# Scene Modelling and Navigation

Alex S O'Donnell

Computer Science, Aberystwyth University, United Kingdom
Alo14@aber.ac.uk

**Abstract.** The three.js library provides many techniques used in computer graphics that can be used to create a lightweight 3D virtual world in WebGL. Developing virtual worlds serves many purposes, from computer games and cartoon movies for entertainment, to architectural modelling and visualization for industry. An example of a 3D virtual world has been created, resembling a house with various objects and features that interact with the user.

**Keywords:** Advanced Computer Graphics, WebGL, Three.js, Texture Mapping, Illumination Models, Geometries.

## 1 How to Run the Code/Controls

- Open the 'model_house.html' file in Mozilla Firefox and mouse left-click anywhere inside the browser window
- The document will then take control of the mouse. From here you can look around the world using the mouse.
- Use the WASD keys to navigate the scene. Toggle the F key to enable flying which will change the WS keys to move you on the Y-axis instead of the Z-axis
- Inside the house there is a model toy car that can be moved using the arrow keys (up arrow accelerates, down arrow reverses and right/left turns the car)

## 2 Cameras, Renderers and the Scene

### 2.1 Cameras

To create and model a 3D house in Three.js, the code requires 3 components to display an image, these include the camera, the renderer and the scene. A Perspective Camera was used for my model house, as it gives a better first person view. This combined with the implemented controls makes navigating the scene easier and less disorienting, especially when moving backwards and forwards along the Z-axis. I set the FOV to 60 degrees, as I discovered that any higher FOV on smaller resolution screens would skew the perspective. The camera has a near/far range of 1 to 1000, anything higher than that would've been unnecessary.

### 2.2 Renderer and the Scene

The renderer is an area of coding that processes objects, lighting, textures, positioning information and how these factors make the objects appear on the screen. It can be thought of as the inner workings of a camera or recorder processing what it is viewing. The performance of the renderer usually defines the performance of the virtual world.

The code uses a WebGL Renderer to render the model house scene, as it is better suited to this task. Anti-aliasing is set to true to make the scene rendered more smoothly. The three.js scene defines an area for the developer to add their objects, lighting and textures to. The scene uses the positioning vectors to locate and place these features in the 3D space.

## 3 The Model House

### 3.1 The World and the House Geometries

Rather than have the world as a floating house, I added a 1000x1000 Plane Buffer Geometry to act as the groundwork. Plane Buffer Geometries stand upright when added to the scene, so I rotated it on its X-axis to make sure it's flat. The main front room is a Box Geometry and is the largest 3D geometry in the scene (220x60x100). The front face of the Box Geometry was removed and replaced with a grid of Plane Buffer Geometries, this was done to implement a transparent Box Geometry representing a window (See section 4.2). Each room has a door which is a 16x32x2 Box Geometry, having a Z depth of 2 ensured that the door would appear on both sides of the walls. The roof of the house is a custom-built geometry using several pushed faces (See section 3.3). The second room is a 110x60x100 Box Geometry to left of and behind the first room, adjacent to this is the third room which is another 110x60x100 Box Geometry. The third room contains an array of Plane Buffer Geometries with different textures inside the Box Geometry and uses different materials then the other rooms (See section 3.2). The back faces for the second and third room are removed and replaced with a grid of Plane Buffer Geometries to again implement a window in between (See section 4.2).

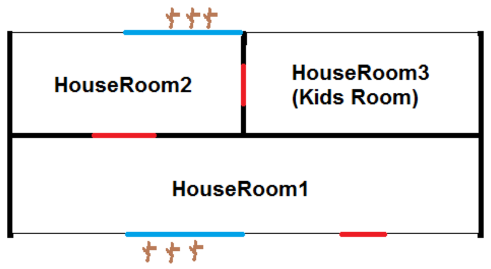


**Fig. 1.** Rough sketch of the house layout, thinner lines are Plane Geometries

## 3.2 Textures and materials

Different texture mapping techniques were used to create the house, for some of the scenarios however limitations were discovered for certain techniques (See section 4.1). UV Mapping is process of using a list of coordinates to assign parts of an image to individual faces on an object. UV Mapping typically involves the use of a sprite sheet or some sort of image collection. This is useful as multiple images can be pulled from the same source file, making the code more efficient.

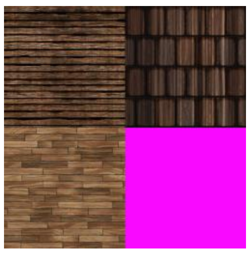


**Fig. 2.** Texture sheet that was used for mapping the wood wall[1] and floorboard[2] textures on the house room geometries

After declaring an array of vectors, I could map different parts of the texture sheet onto the house using the '`faceVertexUvs`' parameter in the geometries. The Box Geometry faces are made up of two triangles, so each face required two sets of corresponding points to map the texture correctly. This technique was also used to map the bed textures, by changing the order of the vertex coordinates you can rotate the textures on the faces. The other method of texture mapping I used was repeat wrapping, which involves tiling the same texture repeatedly onto a mesh. This technique helps prevent the stretching of textures, and is flexible for the developer as they can change the size of the tiles using '`RepeatWrapping`'. This was used to map the grass, kid's room wallpapers/carpet, roof tiles and doors. For some meshes, the textures didn't require repeated patterns, so they were simply mapped to the mesh. These include the rugs, tables, chairs, and the birds.

I sourced my textures in various ways, many were resources from the internet although I attempted to draw some of my own such as the kid's room rug, bed and the birds. The benefit of drawing your own textures is that you know the dimensions of the mesh you are mapping it to, making it fit better. The texture for the tables and chairs was a picture taken with my camera.

I initially used basic materials when testing the code, which helps draw the geometries in a simply shaded way. Combined with ambient lighting, this helped identify any clipping or overlapping geometries when positioning the meshes. Most of these materials were later changed to Lambert materials which creates a diffusely reflecting surface. Phong materials were also tested, which computes the intensities of each point on

[1] Wood wall texture source: http://www.johnsolo.net/tex/tex.php
[2] Floorboard texture source: http://valiet.org/wood-floor-texture/

the surface. The Phong materials were less appealing than the Lambert materials when used on larger meshes such as the house and the grass.

## 3.3 Advanced Modelling, Animations and Movement

The roof of the house is made up of wedge shaped polygons made up of two triangles and two squares (six faces altogether). The bottom face of the polygon is empty as the shape sits on top of the house meshes. I used multiple geometries as building blocks to create the table meshes, a 50x3x25 Box Geometry for the table top and a 2x9x2 Box geometry for the legs. I used Blender to model my own complex geometries, including the chairs, light shades, plant pot, toy car and TV. I implemented the Utah Teapot[3] model to demonstrate what smooth round objects look like in the scene.

An L-system (Lindenmayer system) is a series of production rules for models using alphabetical symbols. Bracketed L-systems were implemented into the code using functions from the practical sessions to create the plants in and around the house. The rules for the trees were changed to represent small leafless branches. The rule was later changed to a different pattern for the potted plant.

The Stemkoski GitHub[4] page provides an example on how to implement texture animation into three.js. This function was used to create a static television screen animation that would appear on the TV mesh as a Plane Buffer Geometry. The function works by taking in the number of images horizontally/vertically in a resource, this divides the resource into a storyboard where each image becomes a frame in the animation.

**Fig. 3.** Animation storyboard used for static. 5 frames horizontally, 1 vertically.

I drew some bird textures that were added to a 32x16 Plane Buffer Geometries. The transparency was enabled to remove the square shape of the geometry. Using the rotation transformation equation, the birds fly around in circles within the scene. The toy car model moves and rotates using three.js' translation functions (See section 3.4).

## 3.4 Controls and Interaction

Various forms of three.js' control sources were tested such as Orbit Controls and Trackball Controls, but Pointer Lock Controls turned out to be the most useful as they synergised well with keyboard inputs. There is little documentation available for the controls systems, but the premise is that each control source changes the way the camera can move in regards to mouse. For example, Orbit Controls pans the camera down when the mouse is dragged down, while Trackball Controls can rotate the camera on

[3] Utah Teapot source: http://tf3dm.com/download-page.php?url=teapot-and-cups-74342

[4] Animation function source: http://stemkoski.github.io/Three.js/Texture-Animation.html

its Z-axis depending on how the mouse is dragged across the screen. Pointer Lock Controls allows the user to move the camera angle using the mouse while using the keyboard to change the camera's positioning.

Keyboard input works by having the controls listen to key press and key releases using the key codes for the arrow keys and WASDF. When a key is pressed, a Boolean is set to true to enable movement in that direction and set back to false when the key is released. Once the walking speed and clock delta is added or subtracted from the corresponding axis, the translation methods from the three.js library are called to move the camera to the correct position. The toy car moves similarly, but rotates when the left/right arrow keys are pressed rather than moving side to side.

### 3.5 Lighting and Shadows

The scene contains a dark grey Ambient Light, which illuminates all object surfaces equally, regardless of position. The aim was to create an overcast day that emphasises the houses inner lighting while having the outside textures remain visible. The house lights use Point Lights which emits light uniformly in all directions while casting shadows. The intensity of the light decays over a set distance. There is a Point Light positioned in every room, with the first room's light having a greater distance than the other lights. Most of the shadows from the meshes worked as intended, however there were some issues when implementing the lighting in certain rooms (See section 4.4).

## 4 Issues and Discoveries

### 4.1 Textures

I initially used UV mapping for the roof tiles[5] texture on the roof polygon, since the top faces of the wedge polygon are quite large, the roof tile texture was severely stretched. The idea was to use UV mapping to keep the entire house textures compact in one resource file, but three.js doesn't support the use of vertex mapping and repeating textures at the same time. Eventually I separated the roof tile texture into its own file, added it to the mesh and used '`RepeatWrapping`' to reduce the size of the texture. Another inconvenience was that Plane Buffer Geometries don't support UV mapping, since '`faceVertexUvs`' is a property of Geometry, not Buffer Geometry.

Certain geometry types were not very practical for applying textures, such as the Extrude Geometry used for the pillow. Textures were disfigured or did not appear at all when mapped to the pillow, so they were unfortunately removed altogether. WebGL relying on texture source dimensions to be to the power of 2 meant that I had to resize many of my textures.

[5] Roof Tiles source: http://www.keyword-suggestions.com/d29vZCByb29mIHRleHR1cmU/

## 4.2 Windows

Implementing windows was tricky, initially the idea was to put the 55x20x1 Box Geometry on top of the house geometries but the textures from the house kept clipping through the window.

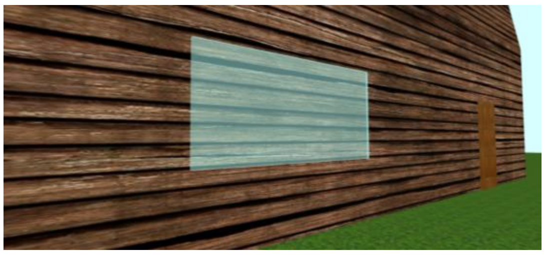


**Fig. 4.** The house texture clipping through the window

My first solution to this was to create a transparent texture image and map it to the window geometry, unfortunately it had the same effect. Even if it had worked I wouldn't have been able to change the opacity of the window (PNG images supports full transparency or no transparency at all). Constructive Solid Geometry is a geometry that can be completely solid on the inside, I created a cube using this geometry, but was unable to view the inside of the house from the other side. I eventually got the windows working as intended by removing the front face of the house completely, and building a 4x3 wall of Plane Buffer Geometries ('`buildHouseWindowWalls`' function) and removing the plane that covers the window.

## 4.3 Models

I initially exported Blender models as *.dae format, but couldn't store the resulting meshes as a variable in three.js which prevented me from transforming meshes such as the toy car later in the code. The Blender models were then exported as JSON files and converted to JavaScript (more secure). Another issue was exporting textures for the models from Blender, in which I had to create new materials in three.js and assign them to the exported models.

## 4.4 Shadows

The Ambient Light was originally a Spot Light, as I wanted to implement moonlight into the scene and have shadows emerging from the plants and the house itself. Unfortunately, the resulting light interfered with the Point Lights inside the house, casting more shadows on objects an illuminating the geometries further.

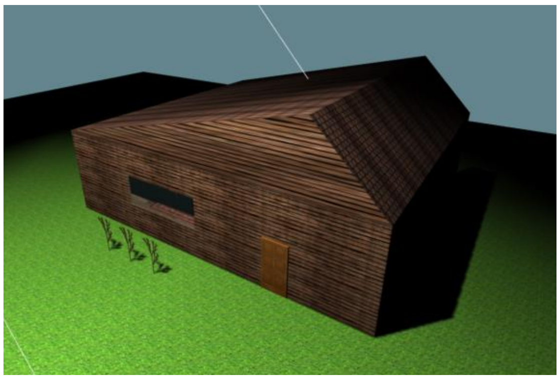


**Fig. 5.** Shadows on the house and plants under the Spot Light

When examining the shadows within the house I came across a strange phenomenon known as shadow acne. Shadow acne is when the shadows on an object become disoriented, forming a checkered pattern. It occurs in the second room of the house, the reason why is unknown. MrDoob, the main contributor to three.js commented in 2015 that they “Still haven’t solved this”[6].

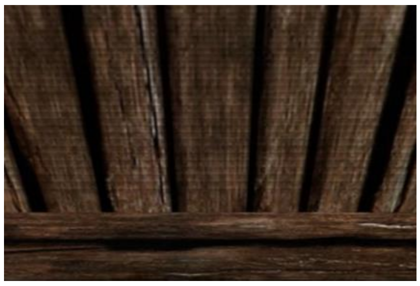


**Fig. 6.** Striped shadows appearing on the roof of room 2

### 4.5 Collision

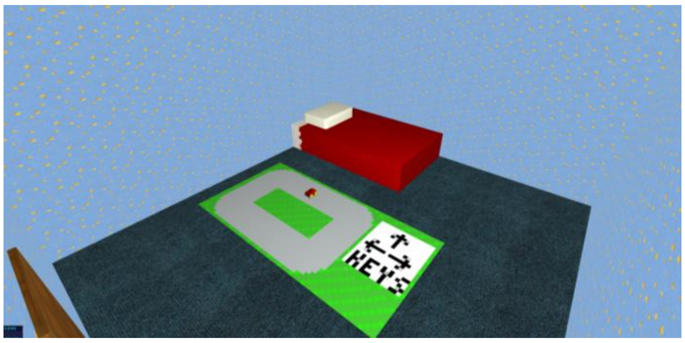
Some aspects of collision were originally in the code, but had to be removed due to being too buggy and frustrating to navigate through. The pseudocode involved having a second controls object, that would test the user’s movements beforehand to see if they collide with a boundary box. Three.js’ translation function made it difficult to predict the user’s movement, the user ended up getting stuck inside the boundary boxes and having to restart the scene.

[6] MrDoob working on shadow acne: https://github.com/mrdoob/three.js/issues/7359

### 4.6 Future Implementations

Three.js provides many useful resources for creativity. If I had more time I planned to have more interactions with the TV, where the user could change the channel and the animations on the screen would change. Bigger and more complex furniture would have made the house more appealing such as bookcases, kitchen tops and fences. 3D bracketed L-systems would have also made the plants more realistic.

## 5 Screenshots



## References & Resources

1. Three.js Documentation: https://threejs.org/docs/
2. Practical information: https://blackboard.aber.ac.uk/
3. Three.js lead contributor's GitHub: https://github.com/mrdoob

- Animation function - http://stemkoski.github.io/Three.js/Texture-Animation.html
- Grass texture - http://seamless-pixels.blogspot.co.uk/2015/04/green-lush-grass-texture.html
- Wood wall texture - http://www.johnsolo.net/tex/tex.php
- Floorboard texture - http://valiet.org/wood-floor-texture/
- Roof tile texture - http://www.keyword-suggestions.com/d29vZCByb29mIHRleHR1cmU/
- Kid's room walls texture - http://www.indoorwallpaper.com/dk5810-walt-disney-kids-toy-story-stars-wallpaper-blue
- Carpet texture- http://carpet.vidalondon.net/carpet-texture-seamless/
- Door texture- http://ancientorange.deviantart.com/art/door-texture-273637443
- Rug texture - https://www.thehomeaccents.com/product/old-world-classics-pazyrk-antique-red-area-rug-couristan
- Utah Teapot: http://tf3dm.com/download-page.php?url=teapot-and-cups-74342