

# Script 02

- Instantiating primitive models.
- Illumination and shadows.
- Animation.
- Perspective camera vs orthographic camera
- Window resizing

## 1.1 Instantiating primitive models

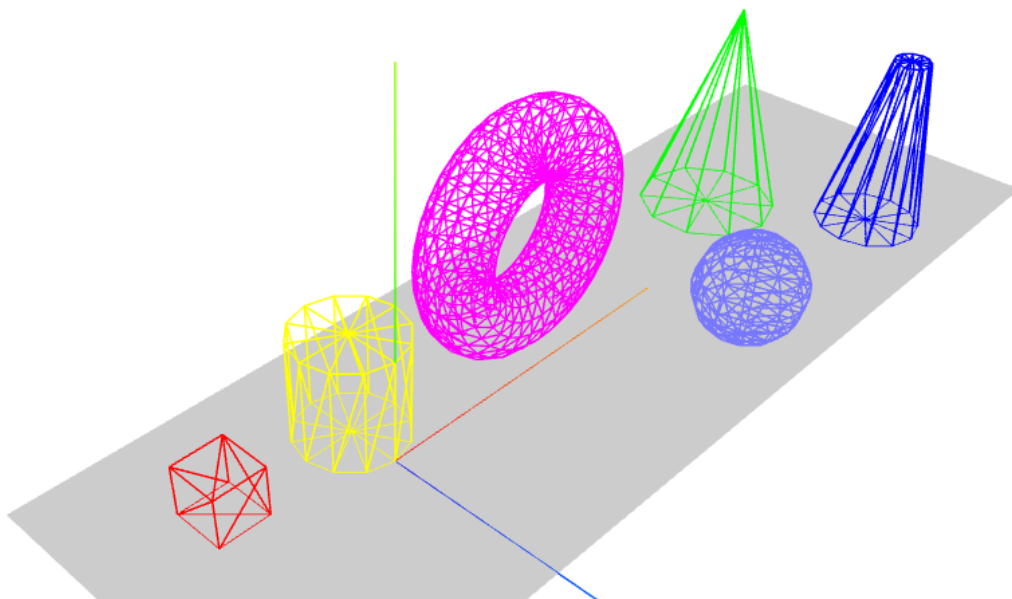
Open the file **threejs\_ex\_02\_01.html**

Analyze the **init()** function:

- How many **models** are defined in the scene?
- Where are they placed? Are any **rotations** applied? Why?
- How is the **camera** looking at the scene?

### Tasks:

- Check the **code comments** and carry out the **suggested tasks**.
- Add **four models** to the scene – with appropriate features – to obtain the scene displayed below.



## 1.2 Illumination and Shadows

Simple illumination effects are easily obtained by **adding a spotlight** to the scene, assigning proper **materials** to the models, and **enabling the rendering of shadows**.

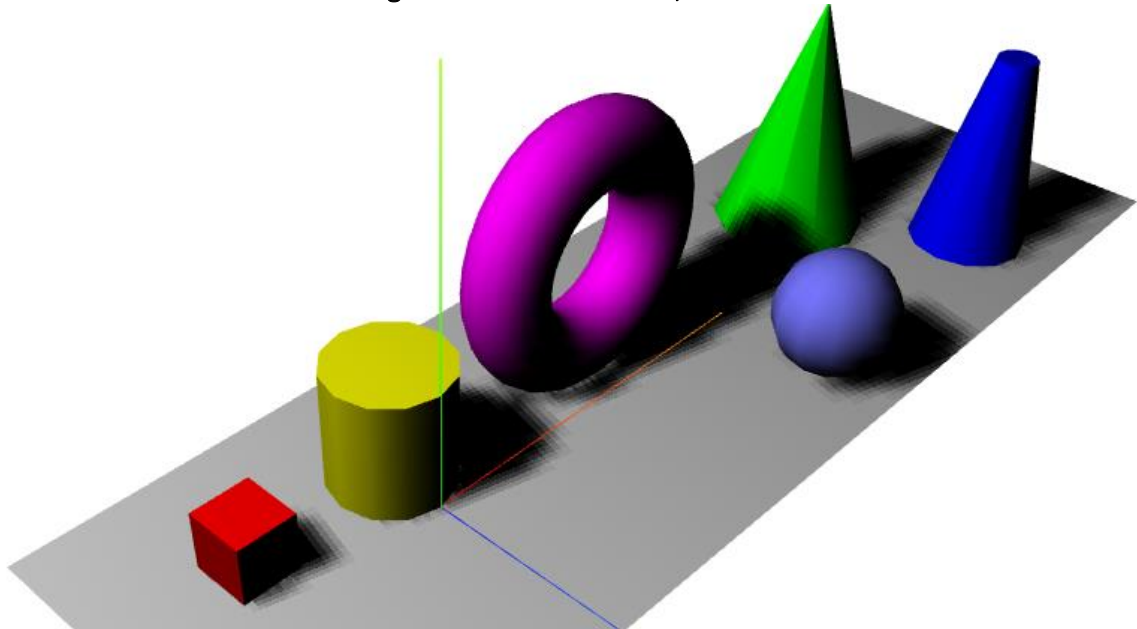
### Tasks:

- Disable the **wireframe** rendering mode. Add a **spotlight** to the scene, placed **at (-40, 60, -10)**. Do you notice any **differences**? Why?
- Change the material defining the models to **Lambert Material**, which computes shading using the Gouraud model. Which **differences** do you notice?
- Enable the rendering of shadows:  

```
renderer.shadowMap.enabled = true;
```
- Enable the models to cast shadows on each other and to receive shadows:  

```
plane.receiveShadow = true;  
cone.castShadow = true;  
cone.receiveShadow = true;
```
- Enable the light source to cast shadows:  

```
light.castShadow = true;
```



- Change the material defining the models to **Phong Material**, which computes shading using the Phong model and simulates the reflection of shiny surfaces. Associate **shinier materials** to some of the models.
- Compare side-by-side the results of the Gouraud and the Phong shading models. Do you notice **any differences**?

### 1.3 Animation

Simple animation effects are easily obtained by **updating model position or rotation angles** just before rendering each frame, and by rendering an appropriate number of frames per second.

#### Tasks:

- Add the following code at the end of the `init()` function:

```
var step = 0;
// Update model features and render the scene
renderScene()

function renderScene() {
    // Rotate the cube around its axes
    cube.rotation.x += 0.02;
    cube.rotation.y += 0.02;
    cube.rotation.z += 0.02;

    step += 0.04;
    // Bounce the sphere up and down
    sphere.position.x = 20 + (10 * Math.cos(step));
    sphere.position.y = 3 + (10 * Math.abs(Math.sin(step)));

    // Render using requestAnimationFrame
    requestAnimationFrame(renderScene);
    renderer.render(scene, camera);
}
```

- What happens?
- Add code to **rotate the torus** around its *XX* axis and to **shuffle the cylinder** back-and-forth in the *ZZ* direction.
- Add code to **displace the camera** back-and-forth along a given direction.

## 1.4 Perspective camera vs orthographic camera

There are two different camera types in Three.js: the orthographic camera (at an indefinite distance to the scene) and the perspective camera (at a finite distance to the scene), which produce different final images.

### Tasks:

- Check the documentation on the Orthographic Camera. How do you **set up** an Orthographic Camera? What is the meaning of the camera **constructor arguments**? How do you define the **viewing direction**?
- Change the camera to the Orthographic Camera. Which **differences** do you notice in a **still image** and in an **animation**?
- Compare side-by-side the images produced by the two cameras. Which camera produces **more realistic** images?

## 1.5 Adding an event listener to handle browser window resizing

Updating the display whenever the browser window is resized can be easily done by registering the corresponding event-handling function:

```
window.addEventListener('resize', onResize, false);
```

In this `onResize()` function, camera aspect ratio and renderer window size are updated as follows:

```
function onResize() {  
    camera.aspect = window.innerWidth / window.innerHeight;  
    camera.updateProjectionMatrix();  
    renderer.setSize(window.innerWidth, window.innerHeight);  
}
```

### Tasks:

- Add the event-handling code to your example file.
- Declare the camera and renderer variables as **global variables**.
- Resize the browser window and see what happens.
- Run the example on your **smartphone**!