Introduction to Computer Graphics
Final Project Submission

The Room

Ivan Zheng - 13536901
Danny Jiang - 13550417
Christian Liu - 12991776
Claudia Skinner - 12586638

Changes to Final Submission

Claudia Skinner: Camera Controls and Model Interaction

The suggestions made in regard to this area of our project were to constrain object dragging to the surface of the floor and to implement object rotation.

The previous version used free dragging movement with DragControls, which seemed unnatural and gave poor control over the object's position in relation to the room.

I have implemented the suggested changes to the controls with a combination of Raycasting and TransformControls. The objects are now detected via Raycasting, and they are dragged across an intersecting plane to simulate movement across a floor. A y-axis rotation gizmo has been added from TransformControls to the selected object. I encountered an issue with OrbitControls and TransformControls being simultaneously active, as the mouse movement would move both the object and camera, so I implemented a switch that allows the user to activate the transform control. Upon first loading the scene, the user may use orbit controls and drag objects in the furniture array. When hovering over an object and pressing "R", orbit controls are disabled, and rotation is enabled. When clicking on another part of the scene, the transform controls will be removed from the object and camera orbit will resume.

```
//create controls
controls = new OrbitControls(camera, renderer.domElement); //camera orbit
transformCtrls = new TransformControls(camera, renderer.domElement);//transform gizmo
transformCtrls.showX = ! transformCtrls.showX;//disable x coordinate
transformCtrls.showZ = ! transformCtrls.showZ;//disable z coordinate
____________________________________________________________________

// event listener for mouse movement
document.addEventListener("mousemove", event => {

  //read the mouse position in relation to window
    mouse.x = ( event.clientX / window.innerWidth ) * 2 - 1;
    mouse.y = - ( event.clientY / window.innerHeight ) * 2 + 1;
    raycaster.setFromCamera(mouse, camera);

    //if an object is being dragged, move it along the plane
    if (isDragging) {
      raycaster.ray.intersectPlane(plane, planeIntersect);
      dragObject.position.addVectors(planeIntersect, shift);
    }
});

//event listener for mouse click
document.addEventListener("mousedown", () => {
 //if the raycaster intersects with an element in the furniture array, then...
  var intersects = raycaster.intersectObjects(furniture, true);
  if (intersects.length > 0) {
    controls.enabled = false; //disable the camera from orbiting
    pIntersect.copy(intersects[0].point);//set the object to the plane
    plane.setFromNormalAndCoplanarPoint(pNormal, pIntersect);
    shift.subVectors(intersects[0].object.position, intersects[0].point);
    isDragging = true; //something is being dragged
    dragObject = intersects[0].object;//add the raycasted object to dragObject variable
    transformCtrls.attach(dragObject);//add transformControls to the dragObject

  } //if the mouse ray intersects with nothing
  else if (intersects.length == 0) {
    controls.enabled=true;//clicking will enable orbit camera
    scene.remove(transformCtrls); //clicking will remove rotation gizmo
  }
} );

//event listener for click release
document.addEventListener("mouseup", () => {
  isDragging = false;//nothing is being dragged
  dragObject = null;//there is nothing in dragObject variable
} );


//event listener for keys
document.body.addEventListener('keydown', keyPressed);
function keyPressed(e){
  switch(e.key) {

        case 'r': //R to enable rotation
        controls.enabled = false; //disable orbit
        scene.add(transformCtrls);//add transform gizmo to scene
        transformCtrls.setMode('rotate');//set it to only rotate
        break;

...
  }
}
```

Additionally I assisted in correcting the loading of textures to our models.

<u>Danny Jiang: Models and Controls alert</u>

I added an alert to show the controls of our room when someone opens the webpage while the previous version did not have information for our controls.

```
alert("---------------------------------------------\n"
 + "Mouse clicks to move and orbit. \n"
 + "R to activate rotation slider.\n"
 + "C to clone object.\n"
 + "---------------------------------------------\n" );
```

I also helped with implementing the suggested GUI and the addition of the models with textures. The textures use the texture loader in the three.js library and loading each obj with the respective MTL file with the textures. For the GUI I helped Ivan with merging the file he used to the main javascript file we were using.

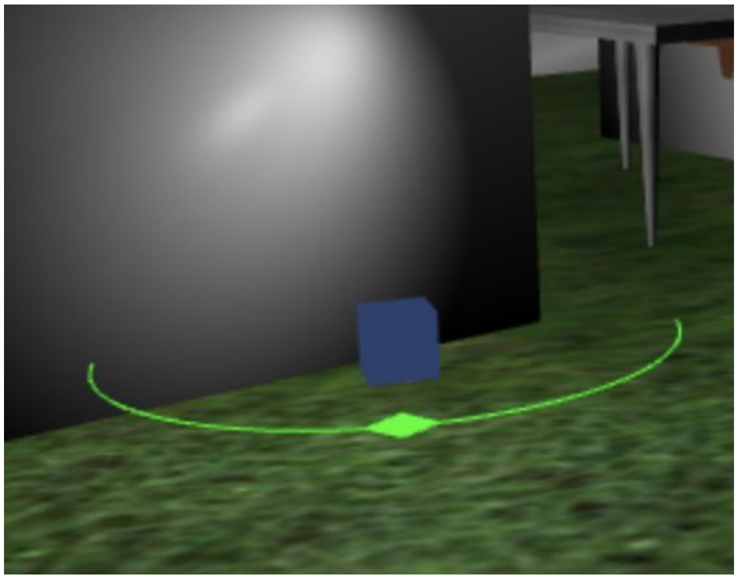
```
function addFurnitures() {

  var mtlLoader = new MTLLoader();
  loadingManager = new THREE.LoadingManager();
  mtlLoader.load('models/house_empty.mtl', function(materials) {
    materials.preload();
    var objLoader = new OBJLoader();
    objLoader.setMaterials(materials);
    objLoader.load('models/house_empty.obj', function(object) {

      object.position.x = 100;
      object.position.y = -1;
      object.position.z = -20;
      object.scale.set(1.5,1.5,1.5)
      scene.add(object);
    });
  });

  mtlLoader.load('models/cybertruck.mtl', function(materials) {
    materials.preload();
    var objLoader = new OBJLoader();
    objLoader.setMaterials(materials);
    objLoader.load('models/cybertruck.obj', function(object) {

      object.position.x = 40;
      object.position.y = -2;
      object.position.z = 0;
      object.scale.set(3,3,3)
      scene.add(object);
      furniture.push(object);
    });
  });

  mtlLoader.load('models/lamp_street_2.mtl', function(materials) {
    materials.preload();
    var objLoader = new OBJLoader();
    objLoader.setMaterials(materials);
    objLoader.load('models/lamp_street_2.obj', function(object) {

      var x = 35;
      var y = -3;
      var z = 15;
      object.position.x = x;
      object.position.y = y;
      object.position.z = z;
      //object.rotation.x = 1.55
      object.rotation.y = -1.5;
      object.scale.set(0.5, 0.5, 0.5)
      var pointLight = new THREE.PointLight( 0xFFFF00, 1, 10 );
      pointLight.position.set( x + 1, y + 10, z + 4 );
      scene.add(pointLight);
      scene.add(object);
    });

  });

  mtlLoader.load('models/table.mtl', function(materials) {
    materials.preload();
    var objLoader = new OBJLoader();
    objLoader.setMaterials(materials);
    objLoader.load('models/table.obj', function(object) {

      object.position.x = -16;
      object.position.y = 0.1;
      object.position.z = -13;
      // object.castShadow = true;
      // object.receiveShadow = true;
      //object.rotation.x = 1.55
      //object.rotation.y = 3.15
      object.scale.set(1,1,1)
      scene.add(object);
      furniture.push(object)
    });

  });

  var textureloader = new THREE.TextureLoader();
  textureloader.load('models/grass_texture/grass.jpg',function(){
  mtlLoader.load('models/grass.mtl', function(materials) {
    materials.preload();
    var objLoader = new OBJLoader();
    objLoader.setMaterials(materials);
    objLoader.load('models/grass.obj', function(object) {

      object.position.x = 11.5;
      object.position.y = -6.5;
      object.position.z = 2.8;

      object.rotation.x = 1.55
      object.rotation.y = 3.15
      object.scale.set(0.5,0.5,0.5)
      scene.add(object);

    });

  });
});
```

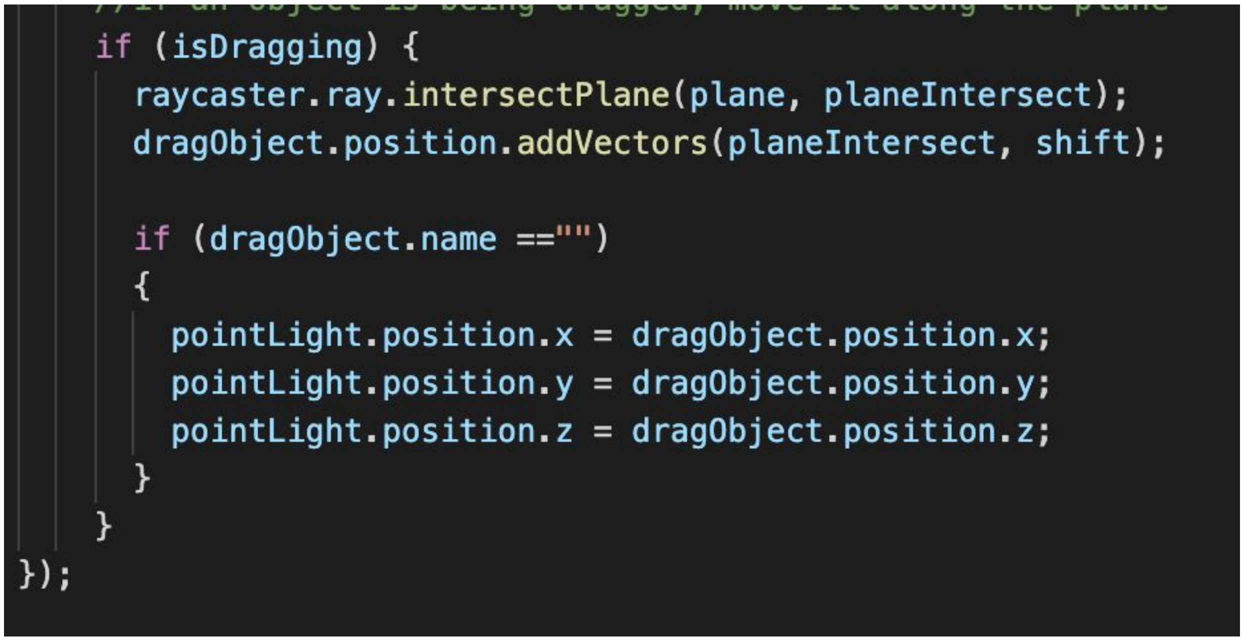
```
var textureloader = new THREE.TextureLoader();
textureloader.load('models/study_chair_cm.jpg',function(){
mtlLoader.load('models/chair1.mtl', function(materials) {
  materials.preload();
  var objLoader = new OBJLoader();
  objLoader.setMaterials(materials);
  objLoader.load('models/chair1.obj', function(object) {

    object.position.x = 10;
    object.position.y = 3;
    object.position.z = -10;
    object.scale.set(7,7,7)
    scene.add(object);
    furniture.push(object);

  });
});
});

var textureloader = new THREE.TextureLoader();
textureloader.load('models/wood.jpg',function(){
mtlLoader.load('models/chair2.mtl', function(materials) {
  materials.preload();
  var objLoader = new OBJLoader();
  objLoader.setMaterials(materials);
  objLoader.load('models/chair2.obj', function(object) {

    object.position.x = -16;
    object.position.y = -1;
    object.position.z = -20;
    object.scale.set(0.1,0.1,0.1)
    scene.add(object);
    furniture.push(object);

  });
});
});

var textureloader = new THREE.TextureLoader();
textureloader.load('models/wood2.jpg',function(){
mtlLoader.load('models/table1.mtl', function(materials) {
  materials.preload();
  var objLoader = new OBJLoader();
  objLoader.setMaterials(materials);
  objLoader.load('models/table1.obj', function(object) {

    object.position.x = 20;
    object.position.y = -2;
    object.position.z = -10;
    object.scale.set(1,1,1)
    scene.add(object);
    furniture.push(object);

  });
});
});
}
```

Ziming Liu: Lighting control, moving lighting object, furnitures clone, skybox,

I have added three types of light, they are

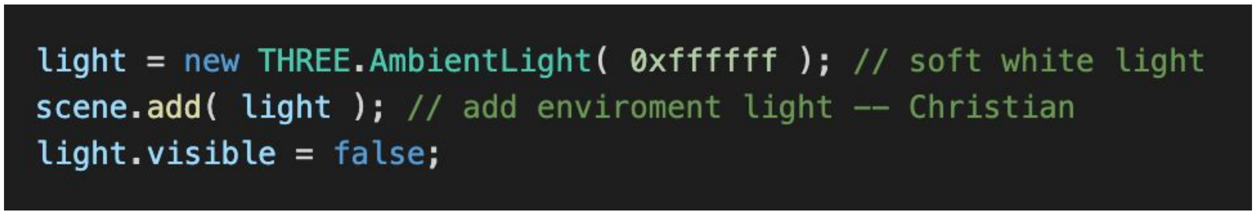
Point light: for the lighting object and street lamp, the light attached to object can be moved with it



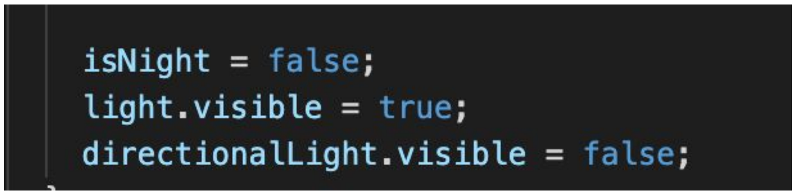


```
    if (isDragging) {
      raycaster.ray.intersectPlane(plane, planeIntersect);
      dragObject.position.addVectors(planeIntersect, shift);

      if (dragObject.name =="")
      {
        pointLight.position.x = dragObject.position.x;
        pointLight.position.y = dragObject.position.y;
        pointLight.position.z = dragObject.position.z;
      }
    }
});
```

Ambient light: for the entire environment, it will be hidden when user switched to night mode

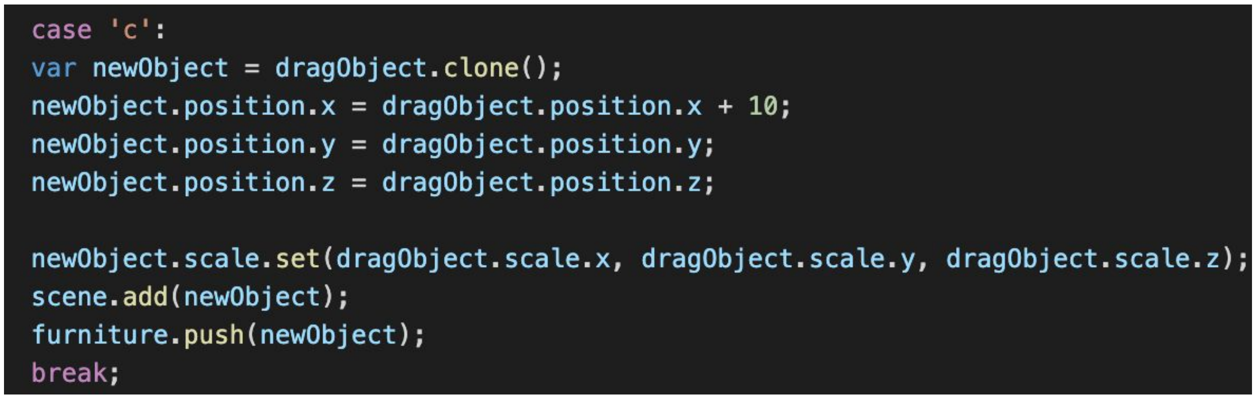


```
light = new THREE.AmbientLight( 0xffffff ); // soft white light
scene.add( light ); // add enviroment light -- Christian
light.visible = false;
```

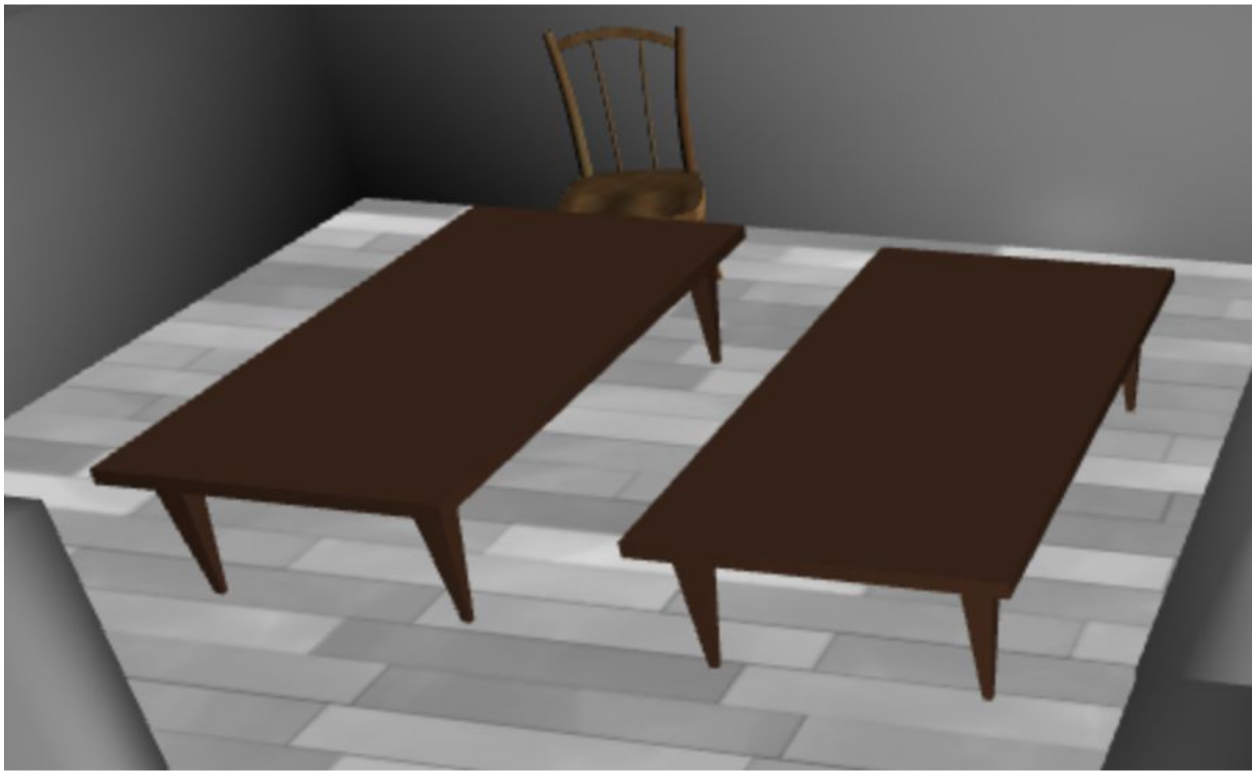


```
    isNight = false;
    light.visible = true;
    directionalLight.visible = false;
```

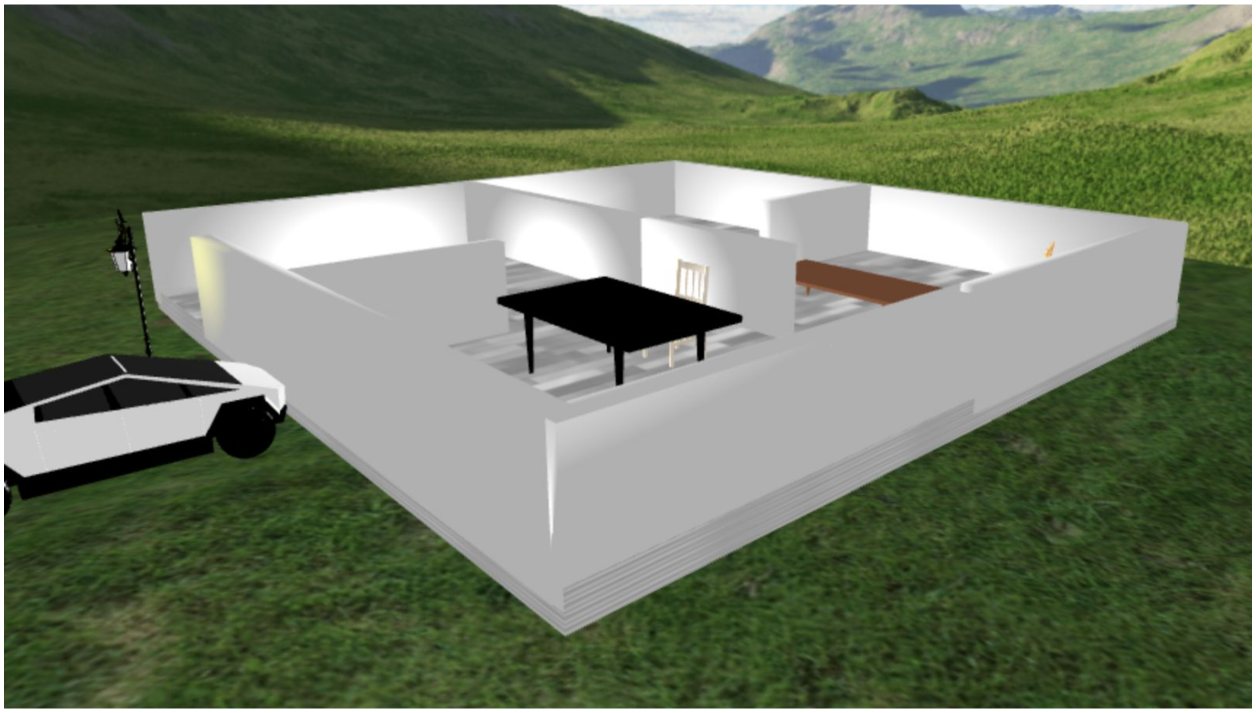
Directional light: for simulating moon’s light
Furnitures clone: when user press ‘c’, the furniture can be cloned

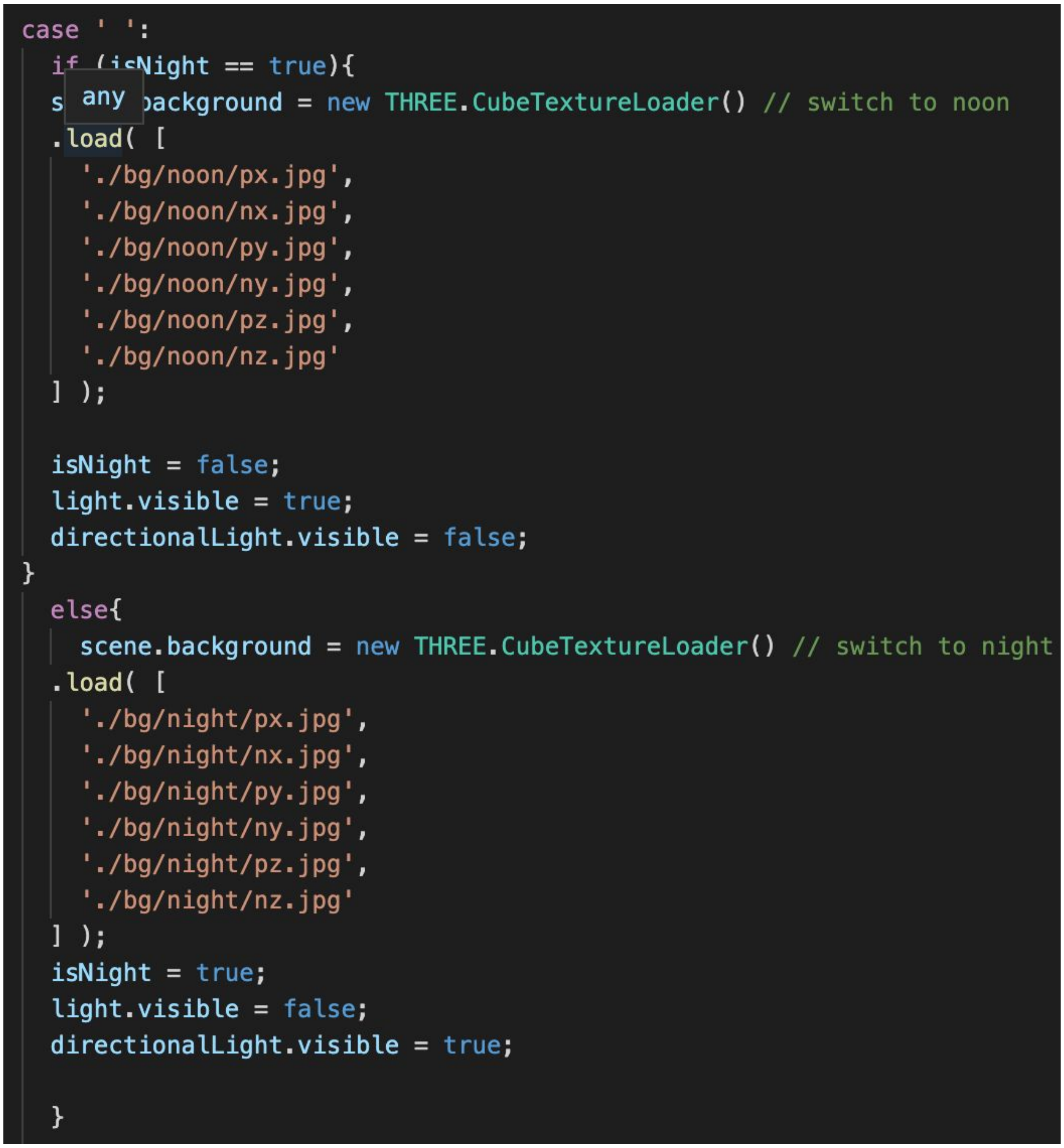


```
case 'c':
var newObject = dragObject.clone();
newObject.position.x = dragObject.position.x + 10;
newObject.position.y = dragObject.position.y;
newObject.position.z = dragObject.position.z;

newObject.scale.set(dragObject.scale.x, dragObject.scale.y, dragObject.scale.z);
scene.add(newObject);
furniture.push(newObject);
break;
```



Skybox: when user press space, the skybox background can be switched between day and night



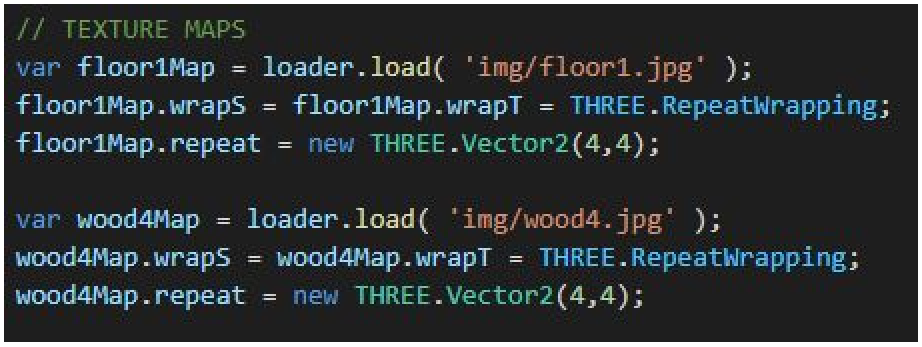


```
case ' ':
  if (isNight == true){
  s any background = new THREE.CubeTextureLoader() // switch to noon
  .load( [
    './bg/noon/px.jpg',
    './bg/noon/nx.jpg',
    './bg/noon/py.jpg',
    './bg/noon/ny.jpg',
    './bg/noon/pz.jpg',
    './bg/noon/nz.jpg'
  ] );

  isNight = false;
  light.visible = true;
  directionalLight.visible = false;
}
  else{
    scene.background = new THREE.CubeTextureLoader() // switch to night
  .load( [
    './bg/night/px.jpg',
    './bg/night/nx.jpg',
    './bg/night/py.jpg',
    './bg/night/ny.jpg',
    './bg/night/pz.jpg',
    './bg/night/nz.jpg'
  ] );
  isNight = true;
  light.visible = false;
  directionalLight.visible = true;

  }
```

<u>Ivan Zheng: Floor textures and GUI</u>

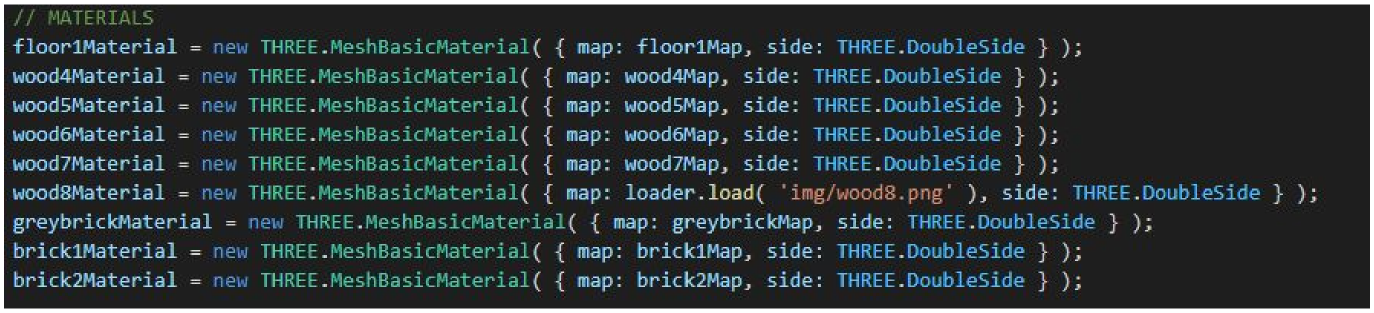
The suggestions that were made in regards to the floor and textures was to include some sort of GUI and convert the existing code into a parametric version, rather than just using keycodes to change textures. The previous version used the keys ‘1’ and ‘2’ to cycle through the floor textures and wall textures respectively. Additionally, only the GUI of the floor textures were implemented in the final version.

Instead of cycling through an array using a function in the previous version, each texture had to be loaded onto a texture map and then repeated as seen in the picture below.



```
// TEXTURE MAPS
var floor1Map = loader.load( 'img/floor1.jpg' );
floor1Map.wrapS = floor1Map.wrapT = THREE.RepeatWrapping;
floor1Map.repeat = new THREE.Vector2(4,4);

var wood4Map = loader.load( 'img/wood4.jpg' );
wood4Map.wrapS = wood4Map.wrapT = THREE.RepeatWrapping;
wood4Map.repeat = new THREE.Vector2(4,4);
```

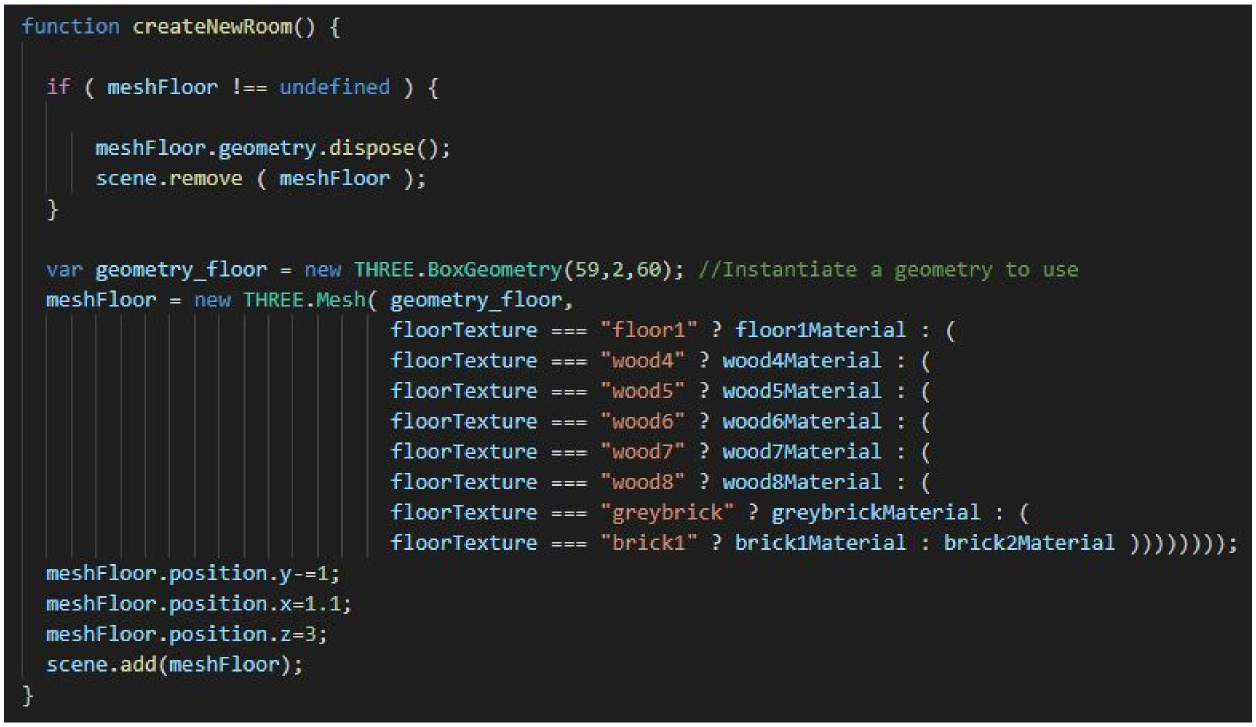
After this, materials for each texture was made with the respective texture map references in it.



```
// MATERIALS
floor1Material = new THREE.MeshBasicMaterial( { map: floor1Map, side: THREE.DoubleSide } );
wood4Material = new THREE.MeshBasicMaterial( { map: wood4Map, side: THREE.DoubleSide } );
wood5Material = new THREE.MeshBasicMaterial( { map: wood5Map, side: THREE.DoubleSide } );
wood6Material = new THREE.MeshBasicMaterial( { map: wood6Map, side: THREE.DoubleSide } );
wood7Material = new THREE.MeshBasicMaterial( { map: wood7Map, side: THREE.DoubleSide } );
wood8Material = new THREE.MeshBasicMaterial( { map: loader.load( 'img/wood8.png' ), side: THREE.DoubleSide } );
greybrickMaterial = new THREE.MeshBasicMaterial( { map: greybrickMap, side: THREE.DoubleSide } );
brick1Material = new THREE.MeshBasicMaterial( { map: brick1Map, side: THREE.DoubleSide } );
brick2Material = new THREE.MeshBasicMaterial( { map: brick2Map, side: THREE.DoubleSide } );
```

A function called ‘setupGUI’ was made in order to initialise the actual GUI and a default texture so that users can change the textures to whatever they prefer using a drop-down menu.

A function called ‘render’ is made to detect whether the option of the chosen texture matches the texture that is shown on the screen and if it doesn’t, it calls a function called ‘createNewRoom’ that removes the current floor and creates a new one with the matching texture. This can be seen in the image below.



```
function createNewRoom() {

  if ( meshFloor !== undefined ) {

      meshFloor.geometry.dispose();
      scene.remove ( meshFloor );
  }

  var geometry_floor = new THREE.BoxGeometry(59,2,60); //Instantiate a geometry to use
  meshFloor = new THREE.Mesh( geometry_floor,
                              floorTexture === "floor1" ? floor1Material : (
                              floorTexture === "wood4" ? wood4Material : (
                              floorTexture === "wood5" ? wood5Material : (
                              floorTexture === "wood6" ? wood6Material : (
                              floorTexture === "wood7" ? wood7Material : (
                              floorTexture === "wood8" ? wood8Material : (
                              floorTexture === "greybrick" ? greybrickMaterial : (
                              floorTexture === "brick1" ? brick1Material : brick2Material ))))))));
  meshFloor.position.y-=1;
  meshFloor.position.x=1.1;
  meshFloor.position.z=3;
  scene.add(meshFloor);
}
```