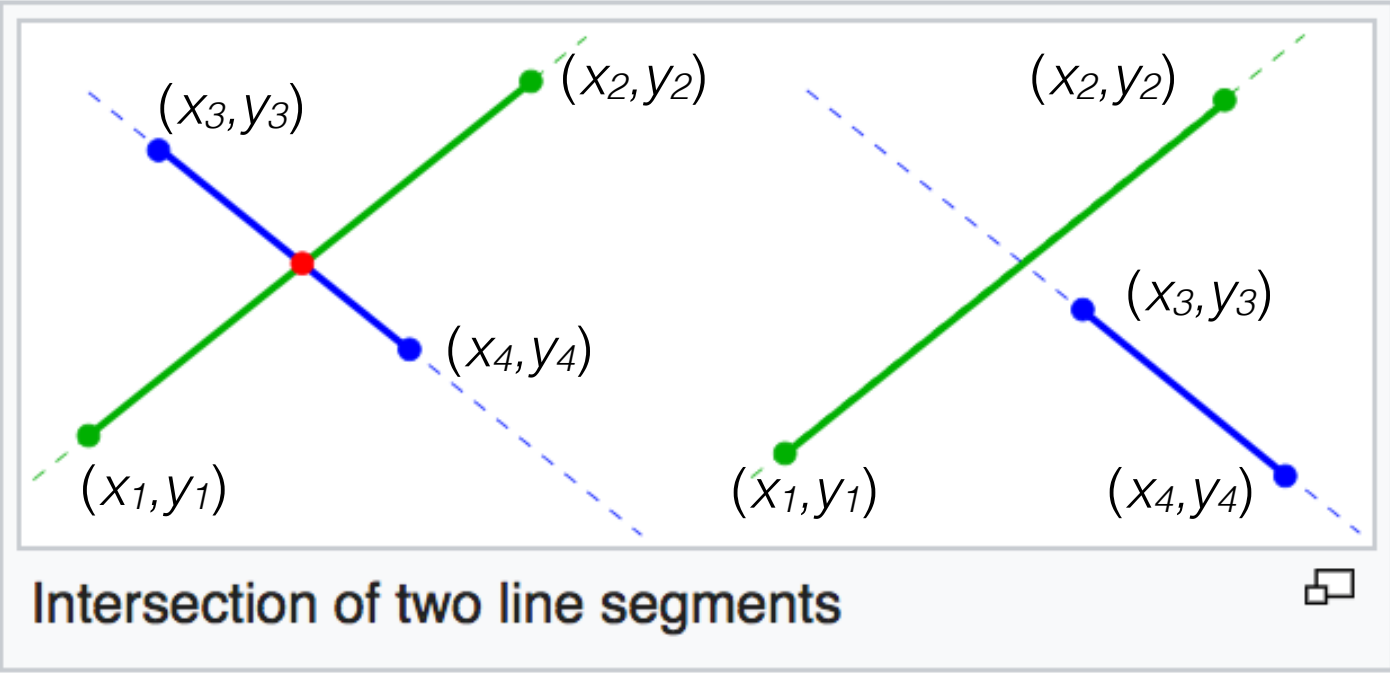
which in matrix format is

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}.$$

Assume $a_1 b_2 - b_1 a_2$ nonzero. Then, with help of determinants, x and y can be found with Cramer's rule as

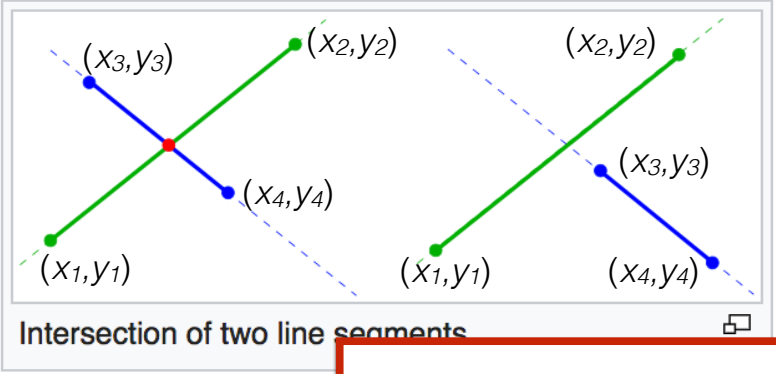
$$x = \frac{\begin{vmatrix} c_1 & b_1 \\ c_2 & b_2 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}} = \frac{c_1 b_2 - b_1 c_2}{a_1 b_2 - b_1 a_2}, \quad y = \frac{\begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}} = \frac{a_1 c_2 - c_1 a_2}{a_1 b_2 - b_1 a_2}.$$

Line Segment Intersection Test



Intersection of two line segments

Line Segment Intersection Test



Intersection of two line segments

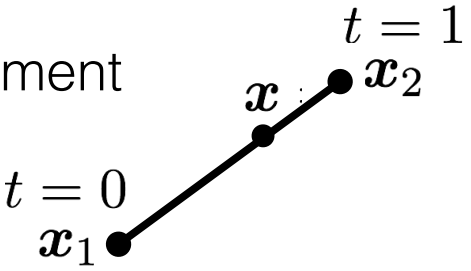
Parametric expression of line segments

$$(x(s), y(s)) = (x_1 + s(x_2 - x_1), y_1 + s(y_2 - y_1))$$
$$(x(t), y(t)) = (x_3 + t(x_4 - x_3), y_3 + t(y_4 - y_3))$$

Segments span parameter interval

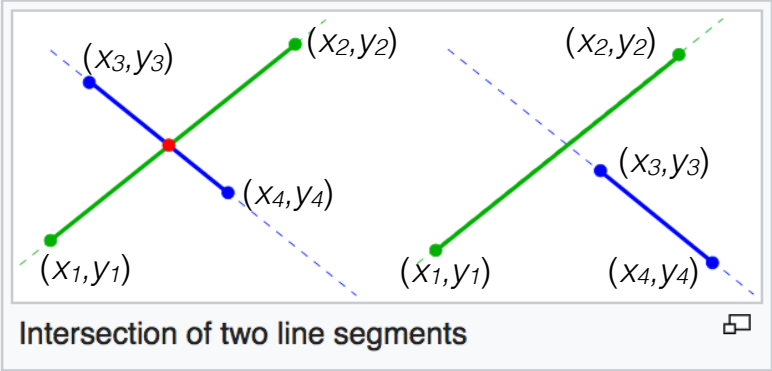
$$0 \leq s, t \leq 1$$

Parametric expression of a line segment



$$\boldsymbol{x} = \boldsymbol{x}_1 + t\boldsymbol{x}_2 = (x_1 + t(x_2 - x_1), y_1 + t(y_2 - y_1)) \text{ , for } 0 \leq t \leq 1$$

Line Segment Intersection Test



Intersection of two line segments

Form linear system s.t. $(\boldsymbol{x}(s), \boldsymbol{y}(s)) = (\boldsymbol{x}(t), \boldsymbol{y}(t))$

$$\begin{bmatrix} \boldsymbol{x}_2 - \boldsymbol{x}_1 & \boldsymbol{x}_4 - \boldsymbol{x}_3 \\ \boldsymbol{y}_2 - \boldsymbol{y}_1 & \boldsymbol{y}_4 - \boldsymbol{y}_3 \end{bmatrix} \begin{bmatrix} s \\ t \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_3 - \boldsymbol{x}_1 \\ \boldsymbol{y}_3 - \boldsymbol{y}_1 \end{bmatrix}$$

Solve for parameters at intersection point using Cramer's rule

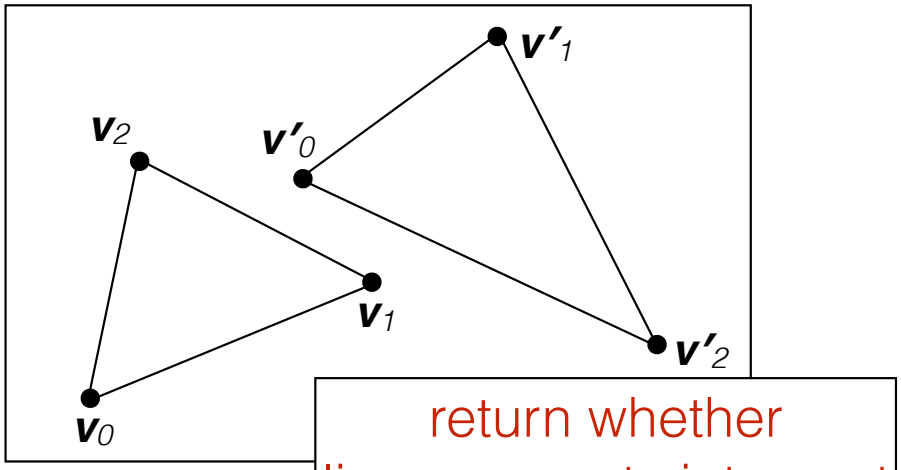
Return intersection, if parameters lie within segments, as $0 \leq s, t \leq 1$

Return non-intersection, otherwise

Three possible cases can occur based on
evaluation of vertices of one triangle against the plane of the other triangle

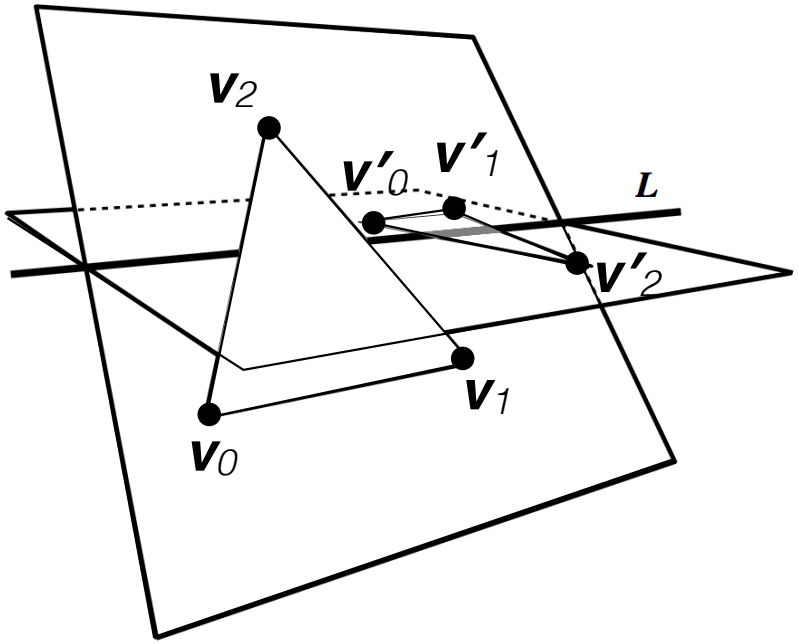
1. Triangle does not intersect plane
(all positive or all negative evaluations)
return non-collision

2. Triangles are coplanar
(all evaluations are zero)



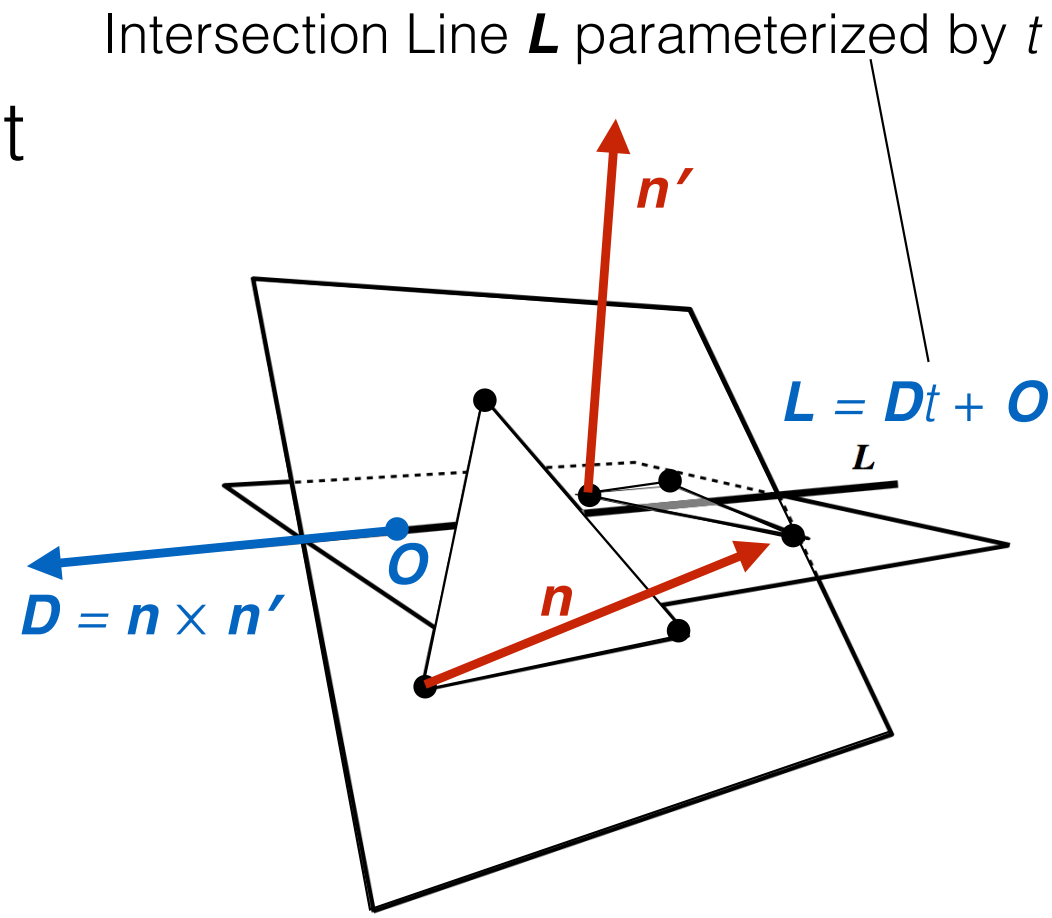
return whether
line segments intersect

3. Triangles are not coplanar
(positive and negative evaluations)



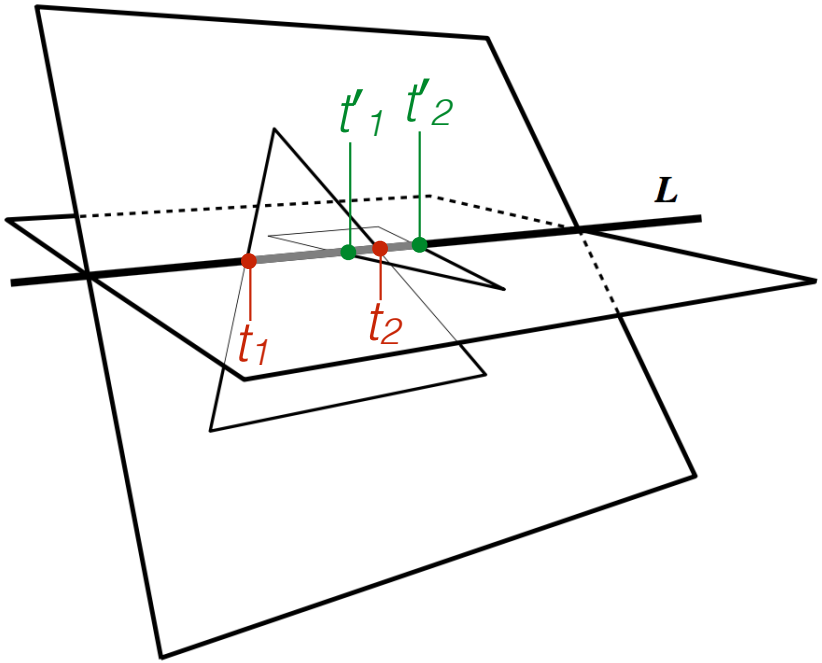
3D Triangle-Triangle Test

1. Compute plane equation of triangle 2.
2. Reject as trivial if all points of triangle 1 are on same side.
3. Compute plane equation of triangle 1.
4. Reject as trivial if all points of triangle 2 are on same side.
5. Compute intersection line and project onto largest axis.
6. Compute the intervals for each triangle.
7. Intersect the intervals.

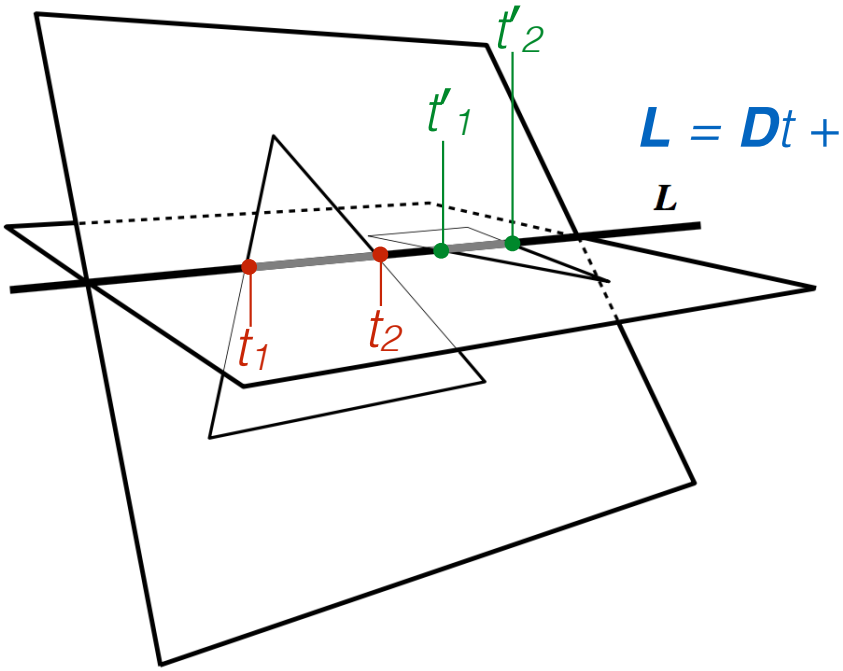


Two possible cases can occur based on
interval spans ($[t_1,t_2]$, $[t'_1,t'_2]$) of triangle intersections with ***L***

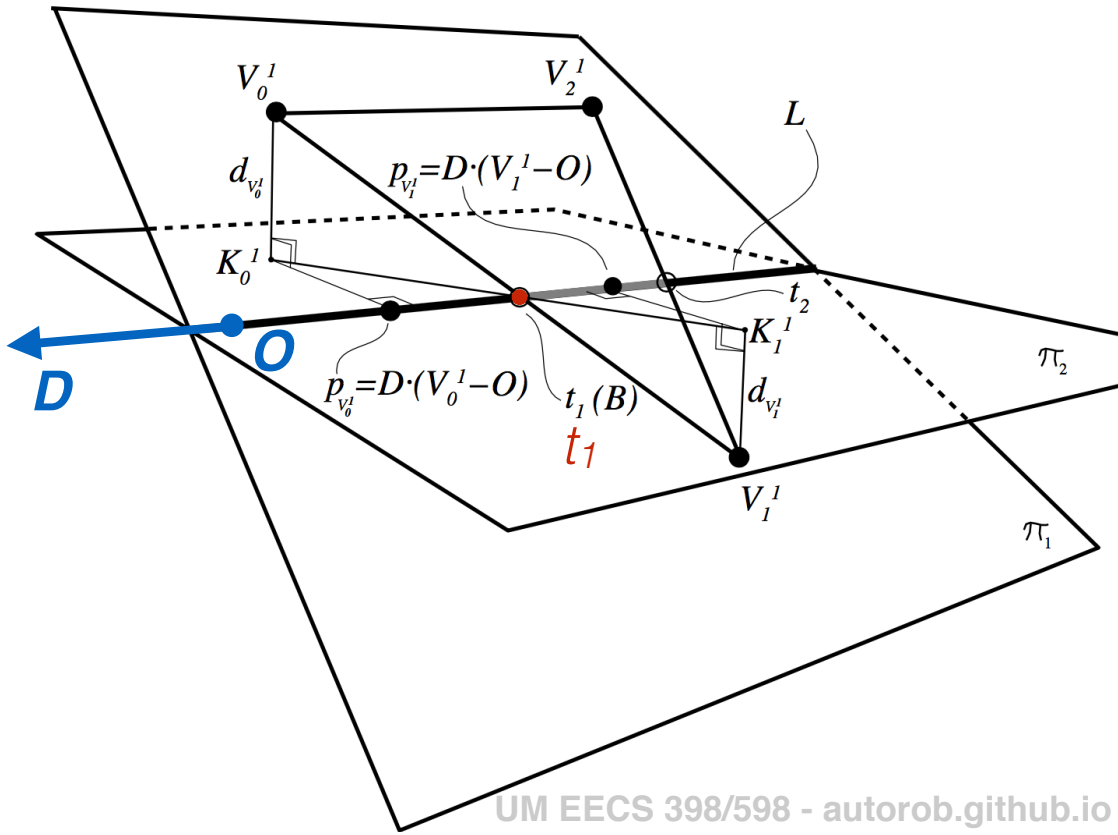
Collision, if intervals overlap



No Collision, if intervals do not overlap



1. Compute plane equation of triangle 2.
2. Reject as trivial if all points of triangle 1 are on same side.
3. Compute plane equation of triangle 1.
4. Reject as trivial if all points of triangle 2 are on same side.
5. Compute intersection line and project onto largest axis.
6. Compute the intervals for each triangle.
7. Intersect the intervals.

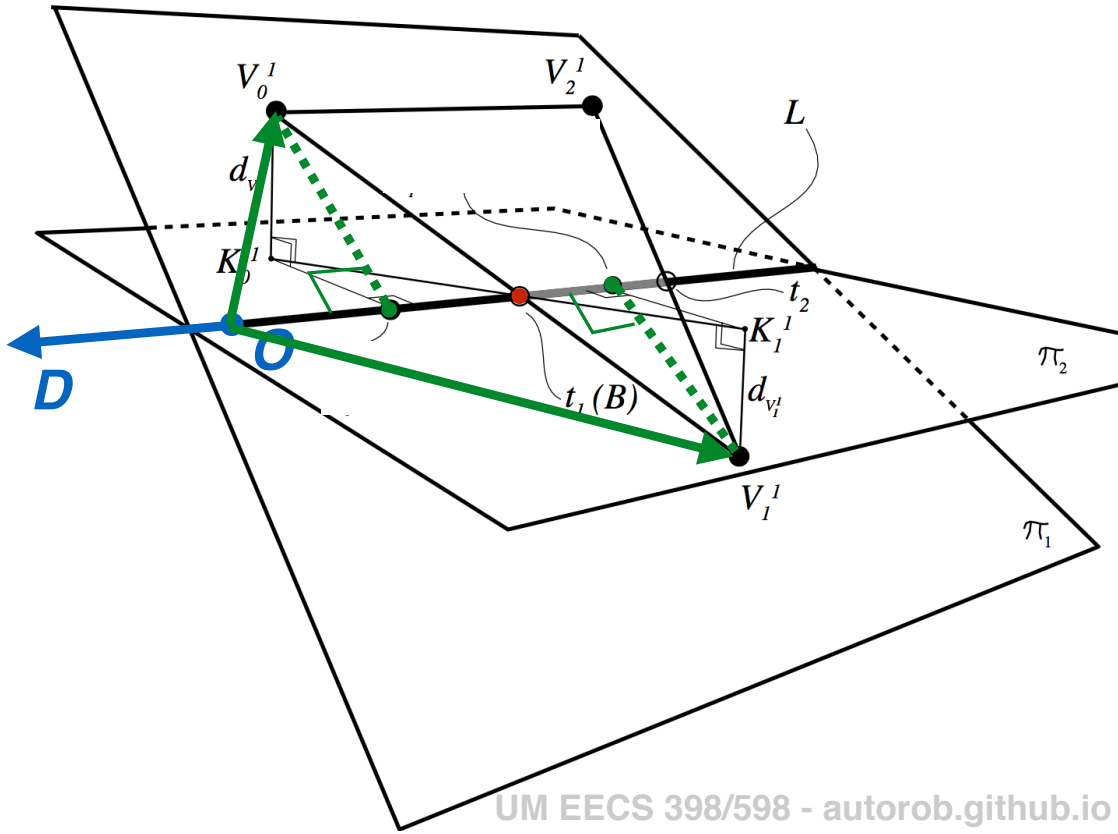


Scalar projection of vertices onto ***L***

$p_0 = \boldsymbol{D} \cdot (v_0 - \boldsymbol{O}) / |\boldsymbol{D}|$

$p_1 = \boldsymbol{D} \cdot (v_1 - \boldsymbol{O}) / |\boldsymbol{D}|$

1. Compute plane equation of triangle 2.
2. Reject as trivial if all points of triangle 1 are on same side.
3. Compute plane equation of triangle 1.
4. Reject as trivial if all points of triangle 2 are on same side.
5. Compute intersection line and project onto largest axis.
6. Compute the intervals for each triangle.
7. Intersect the intervals.



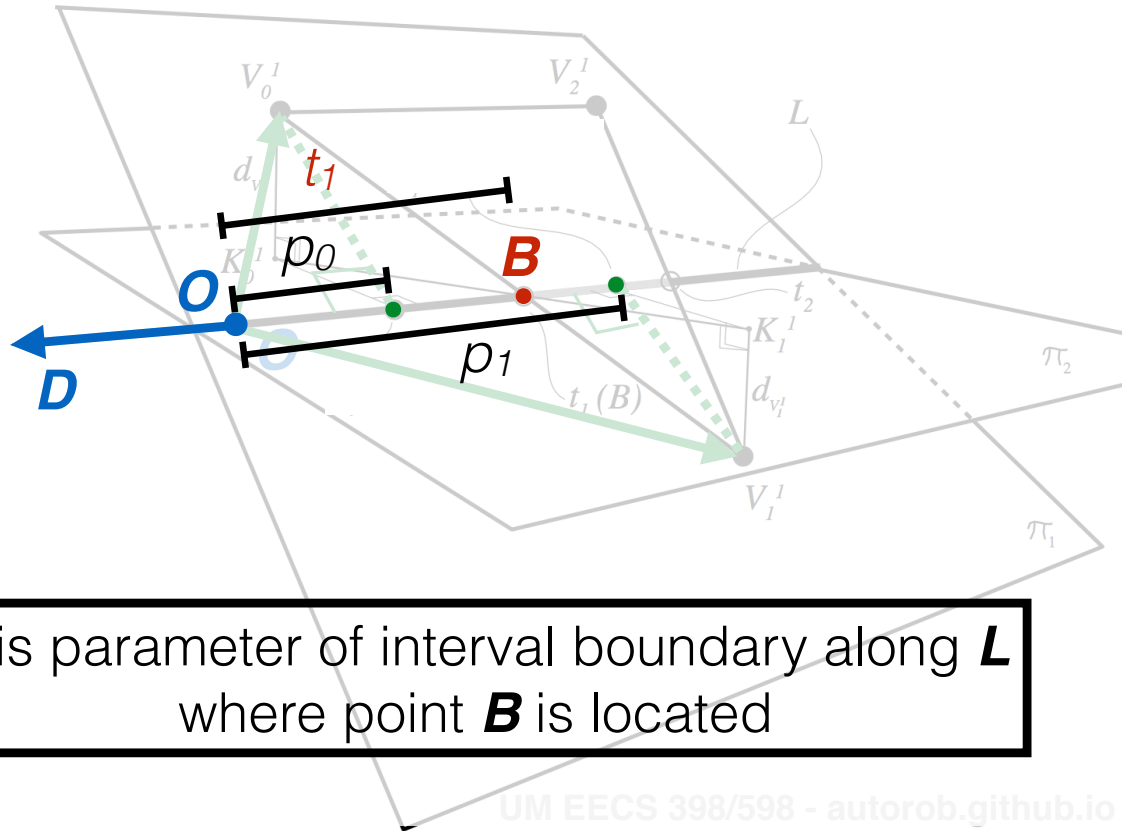
Scalar projection of vertices onto ***L***

$p_0 = \boldsymbol{D} \cdot (v_0 - \boldsymbol{O}) / |\boldsymbol{D}|$

$p_1 = \boldsymbol{D} \cdot (v_1 - \boldsymbol{O}) / |\boldsymbol{D}|$

p_0 and p_1 are parameter values along ***L***

1. Compute plane equation of triangle 2.
2. Reject as trivial if all points of triangle 1 are on same side.
3. Compute plane equation of triangle 1.
4. Reject as trivial if all points of triangle 2 are on same side.
5. Compute intersection line and project onto largest axis.
6. Compute the intervals for each triangle.
7. Intersect the intervals.



t_1 is parameter of interval boundary along ***L*** where point ***B*** is located

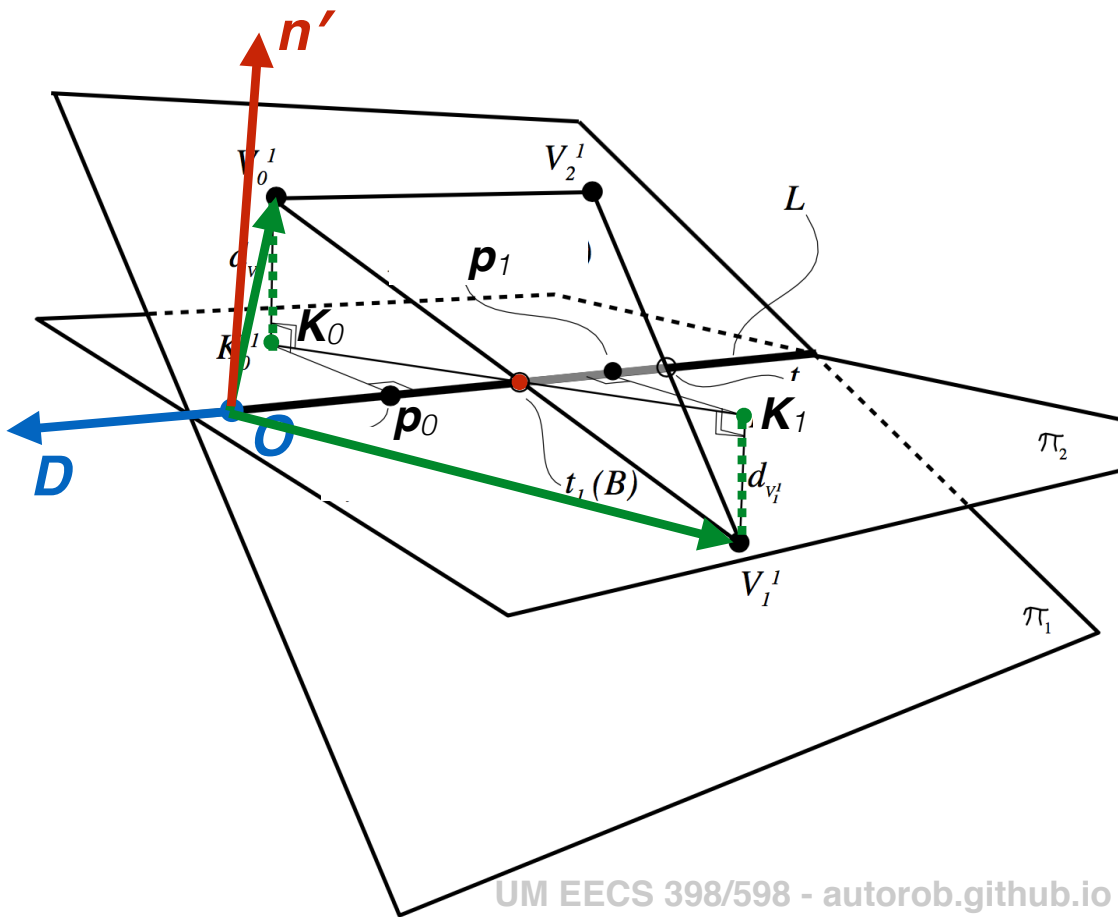
Scalar projection of vertices onto **n′**

$d_0 = \mathbf{n'} \cdot (v_0 - \boldsymbol{O}) / |\mathbf{n'}|$

$d_1 = \mathbf{n'} \cdot (v_1 - \boldsymbol{O}) / |\mathbf{n'}|$

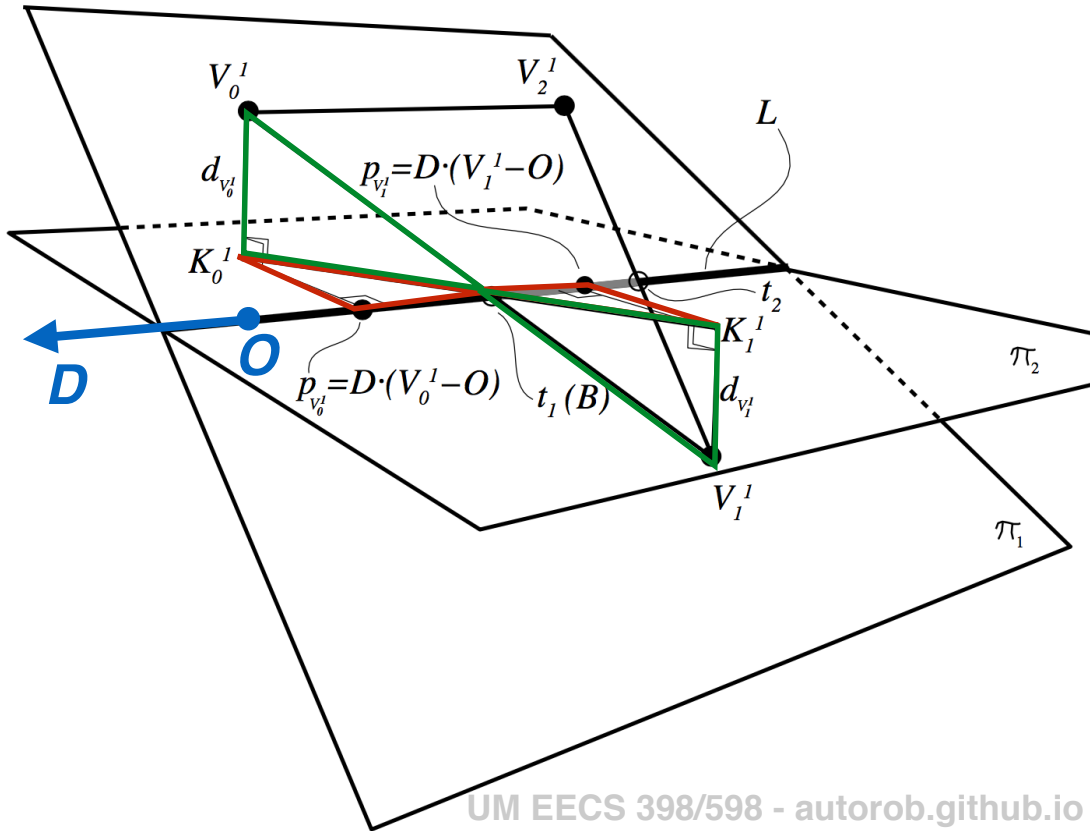
d_0 is distance to point $\boldsymbol{K}_0$ in plane of other triangle T'

1. Compute plane equation of triangle 2.
2. Reject as trivial if all points of triangle 1 are on same side.
3. Compute plane equation of triangle 1.
4. Reject as trivial if all points of triangle 2 are on same side.
5. Compute intersection line and project onto largest axis.
6. Compute the intervals for each triangle.
7. Intersect the intervals.



Assert similar triangles $\triangle \boldsymbol{v_0 K_0 B}$ and $\triangle \boldsymbol{v_1 K_1 B}$
at the point of the interval boundary $\boldsymbol{B}$

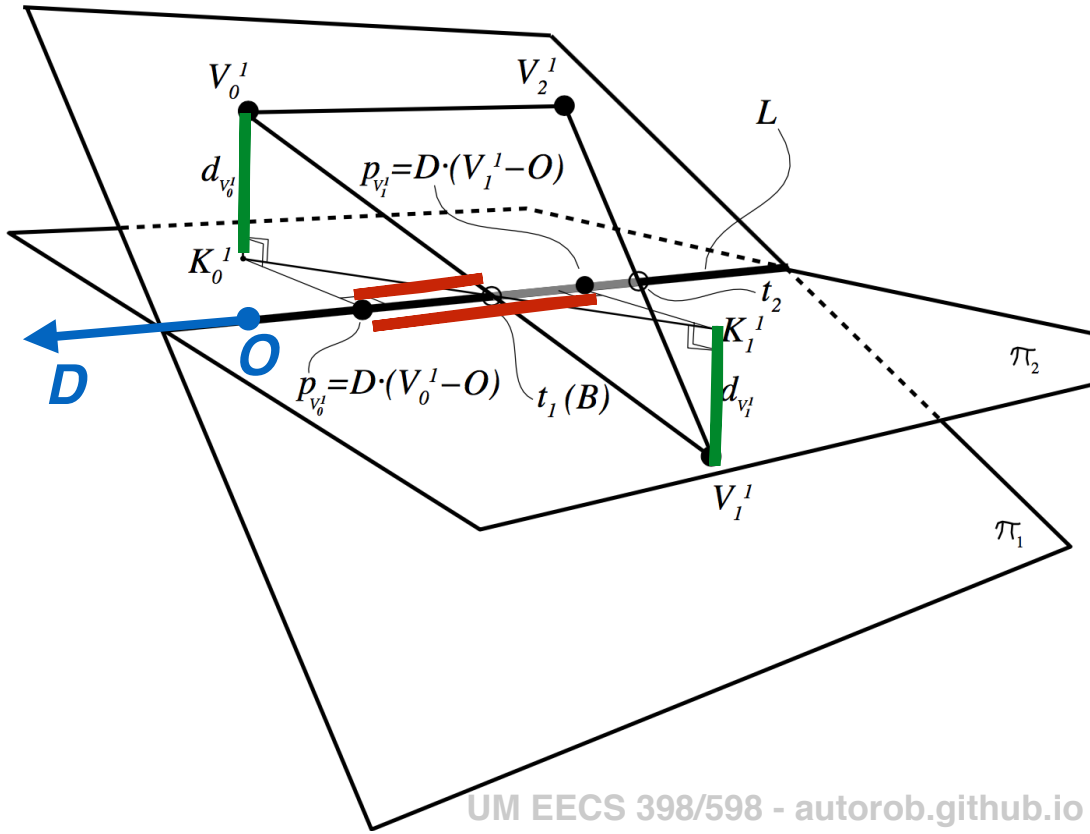
1. Compute plane equation of triangle 2.
2. Reject as trivial if all points of triangle 1 are on same side.
3. Compute plane equation of triangle 1.
4. Reject as trivial if all points of triangle 2 are on same side.
5. Compute intersection line and project onto largest axis.
6. Compute the intervals for each triangle.
7. Intersect the intervals.



Ratios between similar triangles $\triangle \boldsymbol{v_0 K_0 B}$ and $\triangle \boldsymbol{v_1 K_1 B}$ yield relation of t_1 to p_0 and p_1

$$(t_1 - p_0) / d_0 = (p_1 - p_0) / (d_0 - d_1)$$

1. Compute plane equation of triangle 2.
2. Reject as trivial if all points of triangle 1 are on same side.
3. Compute plane equation of triangle 1.
4. Reject as trivial if all points of triangle 2 are on same side.
5. Compute intersection line and project onto largest axis.
6. Compute the intervals for each triangle.
7. Intersect the intervals.

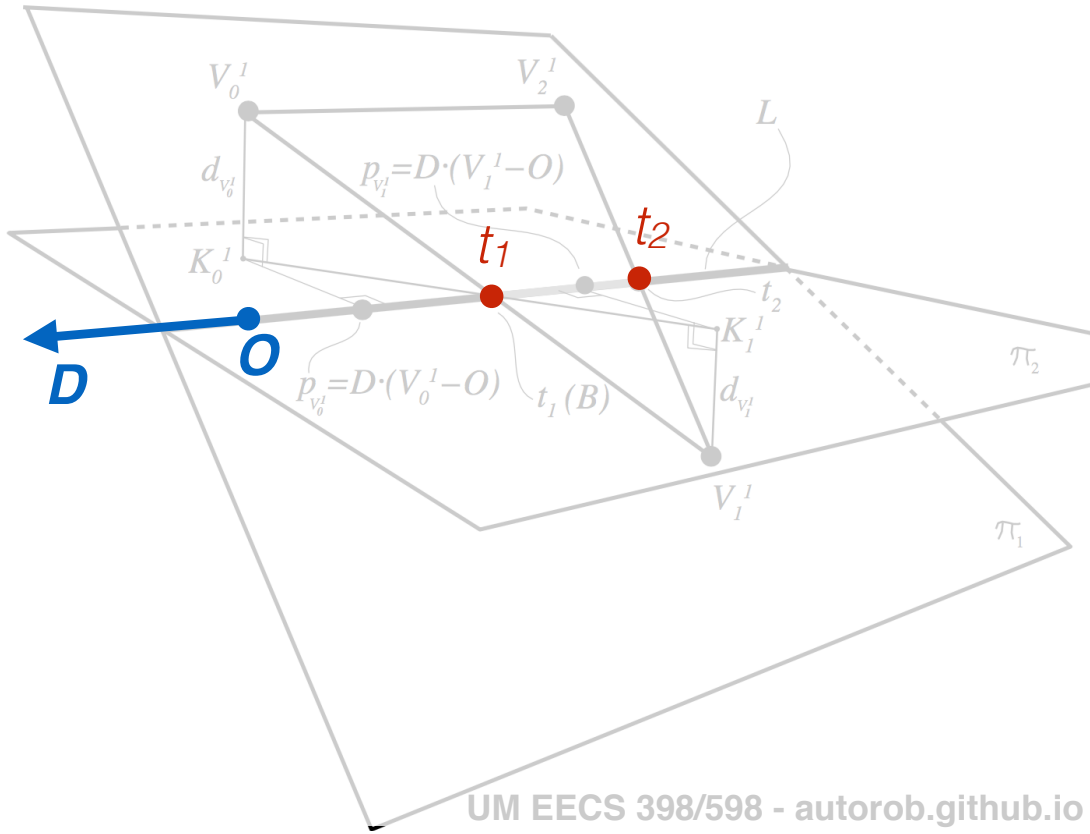


Solve for t_1 to compute interval boundary $\quad t_1 = p_0 + [\,(p_1 - p_0)\, d_0 / (d_0 - d_1)\,]$

Repeat for second vertex $\boldsymbol{v}_2$ of triangle T to compute t_2

Repeat for vertices $\boldsymbol{v}'_1$ and $\boldsymbol{v}'_2$ of triangle T' to compute t'_1 and t'_2

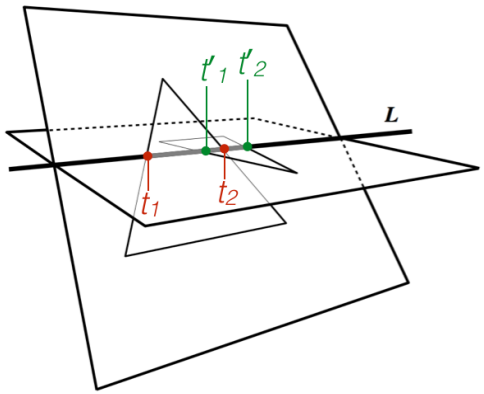
1. Compute plane equation of triangle 2.
2. Reject as trivial if all points of triangle 1 are on same side.
3. Compute plane equation of triangle 1.
4. Reject as trivial if all points of triangle 2 are on same side.
5. Compute intersection line and project onto largest axis.
6. Compute the intervals for each triangle.
7. Intersect the intervals.



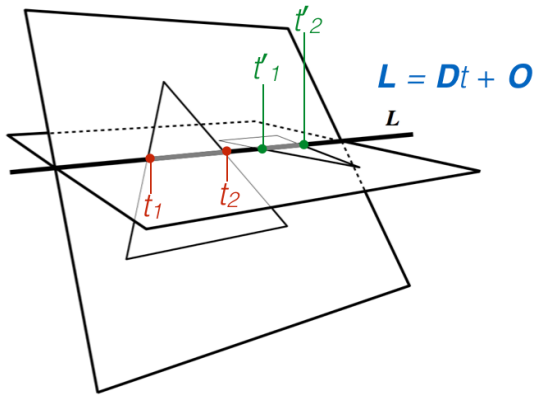
3D Triangle-Triangle Test

1. Compute plane equation of triangle 2.
2. Reject as trivial if all points of triangle 1 are on same side.
3. Compute plane equation of triangle 1.
4. Reject as trivial if all points of triangle 2 are on same side.
5. Compute intersection line and project onto largest axis.
6. Compute the intervals for each triangle.
7. Intersect the intervals.

Collision



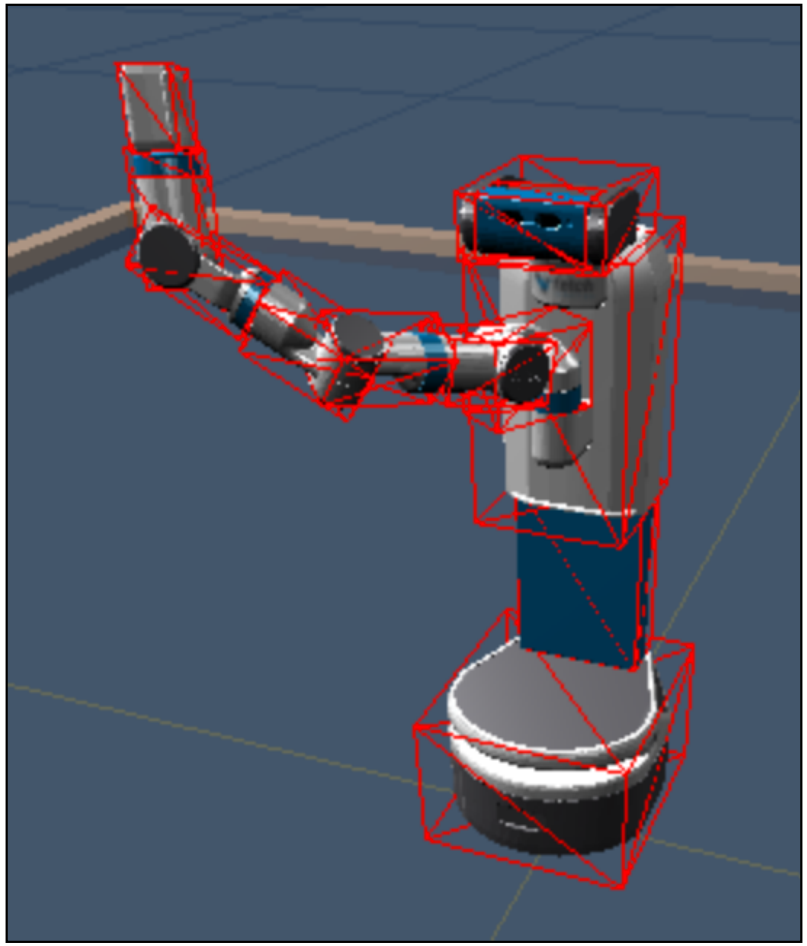
Non-collision

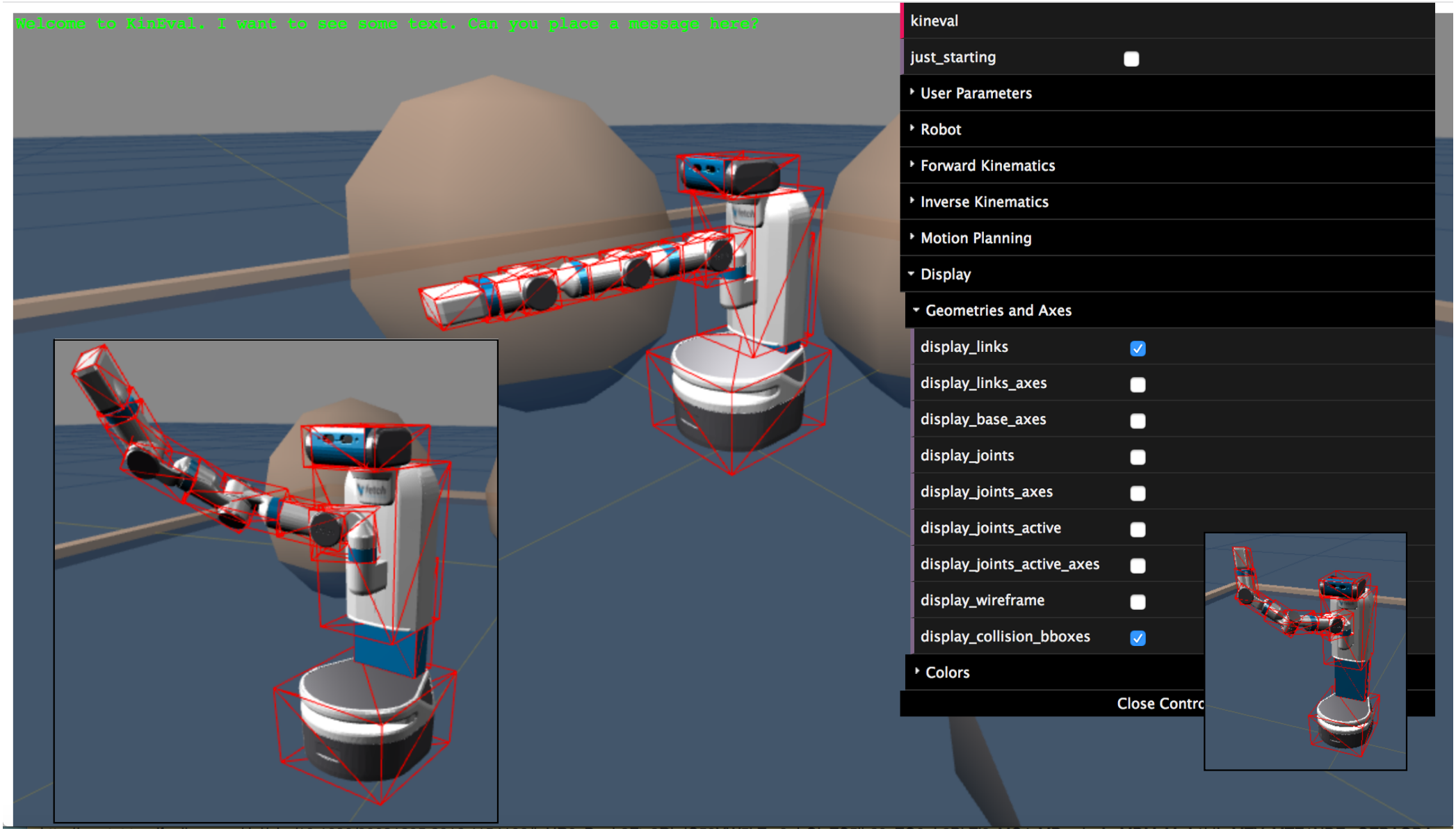


How many triangle tests must be performed?

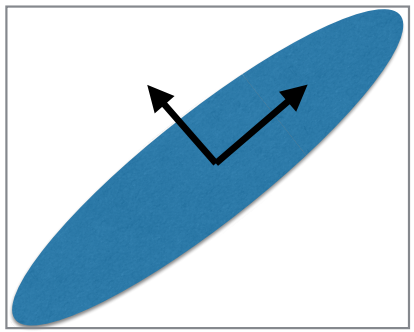
How many triangle tests must be performed?

Can we reduce the number of tests to evaluate?

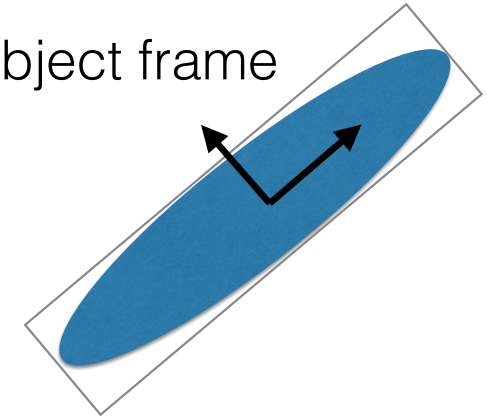



Welcome to KinEval. I want to see some text. Can you place a message here?
kineval
just_starting
User Parameters
Robot
Forward Kinematics
Inverse Kinematics
Motion Planning
Display
Geometries and Axes
display_links
display_links_axes
display_base_axes
display_joints
display_joints_axes
display_joints_active
display_joints_active_axes
display_wireframe
display_collision_bboxes
Colors
Close Contro

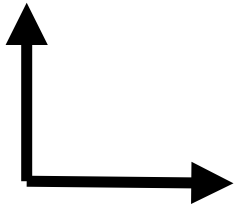
Bounding Boxes



Axis-aligned Bounding Box
(AABB)



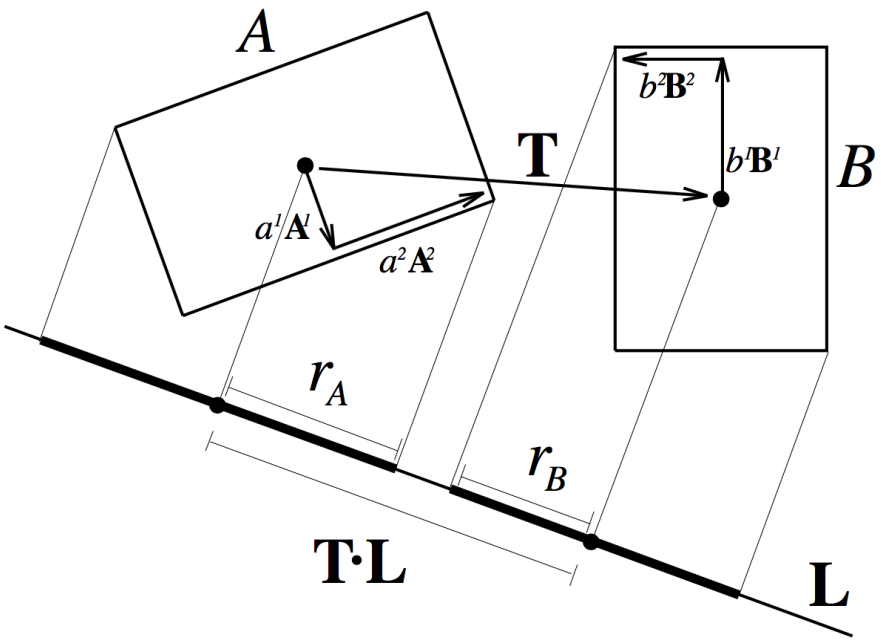
Oriented Bounding Box
(OBB)

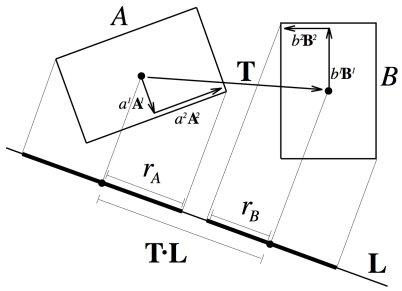


world frame

Gottschalk et al. 1996

Separating Axis Theorem





Hyperplane separation theorem

From Wikipedia, the free encyclopedia

(Redirected from Separating axis theorem)

In geometry, the **hyperplane separation theorem** is a theorem about disjoint convex sets in *n*-dimensional Euclidean space. There are several rather similar versions. In one version of the theorem, if both these sets are closed and at least one of them is compact, then there is a hyperplane in between them and even two parallel hyperplanes in between them separated by a gap. In another version, if both disjoint convex sets are open, then there is a hyperplane in between them, but not necessarily any gap. An axis which is orthogonal to a separating hyperplane is a **separating axis**, because the orthogonal projections of the convex bodies onto the axis are disjoint.

The hyperplane separation theorem is due to Hermann Minkowski. The Hahn–Banach separation theorem generalizes the result to topological vector spaces.

A related result is the supporting hyperplane theorem.

In geometry, a **maximum-margin hyperplane** is a hyperplane which separates two 'clouds' of points and is at equal distance from the two. The margin between the hyperplane and the clouds is maximal. See the article on Support Vector Machines for more details.

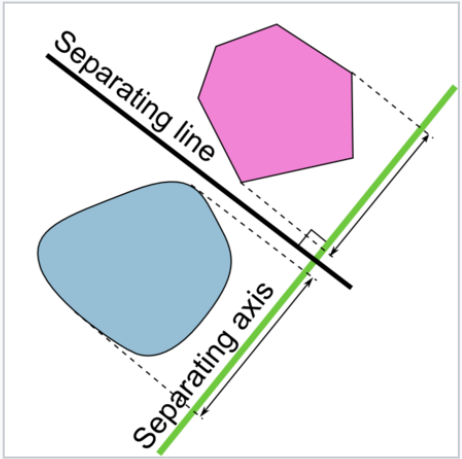
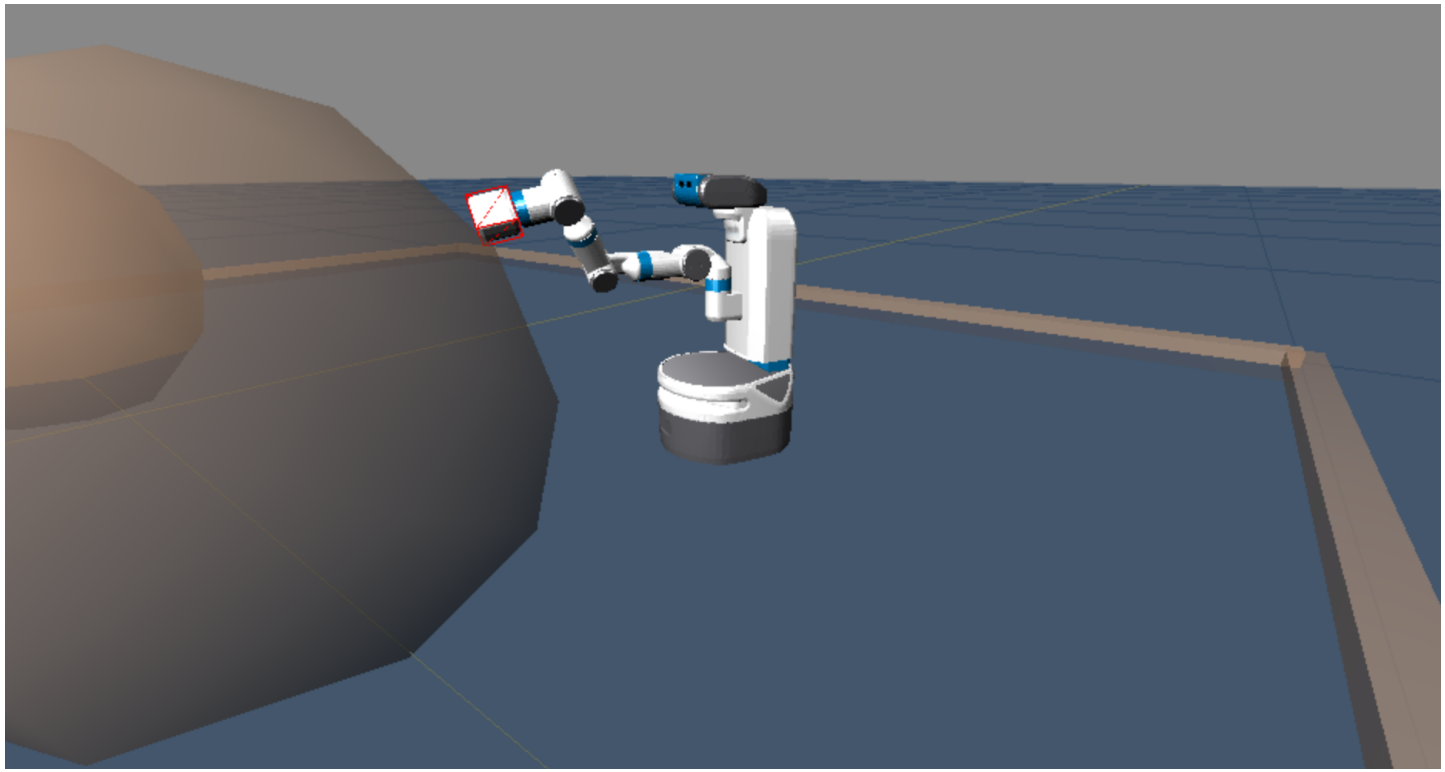


Illustration of the hyperplane separation theorem.

KinEval approximates obstacles with bounding spheres

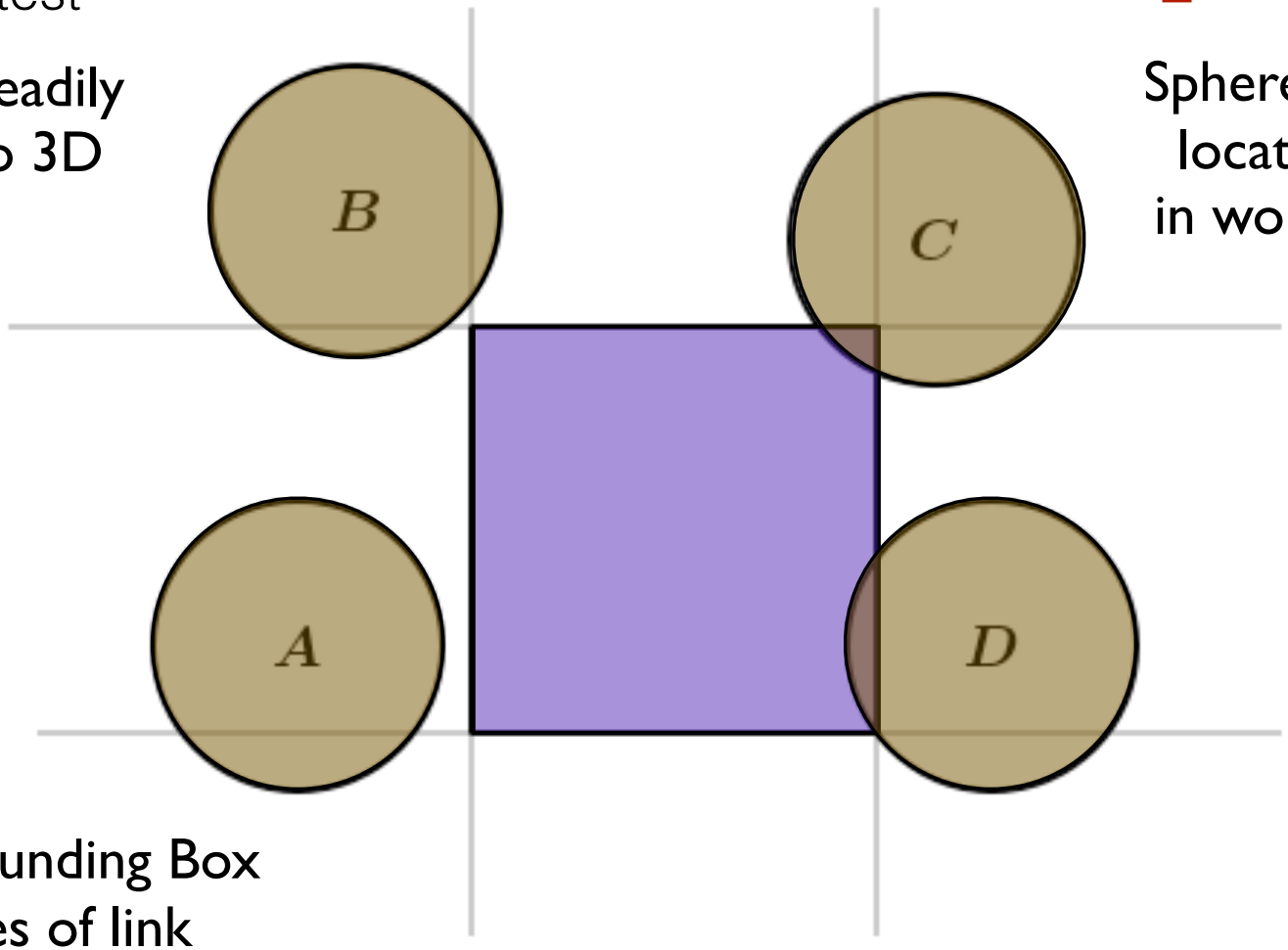


Sphere-bbox test

2D example readily generalizes to 3D

`robot_obstacles[i]`

Sphere obstacles with location and radius in world coordinates



Axis Aligned Bounding Box in coordinates of link

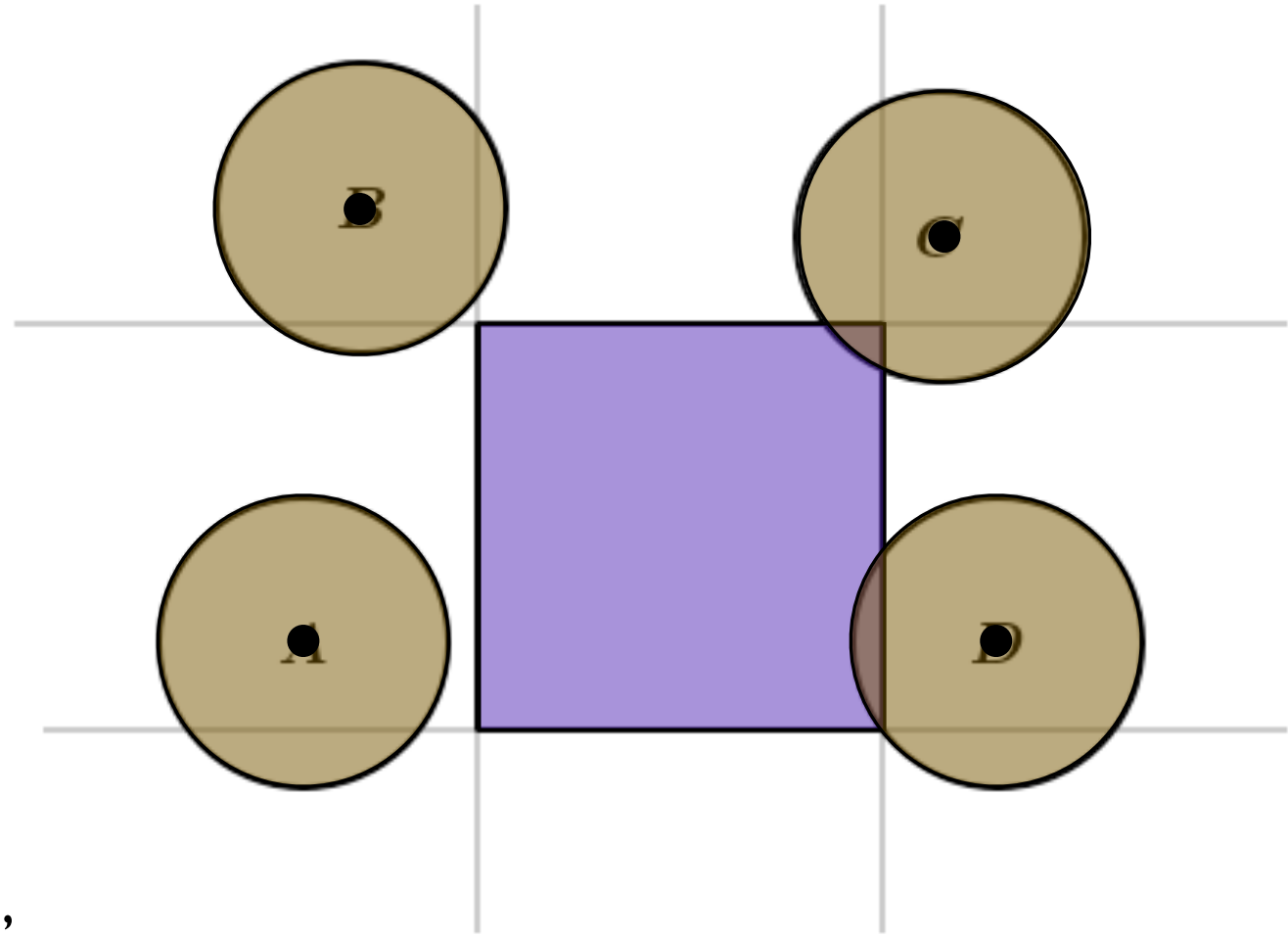
```
robot.links[x].bbox = [[x_min,y_min,z_min], [x_max,y_max,z_max]]
```

Sphere-bbox test

If sphere separable from AABB in any dimension, return no collision

`loc_y-radius>y_max?`

`loc_y+radius<y_min?`

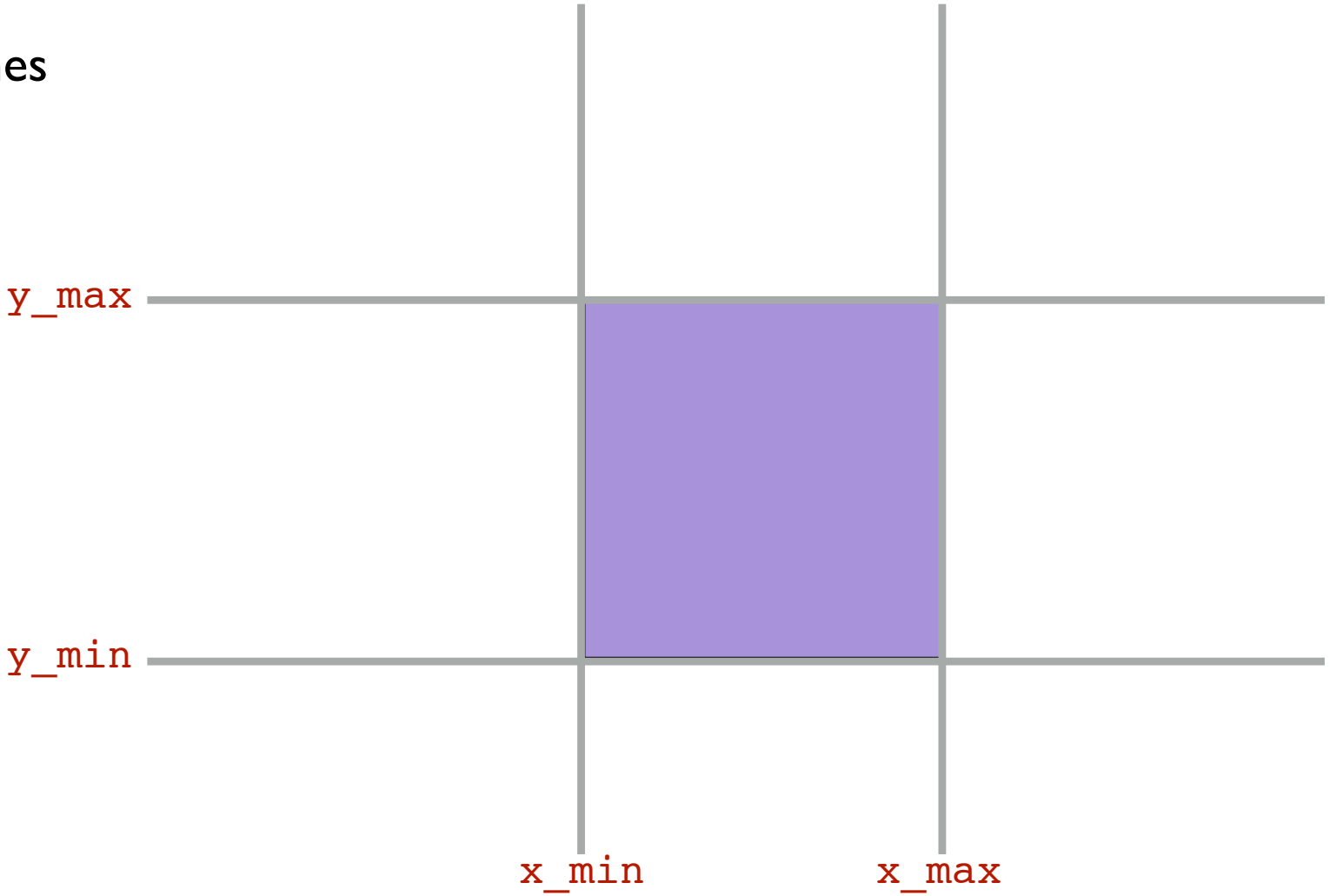


`loc_x+radius<x_min?`

`loc_x-radius>x_max?`

If sphere collides on all tests, return collision

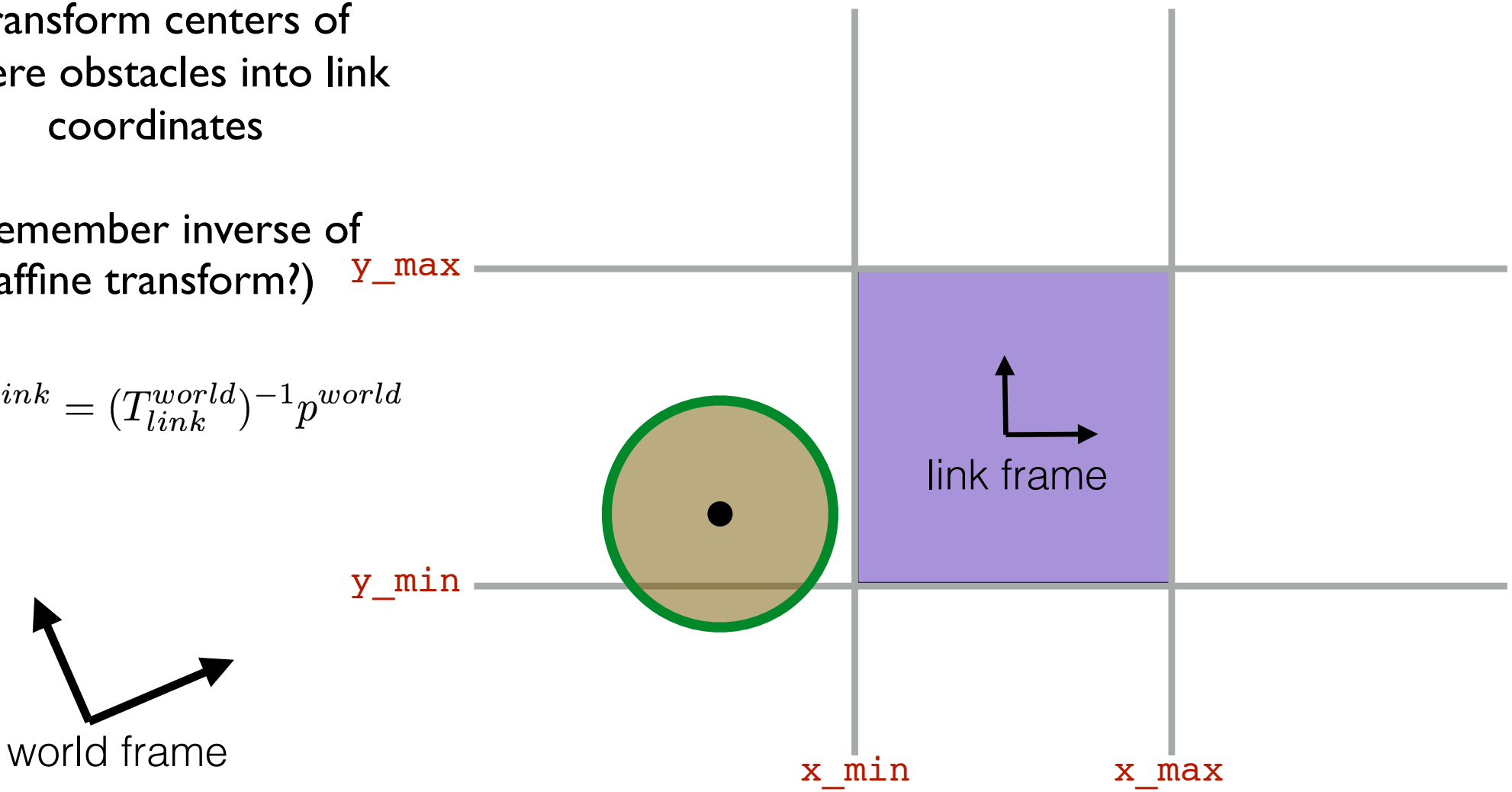
Separating planes



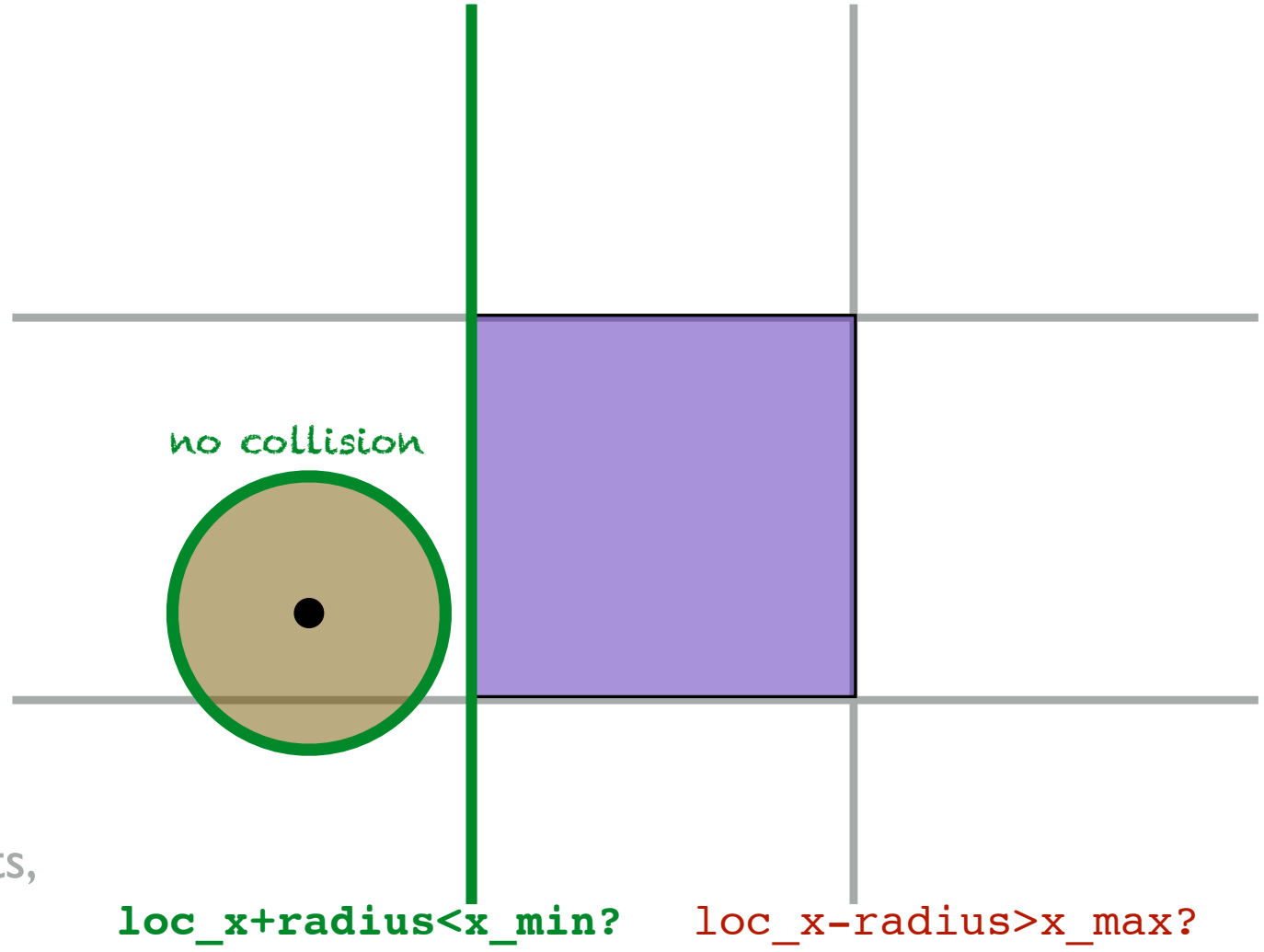
Transform centers of sphere obstacles into link coordinates

(Remember inverse of affine transform?)

$$p^{link} = (T^{world}_{link})^{-1} p^{world}$$

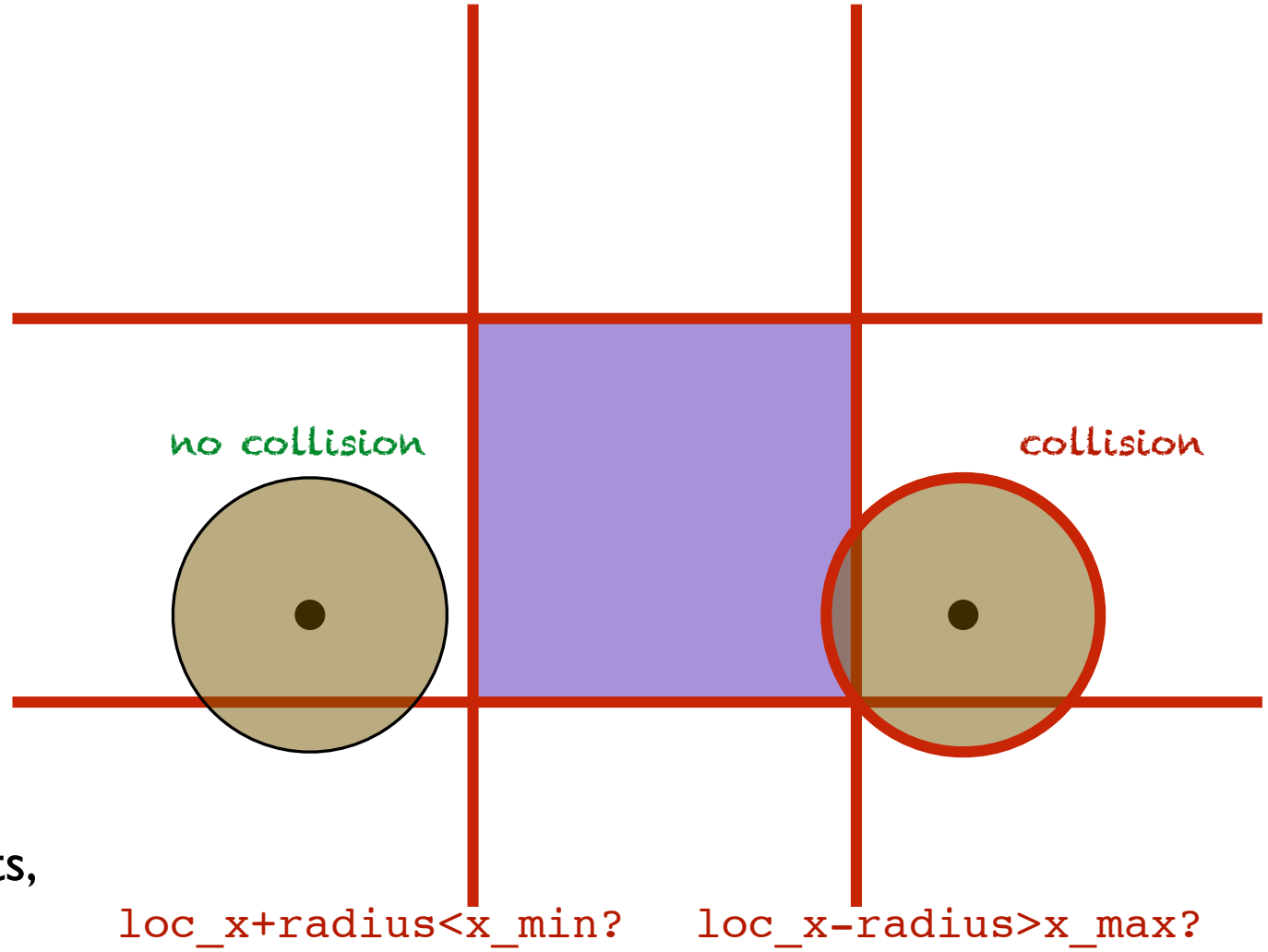


If sphere separable from AABB in any dimension, return no collision



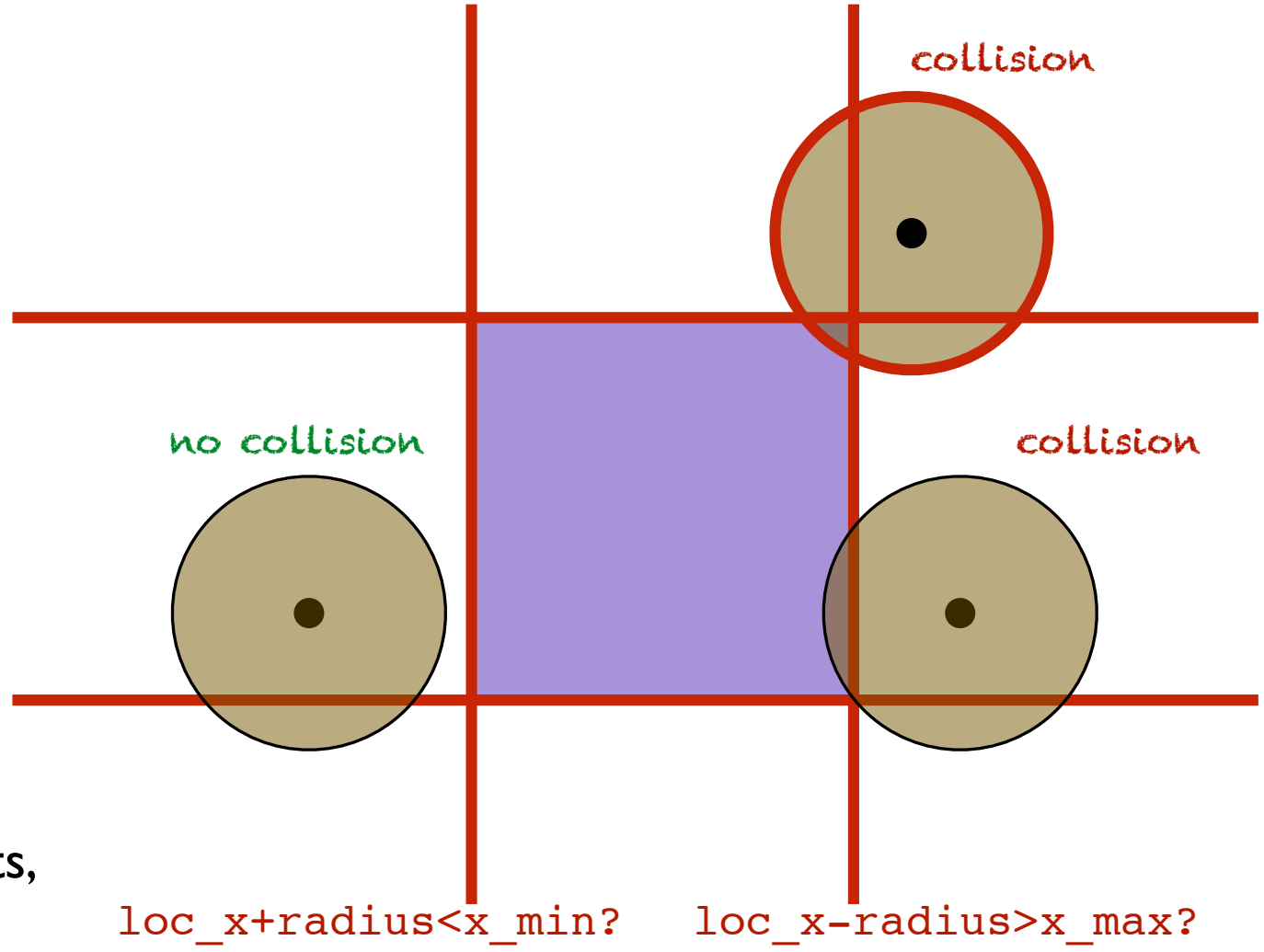
If sphere collides on all tests, return collision

If sphere separable from AABB in any dimension, return no collision



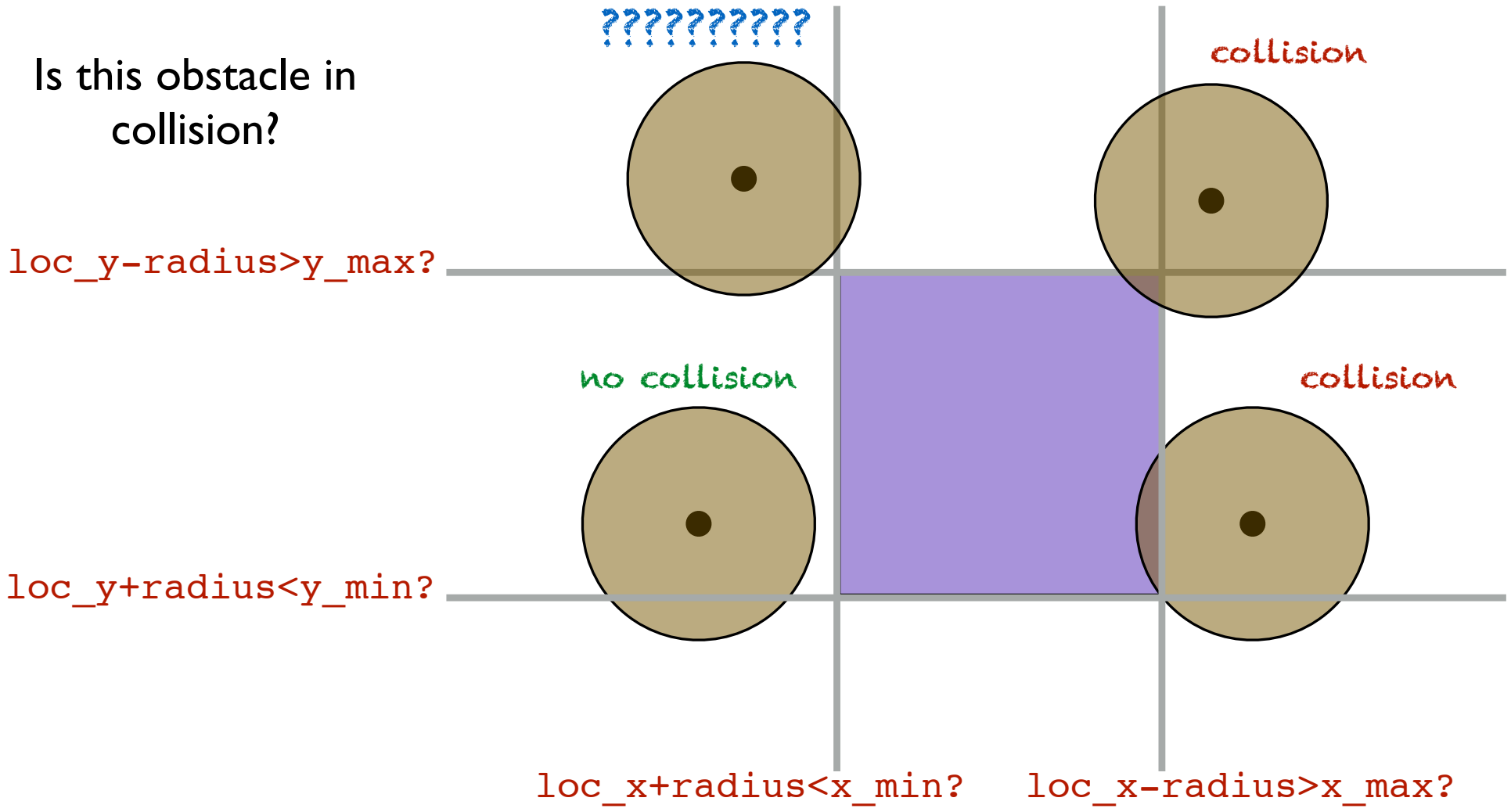
If sphere collides on all tests, return collision

If sphere separable from AABB in any dimension, return no collision

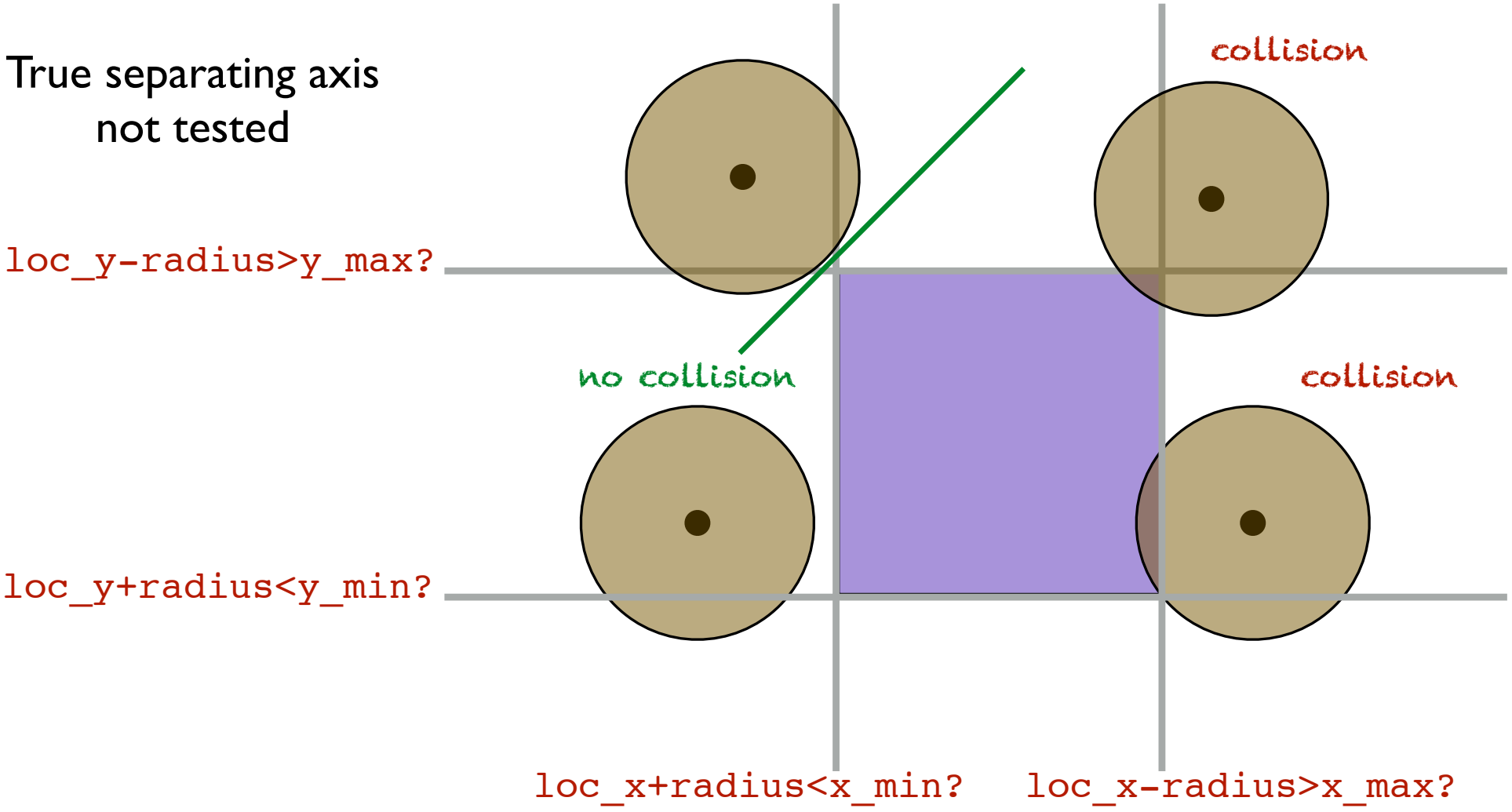


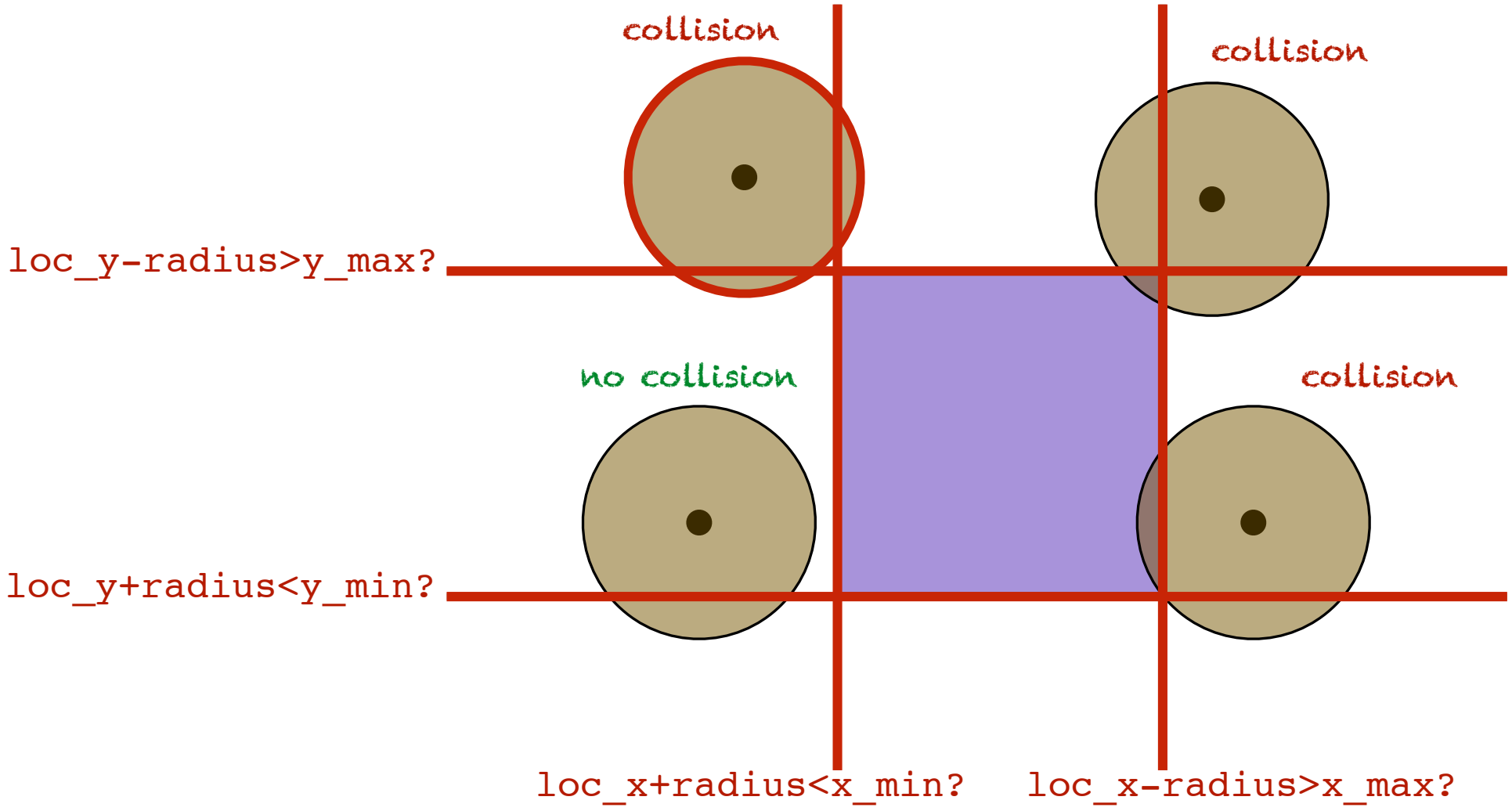
If sphere collides on all tests, return collision

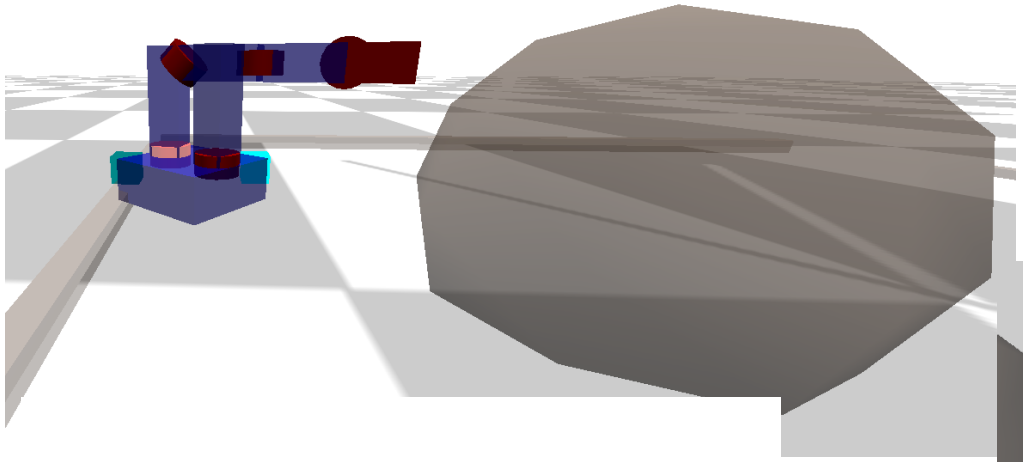
Is this obstacle in collision?

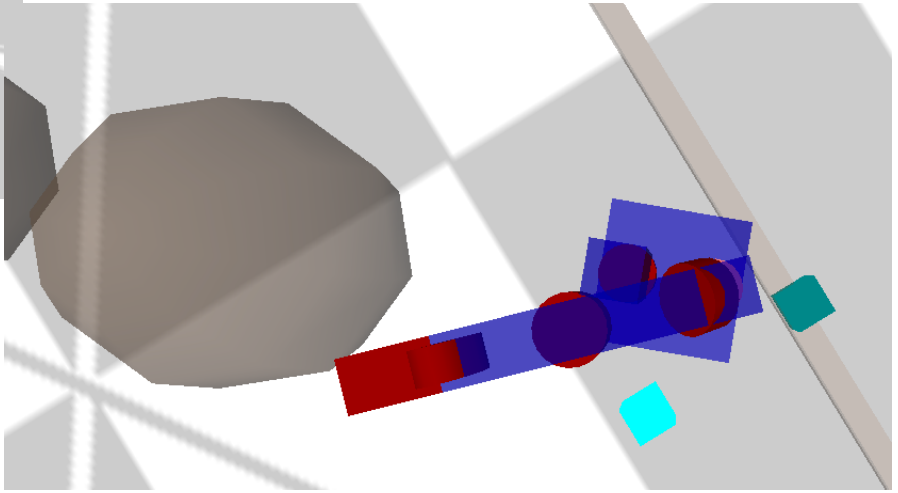


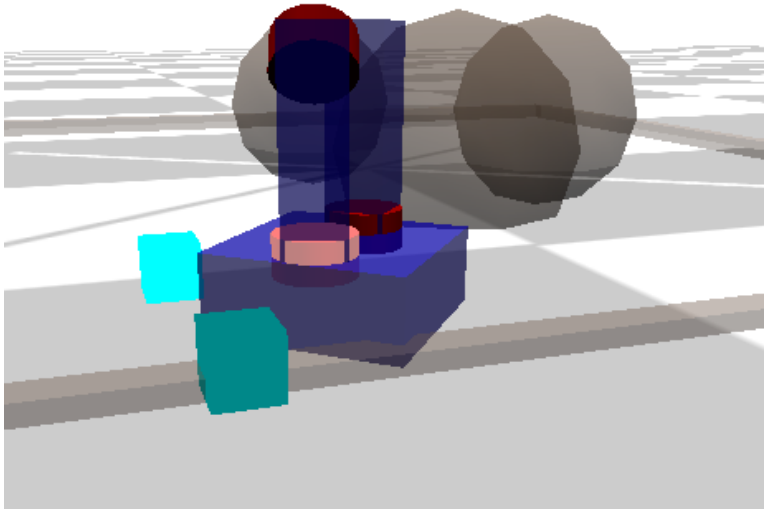
True separating axis not tested




collision
collision
loc_y-radius>y_max?
no collision
collision
loc_y+radius<y_min?
loc_x+radius<x_min?
loc_x-radius>x_max?



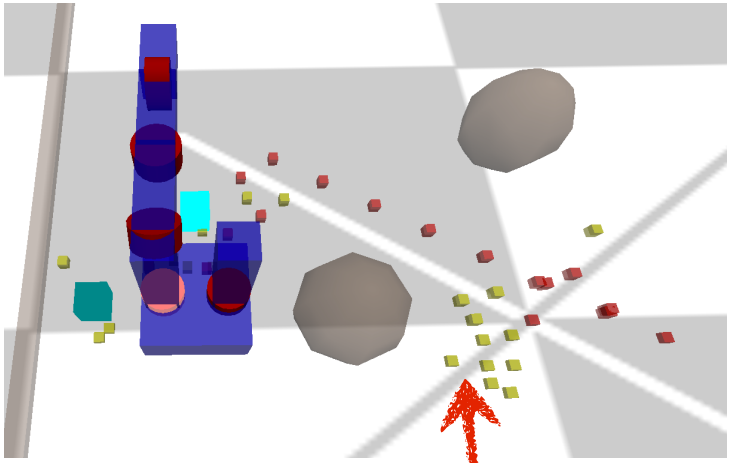




Last notes about planning visualization

kineval_rrt_connect.js

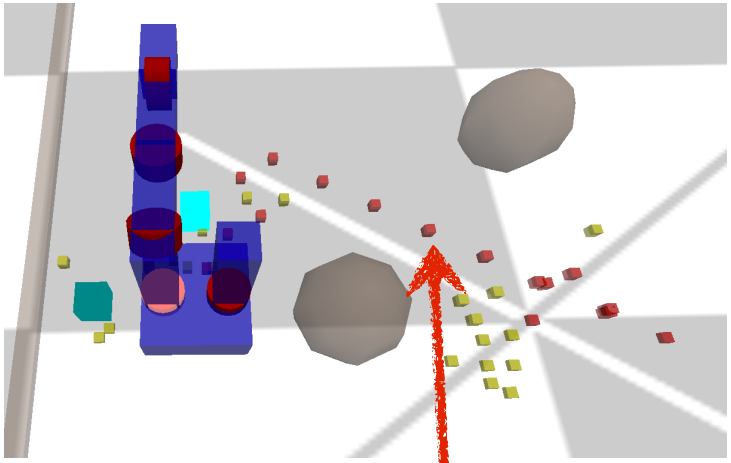
```
function tree_init(q) {

    // create tree object
    var tree = {};

    // initialize with vertex for given configuration
    tree.vertices = [];
    tree.vertices[0] = {};
    tree.vertices[0].vertex = q;
    tree.vertices[0].edges = [];

    // create rendering geometry for base location of vertex configuration
    add_config_origin_indicator_geom(tree.vertices[0]);

    // maintain index of newest vertex added to tree
    tree.newest = 0;

    return tree;
}
```



creates "geom" property of tree vertex with cube at base location for explored tree configuration

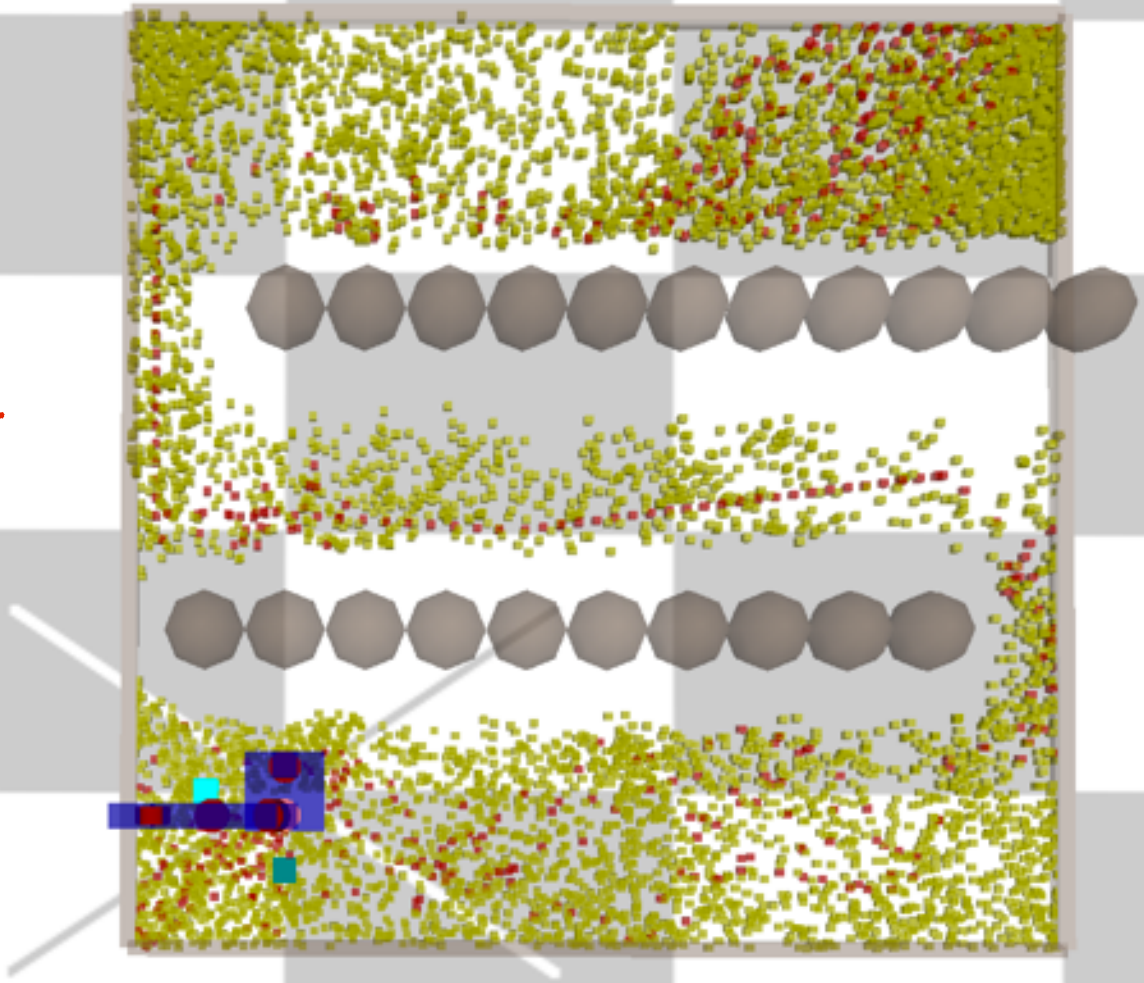
kineval_rrt_connect.js

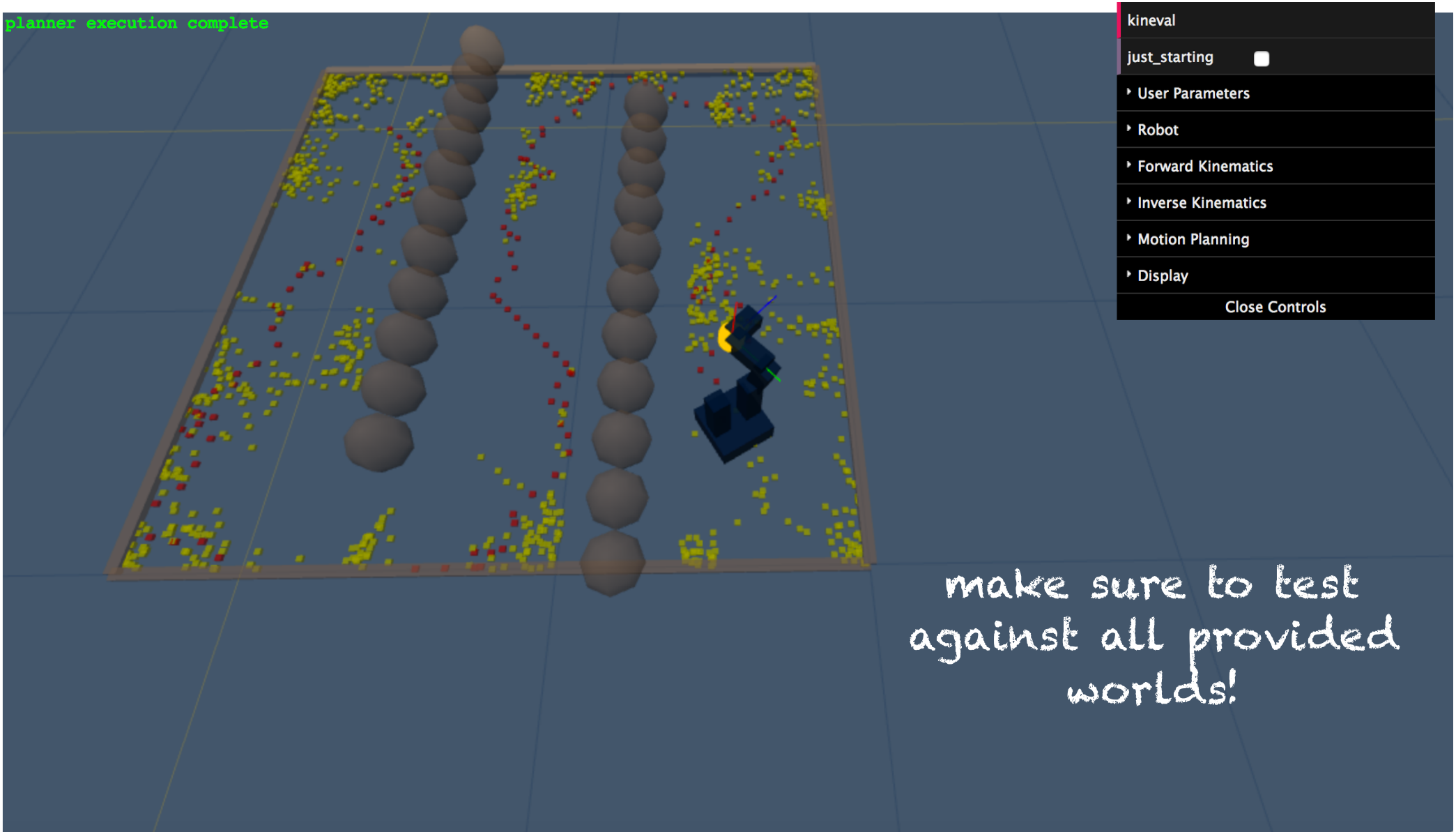
```
for (i=0;i<robot_path.length;i++) {
     robot_path[i].geom.material.color = {r:1,g:0,b:0};
 }
```



found motion path highlighted in red with this code

make sure to test against all provided worlds!




planner execution complete
kineval
just_starting
User Parameters
Robot
Forward Kinematics
Inverse Kinematics
Motion Planning
Display
Close Controls
make sure to test against all provided worlds!