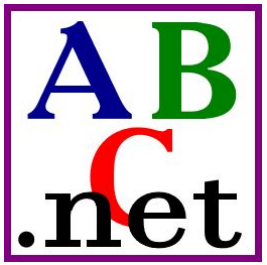
Анимация в системе программирования PascalABC.NET

Михалкович С.С.
Южный федеральный университет,
факультет математики,
механики и компьютерных
наук
miks@math.sfedu.ru

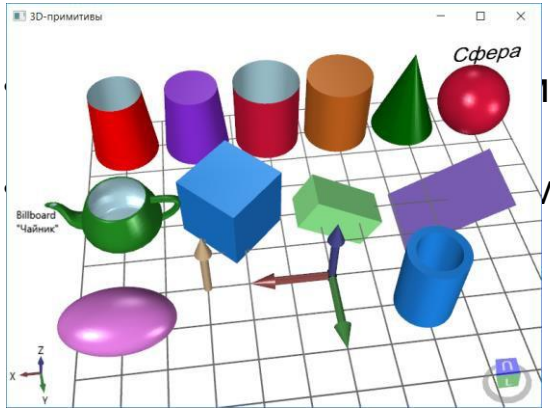


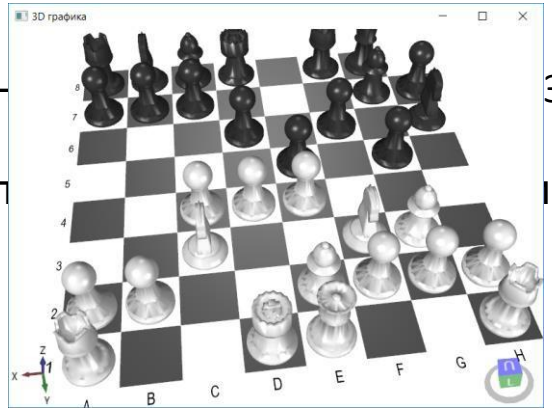
Доклад на учебно-методической конференции
«Использование системы программирования PascalABC.NET и электронного задачника Programming Taskbook в обучении программированию»

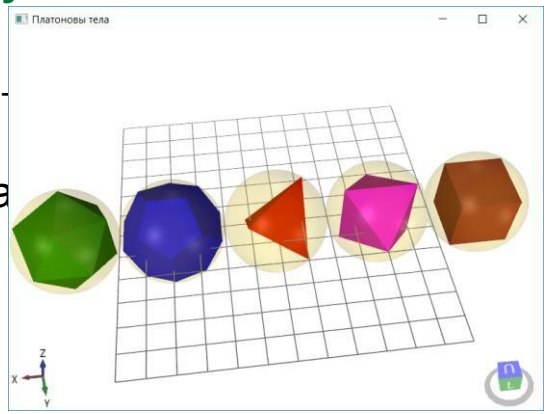
Модуль Graph3D – обзор

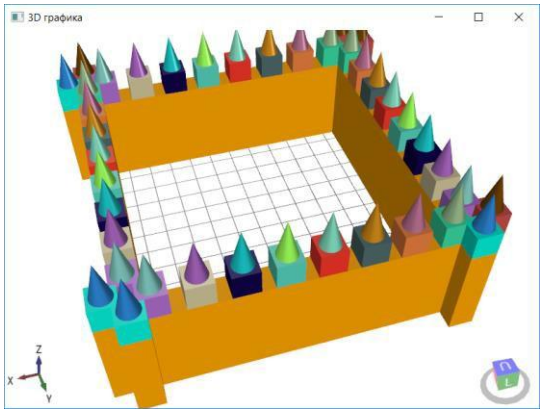
- 3D-программирование – важнейшая сфера разработки программного обеспечения.
- **Graph3D** – простой в освоении и использовании модуль 3D-графики для PascalABC.NET,

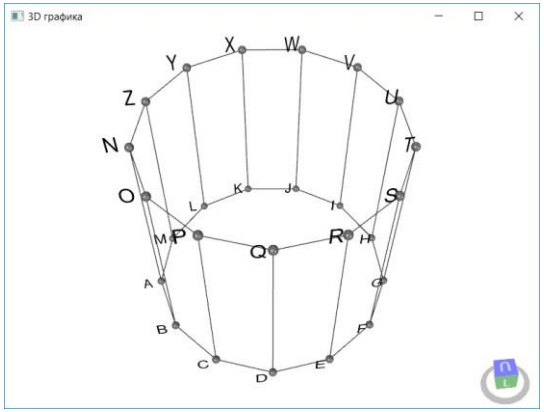
  написанный на основе библиотеки HelixToolkit WPF (**требуется Windows 7 и**

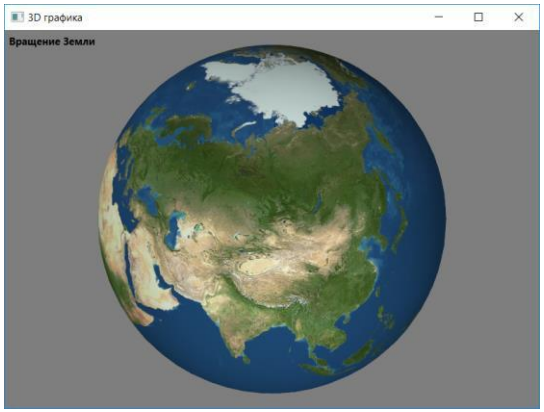












3D-примитивы и их конструирующие функции

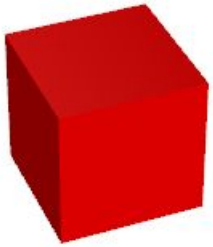
Sphere(x, y, z, Radius, Color)
Ellipsoid(x, y, z, RadiusX, RadiusY, RadiusZ, Color)
Cube(x, y, z, SideLength, Color)
Box(x, y, z, SideX, SideY, SideZ, Color)
Cylinder(x, y, z, Height, Radius, Color)
Tube(x, y, z, Height, Radius, InnerRadius, Color)
Cone(x, y, z, Height, Radius, Color)
Prism(x, y, z, Sides, Height, Radius, Color)
Pyramid(x, y, z, Sides, Height, Radius, Color)
Torus(x, y, z, Diameter, TubeDiameter, Color)
Teapot(x, y, z, Color)
Arrow(x, y, z, vx, vy, vz, Color)
Icosahedron(x, y, z, Radius, Color)
Dodecahedron(x, y, z, Radius, Color)
Tetrahedron(x, y, z, Radius, Color)
Octahedron(x, y, z, Radius, Color)
FileModel3D(x, y, z, FileName, Color)
Lego(x, y, z, col, r, h, Color)
Rectangle3D(p: Point3D, Length, Width: real, Normal, LengthDirection: Vector3D, Color)
Text3D(x, y, z, Text, Height, Color)
Segment3D(p1, p2: Point3D, Thickness, Color)
Segments3D(points: sequence of Point3D, Thickness, Color)
Polyline3D(points: sequence of Point3D, Thickness, Color)
ClosedPolyline3D(points: sequence of Point3D, Thickness, Color)

Sphere, Ellipsoid, Cube, Box, Cylinder, Tube

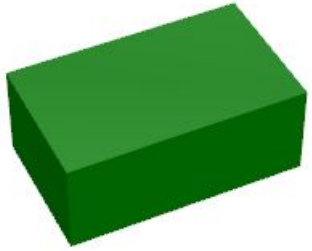
Sphere(x, y, z, Radius, Color)

Ellipsoid(x, y, z, RadiusX, RadiusY, RadiusZ, Color)

Cube(x, y, z, SideLength, Color)



Box(x, y, z, SideX, SideY, SideZ, Color)



Cylinder(x, y, z, Height, Radius, Color)

Tube(x, y, z, Height, Radius, InnerR, Color)

Cone, Torus, Pyramid, Prism, Teapot, Arrow

Cone(x, y, z, Height, Radius, Color)

Torus(x, y, z, Diameter, TubeDiam, Color)

Pyramid(x, y, z, Sides, Height, Radius, Color)

Prism(x, y, z, Sides, Height, Radius, Color)

Teapot(x, y, z, Color)

Arrow(x, y, z, vx, vy, vz, Color)

Платоновы тела, FileModel3D, Lego

Icosahedron(x, y, z, Radius, Color)

Dodecahedron(x, y, z, Radius, Color)

Tetrahedron(x, y, z, Radius, Color)

Octahedron(x, y, z, Radius, Color)

FileModel3D(x, y, z, FileName, Color)

Lego(x, y, z, Rows, Columns, Height, Color)

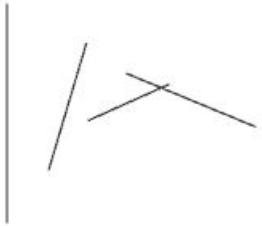
Rectangle3D, Text3D, отрезок, набор отрезков, кривая, замкнутая кривая

Rectangle3D(p, Length, Width, Normal, LD, Color)

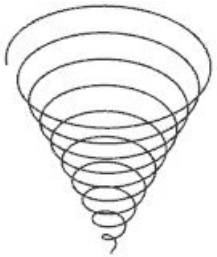
Text3D(x, y, z, Text, Height, Color)

Segment3D(p1, p2, Thickness, Color)

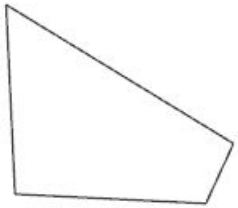
Segments3D(points, Thickness, Color)



Polyline3D(points, Thickness, Color)



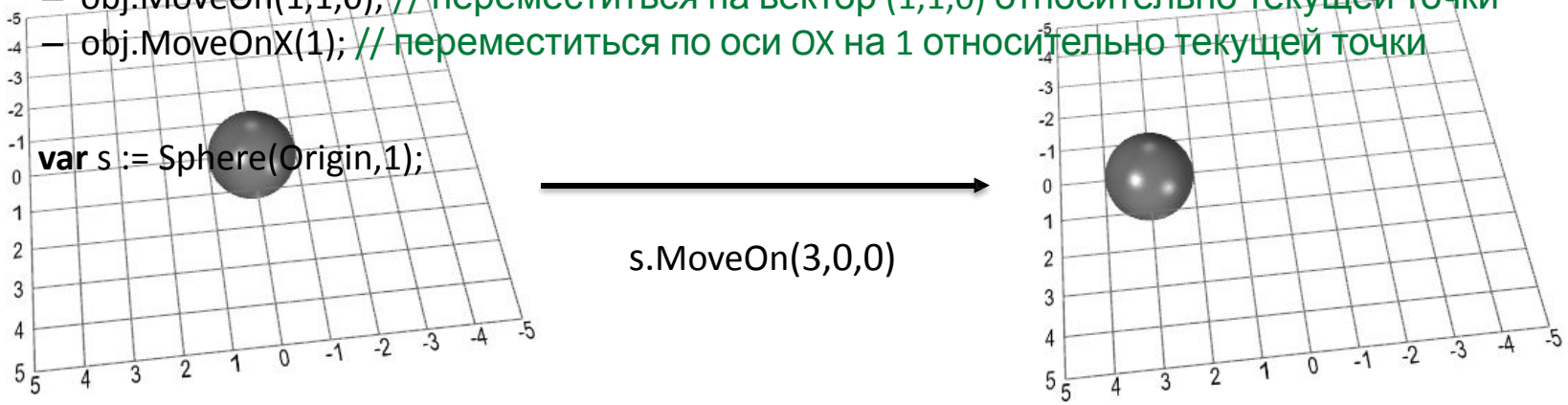
ClosedPolyline3D(points, Thickness, Color)



К оглавлению

Позиция и перемещение 3D-объекта

- Каждый 3D-объект имеет свойство Position типа Point3D – текущие координаты объекта
- Для большинства 3D-объектов Position задаёт центр объекта, для некоторых – опорную точку (например, для конуса и пирамиды, цилиндра, чайника и трубы точка Position задаёт центр основания, для 3D-стрелки – координату начала стрелки, для остальных – некоторую опорную точку).
- Для 3D-объекта можно также обращаться к свойствам X, Y, Z. Очевидно, Position = P3D(X,Y,Z)
- Измененять координаты 3D-объекта obj можно следующими способами:
  - obj.X := 2; obj.Y := obj.Y + 1; obj.Z += 1;
  - obj.Position := P3D(2,3,4); // переместиться к точке (2,3,4)
  - obj.MoveTo(2,3,4); // переместиться к точке (2,3,4)
  - obj.MoveOn(1,1,0); // переместиться на вектор (1,1,0) относительно текущей точки
  - obj.MoveOnX(1); // переместиться по оси OX на 1 относительно текущей точки



К оглавлению

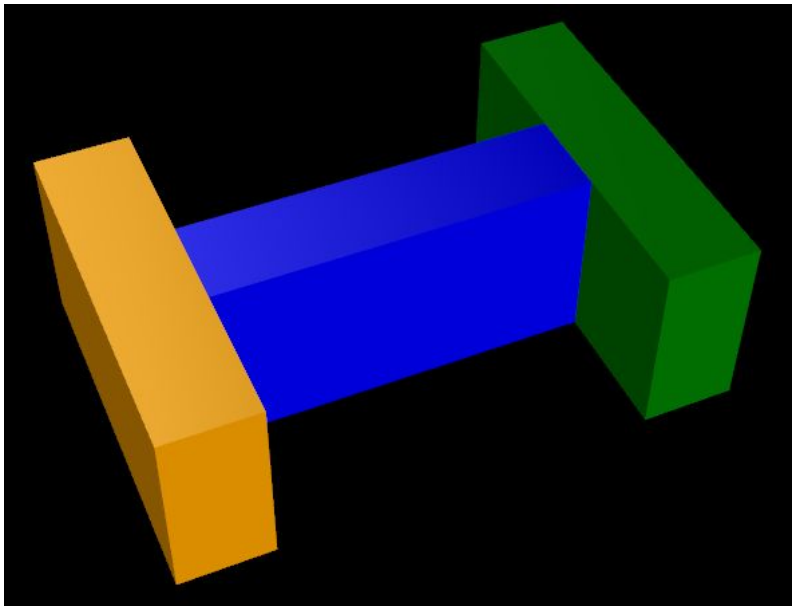
Клонирование

- Любой 3D-объект можно клонировать.
- Клонированный объект будет иметь те же атрибуты, что и исходный.

Код

```
uses Graph3D;
begin
  View3D.BackgroundColor := Colors.Black; View3D.ShowGridLines := False;
  var b := Box(0,0,0,4,1,2,Colors.Blue);
// Клонируем исходный объект
  var b1 := b.Clone;
// Окрашиваем его в другой цвет, поворачиваем и сдвигаем
  b1.Color := Colors.Green;
  b1.Rotate(OrtZ,90).MoveOnX(-2.5);
// Вновь клонируем исходный объект
  var b2 := b.Clone;
// Окрашиваем его в другой цвет, поворачиваем и сдвигаем
// в другую сторону
  b2.Color := Colors.Orange;
  b2.Rotate(OrtZ,-90).MoveOnX(2.5);
```

Скриншот



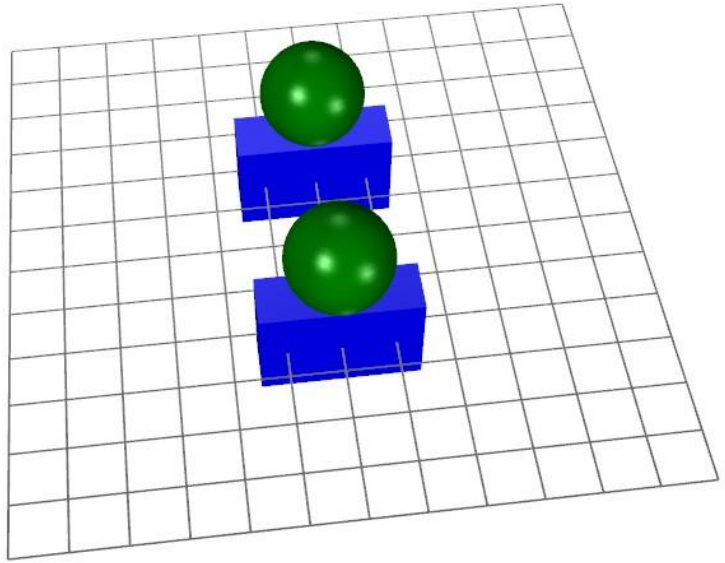
К оглавлению

Группировка

- Несколько 3D-объектов можно группировать. После этого их можно перемещать, вращать, анимировать и клонировать как единое целое.
- Для группировки используется функция Group. Возможна многократная вложенная группировка: группы можно объединять в другие группы.
- В программе ниже мы превращаем паралеллепипед и сферу на нём в группу, после
  чего сдвигаем её по оси OY на -2.
- Затем мы клонируем группу и сдвигаем её по оси OY на 4.

Код Скриншот

```
uses Graph3D;

begin
  var b := Box(0,0,0,3,1,2,Colors.Blue);
  var s := Sphere(0,0,2,1,Colors.Green);

  var g := Group(b,s);
  g.MoveOnY(-2);
  var g1 :=
  g.Clone;
  g.MoveOnY(4);
end.
```



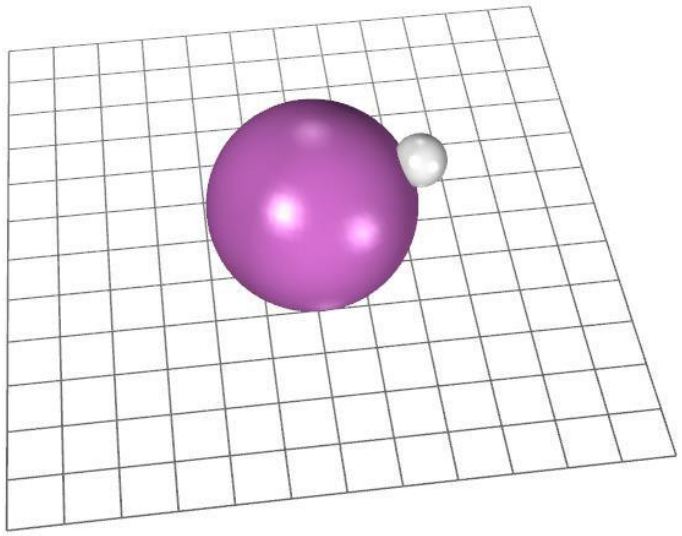
К оглавлению

Дочерние элементы

- Разновидностью группировки являются дочерние элементы. В этом случае в группе всегда есть главный 3D-объект, а к нему присоединяются дочерние элементы.
- Для добавления дочерних элементов используется метод AddChild.
- Важной особенностью является то, что при добавлении дочернего объекта используется локальная система координат родительского объекта.
- В коде ниже дочерний объект добавляется в точку (0,2,1) относительно центра родительского объекта, после чего родительский объект поворачивается как группа

Код Скриншот

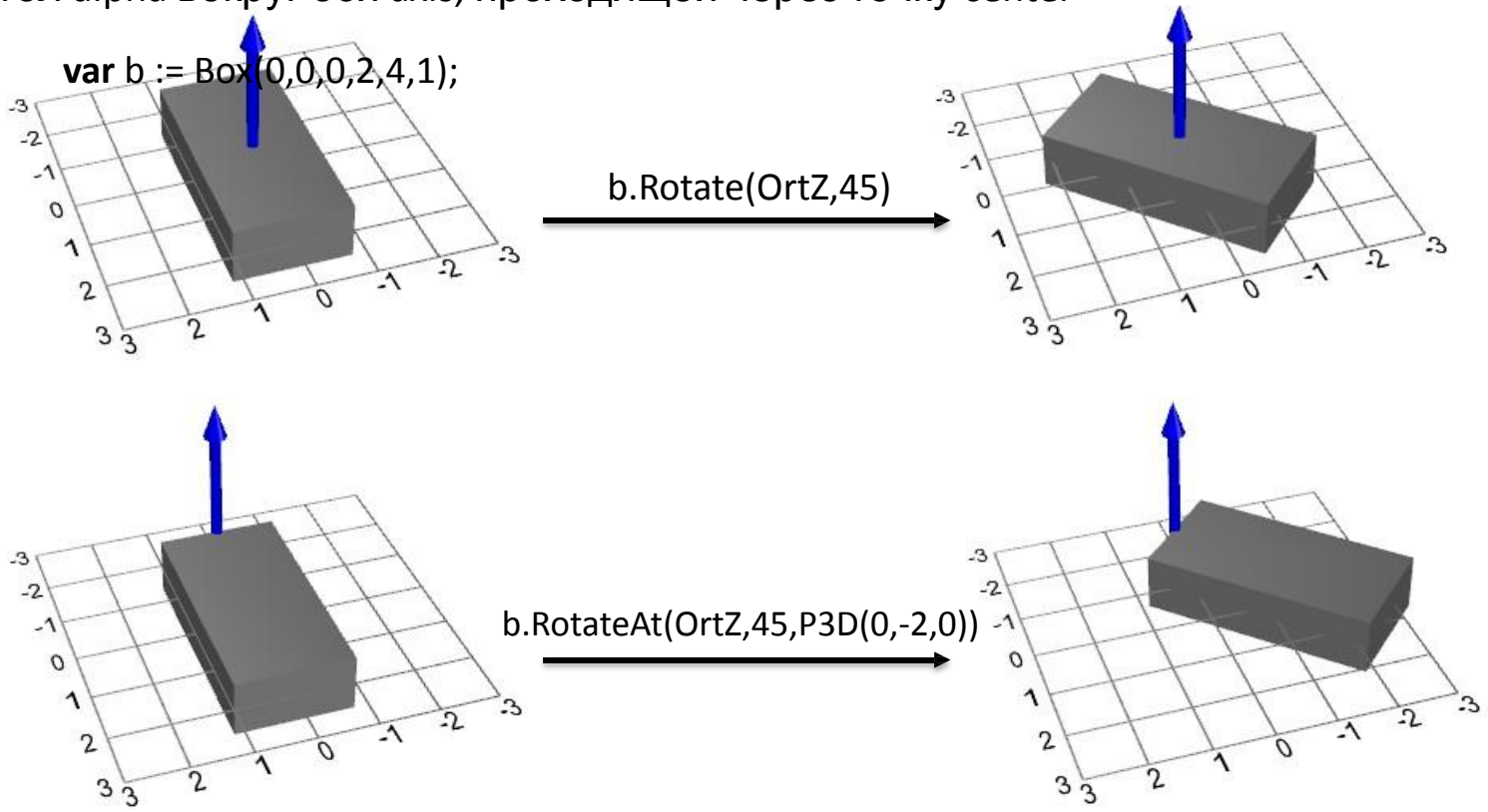
```
uses Graph3D;

begin
  var s := Sphere(0,1,0,2,Colors.Orchid);
  s.AddChild(Sphere(0,2,1,0.5,Colors.White))
  ; s.Rotate(OrtZ,90);
end.
```



К оглавлению

Вращение 3D-объекта

- Каждый 3D-объект имеет метод Rotate(axis, angle), вращающий объект на угол alpha вокруг оси axis, проходящей через центр объекта
- Используя метод RotateAt(axis, angle, center), можно осуществлять вращение объекта на
  угол alpha вокруг оси axis, проходящей через точку center



К оглавлению

Обычная анимация перемещения

- Будем перемещать объект в цикле на малое расстояние и делать паузу после каждого перемещения
- Недостатки: движение скачками; надо вычислять, на сколько двигаться
  и какая пауза

Код

```
uses Graph3D;
begin
  var s :=
  Sphere(Origin,1);

  loop 100 do
  begin
    s.MoveOn(-0.05,0,0);
    Sleep(20);
  end;
end.
```

К оглавлению

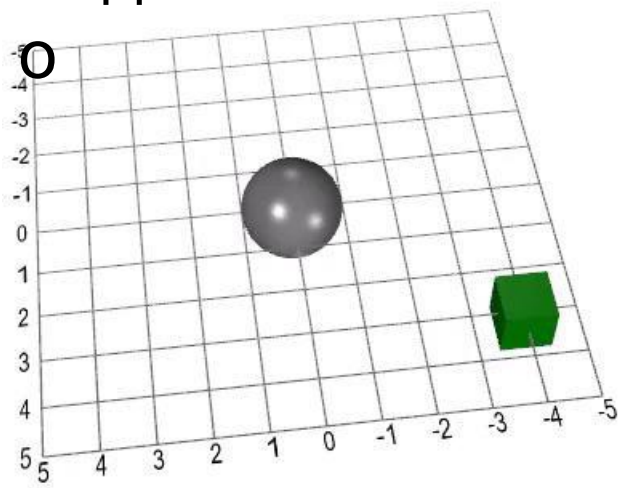
Алгоритмическая анимация перемещения

- Для каждого 3D-объекта имеются стандартные алгоритмические анимации положения, масштабирования и поворота. Такую анимацию можно сконструировать, указав параметры, длительность, после чего запустить. После запуска анимации можно **параллельно выполнять другие действия**

Код

```
uses Graph3D;
begin
  var s := Sphere(Origin,1);
// Формируем анимацию к точке (-5,0,0) за 2 секунды
  var a := s.AnimMoveOn(-5,0,0,2);
// Запускаем анимацию
  a.Begin;
// Создаём куб в точке (-4,3,0) и запускаем его анимацию
// по оси OX на 8 единиц за 3 секунды
  Cube(P3D(-4,3,0),1).AnimMoveOnX(8,3).Begin;
end.
```

Видео



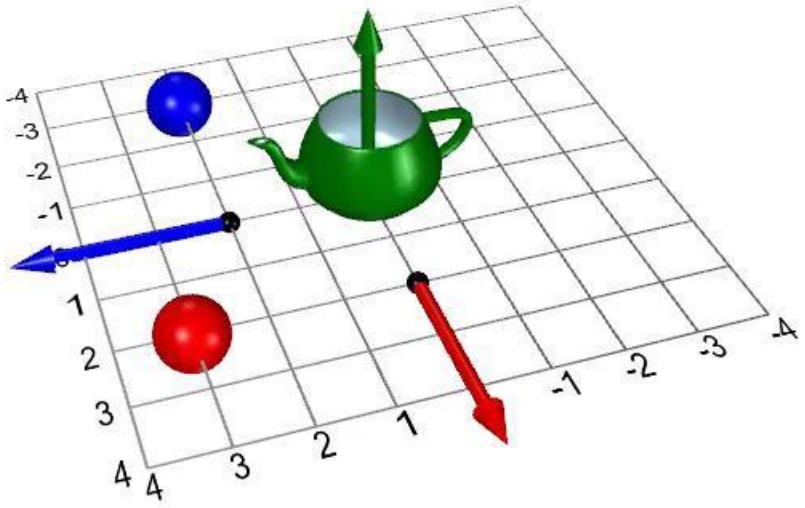
К оглавлению

Алгоритмическая анимация вращения

- Алгоритмическая анимация вращения выполняется с помощью методов AnimRotate(axis, angle, seconds) и AnimRotateAt(axis, angle, center, seconds). Здесь axis – вектор оси, вокруг которой осуществляется анимация вращения, center – смещение центра вращения относительно центра фигуры

Код Видео

```
uses Graph3D;
begin
  var t :=
  Teapot(P3D(0,0,0.9),Colors.Green);  var s
  := Sphere(2,-3,0,0.5,Colors.Blue);  var s1
  := Sphere(3,2,0,0.5,Colors.Red);
// Запускаем анимацию вращения чайника вокруг оси OZ
// на 360 градусов за 4 секунды
  t.AnimRotate(OrtZ,360,4).Begin;
// Запускаем анимацию вращения сферы вокруг оси,
// параллельной оси OX и проходящей через точку (0,3,0)
// на 360 градусов за 4 секунды
  s.AnimRotateAt(OrtX,-360,P3D(0,3,0),4).Begin;
// Запускаем анимацию вращения сферы вокруг оси OY
// на 360 градусов за 4 секунды
  s1.AnimRotateAt(OrtY,360,P3D(-3,0,0),4).Begin
end.
```



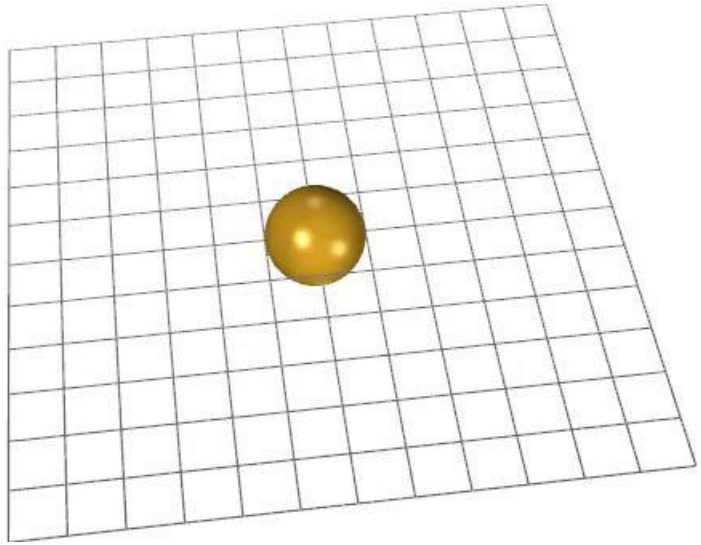
К оглавлению

Комбинирование анимаций (+)

- Две анимации a1 и a2 одного и того же объекта или разных объектов могут быть выполнены параллельно или последовательно. Для этого используются следующие операции:
  - a1 + a2 – последовательное выполнение анимаций
  - a1 * a2 – параллельное выполнение анимаций
- В примере ниже приведена последовательная анимация движения сферы

Код

```
uses Graph3D;
begin
  var s := Sphere(0,0,0,1,Colors.Goldenrod);
// В анимации a храним последовательно выполняющиеся
// анимации перемещения и увеличения сферы
  var a := s.AnimMoveOnX(3) + s.AnimMoveOnY(2) +
           s.AnimMoveOnZ(4);
  a.Begin;
end.
```



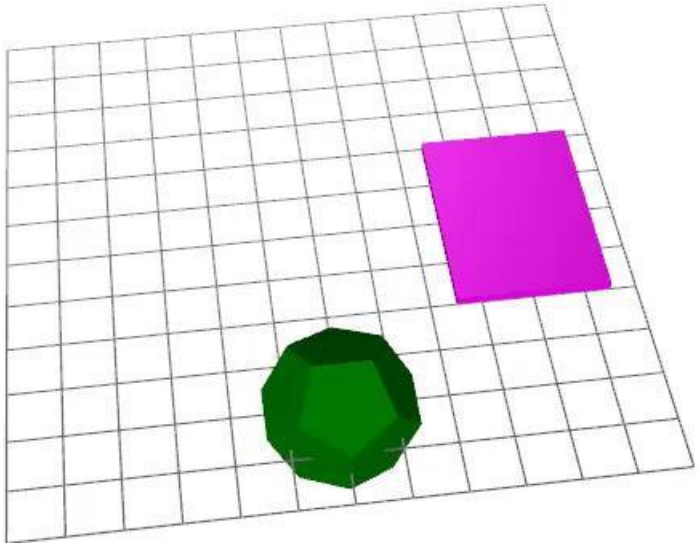
К оглавлению

Комбинирование анимаций (*)

- В примере ниже происходит параллельная анимация, при которой Box перемещается вдоль оси OX и одновременно с этим вращается вокруг оси OY
- Параллельно осуществляется вращение другого объекта – додекаэдра – вокруг оси OZ

Код Видео

```
uses Graph3D;

begin
  var r := Box(-4,0,0.1,3,4,0.2,Colors.Fuchsia);
  var d := Dodecahedron(0,4,0,1.5,Colors.Green);
// В анимации a храним одновременно выполняющиеся
// анимацию перемещения и анимацию вращения вокруг оси OY
// Параллельно выполняется анимация вращения додекаэдра
// вокруг оси OZ
  var a1 := r.AnimMoveOnX(8,2) *
            r.AnimRotate(OrtY,360*2,2)
  ; var a2 :=
  d.AnimRotate(OrtZ,360,2);  var a :=
  a1 * a2;
  a.Begin;
// Можно запустить анимации a1 и a2 отдельно:
// a1.Begin; a2.Begin;
// Это будет равноценно запуску анимации a, равной a1 * a2
```



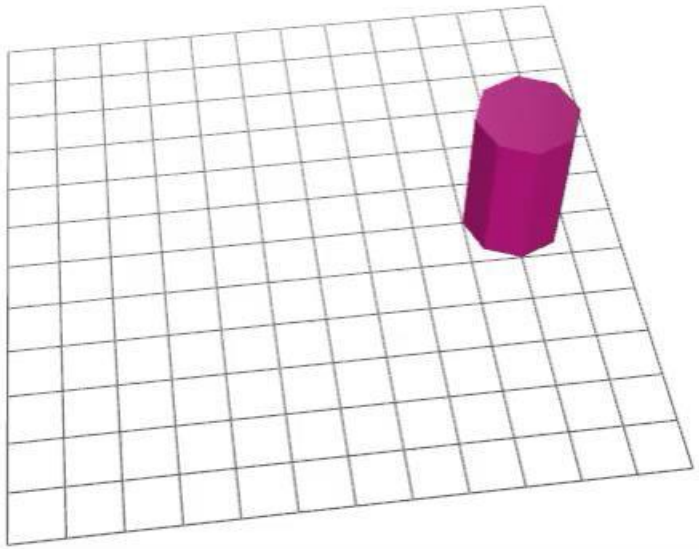
К оглавлению

Комбинирование анимаций (+ и *)

- В примере ниже выполняется комбинированная анимация, заключающаяся в последовательном выполнении анимаций перемещения по оси OX, затем по оси OY (каждая в течение 2 секунд). При этом призма вращается вокруг оси OZ на протяжении всего времени перемещения (4 секунды), что достигается использование параллельной анимации

Код Видео

```
uses Graph3D;

begin
  var p := Prism(-4,0,0,8,3,1);
// В анимации a храним одновременно выполняющиеся
// анимацию перемещения и анимацию вращения вокруг оси OY
// Параллельно выполняется анимация вращения додекаэдра
// вокруг оси OZ
  var a :=(p.AnimMoveOnX(8,2) + p.AnimMoveOnY(4,2)) *
           p.AnimRotate(OrtZ,360*3,4);
  a.Begin;
end.
```



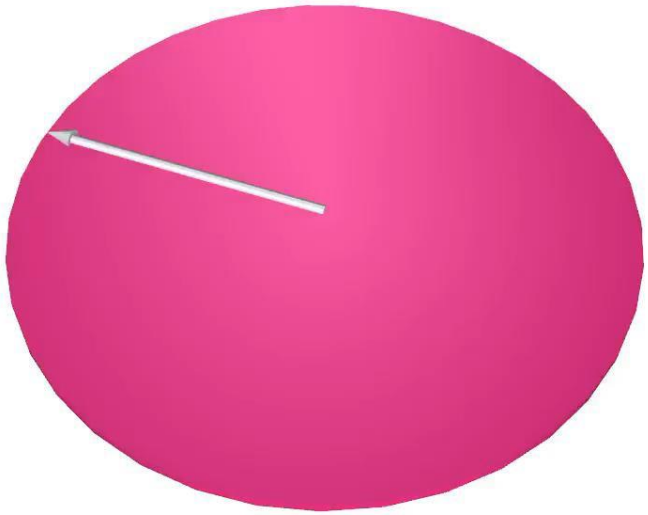
К оглавлению

Модификатор анимаций Forever

- Модификатор a.Forever зацикливает анимацию, выполняя её бесконечно

Код

```
uses Graph3D;
begin
  var disc := Cylinder(0,0,0,0.2,6);
  disc.AddChild(Arrow(0,0,0.2,0,-6,0));
  disc.AnimRotate(OrtZ,-360,6.sec).Forever.Begin;
end.
```



К оглавлению

Анимация на основе кадра

- Если необходимо перемещать объекты интерактивно (с учетом взаимодействий), то алгоритмическая анимация не подходит
- Необходимо использовать анимацию на основе кадра

```
uses Graph3D;

begin
  var c := Sphere(0,0,1,1,Colors.Green);
  c.Velocity := 5;

  OnDrawFrame := dt -> begin
    c.MoveTime(dt);
  end;

  OnKeyDown := k ->
  begin
    case k of
      Key.w,Key.Up:    c.Direction := V3D(0,-1,0);
      Key.s,Key.Down:  c.Direction := V3D(0,1,0);
      Key.a,Key.Left:  c.Direction := V3D(1,0,0);
      Key.d,Key.Right: c.Direction := V3D(-1,0,0);
    end;
  end;

  OnKeyUp := k ->
  begin
    c.Direction := V3D(0,0,0);
  end;
end.
```

В каждом кадре мы перемещаем объект с учетом его скорости c.Velosity, направления c.Direction и времени dt