# Group 5 Week 7

Alex, Terry, Dylan and Matt

# Trying to Collide Without Hardcoding

- In Peer Feedback, someone suggested using Ammo.js/Physi.js for this, since it innately detects collisions and has a ConvexMesh and ConcaveMesh.
- Remembered Group 2 was struggling with Convex/Concave mesh, so thought if we solved it this way, it could also help them.

# So, We Implemented Physics using Physi.js

- First Step was giving player a physi.js mesh

```
playerCapsule = new Physijs.CapsuleMesh(
    new THREE.CylinderGeometry( 2, 2, 2),
    new Physijs.createMaterial(
    new THREE.MeshBasicMaterial({ color: 0x888888 }), 1,.8), 1);
scene.add(playerCapsule);
```

```
playerCapsule.add(camera);
```

# But How Do You Move?

```
camera.getWorldDirection(direction);
direction.y = 0;
direction.x *= 30;
direction.z *= 30;
if(moveForward){
//  controls.getObject().translateZ(-delta);
    playerCapsule.setLinearVelocity(direction);
}
```

```
var onKeyUp = function ( event ) {
    switch( event.keyCode ) {
        case 38: // up
        case 87: // w
            moveForward = false;
            stopPlayer();
            break;
```

```
function stopPlayer(){
    playerCapsule.setLinearVelocity(new THREE.Vector3(0, 0 , 0));

}
```

# Halfway There… … … … … … …?

- At this point, the player was a physics object and collisions worked properly for simple Physi.js objects we added to the scene. The only thing left was to turn our levels into a static Physi.js object.
- Traversing the level’s mesh’s and directly passing a mesh’s geometry attribute to Physi.js didn’t work, because the Physi.js constructors expect a THREE.js geometry.
- At this point, we were stuck again, so...

# Reaching Out For Help

**Mugen87**

1 2d

Have you considered to use a navigation mesh to restrict the movement of the player? This is a very fast approach and it also allows you to author the navigation mesh in a DCC tool like Blender. A live example looks like so: https://mugen87.github.io/yuka/examples/entity/firstperson/ 2

> R Dylan_Hodge:
>
> The problem is that if I traverse the house's meshes and use the bounding box's setFromObject function of each on them, there is no allowance for convexity/concavity:

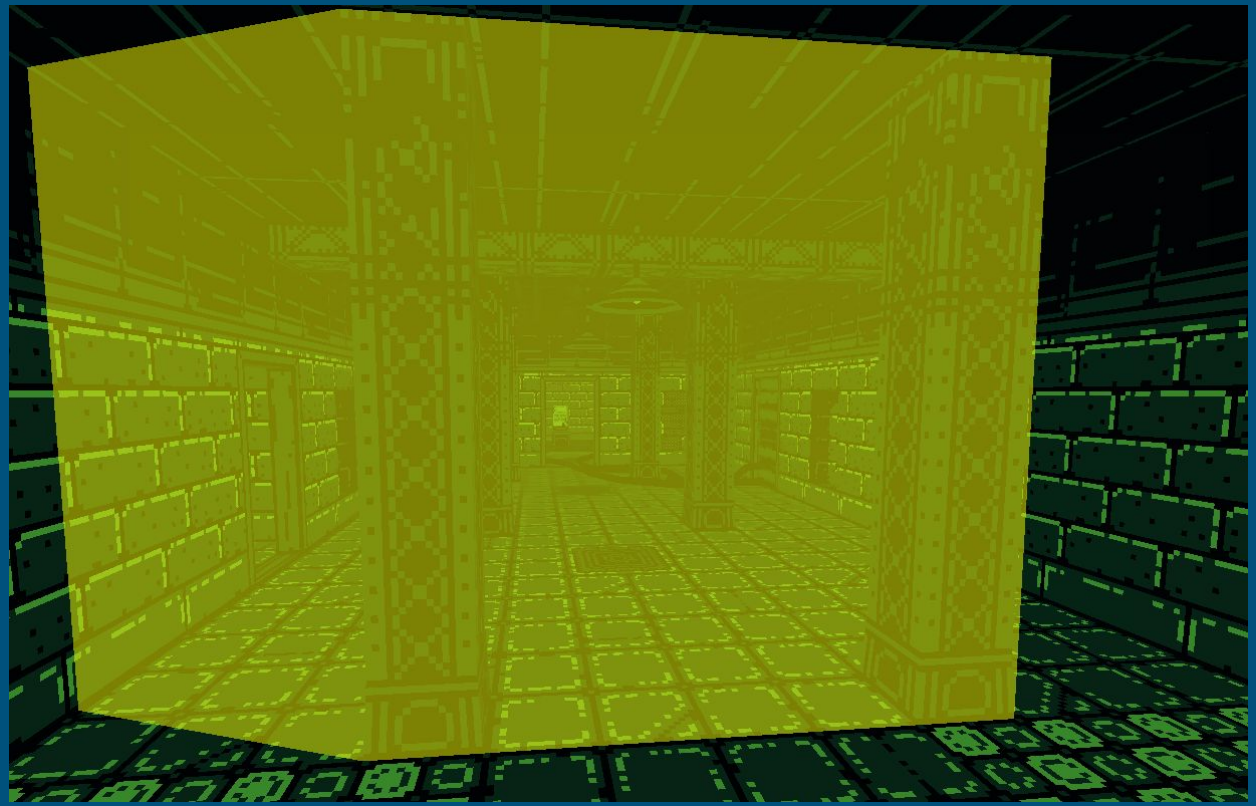
`Box3` is nothing else than an AABB which is always an axis-aligned box. However, `three.js` also provides `ConvexHull` which can be used to generate convex geometries 2 for a given set of points. You can also perform basic ray intersections tests with the convex hull. Intersection tests with other bounding volumes (AABB, OBB, bounding sphere) are not yet supported.

# Convex Hull Seemed like It Could Work...

```
building.traverse( function ( child ) {
            if(child instanceof THREE.Mesh){
                console.log(child);
                let hull = new THREE.ConvexHull();
                hull.setFromObject(child);
                let geom = new THREE.ConvexGeometry(hull);
                let buildingMesh = new Physijs.ConvexMesh(geom, meshMaterial, 0);
                scene.add(buildingMesh);
            }
        });
};
```

# But It Didn’t!

This logic applied to only one mesh:



# Yukas Purpose

- Set Path for doomGuy to follow
  - Follow path behavior
- If doomGuy spots us, he chases
  - Pursuit behavior
  - Combined with NavMesh
- If doomGuy loses sight he returns to his path behavior

# Yuka Initialization

```
entityManager = new YUKA.EntityManager();
//time = new YUKA.Time();
vehicle = new YUKA.Vehicle();

vehicle.setRenderComponent(doomGuy.children[7], sync);

const path = new YUKA.Path();

path.loop = true;
path.add(new YUKA.Vector3(50.20959058991704, -0.5000000000000049, 22.776673254478155));
path.add(new YUKA.Vector3(90.96924973930662, -0.5000000000000169, 76.48312839711204));

vehicle.position.copy(path.current());

const followPathBehavior = new YUKA.FollowPathBehavior(path, 0.5);
//const onPathBehavior = new YUKA.OnPathBehavior( path );
vehicle.steering.add(followPathBehavior);
//vehicle.getWorldPosition

entityManager.add(vehicle);
```

# Troubles with Yuka

- doomGuy isn’t following the path
  - We don’t know why..

```
yukapos = new YUKA.Vector3();
```

```
vehicle.getWorldPosition(yukepos);
```

```
▶Uncaught TypeError: Cannot read property 'extractPositionFromMatrix' of undefined
    at Vehicle.getWorldPosition (yuka.js:3185)
    at animate (gameWshapes.html:802)
```

```
getWorldPosition( result ) {

    return result.extractPositionFromMatrix( this.worldMatrix );

}
```

# Menu

- Added Controls and Volume in the Options menu.
- Changed font of the menu because of character restrictions with other fonts.



# Menu Cont.

```
menuVolume1 = menuSelect.intersectObject(meshVolume1, true);
```

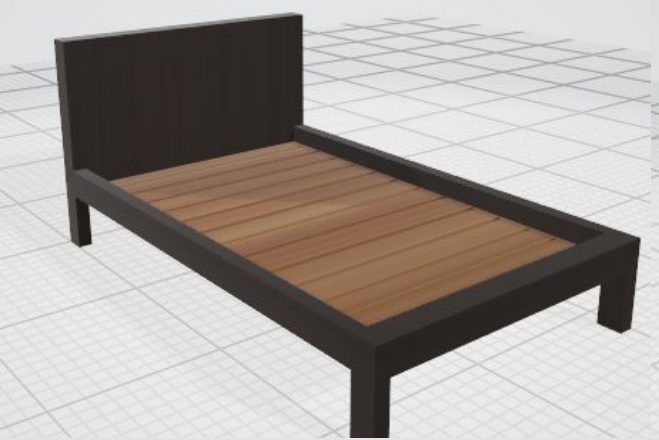
```
doomSound.hasPlayBackControl = true;
```

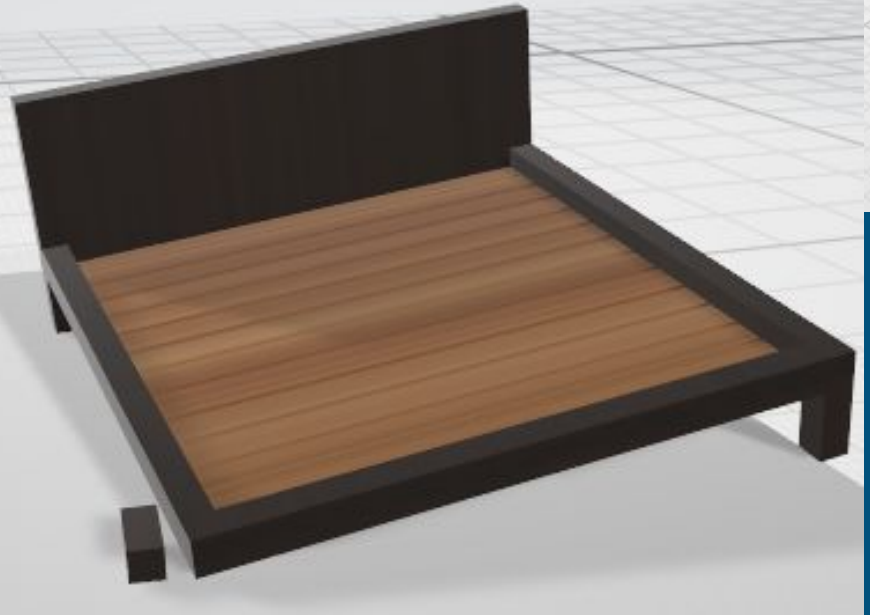
- More of the same type of code to add the different parts of the menu.
- Change the color of the volume setting you are selecting and set the volume accordingly.
- Enable .hasPlaybackControl.

```
else if (menuVolume1[0]) {
    num++;
    meshVolume1.material.color.set(0xFFFFFF);
    if (num > 70) {
        doomSound.setVolume(.2);
    }
}
```

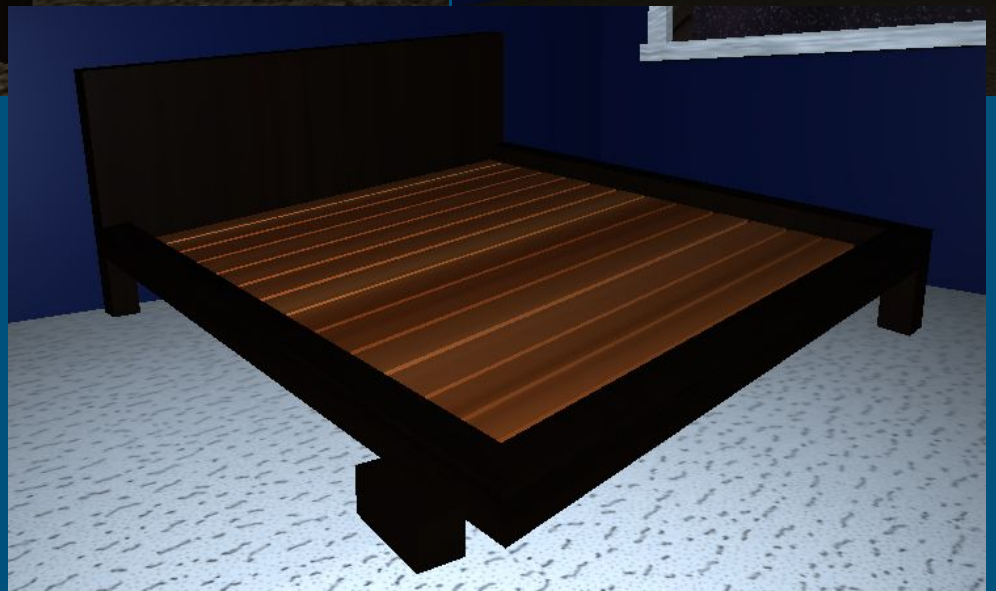
```
let volume1 = new THREE.TextBufferGeometry("1", {
    font: font,
    size: 8,
    height: 1,
    curveSegments: 12,
    bevelThickness: 1,
    bevelSize: .5,
    bevelEnabled: true
});
volume1.computeBoundingBox();

let textMatVolume1 = new THREE.MeshPhongMaterial({ color: 0xff0000, specular: 0xffffff });
meshVolume1 = new THREE.Mesh(volume1, textMatVolume1);
meshVolume1.position.set(-30, 0, -55);
meshVolume1.visible = false;
scene.add(meshVolume1);
```





# Models



# Next Weeks Goals

- Yuka
  - Problem solving vehicle movement
  - NavMesh
- Levels
  - Maze