Project 1:
2D Path Planning

- A-star algorithm for search in a given 2D world
- Implement in JavaScript/HTML5
- Heap data structure for priority queue
- 598: DFS, BFS, Greedy
- Submit through your git repository
AutoRob and JavaScript/HTML5

- AutoRob course projects implemented for web browsers using JavaScript/HTML5
- KinEval code stencil was created for this purpose
- Works in commonly-used modern web browsers (Firefox, Chrome, Opera, …)
Stencil code for KinEval (Kinematic Evaluator) for robot control, kinematics, decision, and dynamics in JavaScript/HTML5

https://github.com/ohseejay/kineval-stencil-fall16
Why JavaScript/HTML5? IMHO

- Spectrum of programming languages with two optimal extremes
  - If performance is the priority, use C (maybe C++)
  - If rapid prototyping ("idea-to-code") is priority, use either
    - JavaScript: highly portable, ubiquitous, great visualization/UI
    - Matlab: wealth of numerical support, vectorizable code, easy viz
  - Everything else is a suboptimal tradeoff of these extremes
+ It’s free! (and open)
+ Reasonable learning curve
+ Runs in almost every browser
+ Excellent UI and visualization (see threejs.org for examples)
+ No complicated build process
+ Translates to C++ style thinking
+ Weak typing
+ Live introspection and coding

- Network access limited to HTTP (for security)
- Limited file I/O (for security)
- Floating point issues (all numbers represented in IEEE 754)
- Speed and efficiency (typically JavaScript is interpreted or compiled to intermediate code)
- Weak typing
- Cryptic debugging messages
What is JavaScript/HTML5

- The essential technologies for creating and rendering web pages are HTML5, JavaScript, and CSS
- These technologies structure the run-time environment of web browsers
What is JavaScript/HTML5

- **HyperText Markup Language** (HTML): a markup language for expressing web pages as documents
  - Web browsers read HTML files and render to display
  - Based on the Document Object Model representation
- **JavaScript** (formally ECMAScript): a high-level, dynamic, untyped, interpreted programming language for making “dynamic” web pages
- **Cascading Style Sheets** (CSS): a style sheet language used for describing the presentation of a document
Document Object Model (DOM)

- HTML document defined by nested markup tags (`<br> <h1>text</h1>`) 
- DOM provides programmatic access to the elements as JavaScript objects 
- “document” is always the root object and contains the global scoping context 
- “window” is a globally defined object relating to the browser viewport 
- CSS can change the visual style of elements
Start with a simple example
Example DOM hierarchy

Document

Root element: <html>

Element: <head>

Element: <title>

Text: "My title"

Element: <body>

Attribute: "href"

Element: <a>

Text: "My link"

Element: <h1>

Text: "My header"
hello example

- http://autorob.github.io/examples/hello.html
hello example

- [http://autorob.github.io/examples/hello.html](http://autorob.github.io/examples/hello.html)
<html><body><h1>Chad</h1>was here</p>some paragraph text</p><canvas id="myCanvas" width="400" height="400"></canvas><script>
// this is a comment in JavaScript. it is ignored
// grab the canvas HTML element for drawing
var canvas = document.getElementById("myCanvas");

// grab the canvas drawing context
var ctx = canvas.getContext("2d");

// draw rectangles
ctx.fillRect(50,50,100,100);
ctx.fillRect(0,0,20,400);
ctx.fillRect(0,0,400,20);
ctx.fillRect(0,380,400,20);
ctx.fillRect(380,0,20,400);
</script></body></html>
Chad was here

some paragraph text

<canvas id="myCanvas" width="400" height="400"></canvas>

var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.fillRect(50,50,100,100);
ctx.fillRect(0,0,20,400);
ctx.fillRect(0,0,400,20);
ctx.fillRect(0,380,400,20);
ctx.fillRect(380,0,20,400);
<html>  
  <body>  
    <!-- this is a comment in HTML. it is ignored -->  
    <!-- say your name big -->  
    <h1>Chad</h1>  
    <!-- say something smaller -->  
    was here  
    <!-- say something smaller -->  
    <!-- start a new paragraph -->  
    some paragraph text  
    <!-- create a element for drawing -->  
    <canvas id="myCanvas" width="400" height="400"></canvas>  
    <!-- create an element with JavaScript code to execute -->  
    <script>  
      // this is a comment in JavaScript. it is ignored  
      // grab the canvas HTML element for drawing  
      var canvas = document.getElementById("myCanvas");  
      // grab the canvas drawing context  
      var ctx = canvas.getContext("2d");  
      // draw rectangles  
      ctx.fillRect(50,50,100,100);  
      ctx.fillRect(0,0,20,400);  
      ctx.fillRect(0,0,400,20);  
      ctx.fillRect(0,380,400,20);  
      ctx.fillRect(380,0,20,400);  
    </script>  
  </body>  
</html>
<html> <body> <!-- this is a comment in HTML. it is ignored --> <h1> Chad </h1> <!-- say your name big --> was here <!-- say something smaller --> <p> <!-- start a new paragraph --> some paragraph text </p> <!-- create a element for drawing --> <canvas id="myCanvas" width="400" height="400"></canvas> <!-- create an element with JavaScript code to execute --> <script> // this is a comment in JavaScript. it is ignored // grab the canvas HTML element for drawing var canvas = document.getElementById("myCanvas"); // grab the canvas drawing context var ctx = canvas.getContext("2d"); // draw rectangles ctx.fillRect(50,50,100,100); ctx.fillRect(0,0,20,400); ctx.fillRect(0,0,400,20); ctx.fillRect(0,380,400,20); ctx.fillRect(380,0,20,400); </script> </body> </html>
Chad was here

Some paragraph text

<canvas id="myCanvas" width="400" height="400"></canvas>

// grab the canvas HTML element for drawing
var canvas = document.getElementById("myCanvas");

// grab the canvas drawing context
var ctx = canvas.getContext("2d");

// draw rectangles
ctx.fillRect(50,50,100,100);
ctx.fillRect(0,0,20,400);
ctx.fillRect(0,0,400,20);
ctx.fillRect(0,380,400,20);
ctx.fillRect(380,0,20,400);

</script>
</body> </html>
Chad

was here

Document

Root element
<html>

Element
<h1>

Text "Chad"

Element
<body>

Text "was here"

Element
<p>

Text "new paragraph"

Element
<canvas>

"myCanvas" drawing region

Element
<script>

JS code

<!-- this is a comment in HTML. it is ignored -->
<h1>Chad</h1>
<!-- say your name big -->

was here  <!-- say something smaller -->

<!-- start a new paragraph -->
some paragraph text

<!-- create a element for drawing -->
<canvas id="myCanvas" width="400" height="400"></canvas>

<!-- create an element with JavaScript code to execute -->
<script>

// this is a comment in JavaScript. it is ignored

// grab the canvas HTML element for drawing
var canvas = document.getElementById("myCanvas");

// grab the canvas drawing context
var ctx = canvas.getContext("2d");

// draw rectangles
ctx.fillRect(50,50,100,100);
ctx.fillRect(0,0,20,400);
ctx.fillRect(0,0,400,20);
ctx.fillRect(0,380,400,20);
ctx.fillRect(380,0,20,400);
</script>
</body> </html>
Chad

was here

Document
  Root element <html>
    Element <h1>
    Element <body>
      Element <p>
      Element <canvas>
        JS code
          // grab the canvas HTML element for drawing
          var canvas = document.getElementById("myCanvas");
          // grab the canvas drawing context
          var ctx = canvas.getContext("2d");
          // draw rectangles
          ctx.fillRect(50, 50, 100, 100);
          ctx.fillRect(0, 0, 20, 400);
          ctx.fillRect(0, 0, 400, 20);
          ctx.fillRect(0, 380, 400, 20);
          ctx.fillRect(380, 0, 20, 400);
          </script>
    </body>
  </html>
Chad

was here

new paragraph

<html>
<body>
<h1>Chad</h1>

was here

<p>some paragraph text</p>

<!-- create a element for drawing -->
<canvas id="myCanvas" width="400" height="400"></canvas>

<!-- create an element with JavaScript code to execute -->
<script>
    // grab the canvas HTML element for drawing
    var canvas = document.getElementById("myCanvas");

    // grab the canvas drawing context
    var ctx = canvas.getContext("2d");

    // draw rectangles
    ctx.fillRect(50,50,100,100);
    ctx.fillRect(0,0,20,400);
    ctx.fillRect(0,0,400,20);
    ctx.fillRect(0,380,400,20);
    ctx.fillRect(380,0,20,400);
</script>
</body>
</html>
Chad

was here

new paragraph

<html>
  <body>
    <!-- this is a comment in HTML. it is ignored -->
    <h1>Chad</h1>  <!-- say your name big -->
    was here  <!-- say something smaller -->
    <!-- start a new paragraph --> some paragraph text </p>
    <!-- create a element for drawing -->
    <canvas id="myCanvas" width="400" height="400"></canvas>
    <!-- create an element with JavaScript code to execute -->
    <script>
      // this is a comment in JavaScript. it is ignored
      // grab the canvas HTML element for drawing
      var canvas = document.getElementById("myCanvas");
      // grab the canvas drawing context
      var ctx = canvas.getContext("2d");
      // draw rectangles
      ctx.fillRect(50,50,100,100);
      ctx.fillRect(0,0,20,400);
      ctx.fillRect(0,0,400,20);
      ctx.fillRect(0,380,400,20);
      ctx.fillRect(380,0,20,400);
    </script>
  </body>
</html>
Chad

was here

new paragraph

Document

Root element <html>

Element <h1>

Text "Chad"

Element <p>

Text "was here"

Element <canvas>

Text "new paragraph"

Element <script>

JS code

"myCanvas" drawing region

---

<html> <body> <!-- this is a comment in HTML. it is ignored -->
<h1>Chad</h1> <!-- say your name big -->
was here <!-- say something smaller -->
<p> <!-- start a new paragraph --> some paragraph text </p>
<!-- create a element for drawing -->
<canvas id="myCanvas" width="400" height="400"></canvas>
<!-- create an element with JavaScript code to execute -->
<script>
    // this is a comment in JavaScript. it is ignored
    // grab the canvas HTML element for drawing
    var canvas = document.getElementById("myCanvas");
    // grab the canvas drawing context
    var ctx = canvas.getContext("2d");
    // draw rectangles
    ctx.fillRect(50,50,100,100);
    ctx.fillRect(0,0,20,400);
    ctx.fillRect(0,0,400,20);
    ctx.fillRect(0,380,400,20);
    ctx.fillRect(380,0,20,400);
</script>
</body> </html>
Chad

was here

new paragraph

<html> <body>

<h1>Chad</h1>

was here

<p>new paragraph</p>

<canvas id="myCanvas" width="400" height="400"></canvas>

<script>

// grab the canvas HTML element for drawing
var canvas = document.getElementById("myCanvas");

// grab the canvas drawing context
var ctx = canvas.getContext("2d");

// draw rectangles
ctx.fillRect(50,50,100,100);
ctx.fillRect(0,0,20,400);
ctx.fillRect(0,0,400,20);
ctx.fillRect(0,380,400,20);
ctx.fillRect(380,0,20,400);

</script>
</body> </html>
Chad

was here

new paragraph

[Image of a square]
Chad

was here

new paragraph
Chad

was here

some paragraph text

```javascript
6+2
8
Math.pow(2,3)
8
"6" + "2"
"62"
new_text = "more things to say"
"more things to say"
for (i=0;i<10;i++) { new_text += " " + i; }
"more things to say 0 1 2 3 4 5 6 7 8 9"
expressions entered here are evaluated live
```
wiped everything out and replaced with more things to say 0 1 2 3 4 5 6 7 8 9

```javascript
for (i=0; i<10; i++) { new_text += " " + i; }

"more things to say 0 1 2 3 4 5 6 7 8 9"
```
Let’s try an animation example
point_x = 30.00 point_y = 410.70

hello_anim example

- [http://autorob.github.io/examples/hello_anim.html](http://autorob.github.io/examples/hello_anim.html)
<html> <body onload=init()> 
<!-- init function will be called when body loaded -->

<div id="text_output"> going to put some text here </div>

<!-- create a element for drawing -->
<canvas id="draw_canvas" width=1000 height="400"></canvas>

<script>
// define a function for initialization as: function name_of_function { function_code }
function init() {

    // create a JavaScript object named "point" with two attributes
    // specifying the horizontal and vertical location of the circle
    point = {x: 50, y: 50}

    // function call to start the animation loop
    animate();
}

function animate() {
    requestAnimationFrame(animate); // requests next time step
    update(); // function call to update the state of the animation
    draw(); // function call to draw the current state of the animation
}
...

</script>
</body> </html>
function animate() {
    requestAnimationFrame(animate); // requests next time step
    update(); // function call to update the state of the animation
    draw(); // function call to draw the current state of the animation
}

animate() [invoked at time t]
  ├─ requestAnimationFrame()
  └─ update()
      └─ draw()

animate() [invoked at time t+dt]
  ├─ requestAnimationFrame()
  └─ update()
      └─ draw()

requestAnimationFrame() will have browser call animate() again.
**IMPORTANT** to avoid code that blocks in animate()
function update() {

    // get a reference to the canvas element "draw_canvas" in the document.
    var canvas = document.getElementById("draw_canvas");

    // update the size of the canvas based on dimensions of browser windows
    //   note: window is a global object for the browser window
    canvas.width = window.innerWidth;
    canvas.height = window.innerHeight-50;

    // move the circle forward by assignment
    point.x = point.x + 5;

    // if statement conditionally executes with roughly this structure:
    //   if (condition) { code } else if (condition) {code} else {code}

    // if the circle is at the extent of the canvas, move it back to the start
    if (point.x > canvas.width) {
        point.x = 0;
    }

    // make the circle look like it bouncing using a sin function
    //   note: the Math object has a number of useful functions
    point.y = (canvas.height-60)-Math.abs((canvas.height/2)*Math.sin(point.x/(canvas.width*0.1)));
}

...
One more animation example
point_x = 1119.00 point_y = 464.82
Many examples available online
Another example:

https://github.com/odestcj/superquadric
Rounded cuboid:

http://odestcj.github.io/superquadric/superquadric.html

Increase shape parameters with corresponding key; decrease using shift+key
a = 2.25 b = 1.00 c = 1.00 r = 10.00 s = 10.00 t = 10.00 (i)crement = 0.05
or try a preset: e(lipse, r(ound)ed cuboid, cor(n)ered cuboid, s(pery), (u)nit scaling
Stencil code for KinEval (Kinematic Evaluator) for robot control, kinematics, decision, and dynamics in JavaScript/HTML5

https://github.com/ohseejay/kineval-stencil-fall16
What is version control?

• Maintains a past history of changes for your code (or any project)

• History of changes maintained in a repository

• Basic workflow

  • Code is “checked out” of a repository and then modified

  • The modified code is then “checked in” to the repository

  • Repository maintains history as the “diff” between the two versions
For example... our TED talk
Henry Evans and Chad Jenkins:

Meet the robots for humanity

TEDxMidAtlantic · 10:21 · Filmed Oct 2013

28 subtitle languages
View interactive transcript

Share this idea
Facebook Twitter Email Embed More

1,145,408 Total views
Share this talk and track your influence!

TED Talks are free thanks to support from Infosys
in-browser drone control
Front-end web interface code associated with teleoperation tutorial for AR.Drone using rosbridge/ROS. This is a simple interface to illustrate basic concepts, and not a maintained release. Please refer to robotwebtools.org for the latest and greatest. http://rosbridge.org/doku.php?do=search&id=ar.drone — Edit
Large open source projects…
3D INTERACTIONS
USING THE LATEST IN WEBGL

MULTI-PLATFORM SUPPORT
HARNESSING THE POWER OF ROS

TOWARDS COMPATIBILITY
MORE BROWSERS, MORE ROBOTS.

ROBOT WEB ARCHITECTURE
BRIDGING ROBOTS AND THE WEB

ROSBRIDGE AS A TRANSPORT
USING JSON TO SPEAK TO YOUR ROBOT

A variety of routes are available for architecting a robot web

While ROS works great for applications on the robot, another layer is
September 2016

523 commits

33 contributors

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>623 commits</td>
<td>47 contributors</td>
</tr>
</tbody>
</table>

- **rosapi**: 0.8.3 (7 hours ago)
- **rosbridge_library**: 0.8.3 (7 hours ago)
- **rosbridge_server**: 0.8.3 (7 hours ago)
- **rosbridge_suite**: 0.8.3 (7 hours ago)
- **.gitignore**: Gitignore vim swapfile (a year ago)
- **.travis.yml**: Cleaning up travis configuration (#283) (2 months ago)
- **AUTHORS.md**: authors and license added (3 years ago)
- **CHANGELOG.md**: update the change log (4 years ago)
- **LICENSE**: authors and license added (3 years ago)
- **README.md**: Update README.md (3 years ago)
changelog updated

rctoris committed on Aug 14, 2015

Showing 4 changed files with 36 additions and 0 deletions.

---

5  rosapi/CHANGELOG.rst

@@ -23,6 +23,11 @@ Changelog for package rosapi

0.7.0 (2014-12-02)

+0.7.13 (2015-08-14)
+ Fix catkin_lint issues
+ Contributors: Matt Vollrath
+
0.7.12 (2015-04-07)

---

14  rosbridge_library/CHANGELOG.rst

@@ -34,6 +34,20 @@ Changelog for package rosbridge_library

 request_id --> id
 Contributors: Russell Toris

+0.7.13 (2015-08-14)
+ Nevermind o_O
+ Add test_depend too (just in case)
+ Add dependency on python bson
+ Get parameter at encode time
Version control options

- Concurrent Versioning System (CVS): very old school
- Subversion (SVN): still in significant use
- Mercurial (hg)
- git: used in AutoRob
git stencil quick start

- Create a git repository from the github or bitbucket website
- Install git on your machine
  - OSX: [https://code.google.com/p/git-osx-installer/](https://code.google.com/p/git-osx-installer/)
- Clone the repository to your machine:
  
  ```bash
  git clone https://github.com/yourid/yourrepo.git
  ```
git stencil quick start

• Clone and copy contents of stencil repository the repository to your machine:
  
git clone https://github.com/ohseejay/kineval-stencil-fall16.git

• Copy contents of stencil repository to your repository
  
  cd <your repository directory>
  cp -r <stencil repository directory>/* .

• Commit these changes and then push them to your remote repository
  
  cd <your repository directory>
  git commit -a -m "initial commit"
  git push
The directory where you are working

commit

add (-u) commit

push

clear

pull or rebase

fetch

revert

checkout HEAD

checkout

diff HEAD

diff

The repository on your local computer

Your repository on gitlab|github|…
Tutorial Examples

Chad

was here
new paragraph
Project 0: Heap sort

My Heap Sort

```
check
to sort: 9499 8320 1472 6409 3865 4921 3157 459 5185 797 4635 9311 2575 100 1165 4631 9772 3413 78 319
heap (insert 5949): 5949
heap (insert 8320): 5949 8320
heap (insert 1472): 8320 1472 5949
heap (insert 4921): 1472 5949 8320 1472 4921 8320 5949 3157 459 3413 78 319
heap (insert 3157): 3157 459 797 1472 3865 6409 4921 8320 5949 3191 2575 100 1165 4631 9772 3413 78 319
heap (insert 459): 5949 8320 1472 4921 3157 3865 6409 4994 4921 8320 5949 3191 2575 100 1165 4631 9772 3413 78 319
heap (insert 3413): 3413 797 1472 3865 6409 4921 3157 459 3191 2575 100 1165 4631 9772 3413 78 319
heap (insert 78): 78 100 459 797 1472 3865 6409 4921 3157 459 3191 2575 100 1165 4631 9772 3413 78 319
heap (insert 3191): 3191 2575 100 1165 4631 9772 3413 78 319
sorted: 78 100 3191 3413 797 1165 1472 2575 3157 3865 6409 4921 3157 459 5185 797 4635 9311 2575
```

```
Project 1: 2D Path Planning
Project 2: Pendularm

System
\( t = 162.00 \quad \Delta t = 0.05 \)
integrator = velocity verlet
\( x = -1.26 \)
\( x_{\text{dot}} = -0.00 \)
\( x_{\text{desired}} = -1.26 \)

Servo: active
\( u = -37.32 \)
\( kp = 1500.00 \)
\( kd = 15.00 \)
\( ki = 150.10 \)

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 - toggle servo
s - disable servo
Projects 3-6: KinEval
git basics: commands

• Push completed project to repository (or just to update)
  • add files to a repository: `git add <file listing>`
  • commit changes to local repo: `git commit -a -m "<msg>"`
  • push local changes to a remote repository: `git push`

• Pull to updates your local repository (and workspace) from remote
  • pull remote changes to a local repository: `git pull`
Branching

• Allows different versions to be modified in parallel

• Helpful for projects with multiple collaborators

• Branches are tagged with a descriptive name

• Conflicts between versions must be resolved in merging (“pull request”)
AutoRob branches

• There will be no collaborator conflicts.

• You contribute code to the master branch. We contribute grading,

• But, you need to keep working while your submitted projects are graded

• You can create a new branch to build upon your code in parallel to grading

• Once complete, your new branch can be merged back into master branch
git basics: commands

- create a local copy of a repository branch:
  - `git clone -b <branch_name> <repo url>`
- switch workspace to a branch of a local repository:
  - `git checkout <branch_name>`
Foreshadowing: Project 3

- create a local copy of a repository: `git clone <repo url>`
  - for KinEval, you should now see repo contents in cloned directory
  - view “home.html”
  - examine “kineval/kineval_startingpoint.js”
  - modify directory to fulfill Project 1

- Note: you might need to clone stencil into a temporary directory and copy into a clone of the repository you have created
Running KinEval

• In Firefox browser, simply open “home.html”

• For other browsers, “home.html” may need to be served to avoid throwing a “security” error

  • If you have python, run the simpleHTTPServer from the directory containing “home.html”

    • python -m SimpleHTTPServer

    • point browser to http://localhost:8000/
Running KinEval

- In Firefox browser, simply open “home.html”

- For other browsers, “home.html” may need to be served to avoid throwing a “security” error

- If you have nodejs, install and run simple-server module from the directory containing “home.html”

  - npm install simple-server
  - node simple-server
  - point browser to http://localhost:3000/home.html
Welcome to Kineval. I want to see some text. Can you place a message here?
Welcome to KinEval. I want to see some text. Can you place a message here?
function startingPlaceholderInit() {
    var local_spacing = 0.9;  // variables declared with "var" are local
    global_spacing = 0.9;  // variables declared with "var" are global
    vert_offset = 1;  // this could be useful later
    jitter_radius = 0.02;  // and this too

    my_object = {}; // objects can be created with braces
    my_object.university = "Michigan"; // create object property with an assignment
    my_object["department"] = "EECS"; // equivalent to my.object.department = "EECS"
    my_object.course_number = 398; // this is a number
    my_object.course_number = my_object.course_number*Math.pow(10,3) + 2;
    // this is a number = 398002; + operator adds number
    my_object.course_number = Math.floor(my_object.course_number/1000).toString() + "-
    + "00" + (my_object.course_number%1000).toString();
    // this is a string; + operator concatenates strings

    var string_containing_the_word_subject = "subject";
    my_object[string_containing_the_word_subject] = "robotics";
    // not equivalent to my_object.string_containing_the_word_subject = "robotics"

    Object { university: "Michigan", department: "EECS", course_number: "398-002", subject: "autonomous_robotics" }
// typeof can be used to test whether an object is defined
if (typeof copied_object === 'undefined') {  // if copied_object does not already exist
  console.log(my_object);  // check it out on the console
  console.log(JSON.stringify(my_object));  // same thing as a string

  // objects are copied by reference
  copied_object = my_object;
  copied_object.subject = "autonomous_robotics";  // what is my_object.subject on the console?
  // view object at the console prompt, type "copied_object" and press enter
}

var empty_array = [];  // this creates an object as an empty array
my_array = [8, 6, 7, 5, 3, 0, 9];  // this creates a 7-element array
my_array[6] = 'ni-i-i-ine';  // replace the sixth element with a string

var i;  // local variable for iterating through array with for loop
for (i=0;i<my_array.length;i++) {
  console.log(my_array[i]);
}
// create a text element to display a message
textbar = document.createElement('div'); // create an empty div element
textbar.style.position = 'absolute'; // set element style parameters
textbar.style.width = window.innerWidth-10;
textbar.style.height = 20;
textbar.style.top = 10 + 'px';
textbar.style.left = 10 + 'px';
textbar.style['font-family'] = "Monospace";
textbar.style.color = "#00ff00";
textbar.innerHTML = "autorob.github.io";
document.body.appendChild(textbar); // put element into document body
function startingPlaceholderAnimate() {
    // keyboard is threejs helper for reading keyboard state
    if (keyboard.pressed("x")) {
        textbar.innerHTML = "moving on up"; // make the pieces move up
        // STENCIL: update the vertical offset variable
    }
    else if (keyboard.pressed("z")) {
        textbar.innerHTML = "relax your mind, let your conscience be free"; // stop jittering the pieces
        // STENCIL: update the radius of the jittering
    }
    else if (keyboard.pressed("shift+1")) {
        textbar.innerHTML = "get a move on"; // increase spacing
        // STENCIL: update the global spacing variable
    }
    else if (keyboard.pressed("1")) {
        textbar.innerHTML = "come together"; // decrease spacing
        // STENCIL: update the global spacing variable
    }
    else {
        // make the pieces jitter, and say something more interesting
        textbar.innerHTML = "Welcome to KinEval. I want to see some text. Can you place a message here?"; // set message text
        vert_offset = 1;
        jitter_radius = 0.2;
    }
}
// CREATE TRANSLATION TRANSFORMATION MATRIX

var jsmat = [
  [1, 0, 0, 0],
  [0, 1, 0, 0],
  [0, 0, 1, 0],
  [0, 0, 0, 1]
];

// this offset will perform the centering along the x-axis
jsmat[0][3] = -Object.keys(robot.joints).length*global_spacing/2;

// iterate over each joint of the robot independently, creating a unique translation matrix for each joint, and setting its 3D transform for (x in robot.joints) {

  // jsmat[1][3] corresponds to the y-coordinate of the position for the 3D object
  jsmat[1][3] = (vert_offset+1)+Math.random()*jitter_radius; // the Math object has lots of helpful functions, such as random number generation

  // jsmat[2][3] corresponds to the z-coordinate of the position for the 3D object

  // add spacing offset for translation of next joint geometry
  jsmat[0][3] += global_spacing;

  // apply matrix to transform of body
  robot.joints[x].xform = matrix_copy(jsmat);
}
Happy hacking!