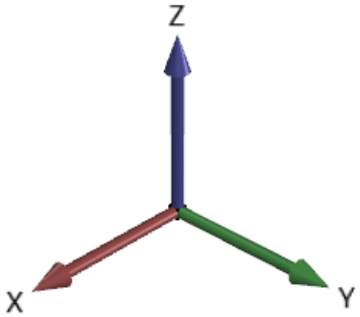
32 слайда



PascalABC.NET 3D-графика. Модуль Graph3D

- 3D-примитивы
- Цвета и материалы
- Перемещение, масштабирование, вращение
- Алгоритмические анимации и их комбинирование
- Группировка и клонирование
- Освещение и камера
- Обработка событий мыши и клавиатуры

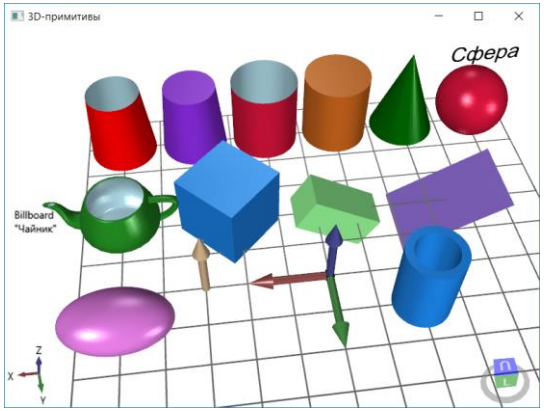
Обновлено 17.06.18

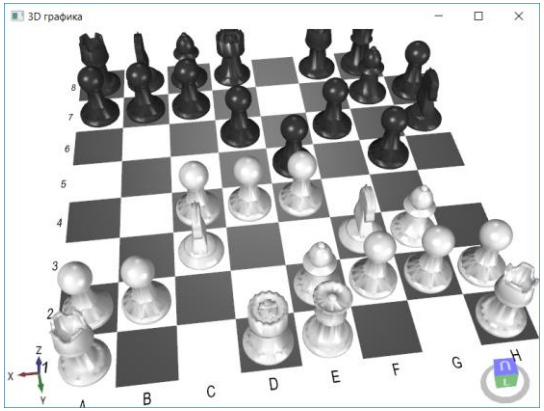
Оглавление

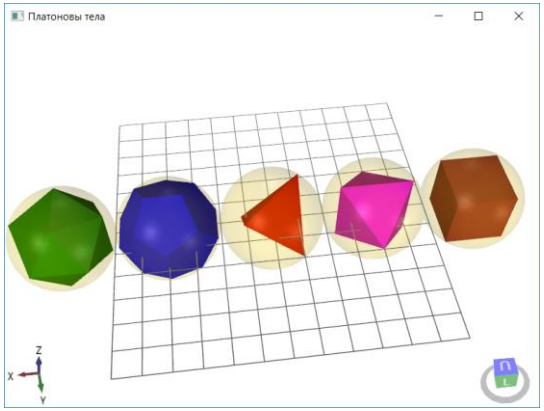
- Слайд 3. Модуль Graph3D – обзор
- Слайд 4. Основные типы Graph3D
- Слайды 5-9. 3D-примитивы и их конструирующие функции
- Слайд 10. Позиция и перемещение 3D-объекта
- Слайд 11. Масштабирование 3D-объекта
- Слайд 12. Вращение 3D-объекта
- Слайд 13. Анимация перемещения
- Слайд 14. Анимация масштабирования
- Слайд 15. Анимация вращения
- Слайд 16. Анимация движения по пути
- Слайды 17-19. Комбинирование анимаций
- Слайд 20. Модификаторы анимаций
- Слайд 21. Клонирование
- Слайд 22. Группировка
- Слайд 23. Дочерние элементы
- Слайд 24. Материалы
- Слайд 25-26. Освещение
- Слайд 27. Обработка событий клавиатуры
- Слайд 28-30. Обработка событий мыши
- Слайд 31. Камера и её свойства
- Слайд 32. Самое важное

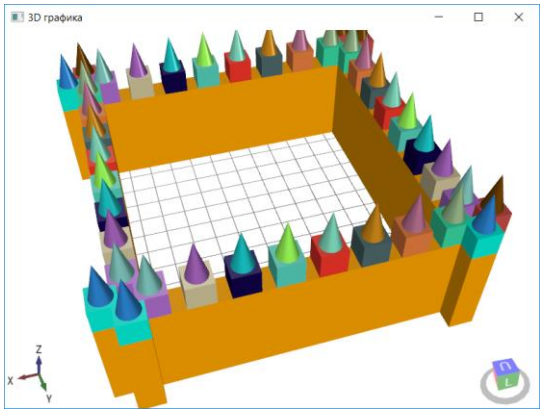
Модуль Graph3D – обзор

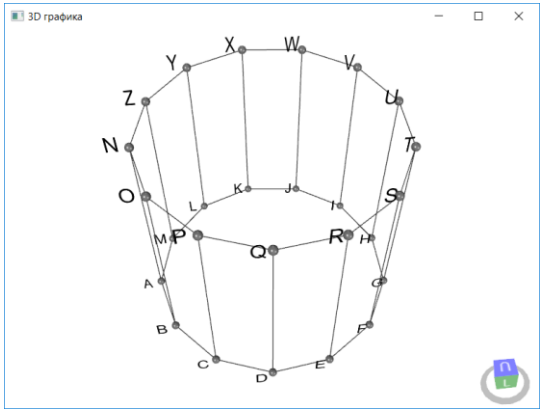
- 3D-программирование – важнейшая сфера разработки программного обеспечения.
- **Graph3D** – простой в освоении и использовании модуль 3D-графики для PascalABC.NET, написанный на основе библиотеки HelixToolkit WPF (**требуется Windows 7 и выше!**).
- Программы для Graph3D манипулируют примитивными 3D-объектами и их свойствами.
- В модуле Graph3D предусмотрены алгоритмические анимации и их комбинаторы.

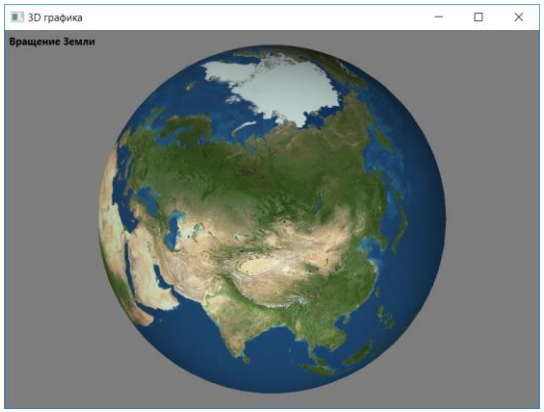












Основные типы Graph3D

- Точка задаётся типом Point3D с полями x,y,z. Для создания объекта типа Point3D используется функция P3D(x,y,z).
- Вектор задаётся типом Vector3D с полями x,y,z. Для создания объекта типа Vector3D используется функция V3D(x,y,z).
- Начало координат задаётся константой **Origin**, имеющей значение P3D(0,0,0)
- Орты осей задаются константами **OrtX** = V3D(1,0,0), **OrtY** = V3D(0,1,0) и **OrtZ** = V3D(0,0,1).
- Цвет имеет тип GColor. Имеется ряд предопределённых цветовых констант в типе **Colors**: Colors.Red, Colors.Blue, Colors.White и т.д.
- Для задания цвета используются также функции **RGB(r,g,b)** и **ARGB(alpha,r,g,b)**, задающие цвет по красной, зелёной и синей цветовым компонентам r,g,b, и по компоненте прозрачности alpha. Имеется также функция **RandomColor**, возвращающая случайный цвет, и функция **GrayColor(b)**, возвращающая серый цвет с интенсивностью b.

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)
Polygon3D(points: sequence of Point3D, Thickness, Color)

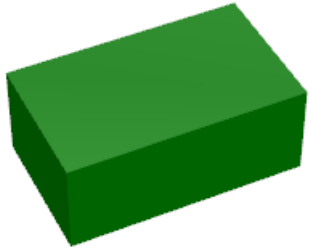
Sphere, Ellipsoid, Cube, Box, Cylinder, Tube

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)

К оглавлению

Платоновы тела, FileModel3D, Lego

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

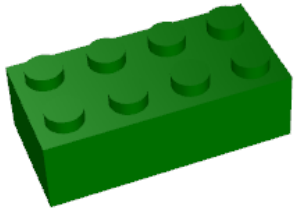
Dodecahedron(x, y, z, Radius, Color)

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)



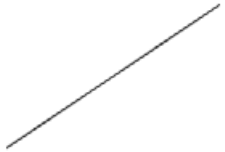
К оглавлению

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

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

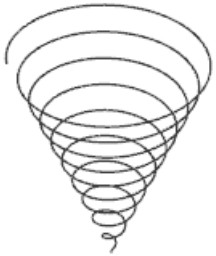
Text3D(x, y, z, Text, Height, Color)

Segment3D(p1, p2, Thickness, Color)

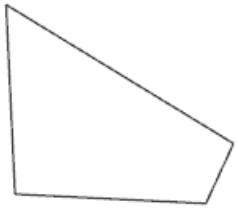


Segments3D(points, Thickness, Color)

Polyline3D(points, Thickness, Color)

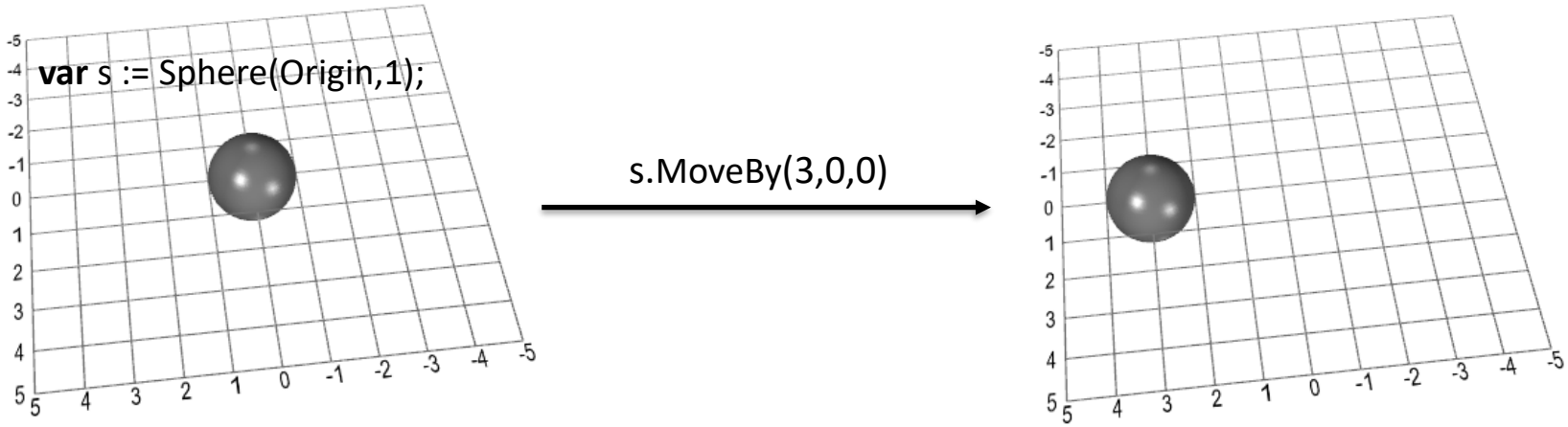


ClosedPolyline3D(points, Thickness, Color)



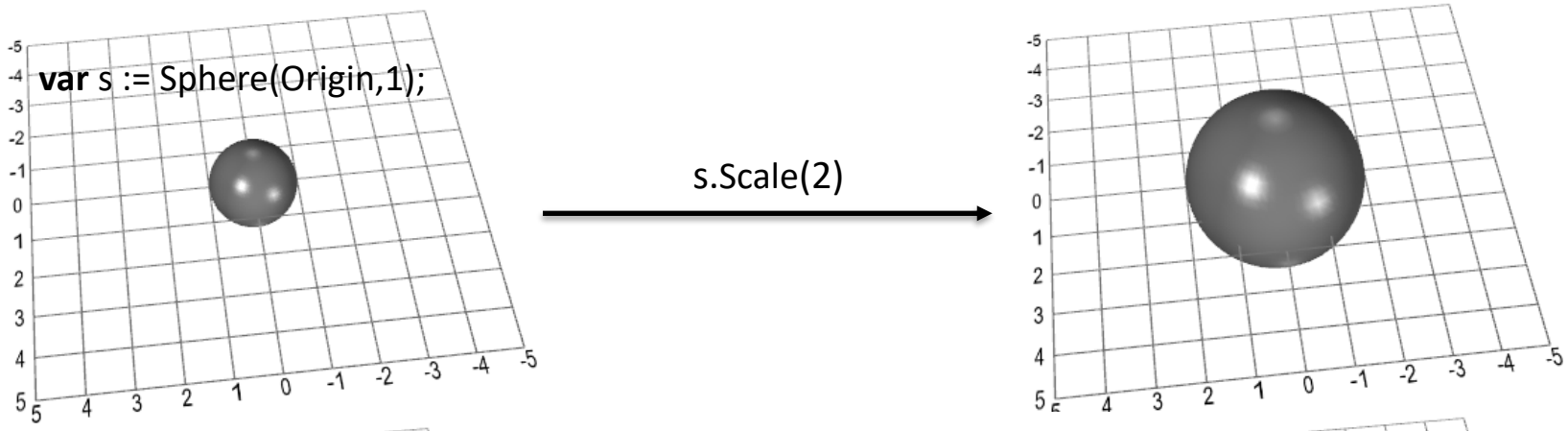
Позиция и перемещение 3D-объекта

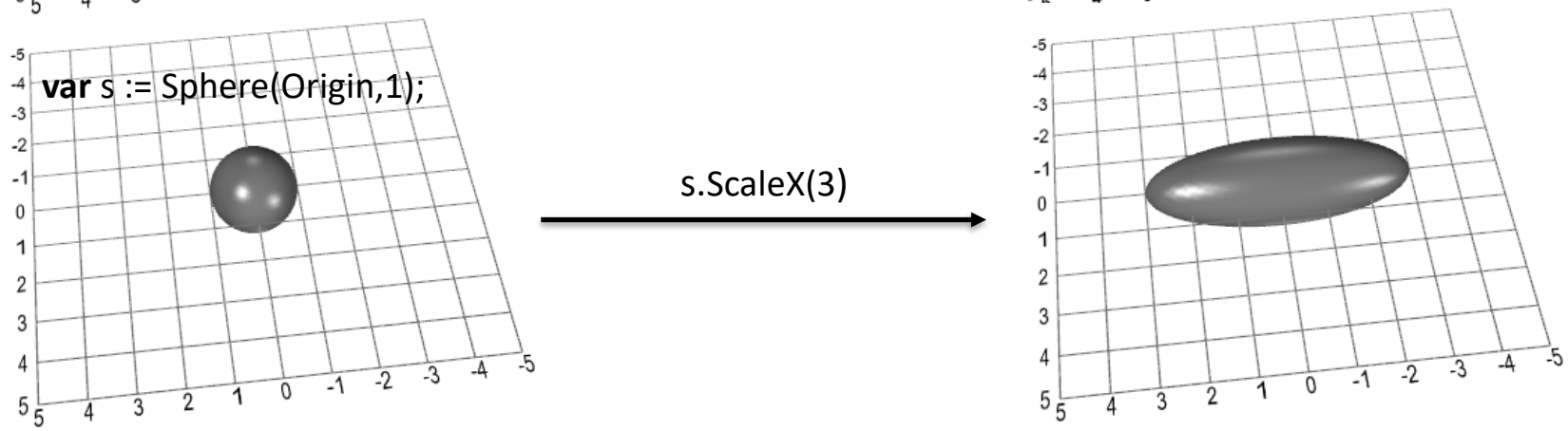
- Каждый 3D-объект имеет свойство Position типа Point3D – текущие координаты объекта
- Для большинства 3D-объектов Position задаёт центр объекта, для некоторых – опорную точку (например, для конуса и пирамиды, цилиндра, чайника и трубы точка Position задаёт центр основания, для 3D-стрелки – координату 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.MoveBy(1,1,0); // переместиться на вектор (1,1,0) относительно текущей точки
  - obj.MoveByX(1); // переместиться по оси OX на 1 относительно текущей точки



Масштабирование 3D-объекта

- Каждый 3D-объект имеет метод Scale(f), масштабирующий объект в f раз. Если f>1, то объект увеличивается, если 0<f<1, то объект уменьшается
- Можно осуществлять масштабирование только по одной из осей. Для этого используются методы ScaleX(f), ScaleY(f), ScaleZ(f)

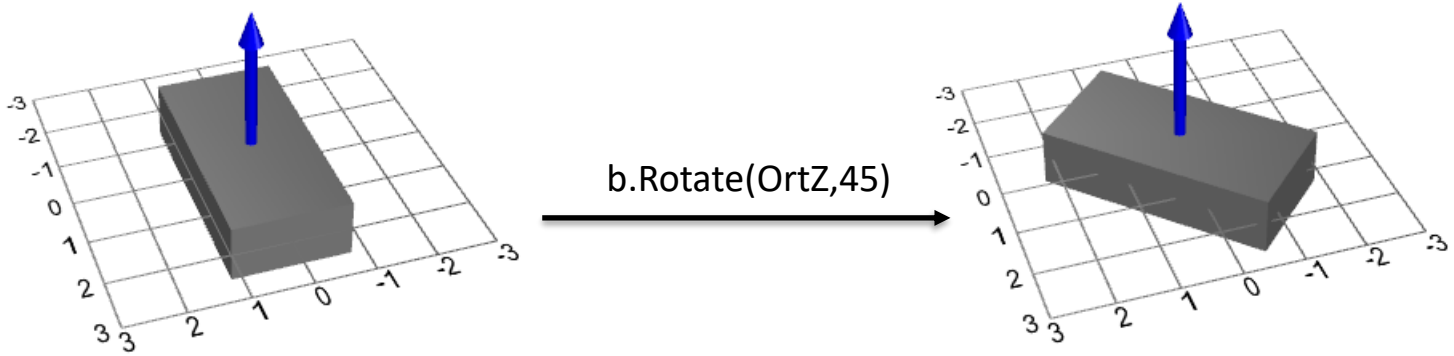


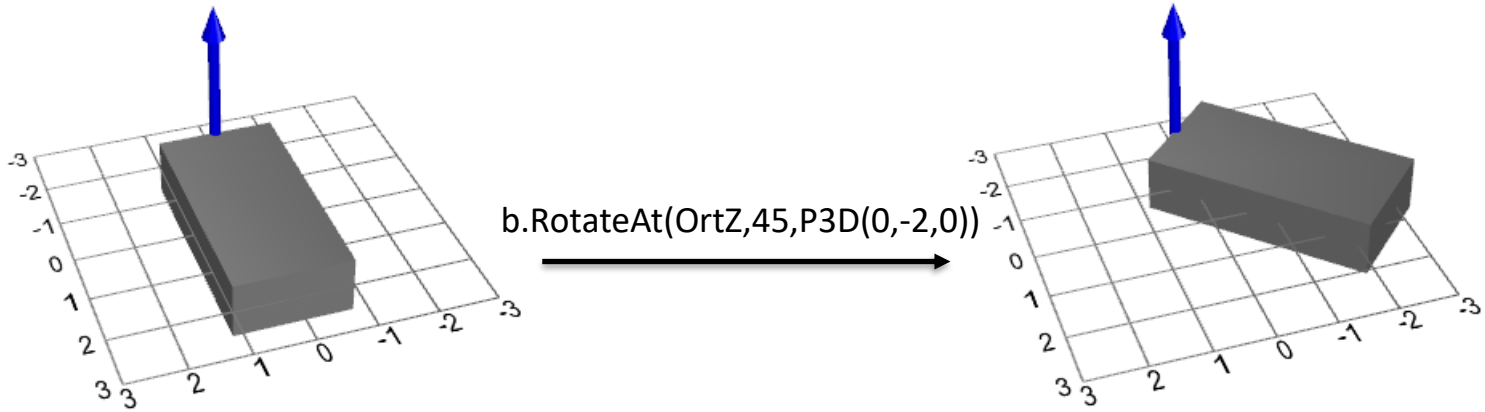


Вращение 3D-объекта

- 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

```
var b := Box(0,0,0,2,4,1);
```





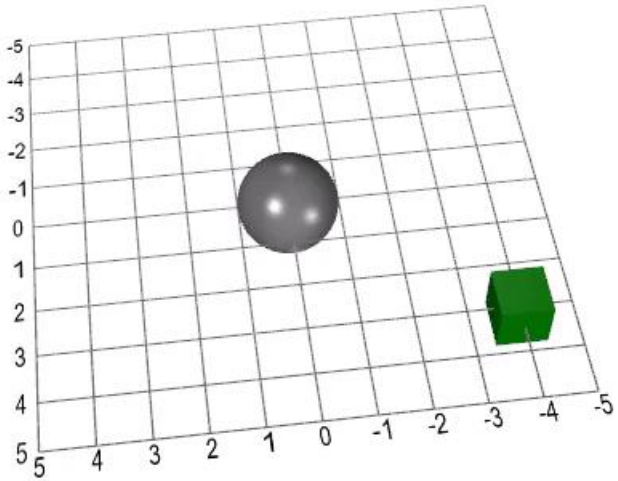
Анимация перемещения

- 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**
- Ниже приводится простейший пример анимации AnimMoveBy перемещения сферы от начала координат на вектор (-4,0,0) за 2 секунды
- Параллельно запускается анимация AnimMoveByX перемещения куба вдоль оси OX на 8 единиц за 3 секунды

Код

```
uses Graph3D;
begin
  var s := Sphere(Origin,1);
// Формируем анимацию к точке (-5,0,0) за 2 секунды
  var a := s.AnimMoveBy(-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).AnimMoveByX(8,3).Begin;
end.
```

Видео



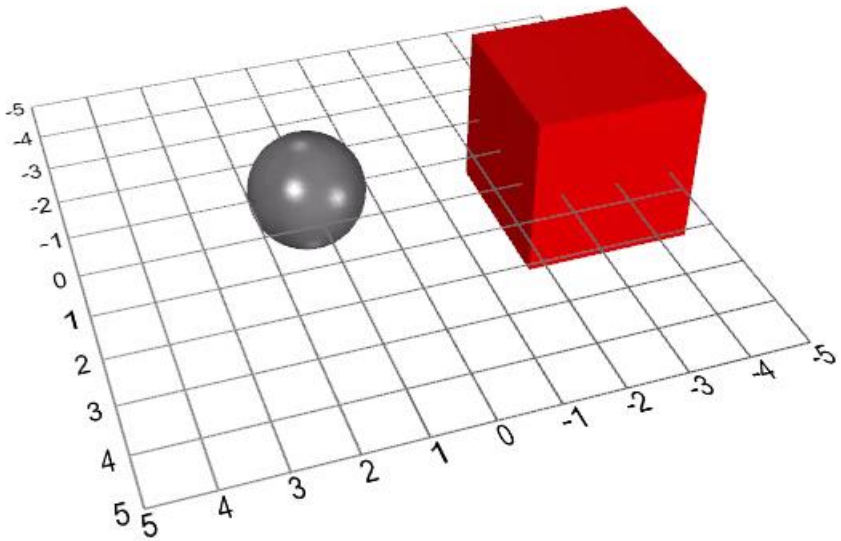
Анимация масштабирования

- Алгоритмическая анимация масштабирования выполняется с помощью методов AnimScale, AnimScaleX, AnimScaleY, AnimScaleZ
- В примере ниже запускается анимация AnimScale масштабирования сферы в 3 раза за 2 секунды
- Одновременно запускается анимация AnimScaleX сплющивания паралеллепипеда в 4 раза за 2 секунды

Код

```
uses Graph3D;
begin
  var s := Sphere(Origin,1);
// Запускаем анимацию масштабирования сферы
// в 3 раза за 2 секунды
  s.AnimScale(3,2).Begin;
  var b := Box(-4,-1,0,3,3,3,Colors.Red);
// Запускаем анимацию сплющивания паралеллепипеда по оси OX
// в 4 раза за 2 секунды
  b.AnimScaleX(0.25,2).Begin;
end.
```

Видео



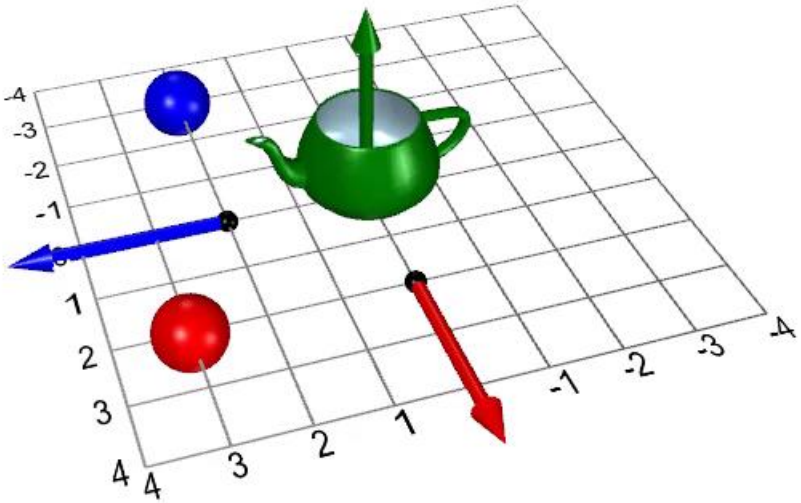
Анимация вращения

- 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
- В примере ниже запускается анимация вращения трёх объектов: чайника относительно своего центра вращения и оси OZ, синей сферы вокруг оси OX и центра, смещённого относительно центра сферы на (0,3,0), и красной сферы вокруг оси OY и центра, смещённого относительно центра сферы на (-3,0,0). Оси вращения изображены стрелками соответствующих цветов, новые центры вращения – маленькими чёрными сферами

Код

```
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.
```

Видео



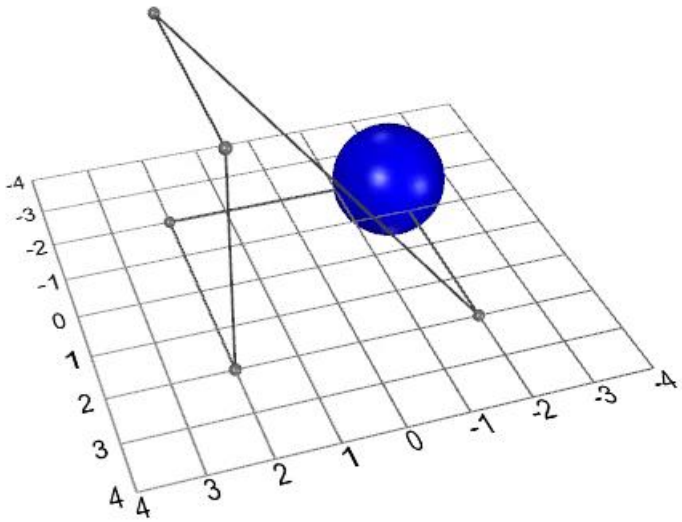
Анимация движения по пути

- Анимация движения по пути выполняется с помощью метода AnimMoveTrajectory(trajectory, seconds), где путь trajectory задаётся в виде последовательности точек
- В примере ниже запускается анимация движения по пути tr, задаваемом массивом точек. Сама траектория движения отображается для наглядности проволочным каркасом, а точки траектории – маленькими сферами

Код

```
uses Graph3D;
begin
  var s := Sphere(P3D(-2,-2,0),1);
  var tr := Arr(P3D(2,-2,0),P3D(2,2,0),P3D(2,2,4),
    P3D(2,-2,4),P3D(-2,2,0),P3D(-2,-2,0));
// Формируем и запускаем анимацию движения по пути,
// задаваемом массивом точек a, в течение 5 секунд
  s.AnimMoveTrajectory(tr,5).Begin;
end.
```

Видео



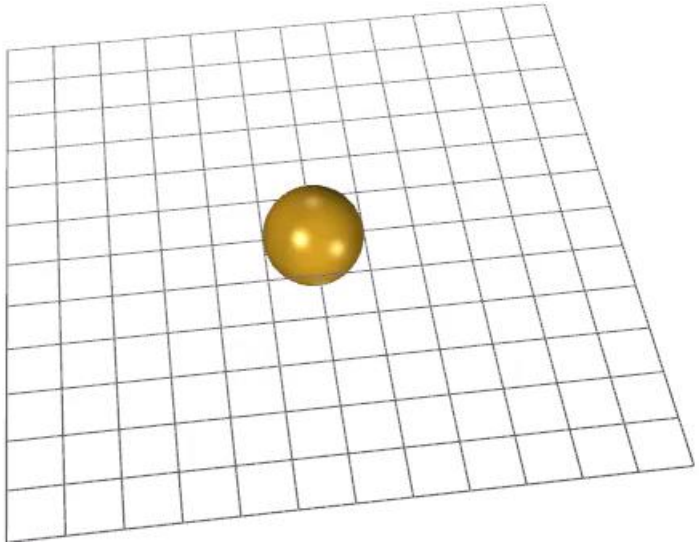
Комбинирование анимаций (+)

- 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, после которой происходит анимация масштабирования

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.AnimMoveByX(3) + s.AnimMoveByY(2) +
           s.AnimScale(2);
  a.Begin;
end.
```

Видео



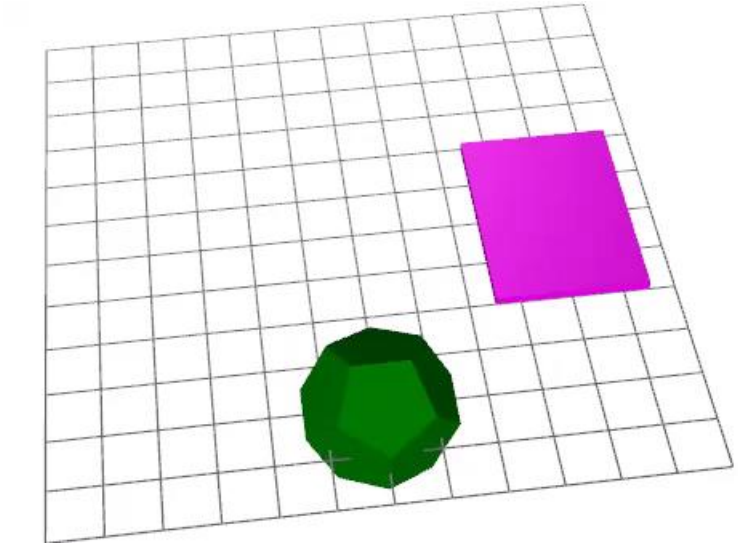
Комбинирование анимаций (*)

- 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

```
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.AnimMoveByX(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
end.
```

Видео



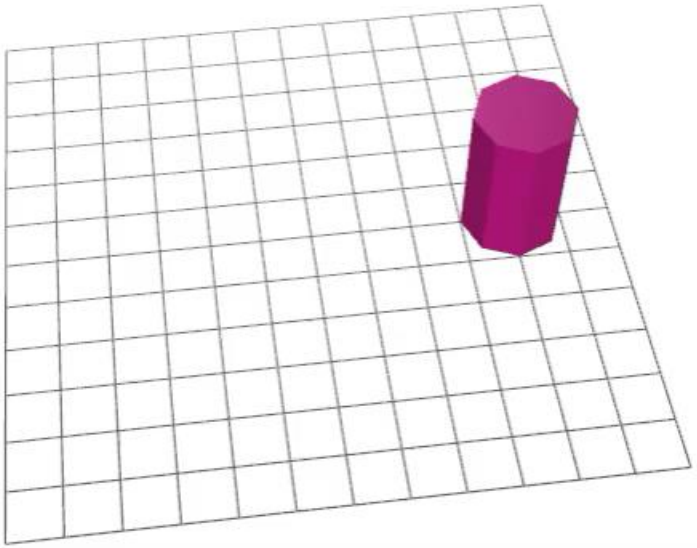
Комбинирование анимаций (+ и *)

- 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

```
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.AnimMoveByX(8,2) + p.AnimMoveByY(4,2)) *
           p.AnimRotate(OrtZ,360*3,4);
  a.Begin;
end.
```

Видео



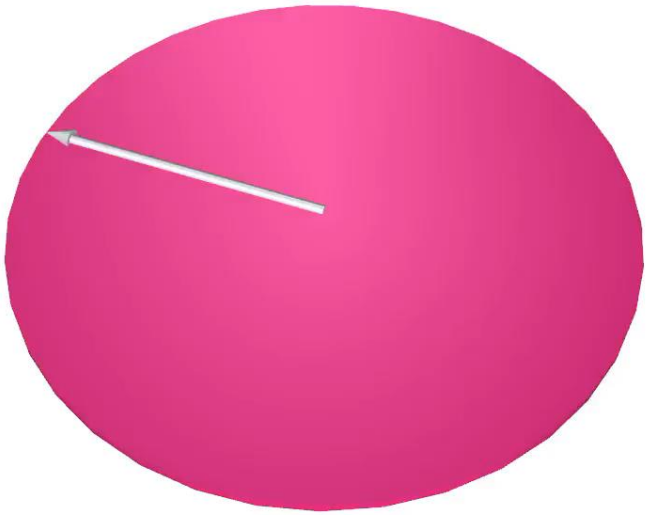
Модификаторы анимаций

- В Graph3D для каждой простой анимации доступны модификаторы AutoReverse, Forever и AccelerationRatio. Они добавляют к основной анимации некоторые свойства
- a.AutoReverse после анимации выполняет эту же анимацию "наоборот" – в результате время анимации удваивается
- a.Forever зацикливает анимацию, выполняя её бесконечно
- a.AccelerationRatio(acc,dec) выполняет анимацию с ускорением acc вначале и замедлением dec в конце (сумма acc и desc не должна быть > 1)

Код

```
uses Graph3D;
begin
  var disc := Cylinder(0,0,0,0.2,6);
  disc.AddChild(Arrow(0,0,0.2,6,0,0);
// Анимация диска со стрелкой
  var a := disc.AnimRotate(OrtZ,360,2);
// К основной анимации добавляем модификаторы,
// возвращающие диск в начальное положение,
// ускоряющие в начале и замедляющие в конце движения
// и зацикливающие этот процесс бесконечно
  a := a.AccelerationRatio(0.5,0.5)
        .AutoReverse.Forever;
  a.Begin;
end.
```

Видео



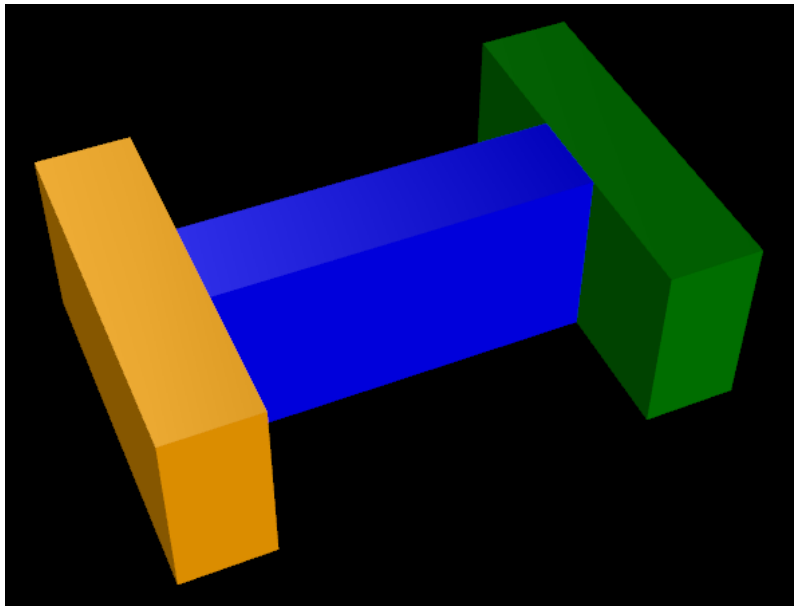
Клонирование

- Любой 3D-объект можно клонировать.
- Клонированный объект будет иметь те же атрибуты, что и исходный. Впоследствии клонированный объект можно сдвинуть, повернуть, изменить цвет и т.д.
- Клонировать объект удобнее чем создавать такой же

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).MoveByX(-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).MoveByX(2.5);
end.
```

Скриншот



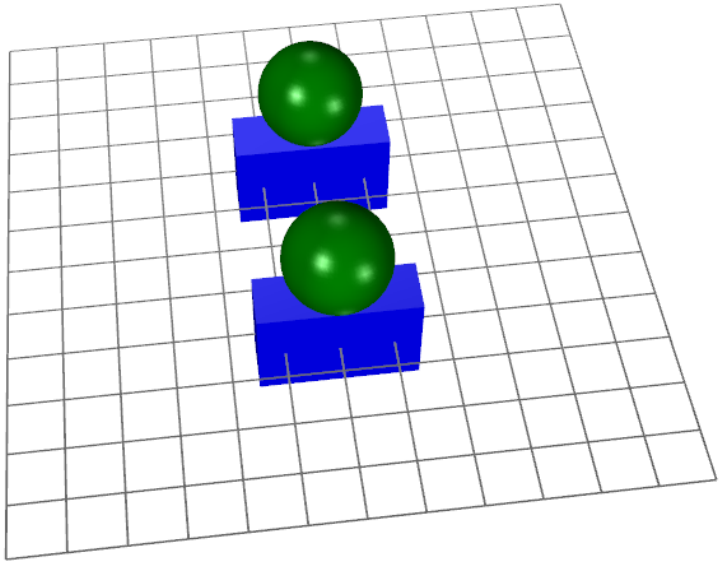
Группировка

- 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

```
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.MoveByY(-2);
  var g1 := g.Clone;
  g.MoveByY(4);
end.
```

Скриншот



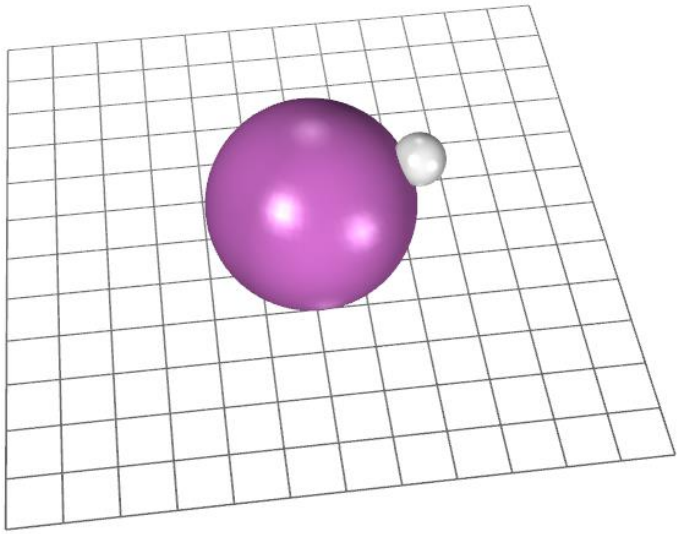
Дочерние элементы

- 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

```
uses Graph3D;

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

Скриншот



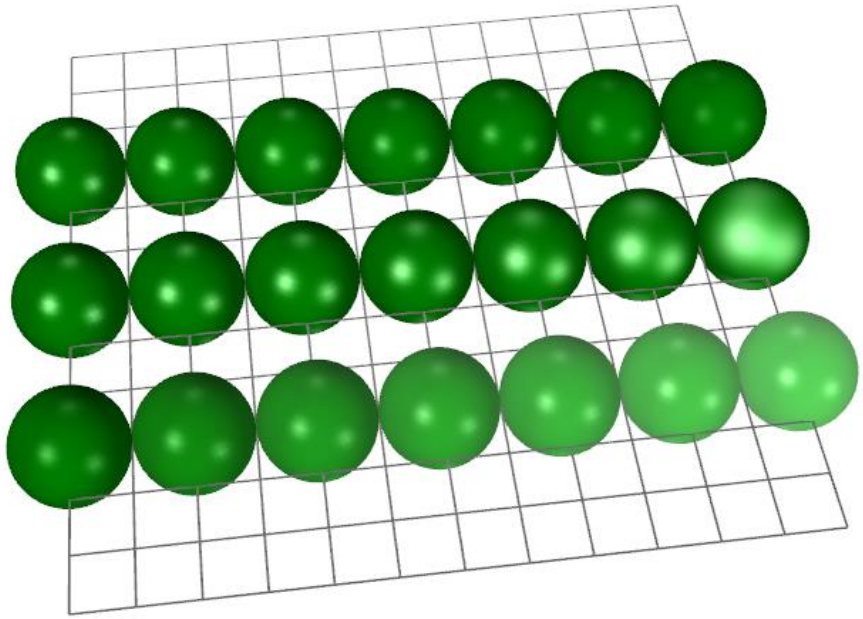
Материалы

- Поверхность 3D-объекта имеет некоторый материал. Простейшим материалом является одноцветный материал, задаваемый цветом. Материал передается как последний параметр при создании 3D-объекта.
- **DiffuseMaterial.** Создает матовую поверхность объекта, на которой свет распространен равномерно во всех направлениях. Его характеристикой является цвет.
- **SpecularMaterial.** Имитирует блестящую поверхность, отражая свет зеркально. Его характеристиками являются яркость и мощность. Его характеристиками являются яркость и размытость
- **EmissiveMaterial.** Создает поверхность, светящуюся заданным светом.
- Материалы можно складывать, используя операцию +
- В примере ниже в первом ряду уменьшается яркость зеркального материала, во втором ряду – увеличивается рассеяние зеркального материала, в третьем – увеличивается интенсивность свечения

Код

```
uses Graph3D;

begin
  for var i:=0 to 6 do
    Sphere(6-2*i,-3,0,1,
      Materials.Diffuse(Colors.Green) +
      Materials.Specular(255-32*i,100));
  for var i:=0 to 6 do
    Sphere(6-2*i,0,0,1,
      Materials.Diffuse(Colors.Green) +
      Materials.Specular(255,100-15*i));
  for var i:=0 to 6 do
    Sphere(6-2*i,3,0,1,
      Materials.Diffuse(Colors.Green) +
      Materials.Specular(128,100) +
      Materials.Emissive(GrayColor(15*i)));
end.
```

Скриншот



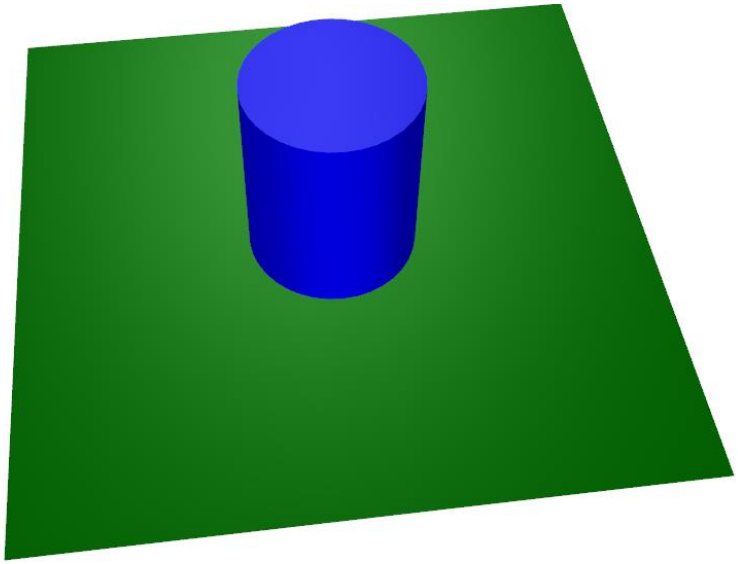
Освещение

- Для задания освещения используются различные типы источников света.
- **AmbientLight** – создаёт источник рассеянного света. Свет исходит со всех сторон.
- **DirectionalLight** – создаёт прямые параллельные лучи в направлении, заданном вектором.
- **PointLight** – создает точечный источник света.
- **SpotLight** – создает источник, который испускает свет на трехмерную сцену по конусу.
- По умолчанию сцена освещается рассеянным светом, дополненным несколькими направленными источниками света
- Будем освещать сцену попеременно (1) DirectionalLight, (2) PointLight и (3) SpotLight

Код

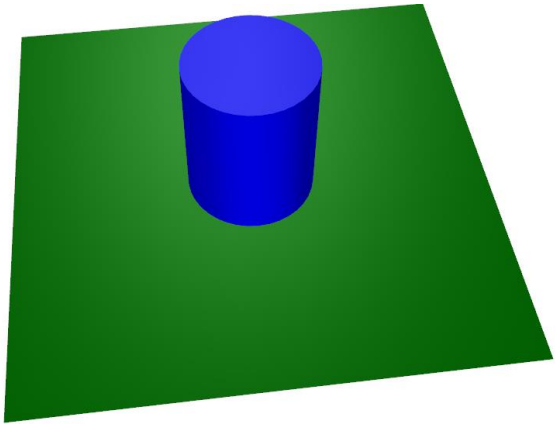
```
uses Graph3D;

begin
  View3D.ShowGridLines := False;
  Cylinder(0,0,0,5,2,True,Colors.Blue);
  Rectangle3D(0,0,-0.01,15,15,OrtZ,OrtX,Colors.Green);
// (1) PointLight
  var p := P3D(-3,3,1);
  Sphere(p,0.3,Colors.White);
  Lights.AddPointLight(GrayColor(64),p);
// (2) DirectionalLight
  var v := V3D(-1,-1,-1);
  Lights.AddDirectionalLight(GrayColor(100),v);
// (3) SpotLight
  var p := P3D(-3,3,1);
  var v := V3D(1,-1,1);
  Sphere(p,0.3,Colors.White);
  Arrow(p,v,Colors.White);
  Lights.AddSpotLight(GrayColor(255),p,v,90,45);
end.
```

Базовая сцена

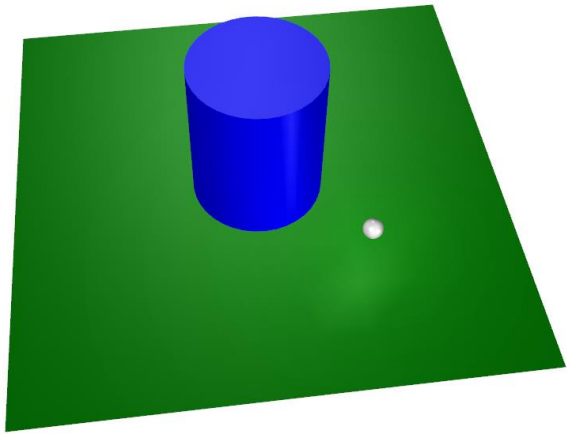


Освещение

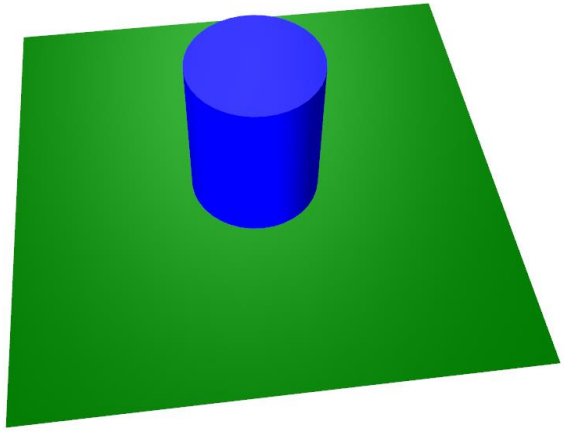
Базовая сцена



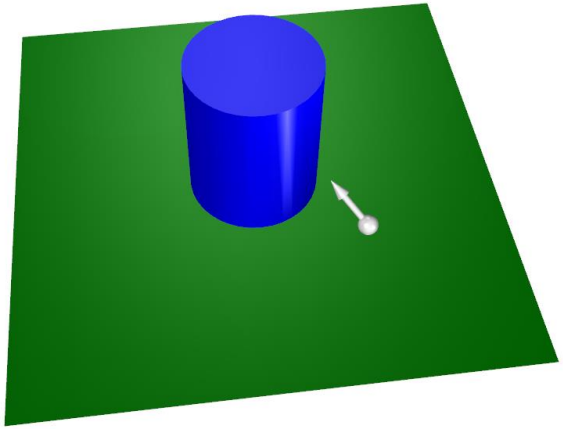
(1) PointLight



(2) DirectionalLight



(3) SpotLight



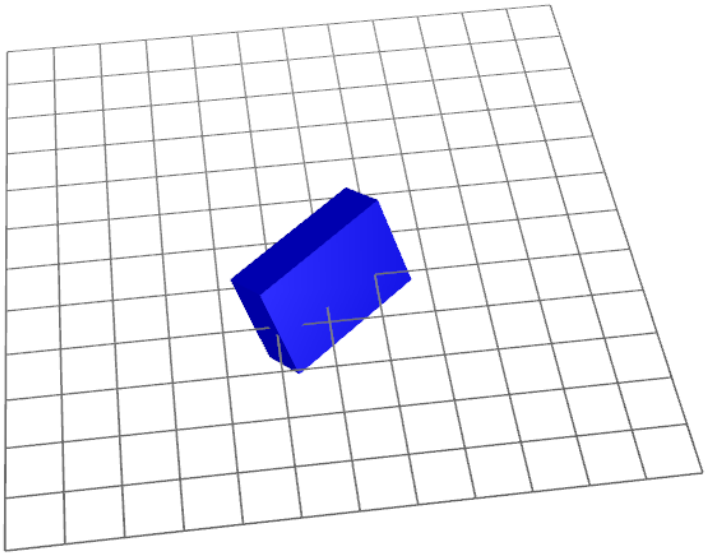
Обработка событий клавиатуры

- Как и в модуле GraphWPF, доступны события OnKeyDown, OnKeyUp.
- Их обработчики – это процедуры с параметром типа Key.
- Лучше всего записывать обработчик в виде лямбда-выражения
- Пример ниже иллюстрирует перемещение объекта стрелками и вращение его вокруг осей при нажатии клавиши с именем оси.

Код

```
uses Graph3D;

begin
  var s := Box(0,0,0,3,1,2,Colors.Blue);
  OnKeyDown := k ->
    case k of
  Key.Left: s.MoveBy(1,0,0);
  Key.Right: s.MoveBy(-1,0,0);
  Key.Up: s.MoveBy(0,-1,0);
  Key.Down: s.MoveBy(0,1,0);
  Key.X: s.Rotate(OrtX,10);
  Key.Y: s.Rotate(OrtY,10);
  Key.Z: s.Rotate(OrtZ,10);
    end;
end.
```

Скриншот



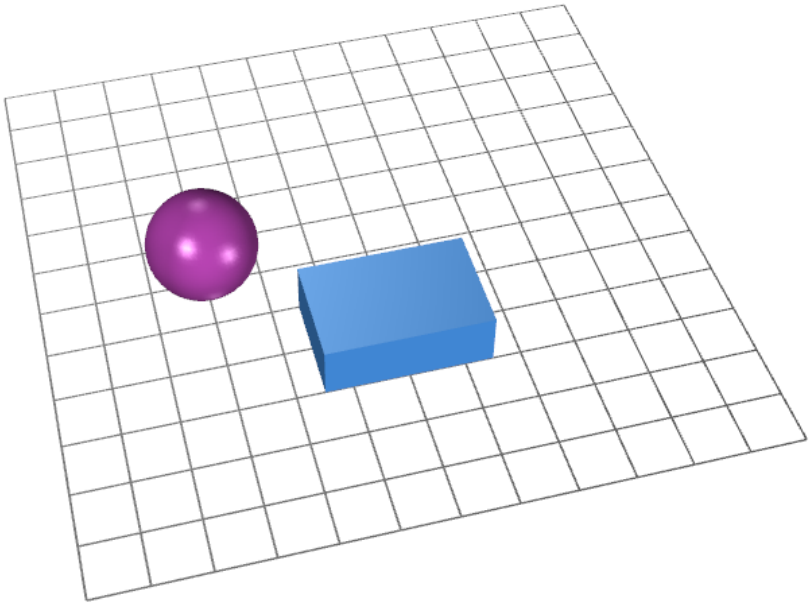
Обработка событий мыши

- Как и в модуле GraphWPF, доступны события OnMouseDown, OnMouseUp, OnMouseMove.
- Их обработчики – это процедуры с параметрами x,y,mb: integer. x,y – это координаты мыши, mb – кнопка мыши: mb = 0 – не нажата, mb = 1 – левая, mb = 2 – правая.
- Пример ниже иллюстрирует выбор ближайшего к точке мыши 3D-объекта с помощью функции FindNearestObject(x,y) и изменение у него цвета на случайный.

Код

```
uses Graph3D;

begin
  var s := Sphere(3,0,1,1);
  var b := Box(0,2,0.5,3,2,1);

  OnMouseDown += procedure (x,y,mb) -> begin
    if mb<>1 then exit;
    var v := FindNearestObject(x,y);
    if v<>nil then
      v.Color := RandomColor;
  end;
end.
```

Скриншот



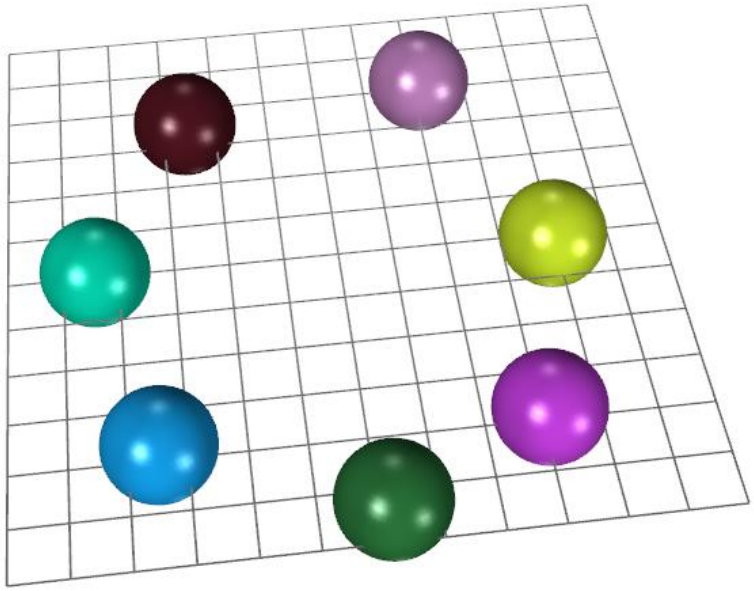
Обработка событий мыши (2)

- Программа ниже иллюстрирует перетаскивание 3D-объектов мышью по плоскости.
- Переменная obj хранит перетаскиваемый объект или nil.
- В обработчике OnMouseDown объект захватывается с помощью FindNearestObject.
- В обработчике OnMouseUp объект освобождается.
- В обработчике OnMouseMove если объект захвачен, то на плоскости OXY ищется точка, соответствующая точке (x,y) под курсором мыши, и если она найдена, к ней перемещается захваченный 3D-объект.

Код

```
uses Graph3D;

begin
  for var i:=-3 to 3 do Sphere(2*i,0,0,1);

  var obj: Object3D;

  OnMouseDown += procedure (x,y,mb) ->
    obj := FindNearestObject(x,y);

  OnMouseUp += procedure (x,y,mb) -> obj := nil;

  OnMouseMove += procedure (x,y,mb) -> begin
    if obj = nil then exit;
    var pp := PlaneXY.PointOnPlane(x,y);
    if pp<>BadPoint then
      obj.Position := pp;
  end;
end.
```

Скриншот



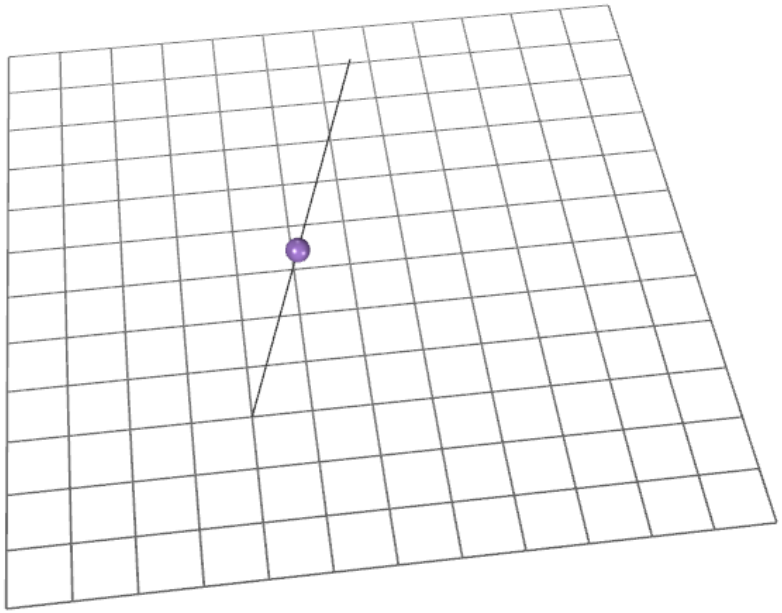
Обработка событий мыши (3)

- Программа ниже иллюстрирует перетаскивание 3D-объекта мышью по прямой.
- В обработчике OnMouseMove для захваченного объекта для прямой, по которой перемещается 3D-объект, ищется ближайшая к курсору мыши точка с помощью метода прямой NearestPointOnLine(x,y). После этого 3D-объект перемещается в эту точку

Код

```
uses Graph3D;

begin
  var p1 := p3D(2,3,0);
  var p := p3D(0,0,5);
  Segment3D(p,p1);
  var s := Sphere(p1,0.2);

  var obj: Object3D;
  OnMouseDown += procedure (x,y,mb) ->
    obj := FindNearestObject(x,y);

  OnMouseUp += procedure (x,y,mb) -> obj := nil;

  OnMouseMove += procedure (x,y,mb) -> begin
    if obj=nil then exit;
    obj.Position := Line(p,p1).NearestPointOnLine(x,y);
  end;
end.
```

Скриншот



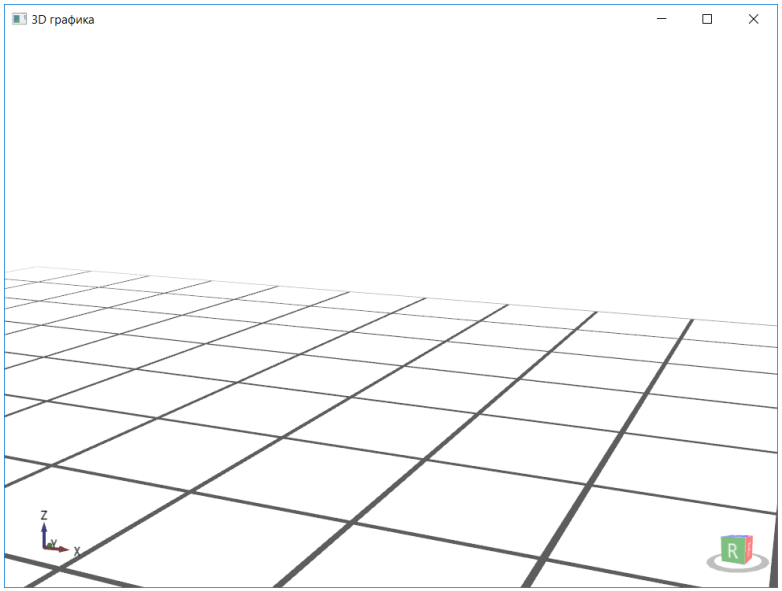
Камера и её свойства

- Камера определяется следующими свойствами:
  Position – точка, из которой мы смотрим на сцену;
  LookDirection – направление просмотра.
- У камеры есть методы:
  MoveBy(dx,dy,dz) – переместиться на вектор (dx,dy,dz);
  Rotate(axis,angle) – повернуть камеру вокруг оси axis на угол angle.
- Ниже демонстрируется пример заготовки игры от первого лица: камера находится внутри героя и может перемещаться и поворачиваться по клавишам.

Код

```
uses Graph3D;

begin
  GridLines.Length := 20;
  GridLines.Width := 20;
  Camera.Position := P3D(0,0,2);
  Camera.LookDirection := V3D(0,10,-2);
  OnKeyDown := k -> begin
    var delta := Camera.LookDirection*0.01;
    case k of
  Key.W: Camera.MoveBy(delta.X,delta.Y,0);
  Key.S: Camera.MoveBy(-delta.X,-delta.Y,0);
  Key.D: Camera.Rotate(OrtZ,1);
  Key.A: Camera.Rotate(OrtZ,-1);
    end;
  end;
end.
```

Скриншот



Самое важное

- Модуль **Graph3D** – простой и мощный модуль трёхмерной графики, содержащий всё необходимое для простейших визуализаций и анимаций.
- Он содержит базовые **трёхмерные примитивы**, такие как сфера, параллелепипед, конус, пирамида, призма, цилиндр, платоновы тела и проч.
- Каждый 3D-объект можно перемещать, поворачивать и менять в размерах. Кроме того, каждый 3D-объект имеет цвет.
- Можно **анимировать** 3D-объекты и **комбинировать анимации**, создавая параллельные и последовательные анимации.
- Можно **группировать** 3D-объекты и **клонировать** как 3D-объекты, так и их группы.
- Каждый 3D-объект имеет **материал**. Можно использовать **диффузный**, **зеркальный** и **светящийся** материалы для создания комбинированного материала.
- Можно задавать освещение сцены, используя **направленный**, **точечный** и **спотовый** источники света.
- Можно задавать свойства камеры, перемещать и вращать её, создавая игры от первого лица.
- В модуле Graph3D доступны обработчики мыши и клавиатуры, позволяющие создавать интерактивные трёхмерные программы. В них используется ряд вспомогательных функций, которые позволяют найти ближайший к данной точке на экране трёхмерный объект, точку на плоскости, соответствующую данной экранной точке и ближайшую точку на прямой относительно данного 3D-луча.
- Модуль Graph3D – один из лучшх способов освоить основные понятия трёхмерной графики.