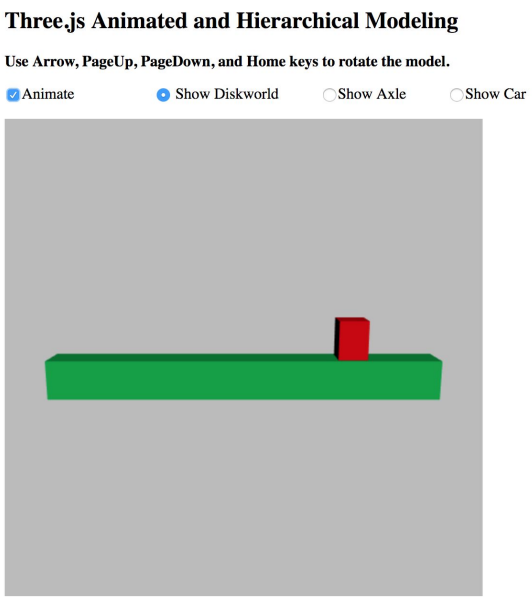
Climbing the Hill

A Colgate-themed platformer

Developing Movement



- Created a Hero object, with parameters for X and Y position and X and Y velocity
- Added a heldKeys list, added and took out keys based on listener
- Added to X and Y velocities based on which keys were being pressed
- Added Jumping and double Jumping, and all the related logic

Modeling and animating the hero

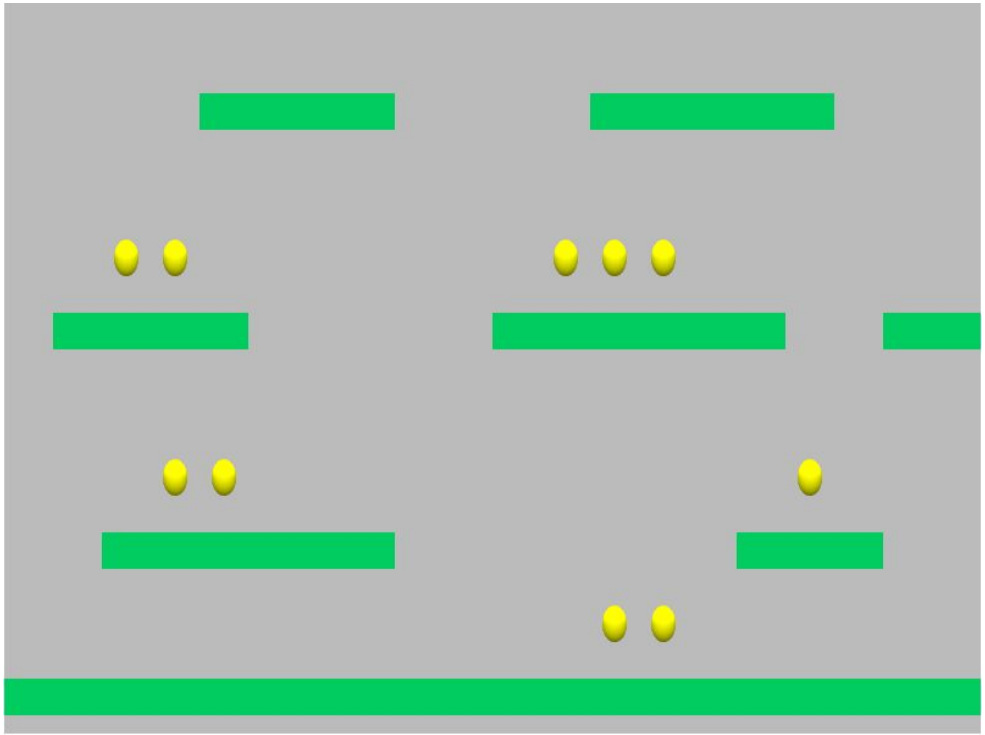
- Modeled hierarchically, so individual parts can be moved in relation to the whole
- Arms swing, legs move, eyes blink
- Head looks up slightly when standing still, as if up at the hill to climb

Level Creation

- Created a level editor to reads level layouts as strings and instantiate game elements in the scene
- Placement of game elements is grid-based
- Grid is used for initial placement only
    - Elements can have motions and interactions based on the game world coordinate system

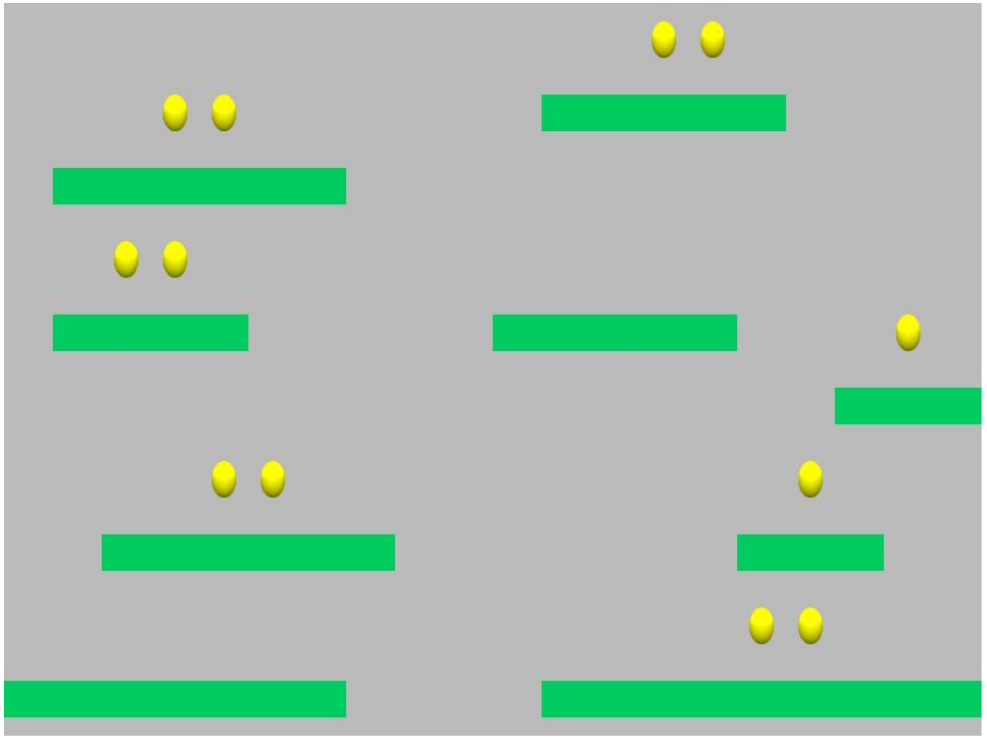
Sample Level

```
var level1 = `
....................
....####....#####...
....................
..oo.......ooo......
.####.....######..##
....................
...oo...........o...
..######.......###..
...........oo......
####################
`;
```



Sample Level

```
var level2 = `
..............oo.....
...oo......#####....
.######.............
..oo................
.####.....#####...o.
..................###
....oo..........o...
..######........###..
...............oo...
#######....#########
`;
```



Collisions

- Tile based, mask based, bounding boxes, penetration resolution
- Ended up using THREE.js raycasting
  - http://stemkoski.github.io/Three.js/Collision-Detection.html
- Drawbacks to raycasting
  - Faces, casting from within a collidable object, high speeds

```
var checkDown = useMin(checkCol(hBoundingBox[2], dirVectors[2], 0, 2),
    checkCol(hBoundingBox[3], dirVectors[2], 0, 2));

if (checkDown) {
  land();
  this.y += 2-checkDown;
}
```

Collisions

```
171
172  function checkCol(pos, dir, near, far) {
173    var ray = new THREE.Raycaster(pos, dir.normalize(), near, far);
174    var collisionResults = ray.intersectObjects( collidableMeshList );
175    if (collisionResults.length > 0) {
176      return collisionResults[0].distance;
177    }
178  }
179
180  function useMin(col1, col2){
181    if (col1 && col2) {
182      return Math.min(col1, col2);
183    }
184    return col1 || col2;
185  }
```