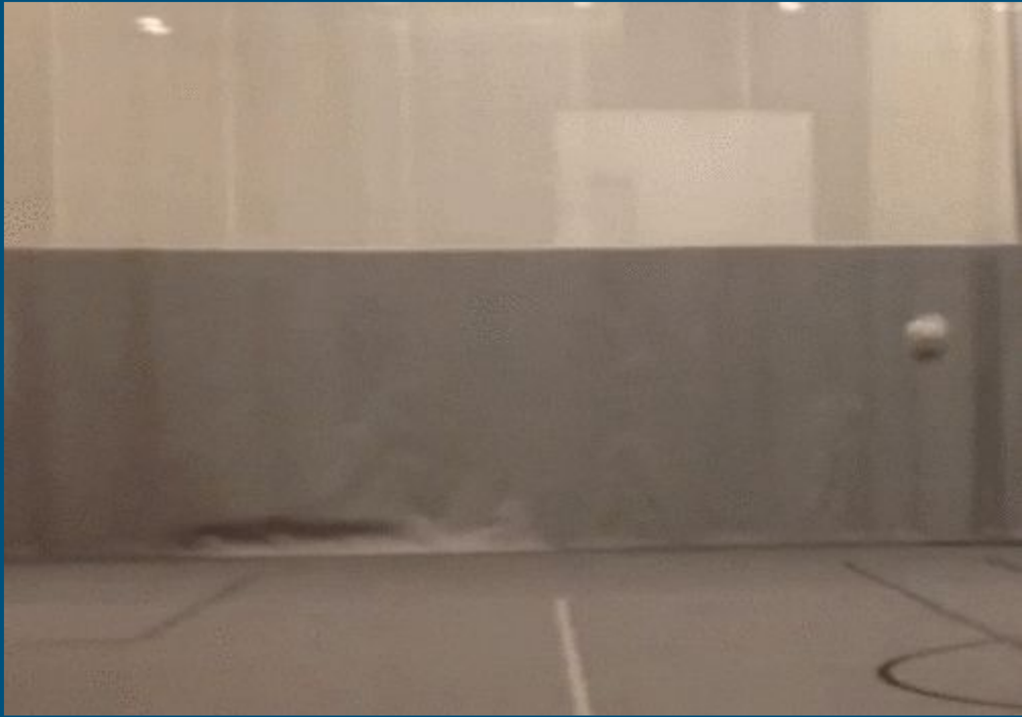
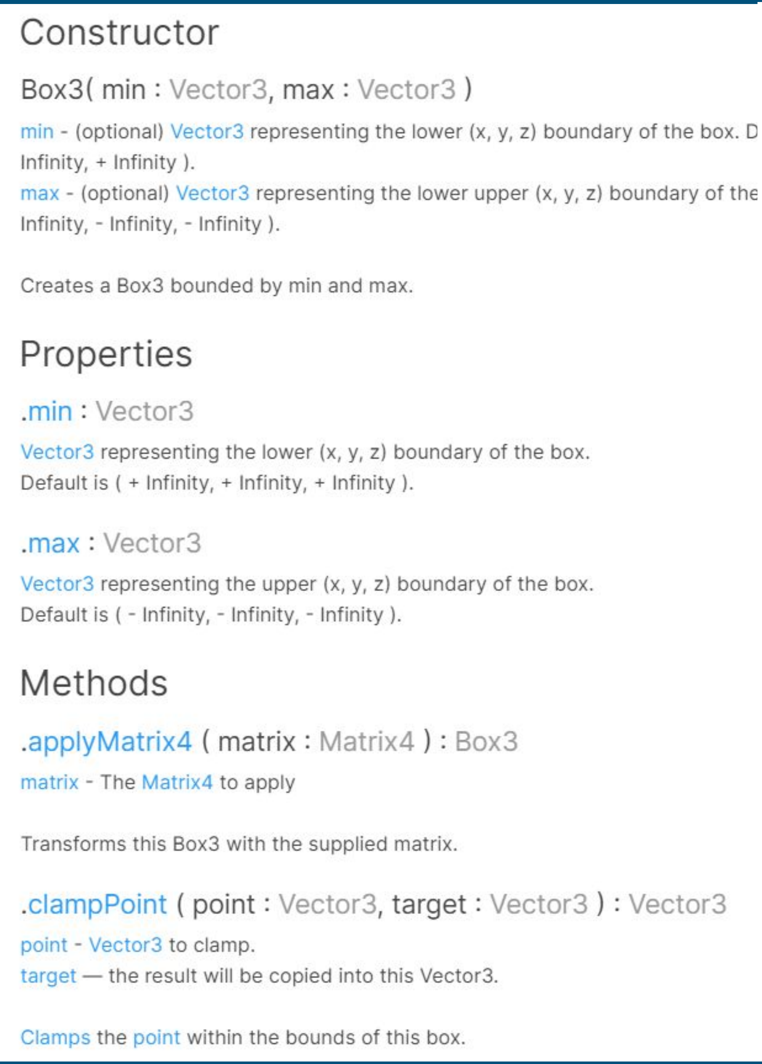
Group 5 Presentation 3

Alex, Matt, Terry and Dylan
Week Three

Object Collision

Because objects in the world should be *objects* and not holograms right?





Constructor

Box3(min : Vector3, max : Vector3)

min - (optional) Vector3 representing the lower (x, y, z) boundary of the box. D
Infinity, + Infinity).
max - (optional) Vector3 representing the lower upper (x, y, z) boundary of the
Infinity, - Infinity, - Infinity).

Creates a Box3 bounded by min and max.

Properties

.min : Vector3

Vector3 representing the lower (x, y, z) boundary of the box.
Default is (+ Infinity, + Infinity, + Infinity).

.max : Vector3

Vector3 representing the upper (x, y, z) boundary of the box.
Default is (- Infinity, - Infinity, - Infinity).

Methods

.applyMatrix4 (matrix : Matrix4) : Box3

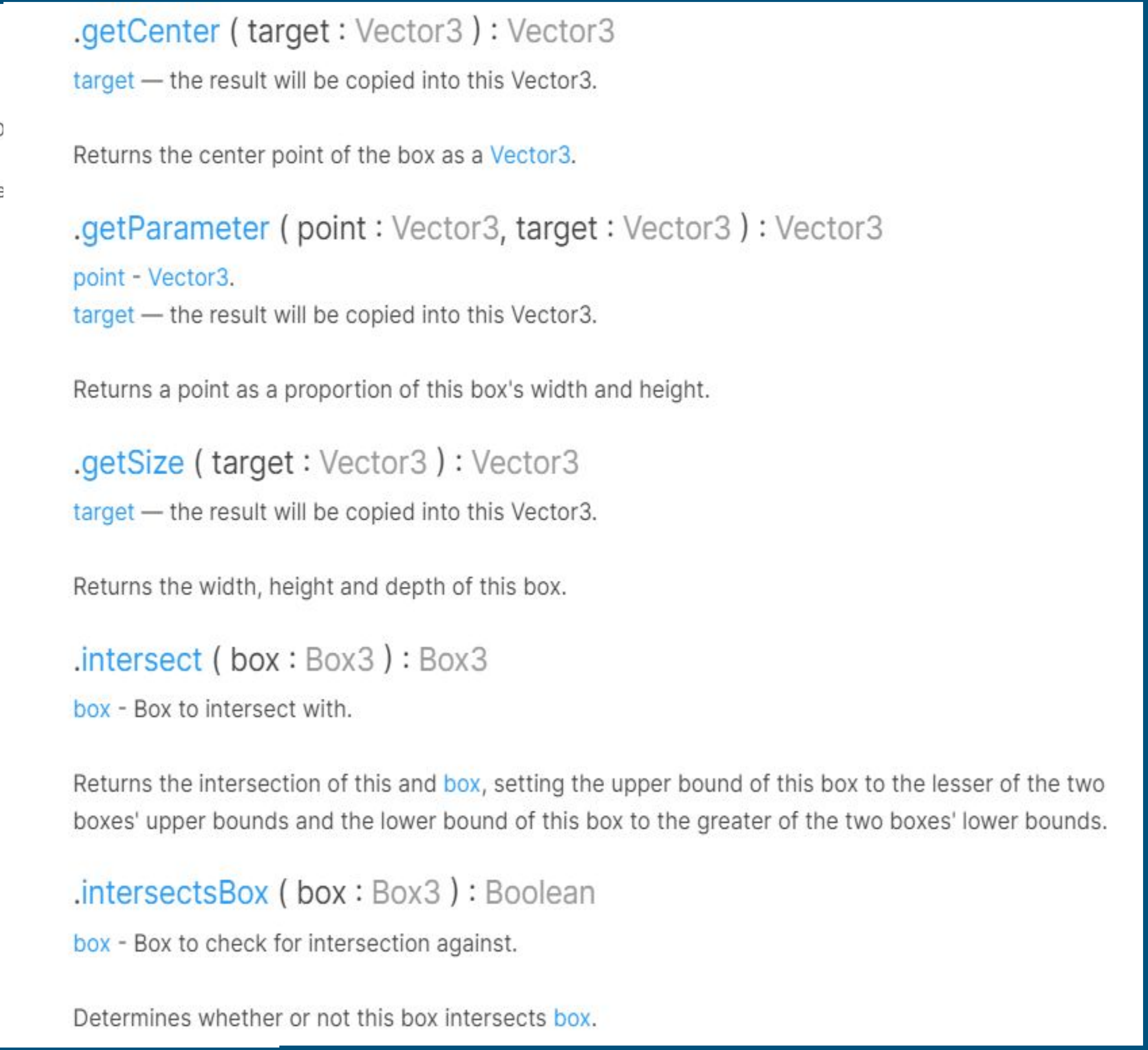
matrix - The Matrix4 to apply

Transforms this Box3 with the supplied matrix.

.clampPoint (point : Vector3, target : Vector3) : Vector3

point - Vector3 to clamp.
target — the result will be copied into this Vector3.

Clamps the point within the bounds of this box.



.getCenter (target : Vector3) : Vector3

target — the result will be copied into this Vector3.

Returns the center point of the box as a Vector3.

.getParameter (point : Vector3, target : Vector3) : Vector3

point - Vector3.
target — the result will be copied into this Vector3.

Returns a point as a proportion of this box's width and height.

.getSize (target : Vector3) : Vector3

target — the result will be copied into this Vector3.

Returns the width, height and depth of this box.

.intersect (box : Box3) : Box3

box - Box to intersect with.

Returns the intersection of this and box, setting the upper bound of this box to the lesser of the two boxes' upper bounds and the lower bound of this box to the greater of the two boxes' lower bounds.

.intersectsBox (box : Box3) : Boolean

box - Box to check for intersection against.

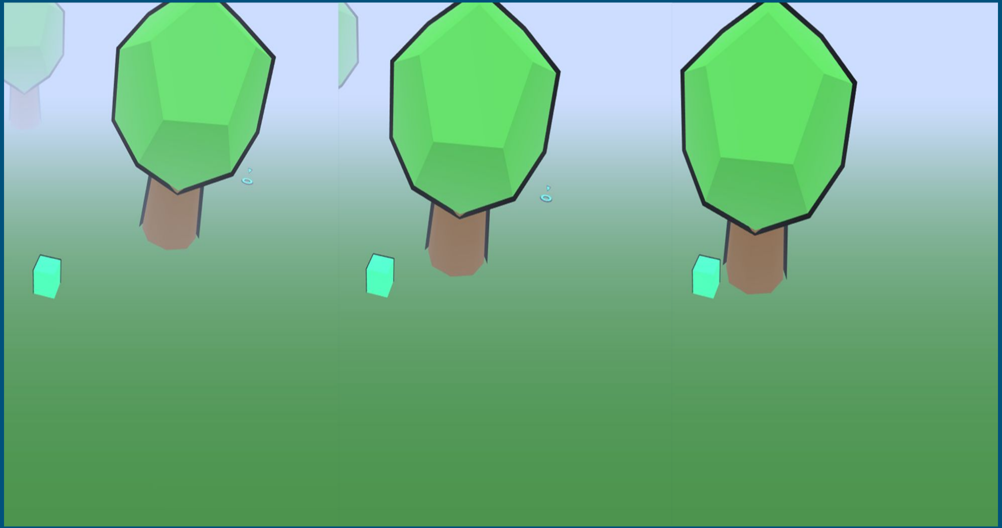
Determines whether or not this box intersects box.

Deeper research into Good BB Examples

There is a cool example some of you might find helpful here:

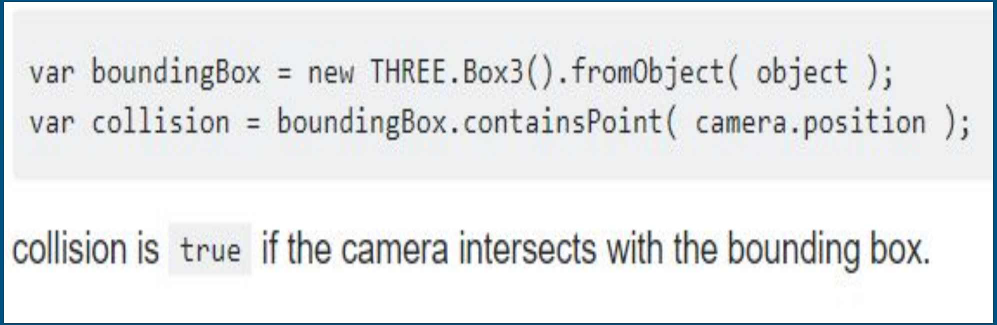
http://www.bryanjones.us/article/basic-threejs-game-tutorial-part-5-collision-detection

(Show Quick Demo)



Ultimately, We HAD Decided Against BB (For Then)

- Approach shown in demo works well for things that are already objects, but for us the player is just a camera, and a camera only occupies a single point.
- Bounding Box Approach Seemed overly complex for our needs



```
var boundingBox = new THREE.Box3().fromObject( object );
var collision = boundingBox.containsPoint( camera.position );
```

collision is `true` if the camera intersects with the bounding box.

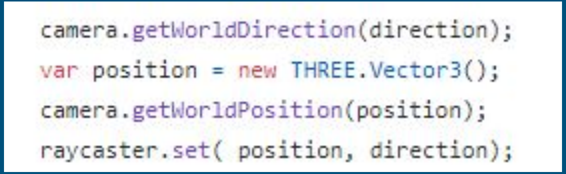
answered May 6 '16 at 11:04

Wilt

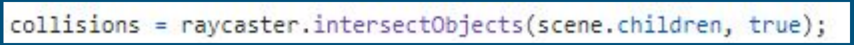
26.7k ● 10 ● 106 ● 150

The Way Monster Freezing Worked At First

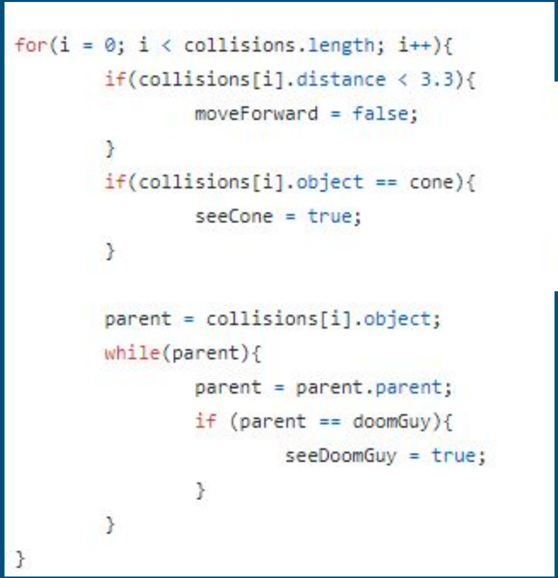
This obviously needed to change, as mentioned on Tuesday, but it worked to at least show the general idea in a demo.



```
camera.getWorldDirection(direction);
var position = new THREE.Vector3();
camera.getWorldPosition(position);
raycaster.set( position, direction);
```

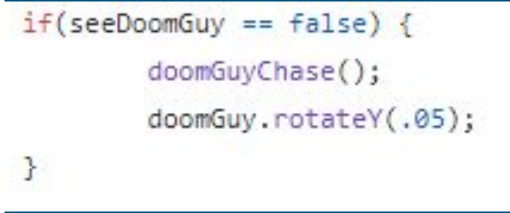


```
collisions = raycaster.intersectObjects(scene.children, true);
```



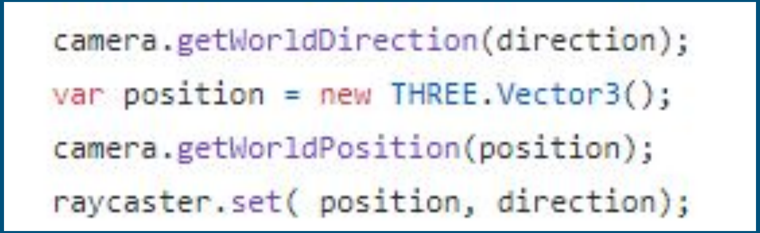
```
for(i = 0; i < collisions.length; i++){
        if(collisions[i].distance < 3.3){
                moveForward = false;
        }
        if(collisions[i].object == cone){
                seeCone = true;
        }

        parent = collisions[i].object;
        while(parent){
                parent = parent.parent;
                if (parent == doomGuy){
                        seeDoomGuy = true;
                }
        }
}
```

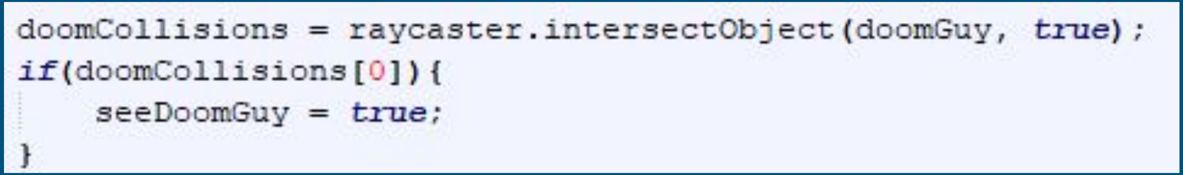


```
if(seeDoomGuy == false) {
        doomGuyChase();
        doomGuy.rotateY(.05);
}
```

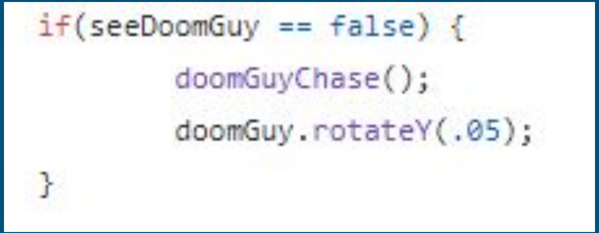
The Way it Works Now (It’s A Lot Better!)



```
camera.getWorldDirection(direction);
var position = new THREE.Vector3();
camera.getWorldPosition(position);
raycaster.set( position, direction);
```

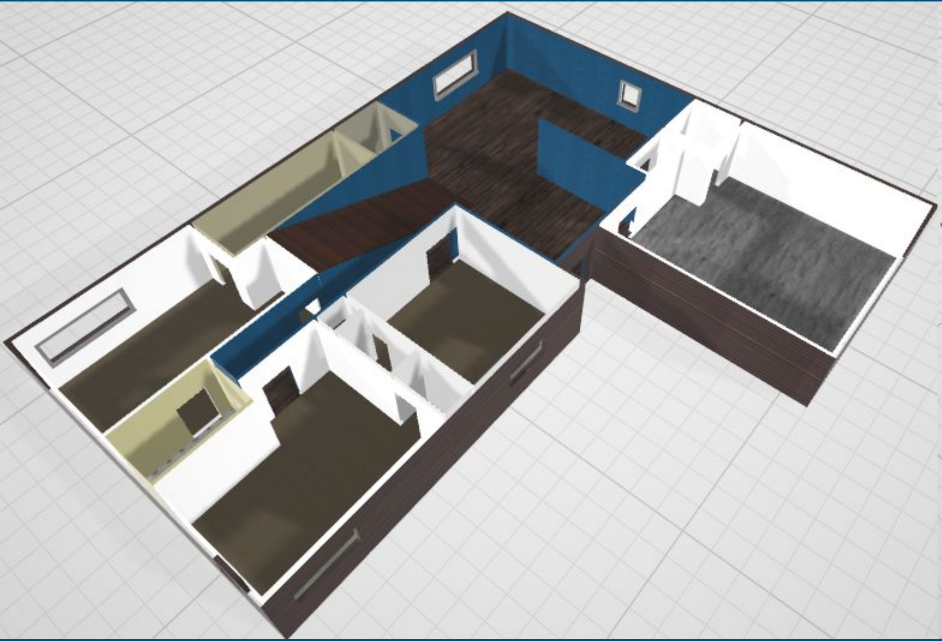
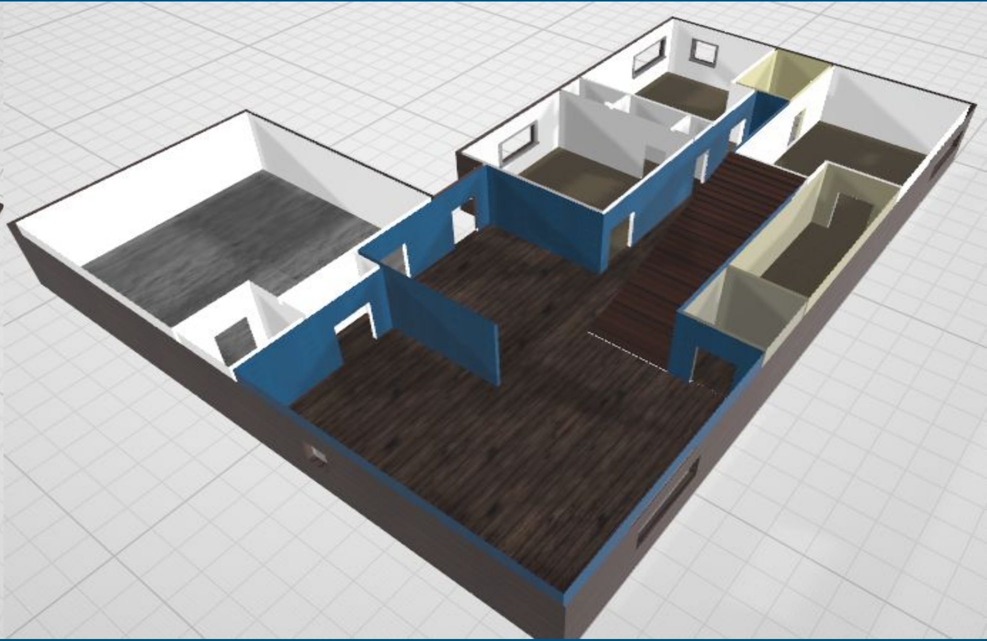


```
doomCollisions = raycaster.intersectObject(doomGuy, true);
if(doomCollisions[0]){
    seeDoomGuy = true;
}
```



```
if(seeDoomGuy == false) {
        doomGuyChase();
        doomGuy.rotateY(.05);
}
```

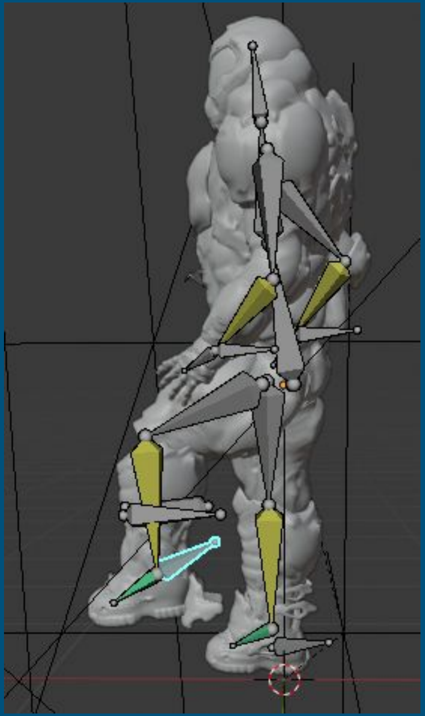
Map Modeling

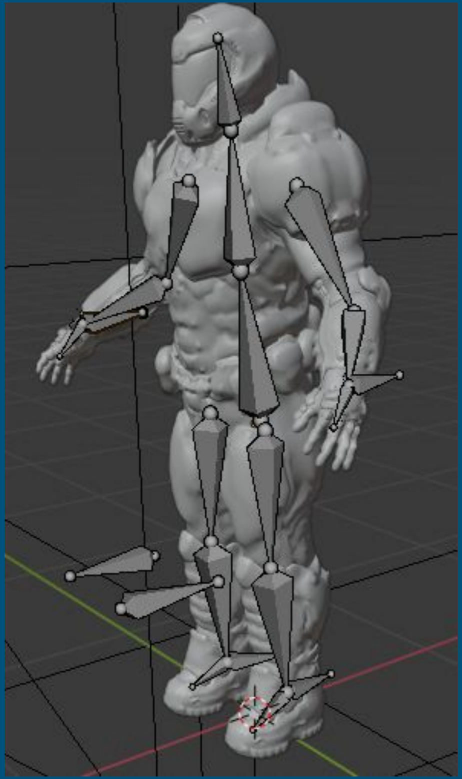



Model Rigging

- Bones
- Inverse Kinematics
  - Bone Constraints
  - The movement of the chain is determined by a "target" bone
- Keep Offset
  - Locks bone with IK bone
- Merge the mesh with the Armature

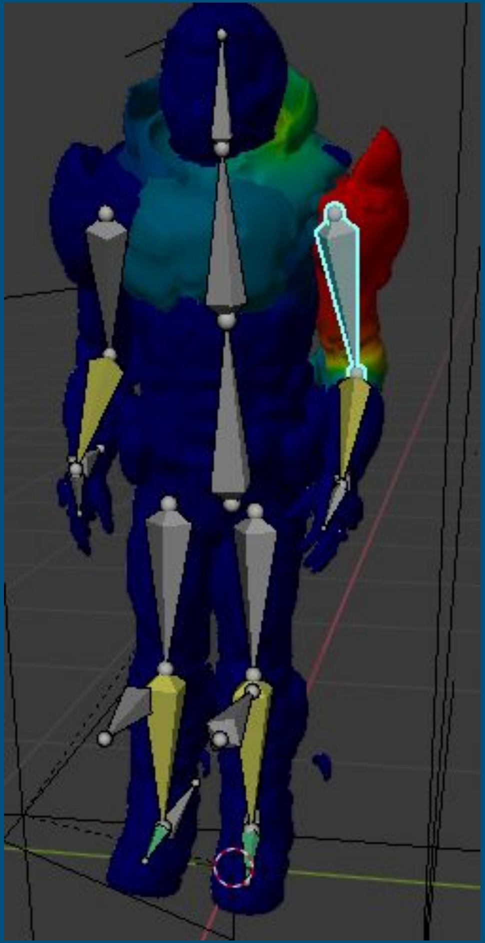
- Enable Infront Setting in Viewport Display
- 'G' to grab, 'S' to scale , 'E' to extrude a bone, and 'R' to rotate
- Ctrl + Tab to scroll through modes.





Animation

- KeyFrames
  - Auto Keying
- Weight Painting
  - Red: High Control
  - Dark Blue: No Control



What Next?

- Second floor and basement of house map
- Adding Collectibles (game objective)
- Adding death if monster touches you
- More Animations, fine tuning animations