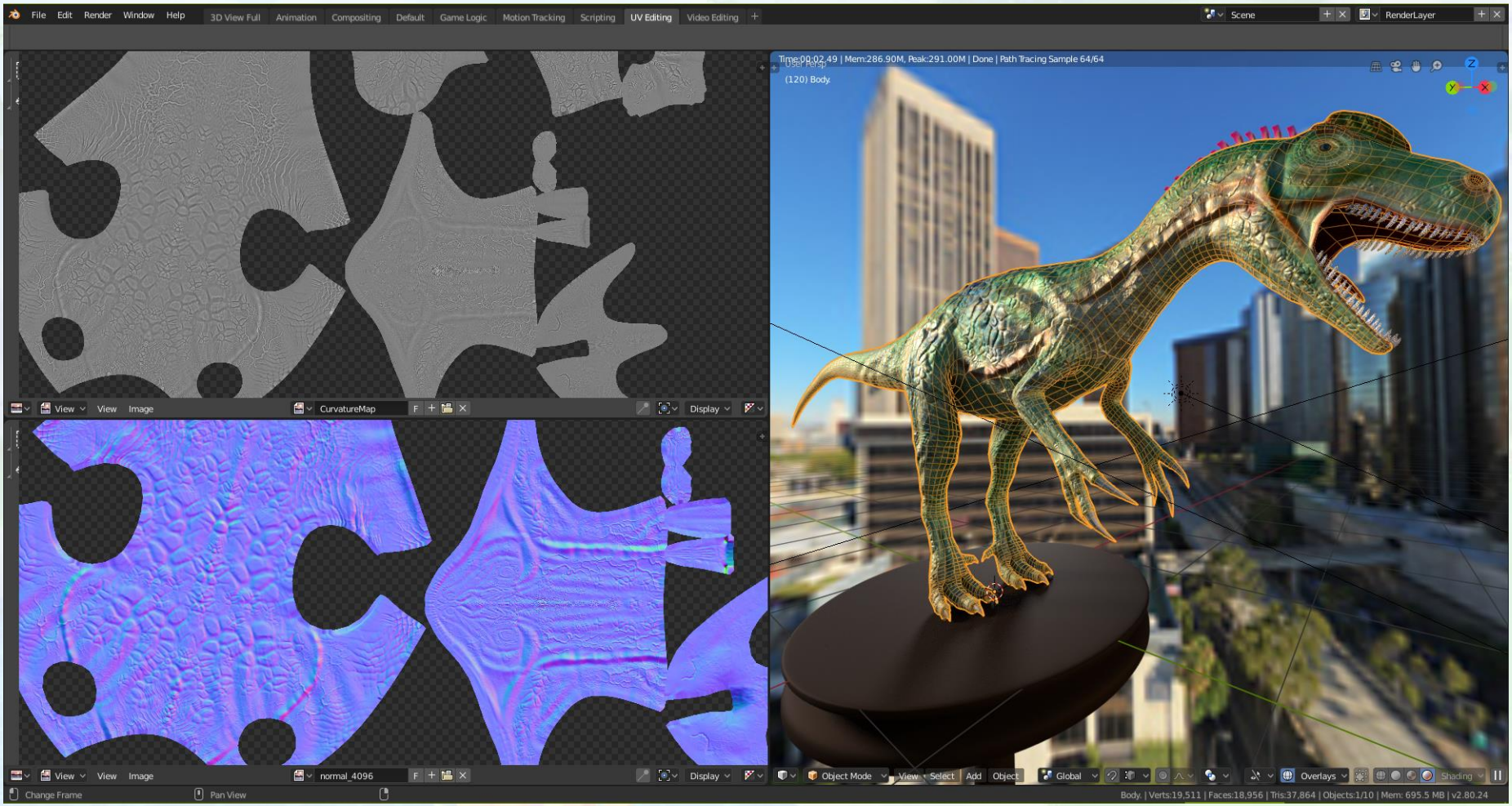
Why 2D is not Enough?

3D Modeling


File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Scripting
Scene
View Layer
Capture2.png
Capture1.png
Display
Object Mode View Select Add Object
Global
Overlays
Shading
User Perspective
(1) Collection | Sphere
Sphere
Eye
Preview
Surface
Use Nodes
Surface Principled BSDF
GGX
Christensen-Burley
Base Color
Subsurface 0.000
Subsurface Radius 1.000 0.200 0.100
Subsurface Color
Metallic 1.000
Specular 0.500
Specular Tint 0.000
Roughness 0.200
Anisotropic 0.000
Anisotropic Rotation 0.000
Sheen 0.000
Sheen Tint 0.500
Clearcoat 0.000
Playback Keying View Marker
Start: 1 End: 250
Select or Deselect All Move Rotate View Call Menu
Collection | Sphere | Verts:447,028 | Faces:447,086 | Tris:894,044 | Objects:0/6 | Mem: 1.13 GB | v2.80.34

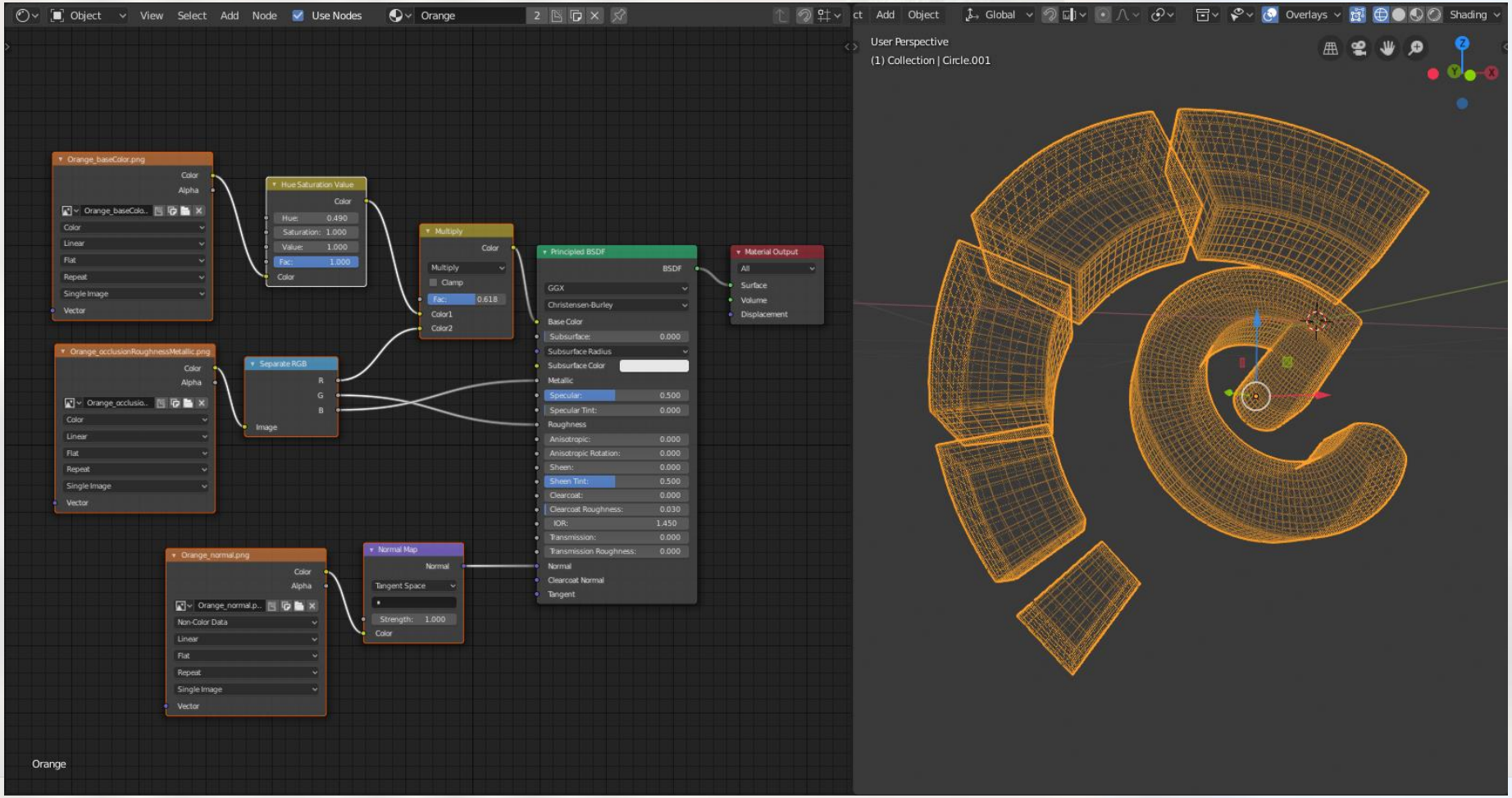
3D Modeling Technologies



Real-time PBR/PBS

AR+VR Technologies

3D Core Concepts



3D Core Concepts

Vertex: a pint in the 3D world (x, y, z)

Line/Edge: connects two points together

Face: three lines connected

Geometry: a collection of faces

Surface: how should the geometry be rendered

Mesh/Model: a geometry and a surface

3D Core Concepts

Scene: The **stage** where every object needs to be added in order to be rendered

Camera: This will control **what the user can see** (and how)

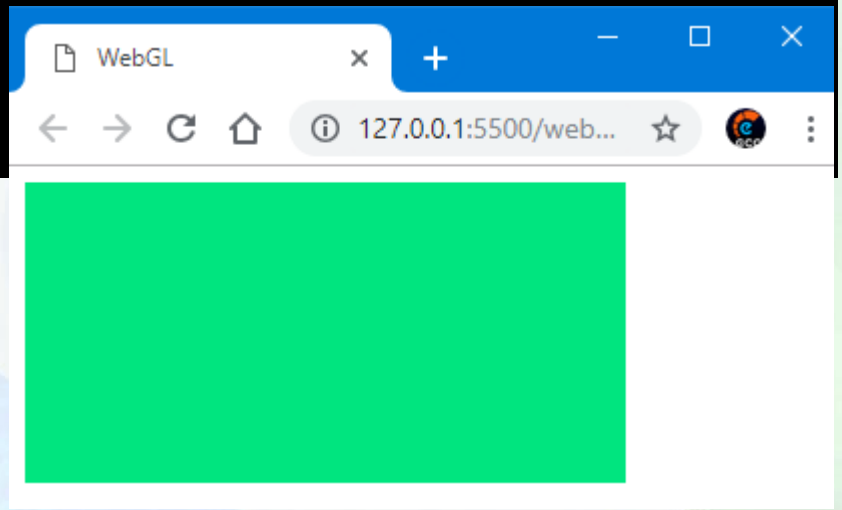
Light: In order for our camera to see anything, we'll need light sources to **illuminate the scene**

Renderer: Displays the scene using WebGL

Objects: The things that will be rendered and animated within the scene

WebGL Context

```html
<!DOCTYPE html>
<head>
    <title>WebGL</title>
</head>
<body>
    <canvas id = 'canvas'></canvas>
    <script>
        var canvas = document.getElementById('canvas');
        var gl = canvas.getContext('webgl');
        gl.clearColor(0.0, 0.9, 0.5, 1);
        gl.clear(gl.COLOR_BUFFER_BIT);
    </script>
</body>
</html>
```

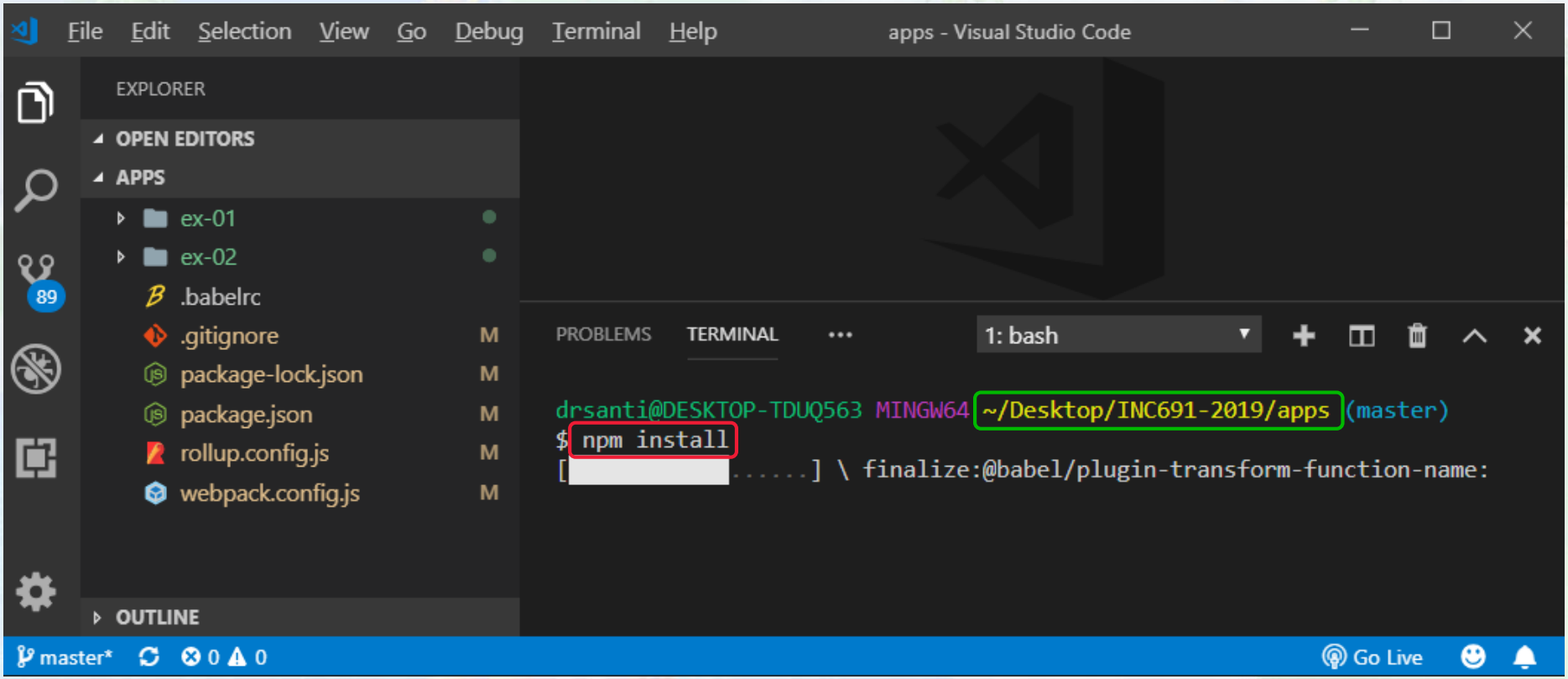


Getting Started

Clone or the project by enter the flowing command in your terminal window

```
git clone https://github.com/drsanti/INC691-2019.git
```

Open the **apps** directory (INC691-2019/apps) and run the command **npm install**



Getting Started

Open the **webpack.config.js** and check/edit the working directory. Save & Close

```
const path = require('path');
const _mode = 'app';

module.exports = {
    entry: (_mode === 'app') ? './ex-02/src/index.js' : './src/index.js',
    mode: 'development',
    output: {
        path: path.resolve(__dirname, 'ex-02/public'),
        filename: 'bundle.js'
    },
    devServer: {
        contentBase: path.join(__dirname, 'ex-02/public'),
        compress: true,
        port: 9000
    },
```

Getting Started

Run the application, by execute the command **npm run start**

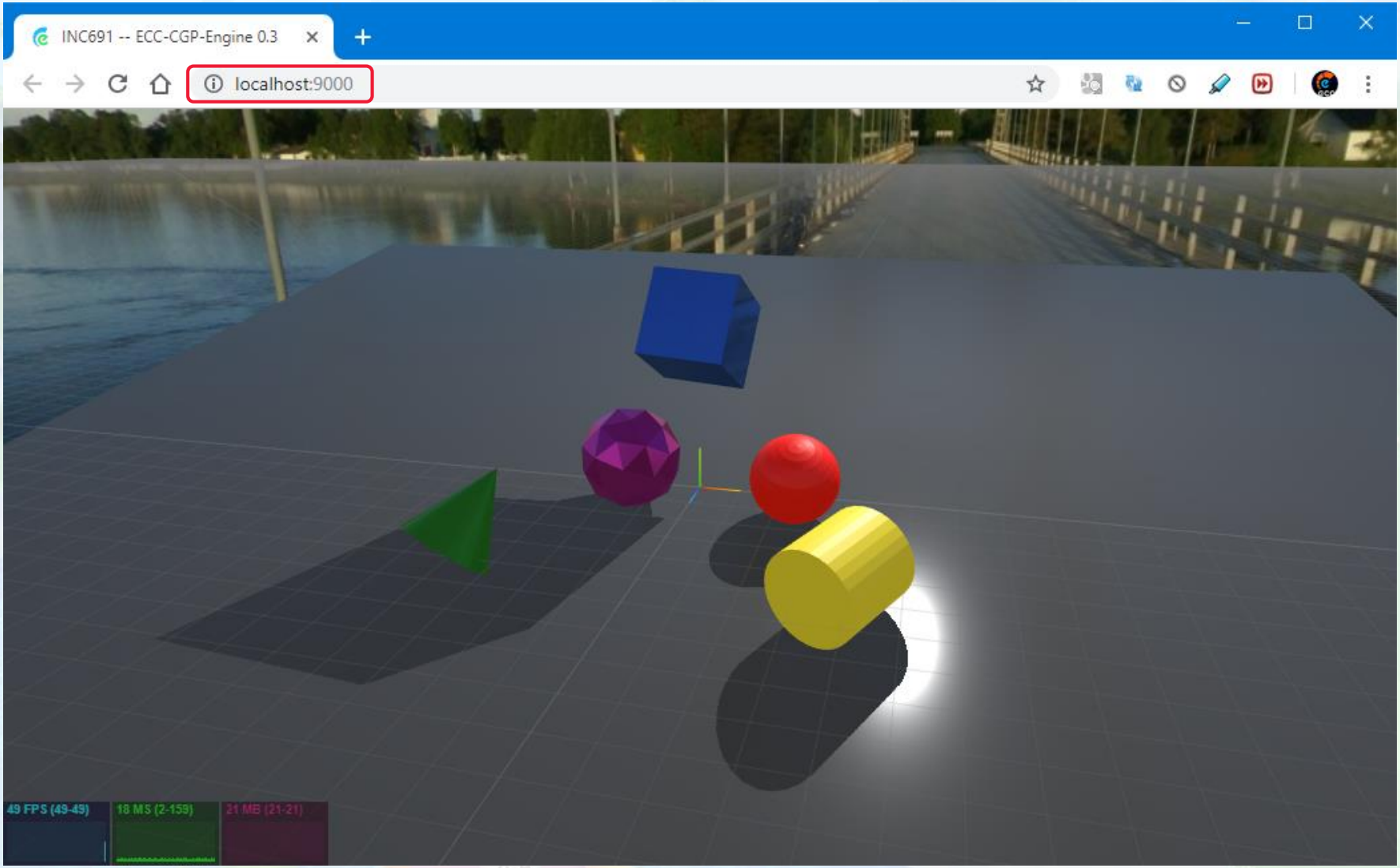
```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL                    1: bash

drsanti@DESKTOP-TDUQ563 MINGW64 ~/Desktop/INC691-2019/apps (master)
$ npm run start

> ecc-cgp-engine@0.0.3 start C:\Users\drsanti\Desktop\INC691-2019\apps
> webpack-dev-server --hot --inline

i ｢wds｣: Project is running at http://localhost:9000/
i ｢wds｣: webpack output is served from /
i ｢wds｣: Content not from webpack is served from C:\Users\drsanti\Desktop\INC691-2019\apps\ex-02\public
i ｢wdm｣: Hash: 9f79df8a1dfada349083
Version: webpack 4.29.3
Time: 2344ms
Built at: 02/14/2019 8:34:40 AM
    Asset      Size  Chunks             Chunk Names
bundle.js  3.28 MiB    main  [emitted]  main
Entrypoint main = bundle.js
```

CTRL+Click to open a browser

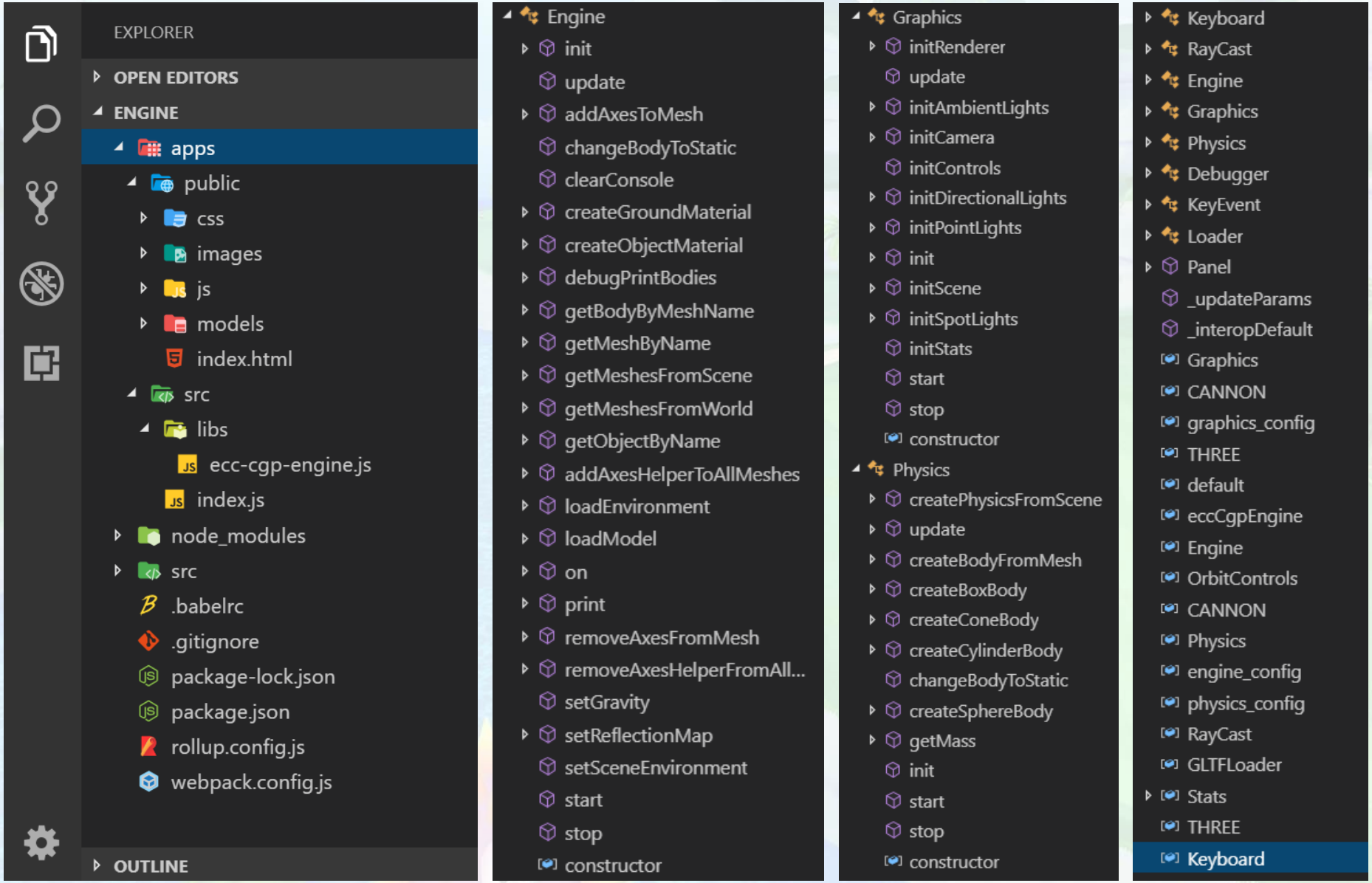
Go Live Ln 16, Col 5 Spaces: 2 UTF-8 CRLF JavaScript Prettier 1

Getting Started

Open a browser and go to **http://localhost:9000/**



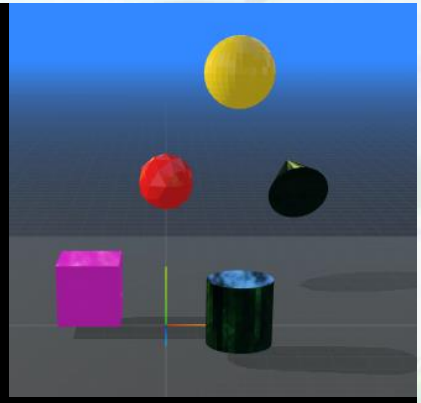
ECC-Computer-Graphics-Physics-Engine



Getting Started

```
import {Engine} from './libs/ecc-cgp-engine';

/* Create the graphics engine */
const engine = new Engine({sceneType: 'fog'});

/* Initialize and start the engine */
engine.init().then( (params) => {
    console.dir(params);          /* Check in the console window */
    engine.start();               /* Start the engine            */
});
```

fog, env, basic

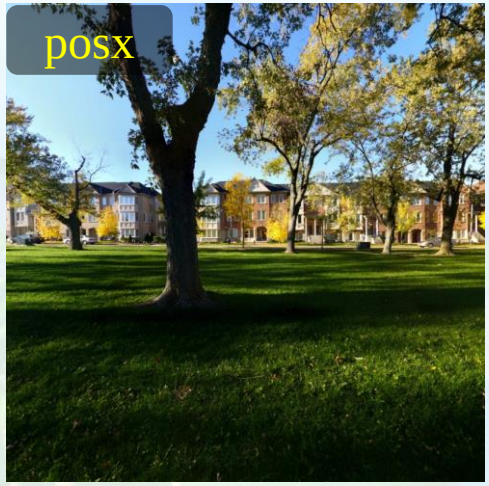


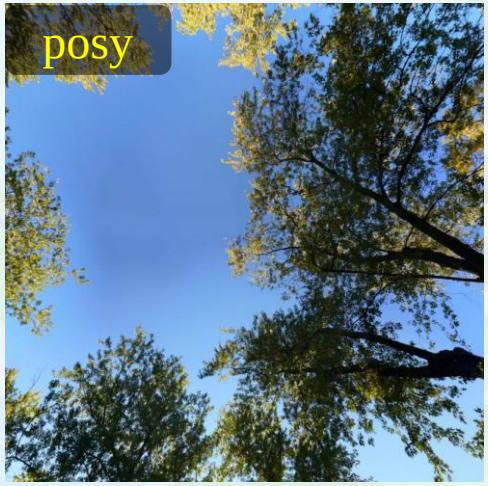
```
▼ Object
  ▶ graphics: Graphics {options: {…}, scene: Scene, camera: PerspectiveCamera, renderer: WebGLRenderer, controls: OrbitControls, …}
  ▶ physics: Physics {options: {…}, graphics: Graphics, timeStep: 0.016666666666666666, scalingFactor: 0.98, debugColor: 16776960, …}
  ▶ scene: Scene {uuid: "98D49B1B-C07D-47C2-A903-290D4A84D4EA", name: "Scene", type: "Scene", parent: Scene, children: Array(8), …}
  ▶ texture: CubeTexture {uuid: "9E06FED2-7C60-413A-8CCD-CB6EDB361939", name: "", image: Array(6), mipmaps: Array(0), mapping: 301, …}
  ▶ __proto__: Object
```

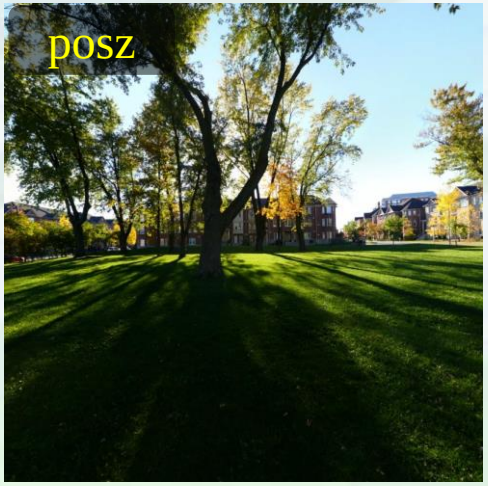
```
/* Try the following commands to check some default configs.*/
console.dir(engine.config);
console.dir(engine.graphics.config);
console.dir(engine.physics.config);
```

Then…
change “config” to “options”
and check their parameters

Environment/Reflection Texture







GL Transmission Format (GLTF)

```
{
    "mesh" : 0,
    "name" : "Sphere",
    "translation" : [
    2.062831401824951,
    8.270650863647461,
    2.418491840362549
    ]
},
{
    "mesh" : 1,
    "name" : "Icosphere",
    "translation" : [
    0,
    4.096539497375488,
    -3.8716928958892822
    ]
},
```

glTF Separate (.gltf + .bin + texture)
glTF Embedded (.gltf)
glTF Binary (.glb)

```
{
    "name" : "Sphere",
    "primitives" : [
        {
            "attributes" : {
                "POSITION" : 0,
                "NORMAL" : 1,
                "TEXCOORD_0" : 2
            },
            "indices" : 3,
            "material" : 0
        }
    ]
}
```

https://www.khronos.org/gltf/

User Initialization

```
import {Engine} from './libs/ecc-cgp-engine';

const engine = new Engine({sceneType: 'fog'});

/* Initialize and start the engine */
engine.init().then( ( params ) => {
    userInit( params ); /* Call the userInit() function */
    engine.start( );
});

/* User initialization function */
function userInit( params ) {
    var mesh = engine.getMeshByName('Cylinder');
    console.log(mesh);
}
```

Engine Callback (Hook)

```
import {Engine} from './libs/ecc-cgp-engine';

const engine = new Engine({sceneType: 'fog', usePhysics: false});

/* Initialize and start the engine */
engine.init().then( ( params ) => {
     userInit( params );
     engine.start( callback ); /* Tell the engine that we need the callback */
});

/* Global variable */
var target = null;

/* User initialization function */
function userInit( params ) {
     target = engine.getMeshByName('Cylinder'); /* Get the desired mesh    */
}

/* Engine's callback/hook function */
function callback( args ) {
     target.rotation.x += Math.PI/100;          /* Rotate along x-axis      */
}
```

Use only graphics

What is the callback function?

What is the args argument?

Engine – Events (KeyDown)

```
import {Engine} from './libs/ecc-cgp-engine';

const engine = new Engine({sceneType: 'fog'});

engine.init( options ).then( ( params ) => {
    userInit( params );
    engine.start( );
});

var body = null; /* Rigid-body*/

function userInit( params ) {
    body = engine.getBodyByMeshName( 'Cube' );    /* Get physics-body  */
    engine.on( 'keydown', onKeyDown );            /* Key-down event    */
}

/* Callback function of the key-down event */
function onKeyDown( event ) {
    if(event.key == 'f') {
        /* Key f is pressed, do something    */
    }
}
```

Target name

Event name

What are the events?

How to detect an event?

What is event handler?

How to utilize key-down event?

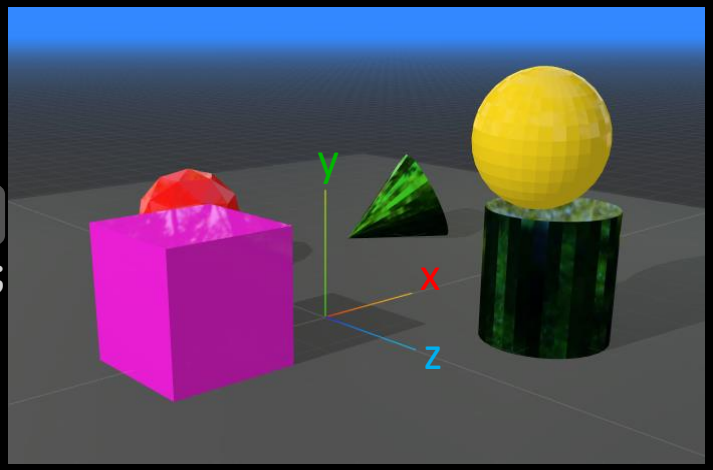
Physics – WorldPoint & Force

```
import {Engine, CANNON} from './libs/ecc-cgp-engine';
/* Other lines of code are here */

var body = null;

function userInit( params ) {
    body = engine.getBodyByMeshName('Cube');
    engine.on('keydown', onKeyDown);
}

function onKeyDown(event) {
    if(event.key == 'f') {
        applyForce();
    }
}
/* Apply force vector to the world-point*/
function applyForce() {
    const worldPoint = new CANNON.Vec3 ( 0, 0, 0 );
    const force = new CANNON.Vec3 ( 1000, 0, 0 );
    body.applyForce(force, worldPoint);
}
```

Target name

Event name

Press f-key to apply the force vector



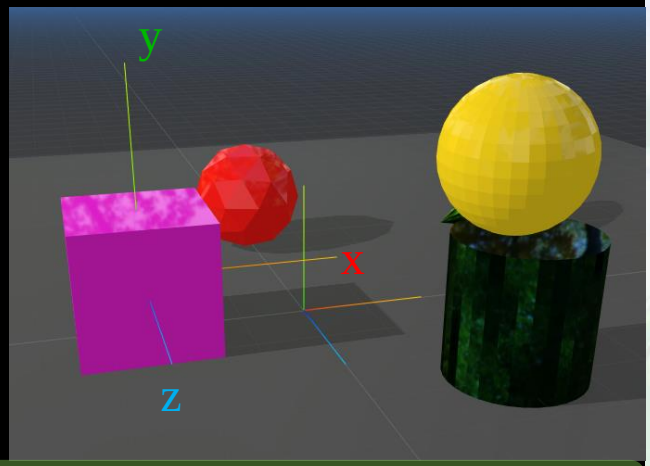
What is the World-Point?

What is the Force Vector?

What will happen after applying the force-vector to the world-point?

Physics – WorldPoint & LocalForce

```
import {Engine, CANNON} from './libs/ecc-cgp-engine';
/* Other lines of code are here */
var body = null;

function userInit( params ) {
    body = engine.getBodyByMeshName('Cube’);
    engine.addAxesToMesh(body.threemesh, 3);
    engine.on('keydown', onKeyDown);
}

function onKeyDown(event) {
    if(event.key == 'f') {
        applyLocalForce();
    }
}
/* Apply local force vector */
function applyLocalForce() {
    const worldPoint = new CANNON.Vec3 ( 0, 0, 0 );
    const force = new CANNON.Vec3 ( 1000, 0, 0 );
    body.applyLocalForce(force, worldPoint);
}
```



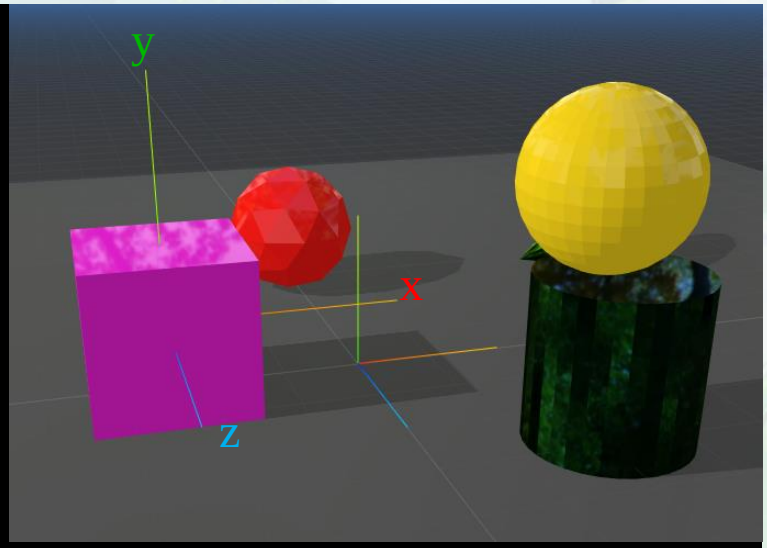
What is the Local Force Vector?

What will happen after applying the local force to the world-point?

How to apply force to center of the rigid body (physical object)?

Physics – Impulse & LocalImpulse

```
/* Other lines of code are here */
var body = null;
function userInit( params ) {
      body = engine.getBodyByMeshName('Cube');
      engine.addAxesToMesh(body.threemesh, 3);
      engine.on('keydown', onKeyDown);
}
/* Key-down event handler */
function onKeyDown(event) {
      if(event.key == 'i') {
            applyImpulse();
      }
      if(event.key == 'l') {
            applyLocalImpulse();
      }
}
/* Apply impulse vector to the world-point */
function applyImpulse() {
      const worldPoint = new CANNON.Vec3 ( 0, 0, 0 );
      const force = new CANNON.Vec3 ( 10, 0, 0 );
      body.applyImpulse(force, worldPoint);
}
/* Apply local impulse vector the world-point */
function applyLocalImpulse() {
      const worldPoint = new CANNON.Vec3 ( 0, 0, 0 );
      const force = new CANNON.Vec3 ( 10, 0, 0 );
      body.applyLocalImpulse(force, worldPoint);
}
```



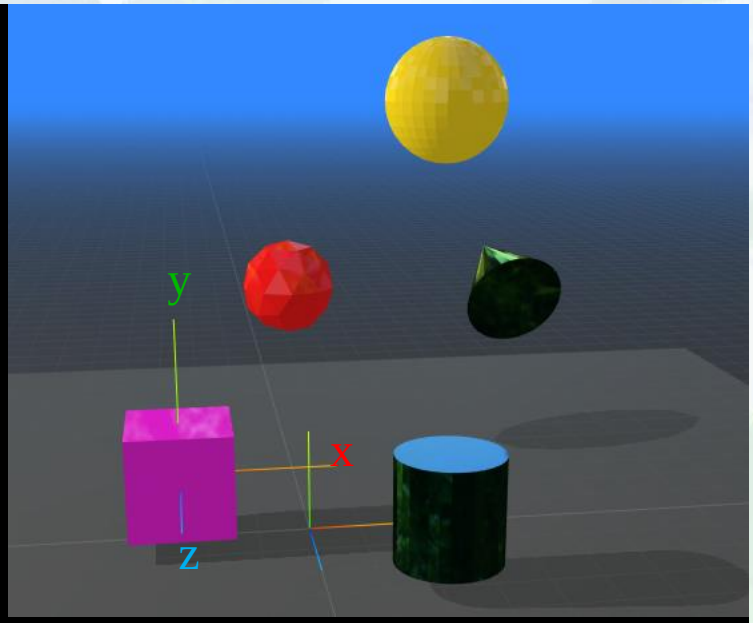
What is the Impulse Vector?

What will happen after applying the impulse to the world-point?

What differences between impulse and local impulse?

Physics – Gravity

```
/* Other lines of code are here */

engine.init().then( ( params ) => {
     userInit( params );
     engine.setGravity(new CANNON.Vec3(0, -9.8, 0));
     engine.start( );
});

var body = null;

function userInit( params ) {
     body = engine.getBodyByMeshName('Cube');
     engine.addAxesToMesh(body.threemesh, 3);
     engine.on('keydown', onKeyDown);
}

function onKeyDown(event) {
     if(event.key == 'l') {
          applyImpulse();
     }
}

function applyImpulse() {
     const worldPoint = new CANNON.Vec3 ( 0, 0, 0 );
     const force = new CANNON.Vec3 ( 0, 10, 0 );
     body.applyLocalImpulse(force, worldPoint);
}
```



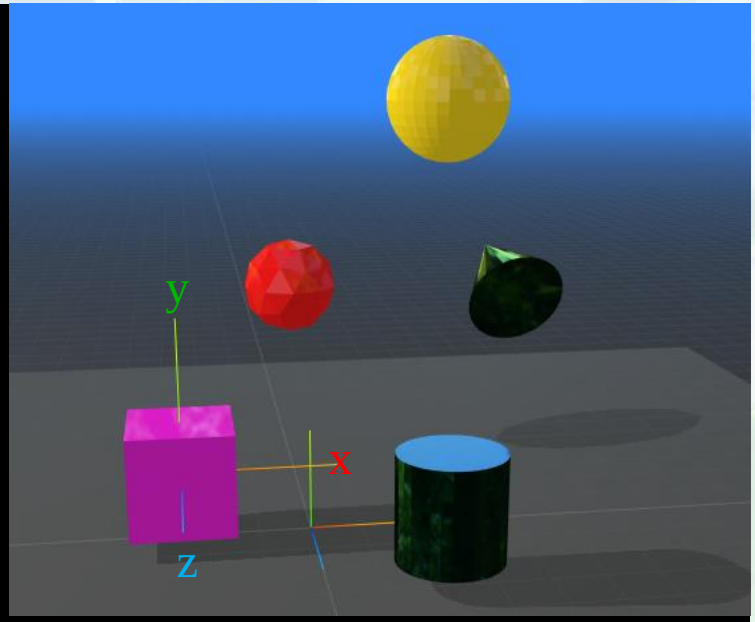
What is the Gravity Vector?

How the gravity vector effects to the rigid body?

What will happen if the direction of the gravity vector is changed?

Physics – Mass

```
/* Other lines of code are here */

engine.init().then( ( params ) => {
     userInit( params );
     engine.setGravity(new CANNON.Vec3(0, -9.8, 0));
     engine.start( );
});
var body = null;

function userInit( params ) {
     body = engine.getBodyByMeshName('Cube');
     body.mass = 5;
     engine.addAxesToMesh(body.threemesh, 3);
     engine.on('keydown', onKeyDown);
}

function onKeyDown(event) {
     if(event.key == 'l') {
          applyImpulse();
     }
}

function applyImpulse() {
     const worldPoint = new CANNON.Vec3 ( 0, 0, 0 );
     const force = new CANNON.Vec3 ( 0, 10, 0 );
     body.applyLocalImpulse(force, worldPoint);
}
```



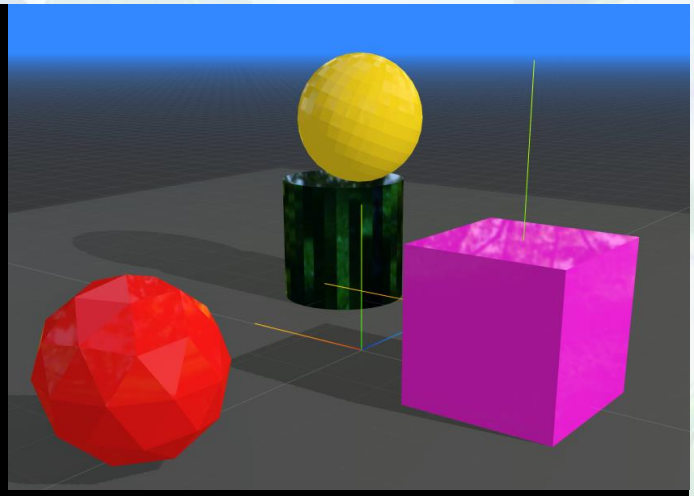
What is the Mass?

How the mass of the object effects to its movement?

What will happen if the mass of the object equals to 0 kg?

Physics – Material (Friction, Restitution)

```
/* Other lines of code are here */



engine.init().then( ( params ) => {
     userInit( params );
     engine.setGravity(new CANNON.Vec3(0, -9.8, 0));
     engine.start( );
});
var body = null;

function userInit( params ) {
     body = engine.getBodyByMeshName('Cube');
     var ground = engine.getBodyByMeshName('StaticCube');

     var groundMat = engine.createGroundMaterial(0.200, 0.5);
     var objectMat = engine.createObjectMaterial(0.001, 0.0, groundMat);

     ground.material = groundMat; /* Apply ground material       */
     body.material   = objectMat; /* Apply object material       */

     body.mass = 10;              /* Change mass of the object   */
     engine.on('keydown', onKeyDown);
}
/* Other lines of code are here */
```



friction restitution