WebGL

Literatura dodatkowa:

1. WebGL Beginners Guide 2012 [literature podstawowa]
2. WebGL Insights 2015 [literature uzupełniająca]
3. *"OpenGL Programming Guide."* 2nd ed., Addison-Wesley Publishing Company
4. ftp://sgigate.sgi.com/pub/opengl/doc/
5. http://www.austin.ibm.com/software/OpenGL/
6. Norman Lin "Linux 3D Graphics Programming". Wordware Publishing 2001
7. http://www.pobox.com/~ndr/glut.html , http://reality.sgi.com/mjk/glut3/glut3.html , http://www.opengl.org , http://www.sgi.com/Technology/openGL

Proponowane biblioteki i frameworki:

a) Proponujemy zastosować babylon.js . Przykładowy kod można znaleźć na podanych poniżej stronach webowych:

https://doc.babylonjs.com/babylon101/first
https://www.babylonjs.com/
https://doc.babylonjs.com/examples/

b) Można zastosować bibliotekę three.js . Przykładowy kod można znaleźć na podanych poniżej stronach webowej
https://threejs.org/docs/index.html#manual/en/introduction/Creating-a-scene
https://github.com/mdn/webgl-examples/tree/gh-pages/tutorial/sample1

Zad 1 – Wprowadzenie

a) Do edycji i uruchamiania kodu źródłowego można wykorzystać stronę internetową

https://www.babylonjs-playground.com/

```
var createScene = function () {

    // Create the scene space
    var scene = new BABYLON.Scene(engine);

    // Add a camera to the scene and attach it to the canvas
    var camera = new BABYLON.ArcRotateCamera("Camera", Math.PI / 2, Math.PI / 2, 2,
BABYLON.Vector3.Zero(), scene);
    camera.attachControl(canvas, true);

    // Add lights to the scene
    var light1 = new BABYLON.PointLight("light1", new BABYLON.Vector3(200, 200, 0), scene);
    var light2 = new BABYLON.PointLight("light2", new BABYLON.Vector3(-200, -200, 0), scene);
    var light3 = new BABYLON.PointLight("light3", new BABYLON.Vector3(0, 0, -200), scene);
    var light4 = new BABYLON.PointLight("light4", new BABYLON.Vector3(0, 0, 200), scene);

    // Add and manipulate meshes in the scene
    var box = BABYLON.MeshBuilder.CreateBox("box", {}, scene);

    return scene;
```

```
};
```

b) Kod źródłowy może być również enkapsulowany wewnątrz strony webowej. W tym przypadku można używać dowolny edytor kodu źródłowego (vi, vim, notepad++, Adobe Brackets, sublime text, MEW, Ultraedit, etc,) oraz przeglądarkę webową do uruchamiania aplikacji. Przykładowa strona webowa została zamieszczona poniżej:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title>Babylon Template</title>
    <style>
      html, body {
        overflow: hidden; width: 100%; height: 100%; margin: 0; padding: 0;
      }
      #renderCanvas {
        width: 100%;height: 100%;touch-action: none;
      }
    </style>
    <script src="https://preview.babylonjs.com/babylon.js"></script>
    <script src="https://preview.babylonjs.com/loaders/babylonjs.loaders.min.js"></script>
    <script src="https://code.jquery.com/pep/0.4.3/pep.js"></script>
  </head>

 <body>

  <canvas id="renderCanvas" touch-action="none">
        Your browser does not support the HTML5 canvas element
  </canvas>

  <script>
    var canvas = document.getElementById("renderCanvas"); // Get the canvas element
    var engine = new BABYLON.Engine(canvas, true); // Generate the BABYLON 3D engine

    /******* Add the create scene function ******/
    var createScene = function () {
      // Create the scene space
      var scene = new BABYLON.Scene(engine);
      // Add a camera to the scene and attach it to the canvas
      var camera = new BABYLON.ArcRotateCamera("Camera", Math.PI / 2, Math.PI / 2, 2, new
BABYLON.Vector3(0,0,5), scene);
      camera.attachControl(canvas, true);
      // Add lights to the scene
      var light1 = new BABYLON.HemisphericLight("light1", new BABYLON.Vector3(1, 1, 0), scene);
      var light2 = new BABYLON.PointLight("light2", new BABYLON.Vector3(0, 1, -1), scene);
      // Add and manipulate meshes in the scene
      var sphere = BABYLON.MeshBuilder.CreateSphere("sphere", {diameter:2}, scene);
      return scene;
    };
    /******* End of the create scene function ******/

    var scene = createScene(); //Call the createScene function
    // Register a render loop to repeatedly render the scene
    engine.runRenderLoop(function () {
        scene.render();
    });
```

```
        // Watch for browser/canvas resize events
        window.addEventListener("resize", function () {
            engine.resize();
        });
    </script>
  </body>
</html>
```

Na podstawie: https://doc.babylonjs.com/babylon101/first

I. Sprawdź i przeanalizuj działanie programu.
Uwaga – podobny szkielet programu wykorzystaj w następnych ćwiczeniach.

a) Zamiast ramiona (czerwony i zielony prostokąt) wstaw sześcian.

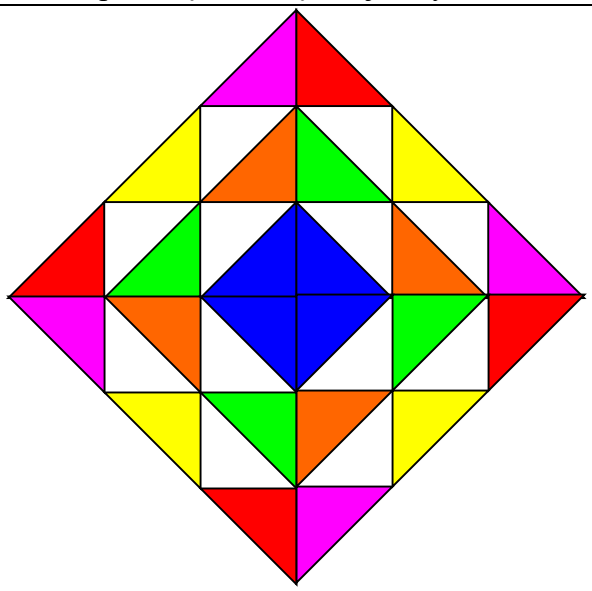
b) Użyj następujących materiałów do opisu ścian w różnych kolorach:

Np. var materialBox = BABYLON.StandardMaterial
materialBox.diffuseColor = new BABYLON.Color3(1, 0, 0);
box1.material = materialBox;
Parametry oświetlenia:
LightAmbient = (0.1, 0.1, 0.1, 0.1);
LightDiffuse = (0.7, 0.7, 0.7, 0.7);
LightSpecular = (0.0, 0.0, 0.0, 0.1);
LightPosition = (5.0, 5.0, 5.0, 0.0);

Jako wynik realizacji tego zadania wstaw końcowy kod źródłowy w języku Python oraz zrzut ekranu przedstawiający uzyskany obiekt 3D.

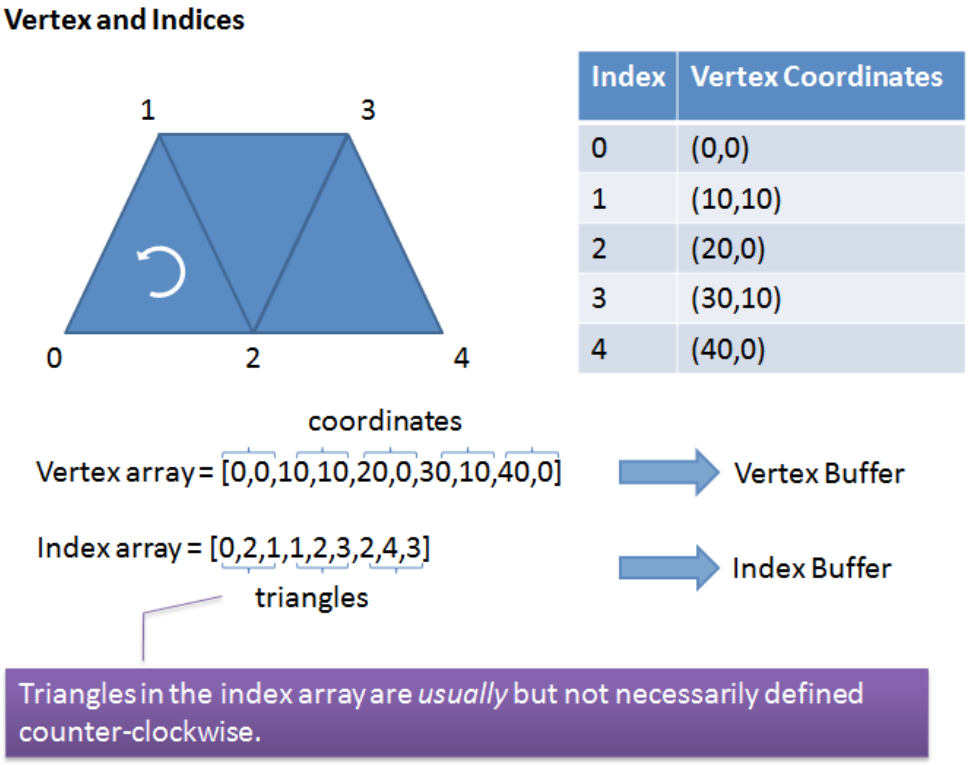
Zad 2 – Budowa podstawowych obiektów oraz opis ich ruchu

Zadanie. Narysować następujący obiekt (zbudowany z 24 trójkątów) oraz wprowadzić go w ruch obrotowy względem początkowego środka ciężkości. Wewnętrzne trójkąty obracają się w przeciwnym kierunku (3 obroty/1obrót całości) oraz oddalają się od początkowego środka ciężkości z prędkością 3 długości przyprostokątnej na 10 obrotów całej figury.
Można pominąć analizę wzajemnych zderzeń trójkątów.



Zobacz również: https://doc.babylonjs.com/how_to/custom

Przykłady konstruowania powierzchni w WebGL:

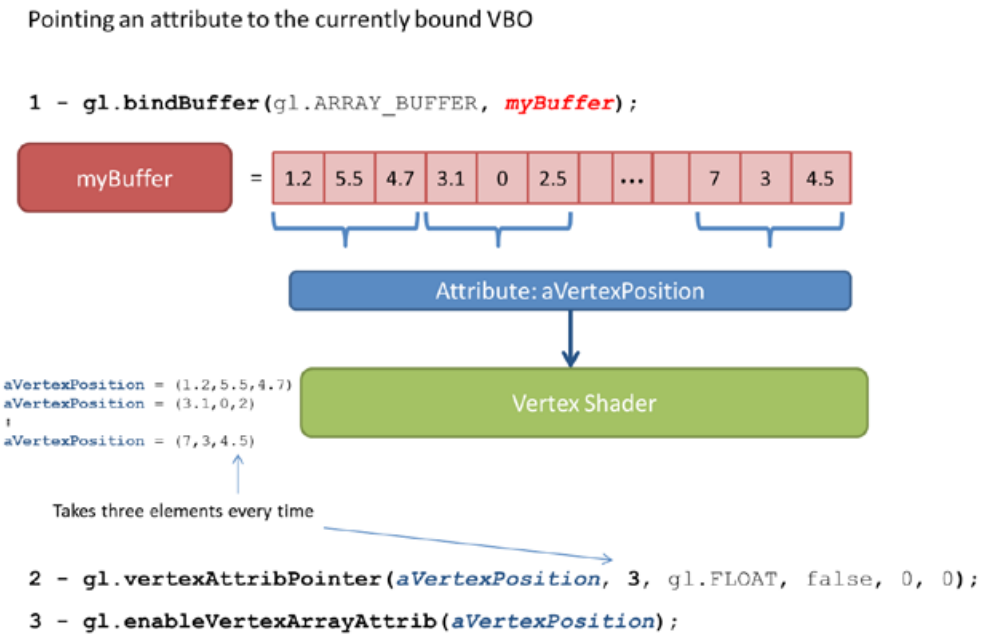


Źródło: WebGL Beginners Guide 2012

Przykładowy kod z w/w książki:

```
var coneVBO = null; //Vertex Buffer Object
var coneIBO = null; //Index Buffer Object
function initBuffers() {
var vertices = []; //JavaScript Array that populates coneVBO
var indices = []; //JavaScript Array that populates coneIBO;
//Vertices that describe the geometry of a cone
vertices =[1.5, 0, 0,
-1.5, 1, 0,
-1.5, 0.809017, 0.587785,
-1.5, 0.309017, 0.951057,
-1.5, -0.309017, 0.951057,
-1.5, -0.809017, 0.587785,
-1.5, -1, 0.0,
-1.5, -0.809017, -0.587785,
-1.5, -0.309017, -0.951057,
-1.5, 0.309017, -0.951057,
-1.5, 0.809017, -0.587785];
//Indices that describe the geometry of a cone
indices = [0, 1, 2,
0, 2, 3,
0, 3, 4,
0, 4, 5,
0, 5, 6,
0, 6, 7,
```

```
0, 7, 8,
0, 8, 9,
0, 9, 10,
0, 10, 1];
coneVBO = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, coneVBO);
gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(vertices),
gl.STATIC_DRAW);
gl.bindBuffer(gl.ARRAY_BUFFER, null);
coneIBO = gl.createBuffer();
gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, coneIBO);
gl.bufferData(gl.ELEMENT_ARRAY_BUFFER, new Uint16Array(indices),
gl.STATIC_DRAW);
gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, null);
}
```



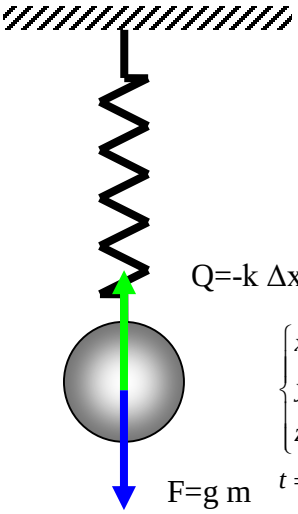
Źródło: WebGL Beginners Guide 2012

https://books.google.pl/books?id=uEmzeCKd8ZEC&pg=PT104&lpg=PT104&dq=%22ch_RenderingModes.html%22&source=bl&ots=rsQQ5FwR6b&sig=ACfU3U0HjJdVXYVwj1Obh1_df235ZpKI_g&hl=pl&sa=X&ved=2ahUKEwjfn52pyIXpAhVnk4sKHXaUDmsQ6AEwAHoECAYQAQ#v=onepage&q=%22ch_RenderingModes.html%22&f=false

Jako wynik realizacji tego zadania wstaw końcowy kod źródłowy w języku Python oraz zrzut ekranu przedstawiający uzyskane obiekty.

Ćwiczenie nr 3 – Modelowanie rzeczywistych obiektów i zjawisk

Zadanie. Zamodelować ruch sprężyny oraz kuli połączonych według zamieszczonego poniżej rysunku z uwzględnieniem praw fizyki i ich rzeczywistego wyglądu.



Dodatkowe założenia:

- Kula wykonana z drewna/szkła
- Sprężyna wykonana z drutu stalowego
- W początkowej chwili sprężyna jest naciągnięta
- Zakładamy zerową prędkość początkowa kuli.
- Model w pełni 3-D, zastosować tekstury do modelowania powierzchni.
- Powierzchnia sprężyny zamodelowana na podstawie wzoru:

$$\begin{cases} x_i = \cos(t_i)\cdot(3+\cos(u_i)) \\ y_i = \sin(t_i)\cdot(3+\cos(u_i)) \\ z_i = 0.6\cdot t_i + \sin(u_i) \end{cases}$$

$$t = 0,\ldots,8\pi;\ u = 0,\ldots,2\pi$$

- Uwzględnić górne i dolne wykończenie sprężyny (sfera-cylinder-sfera-cylinder).

Podpowiedź – dla uproszczenia zastosować równanie ruchu podane w postaci analitycznej.

Jako wynik realizacji tego zadania wstaw końcowy kod źródłowy w języku Python oraz zrzut ekranu przedstawiający uzyskany obiekt 3D.

Projekt

Wykonać model jednego z wymienionych niżej obiektów uwzględniający: kształt obiektu, tekstury, cienie, ruch obiektu, możliwość zmiany szybkości ruchu oraz położenia kamery.

1. Chodzący robot.
2. Rękę wykonującą gesty.
3. Twarz ludzką z mimiką.
4. Poruszające się zwierzę (np. pies, kot).
5. Drzewo poruszające gałęziami.
6. Grupę krzewów poruszających gałęziami.
7. Pięć wahadeł umieszczonych wzdłuż linii i zderzających się ze sobą.
8. Ryba płynąca w wodzie i wyskakująca z niej, co pewien czas.
9. Latający motyl.
10. Łódkę pływającą na falach.
11. Jadący samochód.
12. Lecący samolot.
13. Układ słoneczny.
14. Inne o podobnym poziomie trudności do uzgodnienia z prowadzącym.

Jako wynik realizacji tego zadania wstaw końcowy kod źródłowy w języku Python oraz zrzut ekranu przedstawiający uzyskany obiekt 3D.

Przykład: