Grafika Komputerowa i Komunikacja Człowiek-Komputer

Sprawozdanie z projektu

Łukasz Szumilas, Grupa E02-81o

Zajęcia: 14 stycznia 2018 (odrobione 28 stycznia)

1 Omówienie tematu

Celem ćwiczenia było zapoznanie się z elementem HTML5 *canvas*, który użyty był do rysowania grafiki w postaci 2D i 3D przy użyciu skryptów JavaScript.

. Po zapoznaniu się z instrukcją na stronie laboratorium, korzystając z odpowiednich funkcji trzeba było narysować czworościan foremny. Najważniejszym elementem było odpowiednie zdefiniowanie punktów w przestrzeni (*triangleVertices*), oraz odpowiednie ich połączenie (*triangleFaces*). Zmienić trzeba było także parametry funkcji *vertexAttribPointer*, mającej za zadanie przypisanie bufora do zmiennych atrybutów koloru i pozycji. W funkcji *drawElements* wpisujemy liczbę elementów figury. Najważniejsze fragmenty kodu i efekt wizualny zawarty jest w ***kodzie nr 1 i rysunku nr 1***.

Na zrzucie ekranu widoczne są opcje dotyczące obrotu figury wokół trzech osi, które zostały dodane dzięki instrukcji laboratoryjnej. Problemem było przyspieszanie obrotu figury za każdem naciśnięciem przycisku 'Uruchom'. Przycisk ten uruchamiał funkcję *runWebGL()*, która niepotrzebnie za każdym razem inicjalizowała zbyt wiele rzeczy. Po pierwszym uruchomieniu programu, prosta instrukcja warunkowa za każdym kolejnym razem pobierała już tylko funkcję *getRotation*, która wystarczała do odpowiedniego obrotu figury przy stałej prędkości (***kod nr 2***).

Kolejnym etapem było narysowanie fraktalu 2D przy pomocy elementów canvas. Wybraną przeze mnie figurą był trójkąt, który przy pomocy odpowiedniego algorytmu dzielił się na trzy mniejsze trójkąty, zostawiając w środku wycięty odwrócony trójkąt w wymiarach takich samych jak trzy poprzednie. Bazowało to na dywanie Sierpińskiego, gdzie trójkąt zastępuje w samopodziale inną figurę, kwadrat. Manipulując jedynie zmienną *var najwyzszyPoziom*, trójkąt dochodził do coraz to większego samopodziału. Aby urozmaicić wygląd oglądanego obrazka, zostały zastosowane losowe kolory. Odbyło się to za pomocą odpowiednio podzielonego gradientu, gdzie każdy odcinek przechodził w osobnie wylosowane kolory. Trójkąty zostawały także poddawane perturbacjom zgodnie z ustawioną zmienną *var perturbance*. Najważniejsze fragmenty i efekty zostały przedstawione w ***kodzie nr 3 i rysunku nr 2 i 3***.

2 Omówienie kodu

Kod 1, fragmenty realizujące czworościan foremny.

```
var triangleVertices = [
        1,1,1,  //w
        0,0,0,      //c
        -1,1,-1,    //w
        1,0,0,      //c
        -1,-1,1,     //w
        1,1,0,      //c
        1,-1,-1,    //w
```

```
    0,1,0      //c
  ];

  var triangleFaces = [
    0,1,2,
    0,2,3,
        0,1,3,
        1,2,3
  ];

  //fragmenty funkcji gl_draw()
      gl_ctx.vertexAttribPointer(_position, 3, gl_ctx.FLOAT, false, 4*(3+3), 0);
    gl_ctx.vertexAttribPointer(_color, 3, gl_ctx.FLOAT, false, 4*(3+3), 3*4);

       gl_ctx.drawElements(gl_ctx.TRIANGLES, 12, gl_ctx.UNSIGNED_SHORT, 0);
```

Kod 2, warunek blokujący fragment funkcji

```
      var isRunning= true;

function runWebGL () {
  getRotation();
  if(isRunning)
  {
  gl_canvas = document.getElementById("glcanvas");
  gl_ctx = gl_getContext(gl_canvas);
  gl_initShaders();
  gl_initBuffers();
  gl_setMatrix();
  gl_draw();
  isRunning = false;
  }
}
```

.

Kod 3,fraktal 2D.

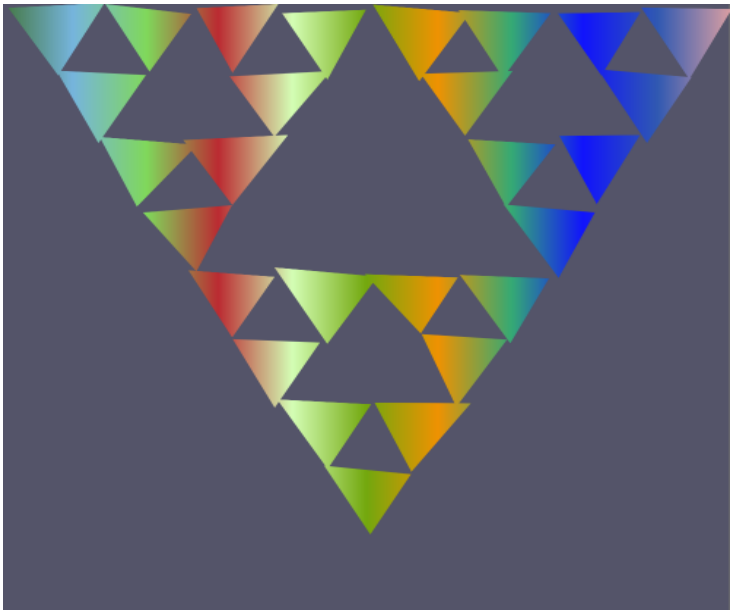
```
 //perturbacje
 var perturbance = 20;

 //najwyzszy poziom samopodzialu
 var najwyzszyPoziom = 3;

 //losowe kolory w postaci gradientu
 var my_gradient=ctx.createLinearGradient(0, 0, 1000, 0);
       my_gradient.addColorStop(0, "#"+((1<<24)*Math.random()|0).toString(16));
       my_gradient.addColorStop(0.1, "#"+((1<<24)*Math.random()|0).toString(16));
       my_gradient.addColorStop(0.2, "#"+((1<<24)*Math.random()|0).toString(16));
       my_gradient.addColorStop(0.3, "#"+((1<<24)*Math.random()|0).toString(16));
       my_gradient.addColorStop(0.4, "#"+((1<<24)*Math.random()|0).toString(16));
       my_gradient.addColorStop(0.5, "#"+((1<<24)*Math.random()|0).toString(16));
       my_gradient.addColorStop(0.6, "#"+((1<<24)*Math.random()|0).toString(16));
       my_gradient.addColorStop(0.7, "#"+((1<<24)*Math.random()|0).toString(16));
       my_gradient.addColorStop(0.8, "#"+((1<<24)*Math.random()|0).toString(16));
       my_gradient.addColorStop(0.9, "#"+((1<<24)*Math.random()|0).toString(16));
       my_gradient.addColorStop(1, "#"+((1<<24)*Math.random()|0).toString(16));
```

```
        //narysowanie fraktalu
        drawTriangles(x1,y1,szerokosc,wysokosc, poziom);

        //funkcja odpowiadajaca za rysunek fraktalu
 function drawTriangles(iks, igrek, szer, wys, ktoryPoziom)
 {
    var width = szer;
        var high = wys;
        var xP = iks;
        var yP = igrek;
        var level = Math.pow(2,ktoryPoziom);

        //odpowiedni podzial i przypisanie nowych wspolrzednych w podzielonym trojkacie
        //przy kolejnym poziomie
        width /=level;
        high /=level;


        var nextX1 = xP;
        var nextY1 = yP;

        var nextX2 = xP + width;
        var nextY2 = yP;

        var nextX3 = (xP + width)/2 + xP/2;
        var nextY3 = yP + high

        //rysowanie
        ctx.beginPath();
        //przechodzimy do rysowania po osiagnieciu najwyzszego poziomu samopodzialu
        if(ktoryPoziom == najwyzszyPoziom)
        {
        ctx.fillStyle = my_gradient//kolor = gradient
    ctx.moveTo(xP + Math.random() * perturbance,yP+ Math.random() * perturbance);
    ctx.lineTo(xP + width+ Math.random() * perturbance, yP+ Math.random() * perturbance);
    ctx.lineTo(xP + (width)/2 + Math.random() * perturbance, yP + high + Math.random() * perturban
        ctx.fill();


        ctx.moveTo(xP + width + Math.random() * perturbance, yP + Math.random() * perturbance);
    ctx.lineTo(xP + width + width + Math.random() * perturbance,yP + Math.random() * perturbance);
    ctx.lineTo((xP + width + width/2 + Math.random() * perturbance), yP + high + Math.random() * p
        ctx.fill();


        ctx.moveTo((xP + width)/2 + xP/2 + Math.random() * perturbance, yP + high + Math.random()
    ctx.lineTo((xP + width + width/2 + Math.random() * perturbance), yP + high + Math.random() * p
    ctx.lineTo((xP + width)+ Math.random() * perturbance, (yP + high + high)+ Math.random() * pert
//
        ctx.fill();
        }
        //rekurencja, dotad az nie dojdziemy do najwyszego poziomu by moc narysowac figury
        else {
                ktoryPoziom++;
                drawTriangles(nextX1,nextY1,szerokosc,wysokosc, ktoryPoziom);
```

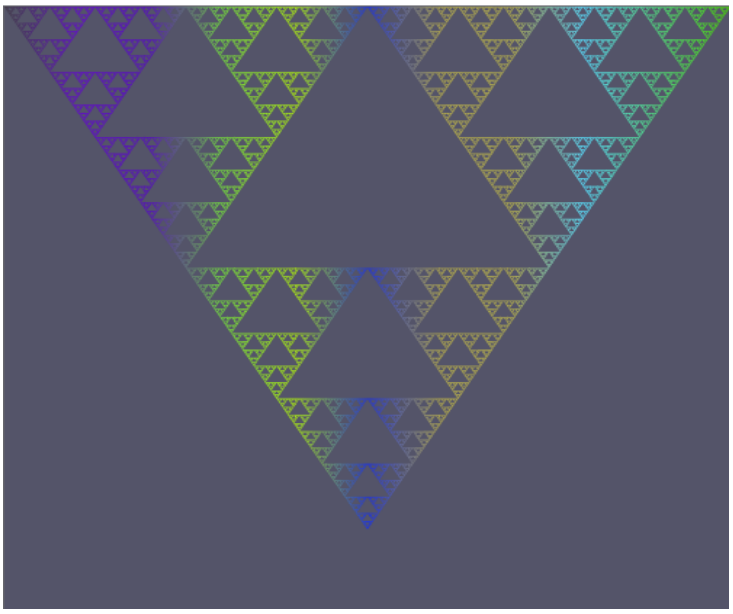
```
                drawTriangles(nextX2,nextY2,szerokosc,wysokosc, ktoryPoziom);
                drawTriangles(nextX3,nextY3,szerokosc,wysokosc, ktoryPoziom);}
}
```

3 Rezultat prac

Rysunek 1: Obracający się czworościan



Rysunek 2: Fraktal. var najwyzszyPoziom = 3, var perturbance = 20



Rysunek 3: Fraktal. var najwyzszyPoziom = 10, var perturbance = 0