Multimedia i interfejsy

Ćwiczenie1 – WebGL, Three.js

WebGL, będący rozszerzeniem możliwości języka JavaScript, służy do renderowania grafiki 2D i 3D. Oparty jest na API OpenGL ES 2.0. Do generowania scen wykorzystuje element canvas występujący w HTML5. Three.js jest jedną z wysokopoziomowych bibliotek umożliwiających tworzenie trójwymiarowych scen bez konieczności używania bezpośrednio API WebGL. Poniższy dokument przedstawia podstawowe informacje niezbędne do utworzenia sceny z wykorzystaniem biblioteki Three.js. Pełen opis jej możliwości można znaleźć m.in. na stronie https://threejs.org/docs/index.html.

Konstrukcja sceny

Prosty skrypt tworzący scenę może mieć następującą postać:

```
<script src = "js/three.js"></script>
     //Dołączenie biblioteki three.js


<script type="text/javascript">
var scene = new THREE.Scene();
     //Utworzenie sceny, do której dodawane będą inne obiekty.
var camera = new THREE.PerspectiveCamera(fov, aspect, near, far);
     //Utworzenie kamery (w tym wypadku stosowane będzie rzutowanie
     //perspektywiczne).
     //fov - kąt widzenia podany w stopniach
     //aspect - stosunek szerokości do wysokości okna; podanie systemowych
     //wartości window.innerWidth/window.innerHeight umożliwia dostosowanie do
     //nich rzutowania
     //near, far - bliska i daleka płaszczyzna odcięcia; tylko obiekty pomiędzy
     //nimi będą renderowane
var renderer = new THREE.WebGLRenderer();
     //Utworzenie obiektu renderer - automatycznie zostanie utworzony element
     //canvas, na którym będzie się odbywać rysowanie
renderer.setSize( width, height);
     //ustalenie rozmiarów obszaru widoku
document.body.appendChild( renderer.domElement );
     //dodanie obiektu renderer do dokumentu; może być dodany do innego
     //kontenera w dokumencie

     //…
     //tworzenie obiektów i dodawanie ich do sceny
     //…


renderer.render(scene, camera);
     //narysowanie wszystkiego, co znajduje się na scenie
</script>
```

Obiekty 2D i 3D

Każdy obiekt zbudowany jest z geometrii definiującej kształt oraz materiału, który określa kolor, przezroczystość, połyskliwość i teksturę. Umieszczając obiekty na scenie trzeba pamiętać o tym, że oś x skierowana jest w prawo, oś y do góry, oś z do przodu.

Przykładowe rodzaje geometrii:

1. 2D
   - PlaneGeometry
   - CircleGeometry
2. 3D
   - CubeGeometry
   - SphereGeometry
   - CylinderGeometry
   - TorusGeometry

Przykładowe rodzaje materiałów:

- MeshBasicMaterial – nie uwzględnia oświetlenia, tylko kolor, teksturę i przezroczystość
- MeshPhongMaterial – wydajny model cieniowania, w którym obecne są refleksy
- MeshLambert – model cieniowania, w którym jasność nie zależy od kąta patrzenia

Tworzenie przykładowego obiektu 3D:

```
var sphere_geometry = new THREE.SphereGeometry(radius);
     //definicja geometrii kuli
var material1 = new THREE.MeshPhongMaterial( {color: 0x0033ff, specular: 0x555555} );
     //definicja materiału
var sphere_mesh = new THREE.Mesh( sphere_geometry, material1 );
     //wstawienie do siatki geometrii i materiału, czyli utworzenie obiektu
     //o zadanym kształcie, pokrytego zadanym materiałem
scene.add( sphere_mesh );
     //dodanie obiektu do sceny
```

Tworzenie obiektu pokrytego teksturą.

```
var cube_geometry = new THREE.CubeGeometry(width, height, depth);
     //definicja prostopadłościanu
var texture = new THREE.TextureLoader().load( 'obraz.bmp' );
     //definicja tekstury pochodzącej z danego pliku
var material2 = new THREE.MeshBasicMaterial( { map: texture } );
     //definicja materiału
var cube_mesh = new THREE.Mesh( cube_geometry, material2 );
scene.add( cube_mesh );
```

Tworzenie obiektu pokrytego różnymi materiałami.

```
var material3 = new THREE.MeshLambertMaterial( {color: 0x550000} );

var material4 = new THREE.MeshBasicMaterial( {color: 0x550055} );

var materials = [material1, material3, material4, material3, material4,
material1];
     //tablica zawierająca 6 materiałów dla sześcianu; kolejne materiały
     //do tablicy można również dodawać w następujący sposób:
     //materials.push[material3];

cube_geometry = new THREE.BoxGeometry( 200, 200, 200 );

cube_mesh = new THREE.Mesh(cube_geometry, new THREE.MeshFaceMaterial(materials));

scene.add( cube_mesh );
```

Tworzenie obiektu pokrytego teksturą z elementu video.

```
myvideo = document.getElementById("myVideo");

var texture = new THREE.VideoTexture(myvideo);

var material = new THREE.MeshBasicMaterial( { map: texture } );

var geometry = new THREE.CubeGeometry( 30, 30, 30);

mesh = new THREE.Mesh( geometry, material );

scene.add( cube_mesh );
```

Operacje na obiektach

Ustalanie położenia

- ```
  obiekt.position.x = x;
  obiekt.position.y = y;
  obiekt.position.z = z;
  ```
- ```
  obiekt.translateX(dx);
  obiekt.translateY(dy);
  obiekt.translateZ(dz);
  ```
- ```
  obiekt.position.set(x, y, z);
  ```

Obracanie

- ```
  obiekt.rotation.x = alfax;
  obiekt.rotation.y = alfay;
  obiekt.rotation.z = alfaz;
  ```
- ```
  obiekt.rotateX(alfax);
  obiekt.rotateY(alfay);
  obiekt.rotateZ(alfaz);
  ```
- ```
  obiekt.rotation.set(alfax, alfay, alfaz);
  ```

Skalowanie

- ```
  obiekt.scale.x = sx;
  obiekt.scale.y = sy;
  obiekt.scale.z = sz;
  ```
  ```

- `obiekt.scale.set(sx, sy, sz);`

Oświetlenie

Biblioteka Three.js umożliwia zdefiniowanie kilku rodzajów oświetlenia:

- AmbientLight – światło otaczające; nie ma pozycji ani kierunku; oświetla scenę równomiernie.
- DirectionalLigth – światło kierunkowe o równoległych promieniach; ma kierunek, ale nie ma pozycji.
- PointLight – światło punktowe; ma pozycję, nie ma kierunku; promienie padają we wszystkich kierunkach na określoną odległość.
- SpotLight – światło reflektorowe; ma pozycję i kierunek; można określić kąt wewnętrzny i zewnętrzny oraz odległość, na jaką sięga

Dla każdego rodzaju światła można określić kolor i intensywność.

Przykładowe oświetlenie:

```
var light = new THREE.DirectionalLight(0xffffff, 0.8);
      //światło kierunkowe o określonej barwie i intensywności

light.position.set(0, 1, 1).normalize;
     //światło kierunkowe nie ma pozycji, w tym wypadku podane parametry
     //służą do obliczenia kierunek padania światła

scene.add( light );
     //dodanie światła do sceny
```

Ustawienia kamery

Położenie kamery można zmieniać w następujący sposób:

```
camera.position.x = x;

camera.position.y = y;

camera.position.z = z;
```

Punkt, na który skierowana jest kamera ustawiamy następująco:

```
camera.lookAt(new THREE.Vector3(x,y,z));
```