Министерство образования республики Беларусь
Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»
Институт информационных технологий

Специальность ПОИТ

КОНТРОЛЬНАЯ РАБОТА

По курсу Алгоритмы компьютерной графики

Вариант № 51

Студент-заочник 3 курса
Группы № 581072
Богданова Кристина Евгеньевна
Тел. +375 (25) 929-99-11

Приняла
преподаватель
Коренская Ирина Николаевна

Минск, 2018

Содержание

Задание № 1 «Анимация и морфинг»

Цель работы: изучить теоретический материал по теме «Анимация и морфинг» и закрепить его путем выполнения индивидуального задания.

При выполнении данной работы необходимо разработать программу, обеспечивающую создание на экране семейства фигур заданной формы. Необходимо обеспечить метаморфозу многоугольника из начальной формы (пятиугольник) в заданную конечную форму (шестиугольник). Узловые точки начальной и конечной фигур берутся с использованием разного направления их обхода в контуре фигуры, например, в исходной фигуре они берутся в порядке направления хода часовой, а в конечной – в направлении против часовой стрелки.

Решение

Исходный код:

index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Задание 1</title>
    <meta charset="utf-8" />
    <script src="script\animation.js"></script>
    <style>
      body { font-family: 'Ubuntu', sans-serif; }
      #startButton {
        font-weight: 700;
        color: white;
        text-decoration: none;
        outline: none;
        margin-left: 40px;
        padding: .8em 1em calc(.8em + 3px);
        border-radius: 3px;
        background: rgb(64,199,129);
        box-shadow: 0 -3px rgb(53,167,110) inset;
        transition: 0.2s;
        user-select: none;
      }
      #startButton:hover { background: rgb(53, 167, 110); }
      #startButton:active {
        background: rgb(33,147,90);
        box-shadow: 0 3px rgb(33,147,90) inset;
      }
    </style>
</head>
<body>
    <div style="position: relative;">
```

```
        <canvas style="position: absolute; left: 0; top: 50px; z-index: 0;"
id="canvas"> </canvas>
    </div>
    <div style="position: absolute; top: 20px; left: 10px;">
        <a id="startButton" onclick="startAnimation()"></a>
    </div>
</body>
</html>
```

animation.js

```
var myFigure, x, y, H, W, pointsB, pointsE, pointsM, pointsS, trails,
useTrail, animate;
window.onload = function () {
    var canvas = document.getElementById('canvas');
    myFigure = canvas.getContext('2d');
    myFigure.beginPath();
    myFigure.fillStyle = "rgba(0, 102, 102, 1)";
    myFigure.moveTo(100, 180);
    myFigure.lineTo(180, 100);
    myFigure.lineTo(260, 180);
    myFigure.lineTo(230, 280);
    myFigure.lineTo(130, 280);
    myFigure.lineTo(100, 180);
    myFigure.lineTo(100, 180);
    myFigure.fill();
    W = canvas.width = window.innerWidth - 30;
    H = canvas.height = window.innerHeight - 60;
    var myPent = canvas.getContext("2d");
    myPent.beginPath();
    myPent.fillStyle = "rgba(0, 102, 102, 1)";
    myPent.moveTo(100, 180);
    myPent.lineTo(180, 100);
    myPent.lineTo(260, 180);
    myPent.lineTo(230, 280);
    myPent.lineTo(130, 280);
    myPent.lineTo(100, 180);
    myPent.lineTo(100, 180);
    myPent.fill();
    pointsB = [
      {_x:100, _y:180},
      {_x:180, _y:100},
      {_x:260, _y:180},
      {_x:230, _y:280},
      {_x:130, _y:280},
      {_x:100, _y:180}
    ];
    pointsE = [
      {_x:1600, _y:700},
      {_x:1600, _y:600},
      {_x:1680, _y:550},
      {_x:1760, _y:600},
      {_x:1760, _y:700},
      {_x:1680, _y:750}
```

```
    ];
    reset();
};

function reset() {
    x = 100;
    y = 80;
    document.getElementById("startButton").textContent = "Начать анимацию";
    pointsM = [
      {_x:100, _y:180},
      {_x:180, _y:100},
      {_x:260, _y:180},
      {_x:230, _y:280},
      {_x:130, _y:280},
      {_x:100, _y:180}
    ];
    pointsS = [
      {_x:100, _y:180},
      {_x:180, _y:100},
      {_x:260, _y:180},
      {_x:230, _y:280},
      {_x:130, _y:280},
      {_x:100, _y:180}
    ];
}

function draw(points) {
    drawHexagon(points, "rgba(0, 102, 102, 1)");
}

function drawHexagon(points, color) {
    myFigure.beginPath();
    myFigure.fillStyle = color;
    myFigure.moveTo(points[0]._x, points[0]._y);
    myFigure.lineTo(points[1]._x, points[1]._y);
    myFigure.lineTo(points[2]._x, points[2]._y);
    myFigure.lineTo(points[3]._x, points[3]._y);
    myFigure.lineTo(points[4]._x, points[4]._y);
    myFigure.lineTo(points[5]._x, points[5]._y);
    myFigure.lineTo(points[0]._x, points[0]._y);
    myFigure.fill();
}

function drawTrail(pointsM) {
    drawHexagon(pointsM, "rgba(255, 255, 255, 1)");
    myFigure.strokeStyle = "rgba(0, 0, 0, 0.5)";
    myFigure.lineWidth = 1;
    myFigure.stroke();
}

function startAnimation() {
    if (animate) {
        document.getElementById("startButton").textContent = "Начать
анимацию";
        clearTimeout(animate);
```

```
        animate = null;
    }
    else {
        reset();
        animate = true;
        document.getElementById("startButton").textContent = "Закончить
анимацию";
        useTrail = false;
        animation();
    }
}

function animation() {
    var currentX = x;
    var currentY = y;
    x += 1500;
    y += 500;
    for (var i = 0; i < 6; i++) {
      pointsS[i]._x = (pointsE[i]._x - pointsB[i]._x) / 500;
      pointsS[i]._y = (pointsE[i]._y - pointsB[i]._y) / 500;
    }
    var stepX = (x - currentX) / 500;
    var stepY = (y - currentY) / 500;
    var k = 0;
    function moveToPoint() {
        if (pointsM[0]._x <= pointsE[0]._x) {
            currentX += stepX;
            currentY += stepY;
            myFigure.clearRect(0, 0, myFigure.canvas.width,
myFigure.canvas.height);
          if (k == 100) {
            trails.push.apply( trails, pointsM );
            k = 0;
          }
            if (useTrail) {
                trails.forEach(function(element) {
                    drawTrail(element);
                }, this);
            }
          for (var i = 0; i < 6; i++) {
            pointsM[i]._x += pointsS[i]._x;
            pointsM[i]._y += pointsS[i]._y;
          }
            draw(pointsM);

            if (animate) {
                window.requestAnimationFrame(moveToPoint);
            }
        }
        else {
            trails.push(pointsM);
            document.getElementById("startButton").textContent = "Начать
анимацию";
            clearTimeout(animate);
            animate = null;
```

```
            }
        }
        moveToPoint();
}
```

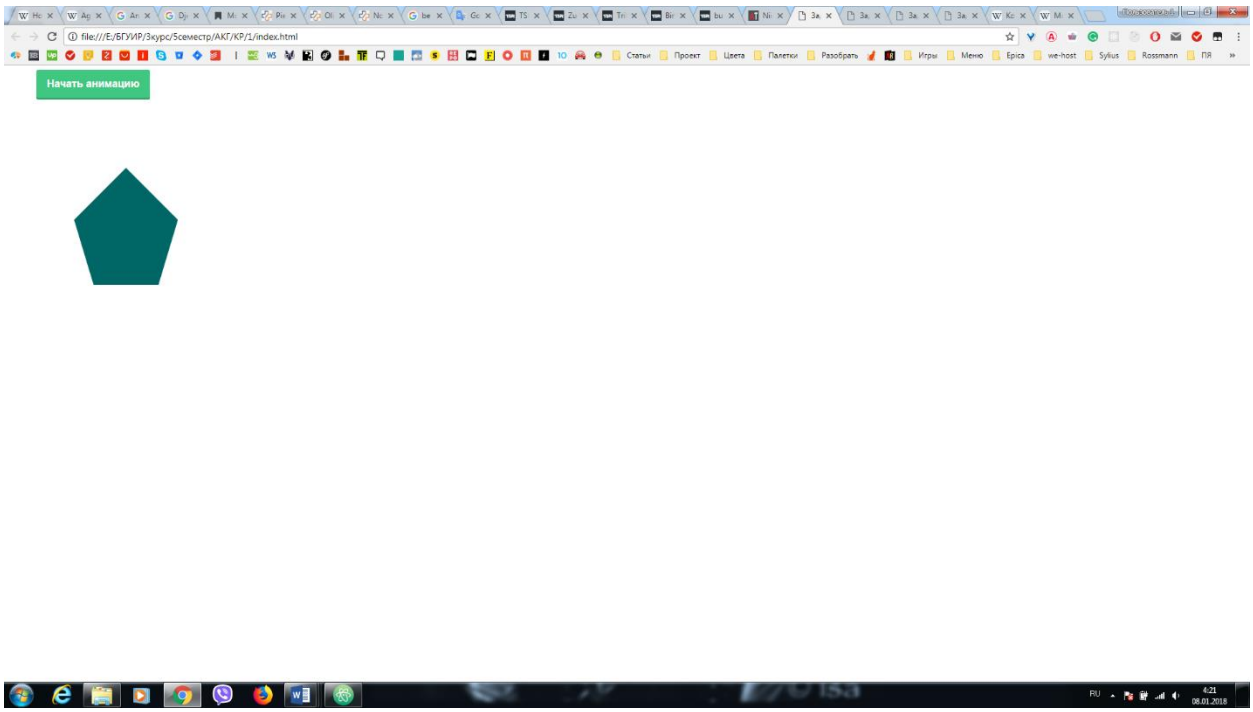


Рисунок 1 – Начальное состояние

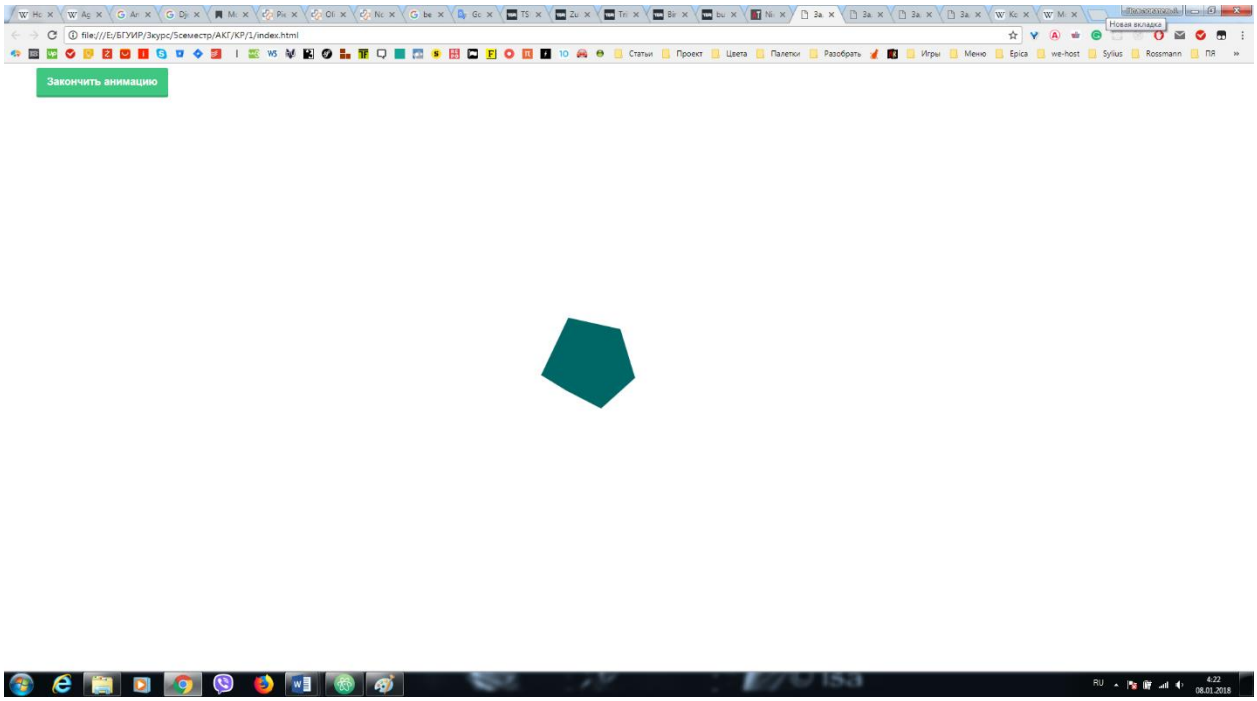


Рисунок 2 – Промежуточное состояние

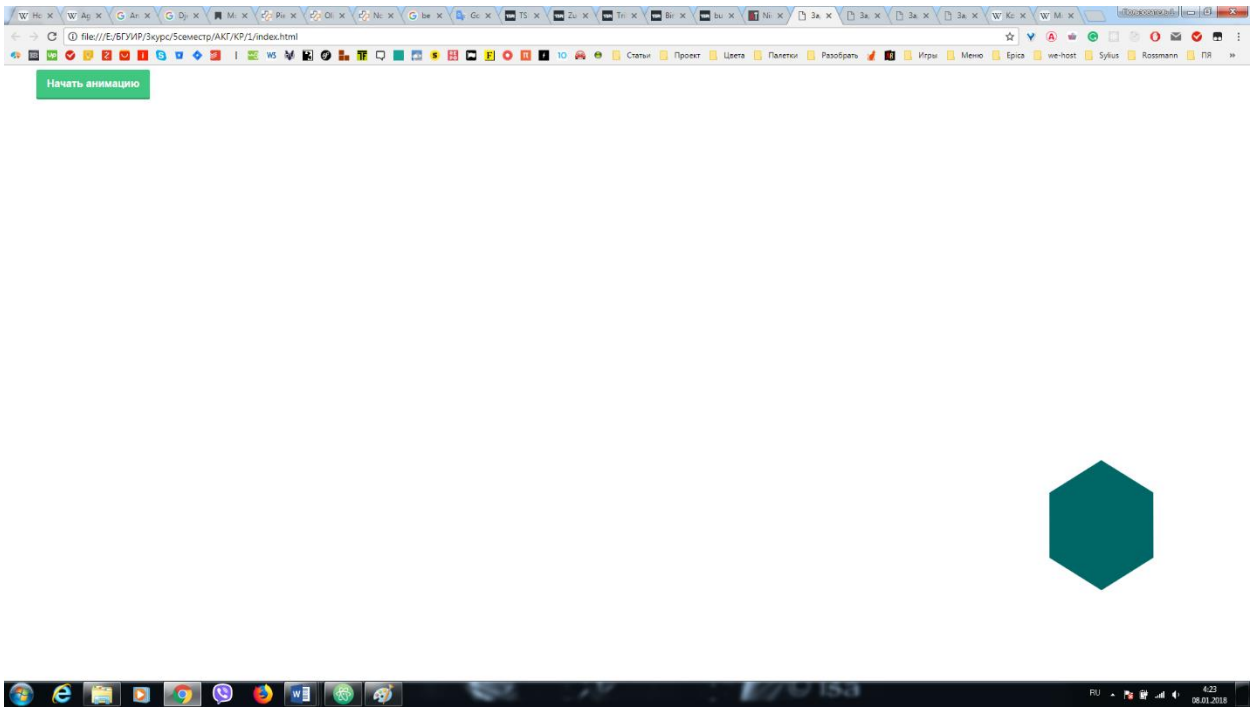


Рисунок 3 – Конечное состояние

Выводы

Анимация (от фр. animation — оживление, одушевление) – компьютерная имитация движения с помощью изменения и перерисовки формы объектов или показа последовательных изображений с фазами движения, подготовленных заранее или порождаемых во время анимации.

Может применяться в компьютерных играх, мультимедийных приложениях (например, энциклопедиях), а также для «оживления» отдельных элементов оформления, например, веб-страниц и рекламы (анимированные баннеры).

Морфинг (англ. morphing, трансформация) — технология в компьютерной анимации, визуальный эффект, создающий впечатление плавной трансформации одного объекта в другой. Для создания эффекта на изображениях задают ключевые точки, которые помогают выполнить правильный морфинг, то есть создать изображения промежуточных состояний (интерполируя имеющиеся данные).

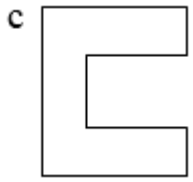
Используется в игровом и телевизионном кино, в телевизионной рекламе. Морфинг также часто используется, если не стоит задача добиться эффекта превращения одного объекта в другой, а нужно выстроить промежуточные состояния между ключевыми положениями анимируемого объекта.

При выполнении данной работы была осуществлена метаморфоза многоугольника из начальной формы (пятиугольник) в заданную конечную форму (шестиугольник) путем анимации и морфинга.

Задание № 2 «Отсечение прямоугольным окном»

Цель работы: изучить теоретический материал по теме «Отсечение прямоугольным окном» и закрепить его путем выполнения индивидуального задания.

При выполнении данной работы необходимо написать программу, выполняющую заданное (внутреннее или внешнее) отсечения окном. Форма окна определяется индивидуальным заданием. Программы должны быть основаны на алгоритме отсечения прямоугольным окном Сазерленда-Коуэна. Описание этого алгоритма приведено в тексте лекций по данной дисциплине. Работу составленной программы необходимо продемонстрировать на примере отсечения перемещающейся фигуры, полученной в результате выполнения задания 1, окном заданной формы.



№ варианта	Вид отсечения	Вид окна
3	Внешнее	3

Решение

Исходный код:

index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Задание 2</title>
    <meta charset="utf-8" />
    <script src="script\animation.js"></script>
    <style>
      body { font-family: 'Ubuntu', sans-serif; }
      #startButton {
        font-weight: 700;
        color: white;
        text-decoration: none;
        outline: none;
        margin-left: 40px;
        padding: .8em 1em calc(.8em + 3px);
        border-radius: 3px;
        background: rgb(64,199,129);
        box-shadow: 0 -3px rgb(53,167,110) inset;
        transition: 0.2s;
        user-select: none;
      }
      #startButton:hover { background: rgb(53, 167, 110); }
```

```
      #startButton:active {
        background: rgb(33,147,90);
        box-shadow: 0 3px rgb(33,147,90) inset;
      }
    </style>
</head>
<body>
    <div style="position: relative;">
        <canvas style="position: absolute; left: 0; top: 50px; z-index: 0;"
id="canvas"> </canvas>
        <canvas style="position: absolute; left: 0; top: 50px; z-index: 1;"
id="canvas1"></canvas>
    </div>
    <div style="position: absolute; top: 20px; left: 10px;">
        <a id="startButton" onclick="startAnimation()"></a>
    </div>
</body>
</html>
```

animation.js

```
var myFigure, myPent1, x, y, H, W, pointsB, pointsE, pointsM, pointsS,
trails, useTrail, animate;
window.onload = function () {
    var canvas = document.getElementById('canvas');
    W = canvas.width = window.innerWidth - 30;
    H = canvas.height = window.innerHeight - 60;
    myFigure = canvas.getContext('2d');
    myFigure.beginPath();
    myFigure.fillStyle = "rgba(0, 102, 102, 1)";
    myFigure.moveTo(100, 180);
    myFigure.lineTo(180, 100);
    myFigure.lineTo(260, 180);
    myFigure.lineTo(230, 280);
    myFigure.lineTo(130, 280);
    myFigure.lineTo(100, 180);
    myFigure.lineTo(100, 180);
    myFigure.fill();
    var myPent = canvas.getContext("2d");
    myPent.beginPath();
    myPent.fillStyle = "rgba(0, 102, 102, 1)";
    myPent.moveTo(100, 180);
    myPent.lineTo(180, 100);
    myPent.lineTo(260, 180);
    myPent.lineTo(230, 280);
    myPent.lineTo(130, 280);
    myPent.lineTo(100, 180);
    myPent.lineTo(100, 180);
```

```
myPent.fill();
pointsB = [
  {_x:100, _y:180},
  {_x:180, _y:100},
  {_x:260, _y:180},
  {_x:230, _y:280},
  {_x:130, _y:280},
  {_x:100, _y:180}
pointsE = [
  {_x:1600, _y:700},
  {_x:1600, _y:600},
  {_x:1680, _y:550},
  {_x:1760, _y:600},
  {_x:1760, _y:700},
  {_x:1680, _y:750}
];
var canvas1 = document.getElementById('canvas1');
W = canvas.width = canvas1.width = window.innerWidth - 30;
H = canvas.height = canvas1.height = window.innerHeight - 60;
var myWin = canvas1.getContext("2d");
myWin.beginPath();
myWin.fillStyle = "rgba(255, 255, 255, 1)";
myWin.moveTo(600, 100);
myWin.lineTo(1200, 100);
myWin.lineTo(1200, 300);
myWin.lineTo(800, 300);
myWin.lineTo(800, 500);
myWin.lineTo(1200, 500);
myWin.lineTo(1200, 700);
myWin.lineTo(600, 700);
myWin.lineTo(600, 100);
myWin.fill();
myWin.strokeStyle = "rgba(0, 0, 0, 1)";
myWin.lineWidth = 1;
myWin.stroke();
var myBack = canvas1.getContext("2d");
myBack.beginPath();
myBack.fillStyle = "rgba(255, 255, 255, 0)";
myBack.moveTo(0, 0);
myBack.lineTo(0, H);
myBack.lineTo(600, H);
myBack.lineTo(600, 100);
myBack.lineTo(1200, 100);
myBack.lineTo(1200, 300);
myBack.lineTo(800, 300);
```

```
    myBack.lineTo(800, 500);
    myBack.lineTo(1200, 500);
    myBack.lineTo(1200, 700);
    myBack.lineTo(600, 700);
    myBack.lineTo(600, H);
    myBack.lineTo(W, H);
    myBack.lineTo(W, 0);
    myBack.fill();
    reset();
};
function reset() {
    x = 100;
    y = 80;
    document.getElementById("startButton").textContent = "Начать анимацию";
    pointsM = [
      {_x:100, _y:180},
      {_x:180, _y:100},
      {_x:260, _y:180},
      {_x:230, _y:280},
      {_x:130, _y:280},
      {_x:100, _y:180}
    ];
    pointsS = [
      {_x:100, _y:180},
      {_x:180, _y:100},
      {_x:260, _y:180},
      {_x:230, _y:280},
      {_x:130, _y:280},
      {_x:100, _y:180}
    ];
}
function draw(points) {
    drawHexagon(points, "rgba(0, 102, 102, 1)");
}
function drawHexagon(points, color) {
    myFigure.beginPath();
    myFigure.fillStyle = color;
    myFigure.moveTo(points[0]._x, points[0]._y);
    myFigure.lineTo(points[1]._x, points[1]._y);
    myFigure.lineTo(points[2]._x, points[2]._y);
    myFigure.lineTo(points[3]._x, points[3]._y);
    myFigure.lineTo(points[4]._x, points[4]._y);
    myFigure.lineTo(points[5]._x, points[5]._y);
    myFigure.lineTo(points[0]._x, points[0]._y);
    myFigure.fill();
```

```
}
function drawTrail(pointsM) {
    drawHexagon(pointsM, "rgba(255, 255, 255, 1)");
    myFigure.strokeStyle = "rgba(0, 0, 0, 0.5)";
    myFigure.lineWidth = 1;
    myFigure.stroke();
}
function startAnimation() {
    if (animate) {
        document.getElementById("startButton").textContent = "Начать
анимацию";
        clearTimeout(animate);
        animate = null;
    }
    else {
        reset();
        animate = true;
        document.getElementById("startButton").textContent = "Закончить
анимацию";
        useTrail = false;
        animation();
    }
}
function animation() {
    var currentX = x;
    var currentY = y;
    x += 1500;
    y += 500;
    for (var i = 0; i < 6; i++) {
      pointsS[i]._x = (pointsE[i]._x - pointsB[i]._x) / 500;
      pointsS[i]._y = (pointsE[i]._y - pointsB[i]._y) / 500;
    }
    var stepX = (x - currentX) / 500;
    var stepY = (y - currentY) / 500;
    var k = 0;
    function moveToPoint() {
        if (pointsM[0]._x <= pointsE[0]._x) {
            currentX += stepX;
            currentY += stepY;
            myFigure.clearRect(0, 0, myFigure.canvas.width,
myFigure.canvas.height);
          if (k == 100) {
            trails.push.apply( trails, pointsM );
            k = 0;
          }
```

```
            if (useTrail) {
                trails.forEach(function(element) {
                    drawTrail(element);
                }, this);
            }
          for (var i = 0; i < 6; i++) {
            pointsM[i]._x += pointsS[i]._x;
            pointsM[i]._y += pointsS[i]._y;
          }
            draw(pointsM);
            if (animate) {
                window.requestAnimationFrame(moveToPoint);
            }
        }
        else {
            trails.push(pointsM);
            document.getElementById("startButton").textContent = "Начать
анимацию";
            clearTimeout(animate);
            animate = null;
        }
    }
    moveToPoint();
}
```

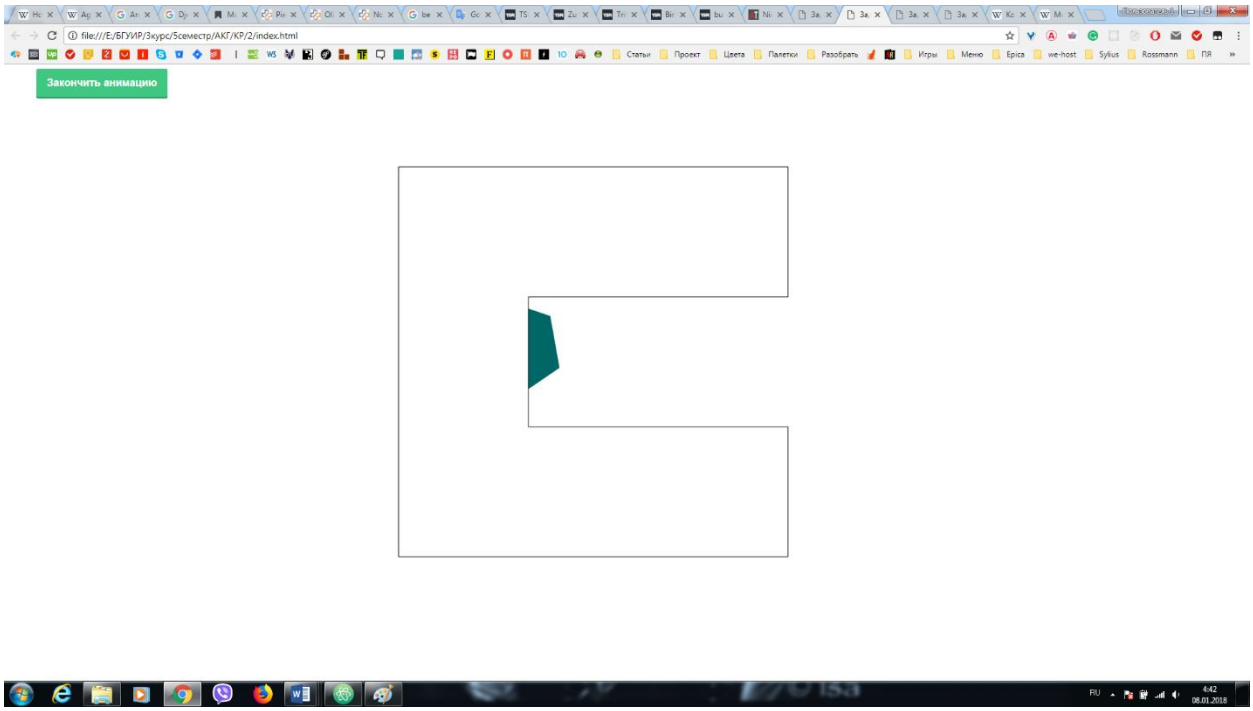


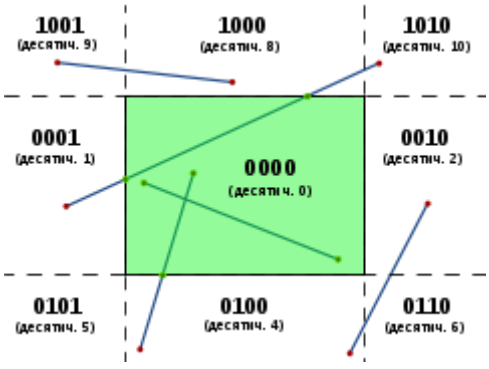
Рисунок 4 – Процесс выполнения

Выводы

Алгоритм Коэна — Сазерленда (англ. Cohen–Sutherland) — алгоритм отсечения отрезков, то есть алгоритм, позволяющий определить часть отрезка, которая пересекает прямоугольник. Алгоритм разделяет плоскость на 9 частей прямыми, которые образуют стороны прямоугольника. Каждой из 9 частей присваивается четырёхбитный код. Биты (от младшего до старшего) значат «левее», «правее», «ниже», «выше». Иными словами, у тех трёх частей плоскости, которые слева от прямоугольника, младший бит равен 1, и так далее.

Алгоритм определяет код концов отрезка. Если оба кода равны нулю, то отрезок полностью находится в прямоугольнике. Если битовое И кодов не равно нулю, то отрезок не пересекает прямоугольник (так как это значит, что оба конца отрезка находятся с одной стороны прямоугольника). В прочих случаях, алгоритм выбирает конец отрезка (или один из концов), имеющий ненулевой код (то есть находящийся вне прямоугольника), находит ближайшую к нему точку пересечения отрезка с одной из прямых, образующих стороны прямоугольника, и использует эту точку пересечения как новый конец отрезка. Укороченный отрезок снова пропускается через алгоритм.

Реализация алгоритма для трёхмерной модели идентична двумерной реализации, за исключением того, что вместо четырёхразрядного кода применяется шестиразрядный (дополнительные два бита глубины).



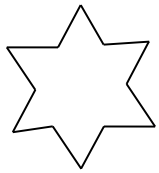
При внутреннем отсечении отображаются только те фрагменты отсекаемой фигуры, которые попадают во внутреннюю область окна, а все остальные отбрасываются. При внешнем отсечении отображается лишь то, что оказывается вне окна.

При выполнении задания было выполнено внешнее отсечение перемещающейся фигуры, полученной в результате выполнения задания 1, окном заданной формы.

Задание № 3 «Отсечение многоугольным окном»

Цель работы: изучить теоретический материал по теме «Отсечение многоугольным окном» и закрепить его путем выполнения индивидуального задания.

При выполнении данной работы необходимо написать программу, выполняющую заданное (внутреннее или внешнее) отсечения окном заданной формы. Форма окна определяется индивидуальным заданием. Программы должны быть основаны на алгоритме отсечения прямоугольным окном Кируса-Бэка. Работу составленной программы необходимо продемонстрировать на примере отсечения перемещающейся фигуры, полученной в результате выполнения задания 1, окном заданной формы.



№ варианта	Вид отсечения	Вид окна
1	внешнее	В

Решение

Исходный код:

index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Задание 3</title>
    <meta charset="utf-8" />
    <script src="script\animation.js"></script>
    <style>
      body { font-family: 'Ubuntu', sans-serif; }
      #startButton {
        font-weight: 700;
        color: white;
        text-decoration: none;
        outline: none;
        margin-left: 40px;
        padding: .8em 1em calc(.8em + 3px);
        border-radius: 3px;
        background: rgb(64,199,129);
        box-shadow: 0 -3px rgb(53,167,110) inset;
        transition: 0.2s;
        user-select: none;
      }
      #startButton:hover { background: rgb(53, 167, 110); }
      #startButton:active {
        background: rgb(33,147,90);
```

```
        box-shadow: 0 3px rgb(33,147,90) inset;
      }
    </style>
</head>
<body>
    <div style="position: relative;">
        <canvas style="position: absolute; left: 0; top: 50px; z-index: 0;"
id="canvas"> </canvas>
        <canvas style="position: absolute; left: 0; top: 50px; z-index: 1;"
id="canvas1"></canvas>
    </div>
    <div style="position: absolute; top: 20px; left: 10px;">
        <a id="startButton" onclick="startAnimation()"></a>
    </div>
</body>
</html>
```

animation.js

```
var myFigure, myPent1, x, y, H, W, pointsB, pointsE, pointsM, pointsS,
trails, useTrail, animate;
window.onload = function () {
    var canvas = document.getElementById('canvas');
    W = canvas.width = window.innerWidth - 30;
    H = canvas.height = window.innerHeight - 60;
    myFigure = canvas.getContext('2d');
    myFigure.beginPath();
    myFigure.fillStyle = "rgba(0, 102, 102, 1)";
    myFigure.moveTo(100, 180);
    myFigure.lineTo(180, 100);
    myFigure.lineTo(260, 180);
    myFigure.lineTo(230, 280);
    myFigure.lineTo(130, 280);
    myFigure.lineTo(100, 180);
    myFigure.lineTo(100, 180);
    myFigure.fill();
    var myPent = canvas.getContext("2d");
    myPent.beginPath();
    myPent.fillStyle = "rgba(0, 102, 102, 1)";
    myPent.moveTo(100, 180);
    myPent.lineTo(180, 100);
    myPent.lineTo(260, 180);
    myPent.lineTo(230, 280);
    myPent.lineTo(130, 280);
    myPent.lineTo(100, 180);
    myPent.lineTo(100, 180);
    myPent.fill();
```

```
pointsB = [
  {_x:100, _y:180},
  {_x:180, _y:100},
  {_x:260, _y:180},
  {_x:230, _y:280},
  {_x:130, _y:280},
  {_x:100, _y:180}
];
pointsE = [
  {_x:1600, _y:700},
  {_x:1600, _y:600},
  {_x:1680, _y:550},
  {_x:1760, _y:600},
  {_x:1760, _y:700},
  {_x:1680, _y:750}
];
var canvas1 = document.getElementById('canvas1');
W = canvas.width = canvas1.width = window.innerWidth - 30;
H = canvas.height = canvas1.height = window.innerHeight - 60;
var myWin = canvas1.getContext("2d");
myWin.beginPath();
myWin.fillStyle = "rgba(255, 255, 255, 1)";
myWin.moveTo(700, 325);
myWin.lineTo(820, 325);
myWin.lineTo(900, 200);
myWin.lineTo(980, 325);
myWin.lineTo(1100, 325);
myWin.lineTo(1050, 450);
myWin.lineTo(1100, 575);
myWin.lineTo(980, 575);
myWin.lineTo(900, 700);
myWin.lineTo(820, 575);
myWin.lineTo(700, 575);
myWin.lineTo(750, 450);
myWin.lineTo(700, 325);
myWin.fill();
myWin.strokeStyle = "rgba(0, 0, 0, 1)";
myWin.lineWidth = 1;
myWin.stroke();
var myBack = canvas1.getContext("2d");
myBack.beginPath();
myBack.fillStyle = "rgba(255, 255, 255, 0)";
myBack.moveTo(0, 0);
myBack.lineTo(0, H);
myBack.lineTo(700, H);
```

```
    myBack.lineTo(700, 325);
    myBack.lineTo(820, 325);
    myBack.lineTo(900, 200);
    myBack.lineTo(980, 325);
    myBack.lineTo(1100, 325);
    myBack.lineTo(1050, 450);
    myBack.lineTo(1100, 575);
    myBack.lineTo(980, 575);
    myBack.lineTo(900, 700);
    myBack.lineTo(820, 575);
    myBack.lineTo(700, 575);
    myBack.lineTo(750, 450);
    myBack.lineTo(700, 325);
    myBack.lineTo(700, H);
    myBack.lineTo(W, H);
    myBack.lineTo(W, 0);
    myBack.fill();
    reset();
};
function reset() {
    x = 100;
    y = 80;
    document.getElementById("startButton").textContent = "Начать анимацию";
    pointsM = [
      {_x:100, _y:180},
      {_x:180, _y:100},
      {_x:260, _y:180},
      {_x:230, _y:280},
      {_x:130, _y:280},
      {_x:100, _y:180}
    ];
    pointsS = [
      {_x:100, _y:180},
      {_x:180, _y:100},
      {_x:260, _y:180},
      {_x:230, _y:280},
      {_x:130, _y:280},
      {_x:100, _y:180}
    ];
}
function draw(points) {
    drawHexagon(points, "rgba(0, 102, 102, 1)");
}
function drawHexagon(points, color) {
    myFigure.beginPath();
```

```
    myFigure.fillStyle = color;
    myFigure.moveTo(points[0]._x, points[0]._y);
    myFigure.lineTo(points[1]._x, points[1]._y);
    myFigure.lineTo(points[2]._x, points[2]._y);
    myFigure.lineTo(points[3]._x, points[3]._y);
    myFigure.lineTo(points[4]._x, points[4]._y);
    myFigure.lineTo(points[5]._x, points[5]._y);
    myFigure.lineTo(points[0]._x, points[0]._y);
    myFigure.fill();
}
function drawTrail(pointsM) {
    drawHexagon(pointsM, "rgba(255, 255, 255, 1)");
    myFigure.strokeStyle = "rgba(0, 0, 0, 0.5)";
    myFigure.lineWidth = 1;
    myFigure.stroke();
}
function startAnimation() {
    if (animate) {
        document.getElementById("startButton").textContent = "Начать
анимацию";
        clearTimeout(animate);
        animate = null;
    }
    else {
        reset();
        animate = true;
        document.getElementById("startButton").textContent = "Закончить
анимацию";
        useTrail = false;
        //useTrail = true;
        animation();
    }
}
function animation() {
    var currentX = x;
    var currentY = y;
    x += 1500;
    y += 500;
    for (var i = 0; i < 6; i++) {
      pointsS[i]._x = (pointsE[i]._x - pointsB[i]._x) / 500;
      pointsS[i]._y = (pointsE[i]._y - pointsB[i]._y) / 500;
    }
    var stepX = (x - currentX) / 500;
    var stepY = (y - currentY) / 500;
```

```
    var k = 0;
    function moveToPoint() {
        if (pointsM[0]._x <= pointsE[0]._x) {
            currentX += stepX;
            currentY += stepY;
            myFigure.clearRect(0, 0, myFigure.canvas.width,
myFigure.canvas.height);
          if (k == 100) {
            trails.push.apply( trails, pointsM );
            k = 0;
          }
            if (useTrail) {
                trails.forEach(function(element) {
                    drawTrail(element);
                }, this);
            }
          for (var i = 0; i < 6; i++) {
            pointsM[i]._x += pointsS[i]._x;
            pointsM[i]._y += pointsS[i]._y;
          }
            draw(pointsM);
            if (animate) {
                window.requestAnimationFrame(moveToPoint);
            }
        }
        else {
            trails.push(pointsM);
            document.getElementById("startButton").textContent = "Начать
анимацию";
            clearTimeout(animate);
            animate = null;
        }
    }
    moveToPoint();
}
```

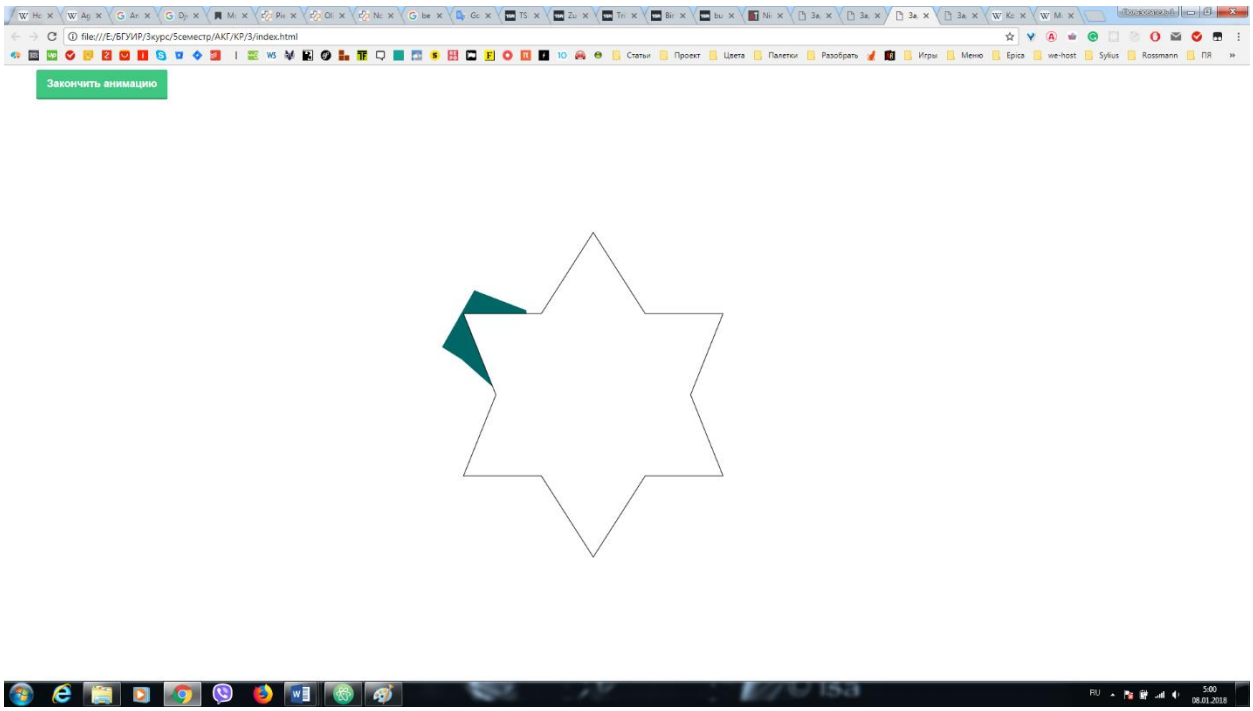


Рисунок 5 – Процесс выполнения

Выводы

Алгоритм Кируса — Бека (англ. Cyrus — Beck) — алгоритм отсечения отрезков произвольным выпуклым многоугольником. Был предложен в качестве более эффективной замены алгоритма Коэна — Сазерленда, который выполняет отсечение за несколько итераций.

Отсекаемые отрезки представляются в параметрическом виде:

$$p(t) = p_0 + t(p_1 - p_0), t \in [0, 1]$$

где

p_0, p_1 — координаты начала и конца отрезка соответственно,

t — параметр.

Каждый отсекаемый отрезок содержит координаты начала и конца, а также два параметра tA и tB, соответствующие началу и концу отрезка.

Для каждого отсекаемого отрезка P выполняются следующие действия:

Рёбра отсекающего многоугольника обходятся против часовой стрелки. Для каждого ребра E вычисляется параметр tE, описывающий пересечение E с прямой, на которой лежит отрезок P. Вычисляется скалярное произведение вектора E и внешней нормали N, в зависимости от знака которого возникает одна из следующих ситуаций:

E · N < 0 — отрезок P направлен с внутренней на внешнюю сторону ребра E. В этом случае параметр tA заменяется на tE, если tE > tA.

E · N > 0 — отрезок P направлен с внешней на внутреннюю сторону ребра E. В этом случае параметр tB заменяется на tE, если tE < tB.

E · N = 0 — отрезок P параллелен ребру E. Отрезок P отбрасывается как невидимый, если находится по правую сторону от E.

Если t_A <= t_B, то заданная параметрами tA и tB часть отрезка P видима. В противном случае отрезок P полностью невидим.

При выполнении задания было выполнено внешнее отсечение перемещающейся фигуры, полученной в результате выполнения задания 1, окном заданной формы.

Задание № 4 «Построение проекции трехмерного объекта»

Цель работы: изучить теоретический материал по теме «Построение проекции трехмерного объекта» и закрепить его путем выполнения индивидуального задания.

При выполнении данной лабораторной работы необходимо построить заданную проекцию заданной фигуры.

№ вар.	ди-на-мика	Вид проекции	осн. при-змы	параметры проекции			
				ρ	θ (a,U_y)	φ(b,U_x)	d
3	$X_в$	изомет.	6				

Решение

Исходный код:

index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Задание 4</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, user-scalable=no, minimum-scale=1.0, maximum-scale=1.0">
    <style>
        body {
            color: #ffffff;
            font-family: Monospace;
            font-size: 13px;
            text-align: center;
            font-weight: bold;
            background-color: #000000;
            margin: 0px;
            overflow: hidden;
        }
        #info {
            color: #fff;
            position: absolute;
            top: 0px;
            width: 100%;
            padding: 5px;
            z-index: 100;
```

```
        }
    </style>
</head>

<body>
    <script src="three.js"></script>
    <script>
        var arrayLand = Array(6);
        var arrayEdges = Array(6);
        var arrayBottom = Array(6);
        var height = 100;
        var rotation = 0;
        var size = 2;
        var camera, scene, renderer;
        var mesh;
        var line;
        var line2;
        var dash = 1;
        var geometry;
        var geometry2;
        var material;
        var material2;
        var arr = Array(1);
        var arrayPoints = [[0, 50, 0], [50, 50, -75], [125, 50, -75], [175,
50, 0], [125, 50, 75], [50, 50, 75], [0, 50, 0]];
        var tmp;
        init();
        animate();
        function init() {
            camera = new THREE.OrthographicCamera(window.innerWidth / -4,
window.innerWidth / 4, window.innerHeight / 4, window.innerHeight / -4, 1,
10000);
            camera.position.z = 800;
            camera.position.y = 200;
            camera.position.x = 200;
            scene = new THREE.Scene();
            camera.lookAt(scene.position);
            material2 = new THREE.LineDashedMaterial({
                color: 0x666666,
                dashSize: dash,
                gapSize: 0.00001
            });
            tmp = new THREE.Geometry();
            tmp.vertices.push(new THREE.Vector3(0, 0, 0));
            tmp.vertices.push(new THREE.Vector3(0, 0, 800));
```

```
            tmp.computeLineDistances();
            arr[5] = new THREE.Line(tmp, material2);
            scene.add(arr[5]);

            tmp = new THREE.Geometry();
            tmp.vertices.push(new THREE.Vector3(0, 0, 0));
            tmp.vertices.push(new THREE.Vector3(0, 400, 0));
            tmp.computeLineDistances();
            arr[4] = new THREE.Line(tmp, material2);
            scene.add(arr[4]);
            tmp = new THREE.Geometry();
            tmp.vertices.push(new THREE.Vector3(0, 0, 0));
            tmp.vertices.push(new THREE.Vector3(400, 0, 0));
            tmp.computeLineDistances();
            arr[3] = new THREE.Line(tmp, material2);
            scene.add(arr[3]);
            for (var i = 0; i < 6; i++) {
                arrayLand[i] = {
                    geometry: new THREE.Geometry(),
                    material: new THREE.LineDashedMaterial({
                        color: 0x006666,
                        dashSize: 2,
                        gapSize: 0.0001,
                        linewidth: 2
                    }),
                    line: null
                };
                arrayLand[i].geometry.vertices.push(new
THREE.Vector3(arrayPoints[i][0], arrayPoints[i][1], arrayPoints[i][2]));
                arrayLand[i].geometry.vertices.push(new
THREE.Vector3(arrayPoints[i + 1][0], arrayPoints[i + 1][1], arrayPoints[i +
1][2]));
                arrayLand[i].geometry.computeLineDistances();
                arrayLand[i].line = new
THREE.LineSegments(arrayLand[i].geometry, arrayLand[i].material);
                scene.add(arrayLand[i].line);
                arrayBottom[i] = {
                    geometry: new THREE.Geometry(),
                    material: new THREE.LineDashedMaterial({
                        color: 0x006666,
                        dashSize: 2,
                        gapSize: 0.0001,
                        linewidth: 2
                    }),
                    line: null
```

```
                };
                arrayBottom[i].geometry.vertices.push(new
THREE.Vector3(arrayPoints[i][0], arrayPoints[i][1] + height,
arrayPoints[i][2]));
                arrayBottom[i].geometry.vertices.push(new
THREE.Vector3(arrayPoints[i + 1][0], arrayPoints[i + 1][1] + height,
arrayPoints[i + 1][2]));
                arrayBottom[i].geometry.computeLineDistances();
                arrayBottom[i].line = new
THREE.LineSegments(arrayBottom[i].geometry, arrayBottom[i].material);
                scene.add(arrayBottom[i].line);
                arrayEdges[i] = {
                    geometry: new THREE.Geometry(),
                    material: new THREE.LineDashedMaterial({
                        color: 0x006666,
                        dashSize: 2,
                        gapSize: 0.0001,
                        linewidth: 2
                    }),
                    line: null
                };
                arrayEdges[i].geometry.vertices.push(new
THREE.Vector3(arrayPoints[i][0], arrayPoints[i][1], arrayPoints[i][2]));
                arrayEdges[i].geometry.vertices.push(new
THREE.Vector3(arrayPoints[i][0], arrayPoints[i][1] + height,
arrayPoints[i][2]));
                arrayEdges[i].geometry.computeLineDistances();
                arrayEdges[i].line = new
THREE.LineSegments(arrayEdges[i].geometry, arrayEdges[i].material);
                scene.add(arrayEdges[i].line);
            }
            for (var i = 0; i < 6; i++) {
                var tm = 0.2;
                arrayLand[i].line.rotation.y = tm;
                arrayEdges[i].line.rotation.y = tm;
                arrayBottom[i].line.rotation.y = tm;
            }
            renderer = new THREE.WebGLRenderer({
                antialias: true
            });
            renderer.setClearColor(0xffffff, 1);
            renderer.setPixelRatio(window.devicePixelRatio);
            renderer.setSize(window.innerWidth, window.innerHeight);
            document.body.appendChild(renderer.domElement);
            window.addEventListener('resize', onWindowResize, false);
        }
```

```
function onWindowResize() {
    camera.aspect = window.innerWidth / window.innerHeight;
    camera.updateProjectionMatrix();

    renderer.setSize(window.innerWidth, window.innerHeight);
}
function animate() {
    requestAnimationFrame(animate);
    for (var i = 0; i < 6; i++) {
        arrayLand[i].line.rotation.x = rotation;
        arrayEdges[i].line.rotation.x = rotation;
        arrayBottom[i].line.rotation.x = rotation;
    }
    rotation = (rotation >= 6.28 ? 0 : rotation + 0.005)
    if (rotation < 3.14) {
        mid = rotation < 1.57;
    } else {
        mid = rotation < 4.71;
    }
    for (var i = 0; i < 6; i++) {
        arrayLand[i].material.gapSize = 0.0001;
        arrayEdges[i].material.gapSize = 0.0001;
        arrayBottom[i].material.gapSize = 0.0001;
    }
    if (rotation < 1.05 || rotation >= 6.037) {
        arrayLand[0].material.gapSize = size;
        arrayLand[1].material.gapSize = size;
        arrayLand[2].material.gapSize = size;
        arrayEdges[1].material.gapSize = size;
        arrayEdges[2].material.gapSize = size;
    }
    if (rotation >= 1.05 && rotation < 1.38) {
        arrayLand[0].material.gapSize = size;
        arrayLand[1].material.gapSize = size;
        arrayLand[5].material.gapSize = size;
        arrayEdges[1].material.gapSize = size;
        arrayEdges[0].material.gapSize = size;
    }
    if (rotation >= 1.38 && rotation < 2.13) {
        arrayLand[0].material.gapSize = size;
        arrayLand[4].material.gapSize = size;
        arrayLand[5].material.gapSize = size;
        arrayEdges[5].material.gapSize = size;
        arrayEdges[0].material.gapSize = size;
    }
```

```
            if (rotation >= 2.13 && rotation < 2.9) {
                arrayLand[3].material.gapSize = size;
                arrayLand[4].material.gapSize = size;
                arrayLand[5].material.gapSize = size;

                arrayEdges[5].material.gapSize = size;
                arrayEdges[4].material.gapSize = size;
            }
            if (rotation >= 2.9 && rotation < 3.65) {
                arrayBottom[3].material.gapSize = size;
                arrayBottom[4].material.gapSize = size;
                arrayBottom[5].material.gapSize = size;
                arrayEdges[5].material.gapSize = size;
                arrayEdges[4].material.gapSize = size;
            }
            if (rotation >= 3.65 && rotation < 4.42) {
                arrayBottom[0].material.gapSize = size;
                arrayBottom[4].material.gapSize = size;
                arrayBottom[5].material.gapSize = size;
                arrayEdges[5].material.gapSize = size;
                arrayEdges[0].material.gapSize = size;
            }
            if (rotation >= 4.42 && rotation < 4.75) {
                arrayBottom[0].material.gapSize = size;
                arrayBottom[1].material.gapSize = size;
                arrayBottom[5].material.gapSize = size;
                arrayEdges[1].material.gapSize = size;
                arrayEdges[0].material.gapSize = size;
            }
            if (rotation >= 4.75 && rotation < 6.037) {
                arrayBottom[0].material.gapSize = size;
                arrayBottom[1].material.gapSize = size;
                arrayBottom[2].material.gapSize = size;
                arrayEdges[2].material.gapSize = size;
                arrayEdges[1].material.gapSize = size;
            }
            renderer.render(scene, camera);
        }
    </script>
</body>
</html>
```

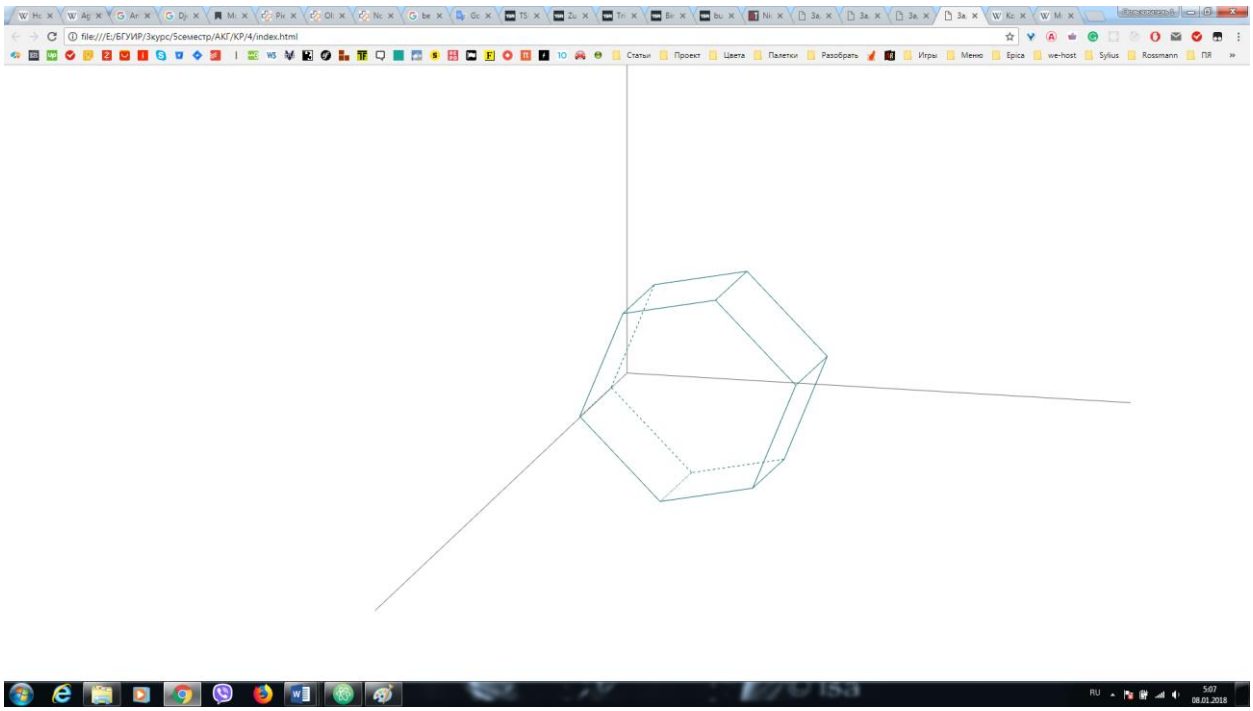


Рисунок 6 – Процесс выполнения

Выводы

Проецирование, в общем случае называется процедура представления n-мерных объектов в (n-1)-мерном пространстве. Проецирование 3-х мерных объектов, в этой связи, представляет собой процедуру их отображения на двумерной поверхности. В этом случае процесс проецирования отдельных точек исходного объекта заключается в нахождении точки пересечения линии проецирования, проходящей через эти точки, с поверхностью проецирования.

Перспективные проекции - это такие проекции, у которых все линии проецирования сходятся в одной точке - центре проецирования (или точки наблюдения). Эти проекции наиболее полно отражают специфику человеческого зрения.

Параллельные проекции - это такие проекции, у которой все линии проецирования параллельны друг другу. Параллельные проекции можно рассматривать как частный случай перспективной проекции, у которой центр проецирования (точка наблюдения) удален в бесконечность.

Перспективные проекции по форме поверхности проецирования подразделяются на плоские, цилиндрические и сферические.Параллельные проекции подразделяются на аксонометрические и косоугольные.

Аксонометрические проекции характеризуются тем, что у них линии проецирования перпендикулярны плоскости проецирования. Они делятся на:

- изометрию (на проекции углы между тремя парами осей координат трехмерной системы одинаковые);
- диметрия (на проекции углы только между двумя парами осей координат трехмерной системы одинаковые);
- триметрию (на проекции углы между всеми тремя парами осей координат трехмерной системы разные).

Косоугольные проекции характеризуются тем, что у них линии проецирования пересекаются с плоскостью проецирования не под прямым углом. Они подразделяются на свободные и кабинетные.

Свободные проекции характеризуются тем, что линии проецирования пересекаются с плоскостью проецирования под углом 450.

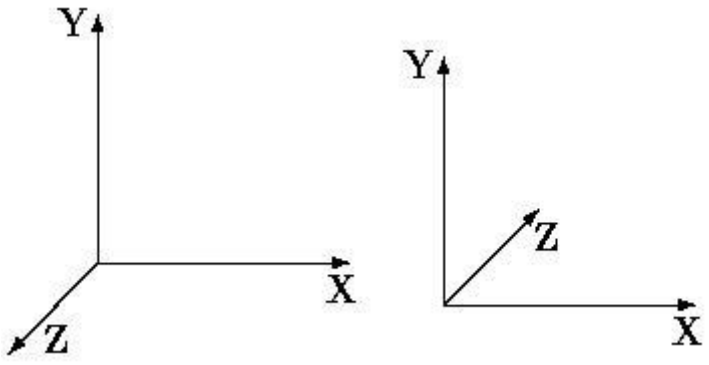
Кабинетные проекции отличаются от свободных тем, что у них масштаб по третьей оси (как равило, оси Z) уменьшен в два раза.

Задача проецирования графического объекта, в конечном счете, сводится к определению координат X,Y отдельных точек объекта на плоскости проецирования, которые изначально заданны тремя координатами в мировой системе координат.

Задача перспективного проецирования разбивается на две задачи преобразования координат:

- преобразование для перехода из мировой системы координат в видовую систему координат
- преобразования для перехода из видовой системы координат в координаты на плоскости проекции.

Мировая и видовая системы координат:



При выполнении задания была постоена заданная проекция заданной фигуры, выполнена анимация объекта. В качестве объекта для проецирования используется прямоугольная призма, в качестве основания которой используется шестиугольник.