Программирование графических приложений

Тема 13
Поверхности вращения и параметрические поверхности

Цель изучения темы. Изучение методов работы с поверхностями вращения и параметрическими поверхностями в WebGL.

13.1. Поверхности вращения

Тело вращения получается вращением кривой вокруг оси Y. Для создания поверхности вращения используется класс LatheGeometry. Конструктор класса:

```
LatheGeometry( points, segments, phiStart, phiLength );
```

Указывается множество точек points для построения вращаемой кривой, которые просто соединяются точками, а также количество сегментов segments радиальной окружности (по умолчанию равно 12). Также можно указать начальный угол phiStart и величину угла вращения phiLength.

13.2. Поверхность вращения на основе графика

Поверхность вращения проще всего построить на основе графика функции в качестве образующей (если, конечно, образующая заданной поверхности может быть описана аналитически).

Нарисуем, например, шахматную пешку. Такой объект удобнее всего описать как составной с использованием класса Object3D библиотеки Three.js. Одна из частей (сложный изгиб) будет представлена поверхностью вращения. Для построения изгиба за основу возьмем функцию

$$f = 10 + 20e^{(-x^2)}.$$

Ее график:

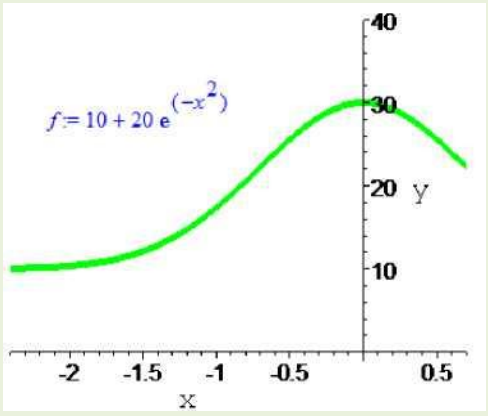

Рис. 13.1

Детали пешки будем собирать в трехмерном объекте pawn. Опишем также материал объекта:

```
pawn = newTHREE.Object3D;
var material = new THREE.MeshPhongMaterial ( { color: 0xdaa520, specular: 0x00b2fc,
        shininess: 50, blending: THREE.NormalBlending, depthTest: true } );
```

Теперь заполняем массив вершин:

```
varpoints = [];
for ( var i = - 2.4; i < 0.7; i = i + 0.1 )
  {
   points.push( new THREE.Vector3( 10 + 20*Math.exp( -i*i ), 0, 24*i ) );
```

```
}
```

Добавляем вершины в объект и поворачиваем:

```
var geometry = new THREE.LatheGeometry( points, 32 );
object = new THREE.Mesh( geometry, material );
object.position.set( 0, 26, 0 );
object.rotation.x = Math.PI/2;
pawn.add( object );
```

Результат:

Рис. 13.2

Нарисуем голову пешки в виде сферы с тем же материалом:

```
var geometry = new THREE.SphereGeometry(16, 50, 50);
var Sphere1 = new THREE.Mesh( geometry, material );
Sphere1.position.set( 0, 102, 0 );
pawn.add( Sphere1 );
```

Добавим два диска (цилиндра) для придания большего сходства с пешкой:

```
// "подголовник" пешки
var geometry = new THREE.CylinderGeometry( 22, 22, 6, 32 );
var disc1 = new THREE.Mesh( geometry, material );
disc1.position.set( 0, 86, 0 );
pawn.add( disc1 );
// основание пешки
var geometry = new THREE.CylinderGeometry( 32, 32, 8, 32 );
var disc2 = new THREE.Mesh( geometry, material );
disc2.position.set( 0, 10, 0 );
pawn.add( disc2 );
```

И, наконец, ткань на основании пешки (опять-таки в виде диска), с темным материалом:

```
var material = new THREE.MeshPhongMaterial( { color: 0x4c3c18, blending:
        THREE.NormalBlending, depthTest: true } );
var geometry = new THREE.CylinderGeometry( 32, 32, 6, 32 );
var disc3 = new THREE.Mesh( geometry, material );
disc3.position.set( 0, 3, 0 );
pawn.add( disc3 );
scene.add( pawn );
```

Полный текст программы содержится в файлах **ex13_01.html**, **ex13_01.js**.
Результат:

Рис. 13.3

13.3. Поверхность вращения с комбинированной образующей

Рассмотрим типичную ситуацию при построении поверхности вращения: образующая составлена из нескольких частей, описываемых по-разному – аналитическими функциями, массивом точек, различными аппроксимациями.

В качестве примера такого объекта построим бокал. Основная задача – подобрать точки на кривой вращения, образующей поверхность бокала. Конечно, можно подобрать эти точки массива наугад. Но лучше сформировать массив точек, подобрав графики функций. Для нашего бокала можно взять за основу объединение двух функций:

- для чаши - кривую Безье,
- для ножки - функцию $f = 5 + 30\,\mathrm{arctg}(e^{3x})$:

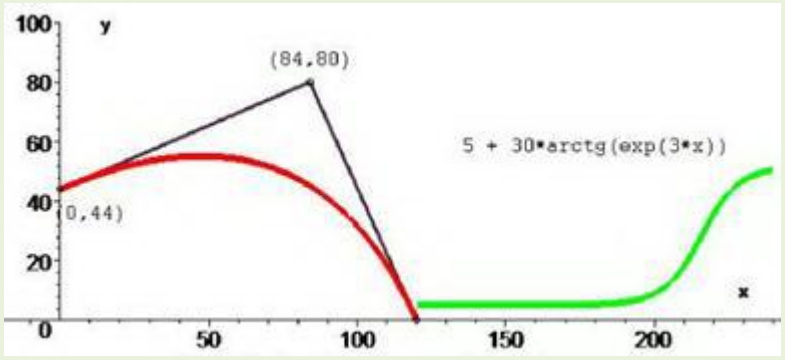

Рис. 13.4

Итак, мы должны создать массив points и записать в него точки двух кривых:

```
var points = [];
v0 = new THREE.Vector2( 0, 44 );
v1 = new THREE.Vector2( 84, 80 );
v2 = new THREE.Vector2( 120, 0 );
var BezierCurve = new THREE.QuadraticBezierCurve( v0, v1, v2 );
for ( var i = 0; i < 1; i = i + 0.05 ) {points.push( new THREE.Vector3 (
        BezierCurve.getPoint( i ).y, 0, 236 - 120*i ) ); }
for ( var i = - 4; i < 1.2; i = i + 0.1 ) {points.push( new THREE.Vector3 ( 5 +
        30*Math.atan(Math.exp(3*i)), 0, 24 - 24*i ) );}
```

Описываем геометрию и материал, создаем бокал:

```
var geometry = new THREE.LatheGeometry( points, 32 );
var material = new THREE.MeshPhongMaterial ({ color: 0xd53044, specular: 0x00b2fc,
        shininess: 50,
side: THREE.DoubleSide, blending: THREE.NormalBlending, depthTest: true } );
var glass = new THREE.Mesh( geometry, material );
glass.position.set( 0, -100, 0 );
glass.rotation.x = -Math.PI/2;
scene.add( glass );
```

Полный текст программы содержится в файлах **ex13_02.html**, **ex13_02.js**.
Результат:

Рис. 13.5

13.4. Параметрические поверхности

Для изображения поверхностей, заданных своими параметрическими уравнениями, используется класс ParametricGeometry. Синтаксис вызова:

ParametricGeometry(func, slices, stacks, useTris),

где func - функция, возвращающая координаты точки поверхности для заданных параметров **u** и **v**; slices - уровень детализации (первой) **u**-координаты; stacks - уровень детализации (второй) **v**-координаты.

Изобразим, например, эллиптический параболоид $z = x^2 + y^2$.

В параметрическом виде эту поверхность можно записать как $x=\rho\cos\theta$, $y=\rho\sin\theta$, $z=\rho^2$.

При этом параметры функции можно обозначать как угодно (а не обязательно **u** и **v**). Кроме того, нужно учесть, что оба параметра в ParametricGeometry меняются от нуля до единицы.

Итак, опишем наш параболоид. Объявим глобальные переменные:

```
var segments = 32;
var graphGeometry;
var graphMesh;
```

Создадим функцию для ParametricGeometry:

```
ParamFunction = function(rho, teta)
{
rho = 100*rho;
teta = 2*Math.PI*teta;
```

```
x = rho*Math.cos(teta);
y = rho*Math.sin(teta);
varz = rho*rho/40;
return new THREE.Vector3(y, z, x);
};
```

При этом мы растянули область изменения переменной rho от нуля до 100, и teta от нуля до 2π, сжали функцию по оси *Oz*. Кроме того, учли расположение осей и вместо вектора (x, y, z) взяли (y, z, x), чтобы поверхность была направлена привычным нам образом.

Осталось создать геометрию поверхности, материал, фигуру (сеть mesh), и добавить на сцену:

```
graphGeometry = new THREE.ParametricGeometry ( ParamFunction, segments,
        segments, false );
var material = new THREE.MeshNormalMaterial ( {color: 0x33CCFF,
        side:THREE.DoubleSide });
graphMesh = new THREE.Mesh (graphGeometry, material );
graphMesh.doubleSided = true;  // двусторонняя поверхность
graphMesh.position.set(0,0,0);
scene.add(graphMesh);
```

Добавим также оси координат. Полный текст программы содержится в файлах **ex13_03.html**, **ex13_03.js**.

Результат:

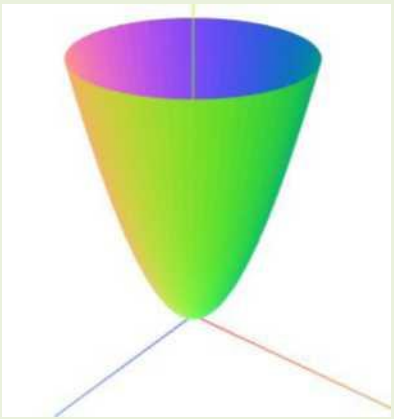

Рис. 13.6

Наложим на изображение объекта текстуру. Файл **square.png** – белое поле, ограниченное квадратом для будущей текстуры:

Рис. 13.7

Идея в том, чтобы наложить эту текстуру на каждый сегмент изображения поверхности. Для этого достаточно изменить материал; код изменится так:

```
graphGeometry = newTHREE.ParametricGeometry ( ParamFunction, segments, segments,
        false );
```

```
var Texture = new THREE.ImageUtils.loadTexture('square.png' );
Texture.wrapS = Texture.wrapT = THREE.RepeatWrapping;
Texture.repeat.set( 40, 40 );
var material = new THREE.MeshBasicMaterial( { map: Texture, vertexColors:
        THREE.VertexColors, side:THREE.DoubleSide } );
material.map.repeat.set( segments, segments );
graphMesh = new THREE.Mesh(graphGeometry, material );
graphMesh.position.set(0,0,0);
scene.add(graphMesh);
```

Полный текст программы содержится в файлах **ex13_04.html**, **ex13_04.js**.
Результат:

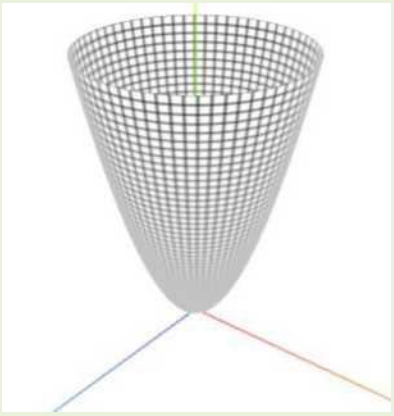

Рис. 13.8

Также можно раскрасить каждый участок в свой цвет, при этом сделать участки одинаковой высоты одного цвета. Для этого добавим такой код:

```
graphGeometry.computeBoundingBox();
varyMin = graphGeometry.boundingBox.min.y;
var yMax = graphGeometry.boundingBox.max.y;
var yRange = yMax - yMin;
var color, point, face, numberOfSides, vertexIndex;
var faceIndices = [ 'a', 'b', 'c', 'd' ];
for ( var i = 0; i < graphGeometry.vertices.length; i++ )
  {
   point = graphGeometry.vertices[ i ];
   color = new THREE.Color( 0x0000ff );
   color.setHSL( 0.7 * (yMax - point.y) / yRange, 1,0.5 );
   graphGeometry.colors[i] = color;
  }
for ( var i = 0; i < graphGeometry.faces.length; i++ )
  {
   face = graphGeometry.faces[ i ];
   numberOfSides = ( face instanceof THREE.Face3 ) ? 3 : 4;
   for( var j = 0; j < numberOfSides; j++ )
     {
      vertexIndex = face[ faceIndices[ j ] ];
      face.vertexColors[ j ] = graphGeometry.colors[ vertexIndex ];
     }
  }
```

Полный текст программы содержится в файлах **ex13_05.html**, **ex13_05.js**.

Результат:

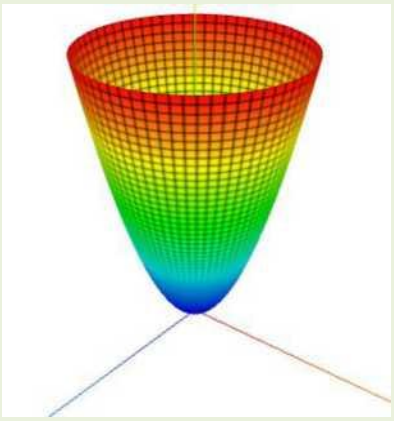

Рис. 13.9

Контрольные вопросы

1. Поверхности вращения
2. Параметрические поверхности