Contents

[Camera 2](#_Toc507965563)

[Camera Controls: 2](#_Toc507965564)

[Features: 2](#_Toc507965565)

[Jump Feature: 2](#_Toc507965566)

[Canvas: 2](#_Toc507965567)

[Resize window 2](#_Toc507965568)

[Skybox: 2](#_Toc507965569)

[Designer Tools 2](#_Toc507965570)

[Lighting: 2](#_Toc507965571)

[Models 4](#_Toc507965572)

# Camera

## Camera Controls:

WASD = Forward/Left/Backward/Right

R/F = Up/Down

Right Click (long press) = Change camera direction to the mouse pointer

# Features:

## Jump Feature:

**Note**: *This feature is possibly bugged. Further investigation required*. In development version the jump feature is turned off by default. To enable this feature, uncomment the code inside the “START JUMP FUNCTIONALITY” and “END JUMP FUNCTIONALITY” comments.

## Canvas:

### Resize window

Update viewport when the browser window is resized:

function resizeViewport(renderer);

## Skybox:

We create an enormous 3D cube and we are putting textures inside it. We have downloaded multiple images which when placed together can resemble a real 3D environment. Each picture is carefully placed in a specific interior side of the cube. Inside this cube we will place our normal objects, and because the cube is so large it can fool the user that this is the whole sky and not just a cube side. For the picture used in the skybox consult ‘resources’

# Designer Tools

## Lighting:

All lights are created in the loadLights.js file. In the table below, listed are all the **default** properties for every type of light. When creating a light, it is not necessary to manually put together all the properties. Instead, you can avoid putting some light properties if you are comfortable with its default values.

For more information on what the different kinds of lights and their properties do please consult the three.js lighting documentation.

* Create objects for all the lights to be loaded into the scene and create these lights

function loadAllLights();

* Inside the loadAllLights function, create the lights by calling the function createLight and passing it the light object you created in the loadAllLights function

function createLight(lightParam);

|  |  |
| --- | --- |
| **Default Light Objects** | |
| **Lights** | **Default properties and values** |
| PointLight | "type": "PointLight",  "color": 0xffffff,  "intensity": 1,  "distance": 0,  "decay": 1,  "size": 2,  'position': {  'x': 0,  'y': -20,  'z': 0  } |
| AmbientLight | 'type': 'AmbientLight',  'color': 0xffffff,  'intensity': 1 |
| DirectionalLight | 'type': 'DirectionalLight',  'color': 0xffffff,  'intensity': 1,  'size': 2,  'position': {  'x': 0,  'y': 1,  'z': 0  },  'target': {  'x': 0,  'y': 10,  'z': 10  },  'castShadow': false |
| HemisphereLight | 'type': 'HemisphereLight',  'skyColor': 0xffffff,  'groundColor': 0xffffff,  'intensity': 1,  'size': 2,  'position': {  'x': 0,  'y': 1,  'z': 0  } |
| SpotLight | 'type': 'SpotLight',  'color': 0xffffff,  'intensity': 1,  'distance': 100,  'angle': Math.PI/3,  'penumbra': 0,  'decay': 1,  'position': {  'x': 0,  'y': -20,  'z': 0  },  'castShadow': false,  'shadowDarkness': 0,  'shadowCameraVisible': false,  'position': {  'x': 0,  'y': 1,  'z': 0  } |

## Models

All models are created in the loadModels.js file. Under the function loadAllModels , in a similar fashion to creating lights, create a model object passing it its properties. In the table below are all the available properties you can use.

|  |  |  |
| --- | --- | --- |
| **Property** | **Example Value** | **Description** |
| ‘modelPath’ | (string)'models/desk.dae' | (**Required**) Sets the path where the model is stored |
| ‘name’ | (string)’Wall’ | (*Optional* but becomes required when description is set to true) Is the name under which the description is in the xml database. |
| ‘collision' | (Boolean) true | (*Optional*) Enable collision for this model. When this is set to true, the player will be blocked from passing through the model |
| ‘interaction’ | (Boolean) true | (*Optional*) Enable interaction for this model. When this is set to true, if the player presses “E” while looking it, its animation will play |
| ‘description’ | (Boolean)true | (*Optional*)Enable retrieving the description of the model from the description database. When this is set to true, if the player presses “E” while looking at the model, it will retrieve its description from the ml description database. *Note*: The name property becomed (Required) when this option is set to true, also the name MUST exist in the database |
| ‘animation’ {  ‘autoplay’  ‘loop’  ‘clips’  } | (Boolean) true  (Boolean) true  (2D double array) [[0,0.6], [0.6,1.2]] | (*Optional* but becomes required when interaction is set to true) **Autoplay**: When is set to true, the model will play its animation immediately after the model is loaded. False, the model will not play unless implicitly called by the player  **Loop:** When is set to true, when the animation is played it will play forever, making infinite loops. When it set to false, it will only play once  **Clips:** Play specific parts of an animation. Put how many clips inside the array (e.g [0,0.6], start -> 0 finish -> 0.6). Please not there is only one decimal point precision available. |
| 'trigger'{  'animationTrigger'  } | (string)"Ceiling Fan" | (*Optional*) Triggers the animation of another model (the name of the other model is the value of this property). When the player presses “E” while looking this model, it will trigger the animation of another model |
| 'position': {  'x'  'y'  'z'  } | (Integer)-10  (Integer)10  (Integer)130 | (**Required**) Positions the model in a 3D setting  (x : forward/ backward) (e.g +15/-15)  (y: top/bottom) (e.g +10/-10)  (z: left/right) (e.g +20/-20) |
| 'rotate': { 'x' 'y' 'z'  } | (Integer)  (Integer)  (Integer) | (*Optional*) Rotates the model on a 3D space along the three axis (x, y, z) The values provided for each axis are integers Degrees. Not radians. |
| ‘scale’ | (Float)2.3 | (**Required**) Scales(increases/decreases) the model’s size.  Normal size as imported: 1.0 |