Анимация в Mathematica

Юдинцев В. В.
Самарский университет
Кафедра теоретической механики

Стили по умолчанию...

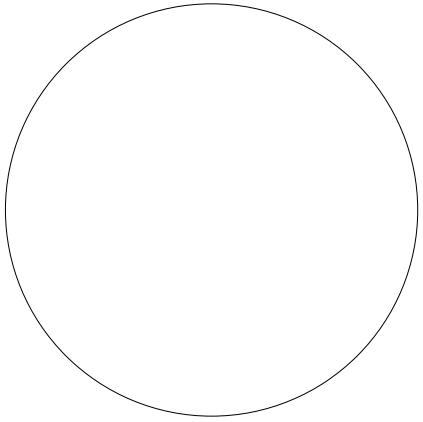
In[•]:=
```
SetOptions[Graphics, BaseStyle → {14, FontFamily → "Helvetica"}];
```

Геометрические объекты в Mathematica

Окружность

In[•]:=
```
Graphics[Circle[{0, 0}, 2]]
```

Out[•]=

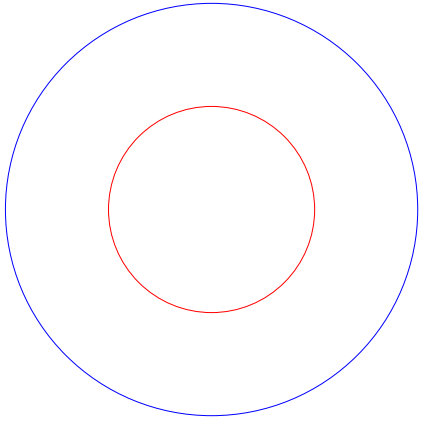


Graphics

Функция Graphics отображает графические примитивы, передаваемые ей в виде списка. Графические примитивы могут чередоваться описаниями их атрибутов (цвет и толщина линий, тип линий, ...)

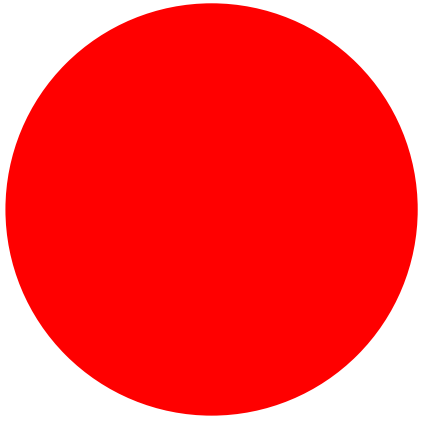
In[ ]:=
```
Graphics[{Red, Circle[{0, 0}, 1], Blue, Circle[{0, 0}, 2]}]
```

Out[ ]=



In[ ]:=
```
Graphics[{Red, Disk[{0, 0}, 1]}]
```

Out[ ]=

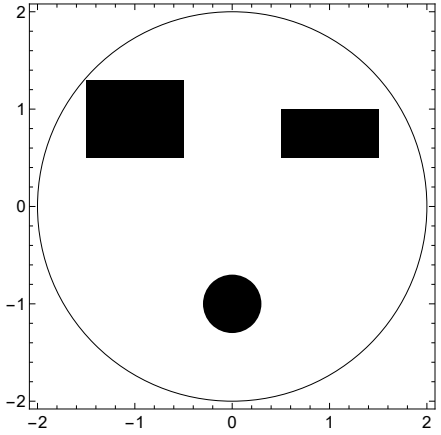


Несколько объектов

Несколько объектов $\{g_1, g_2, g_3, ..., g_n\}$

In[•]:=
```
Graphics[{
  Circle[{0, 0}, 2],
  Rectangle[{0.5, 0.5}, {1.5, 1}],
  Rectangle[{-1.5, 0.5}, {-0.5, 1.3}],
  Disk[{0, -1}, 0.3]
 },
 Frame → True]
```

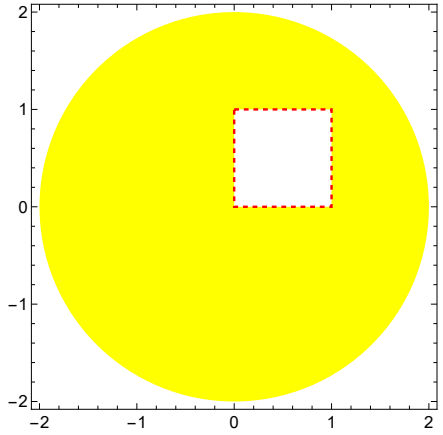
Out[•]=



Контур и заливка

In[•]:=
```
Graphics[{
  Yellow, (* Желтый *)
  Disk[{0, 0}, 2], (* Диск *)
  White, (* Белый *)
  EdgeForm[{Thick, Red, Dashed}], (* Толстая красная рамка пунктиром *)
  Rectangle[{0, 0}, {1, 1}] (* Прямоугольник *)
},
Frame → True]
```

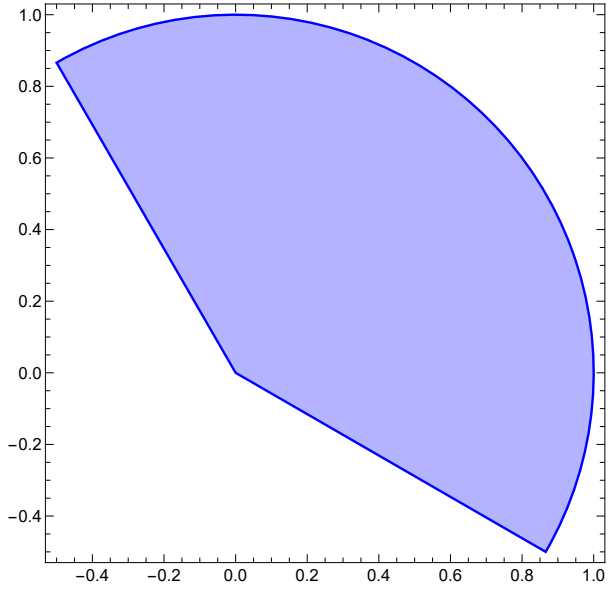
Out[•]=



Сектор

In[•]:=
```
Graphics[
 {
  Lighter[Blue, 0.7], (* Светлее на 70 %. 100 % -- белый *)
  EdgeForm[{Thick, Blue}], (* толстая голубая линия *)
  Disk[{0, 0}, 1, {-30 °, 120 °}]
 },
 Frame → True (* Показать рамку рисунка с надписями осей *),
 ImageSize → 500
]
```

Out[•]=

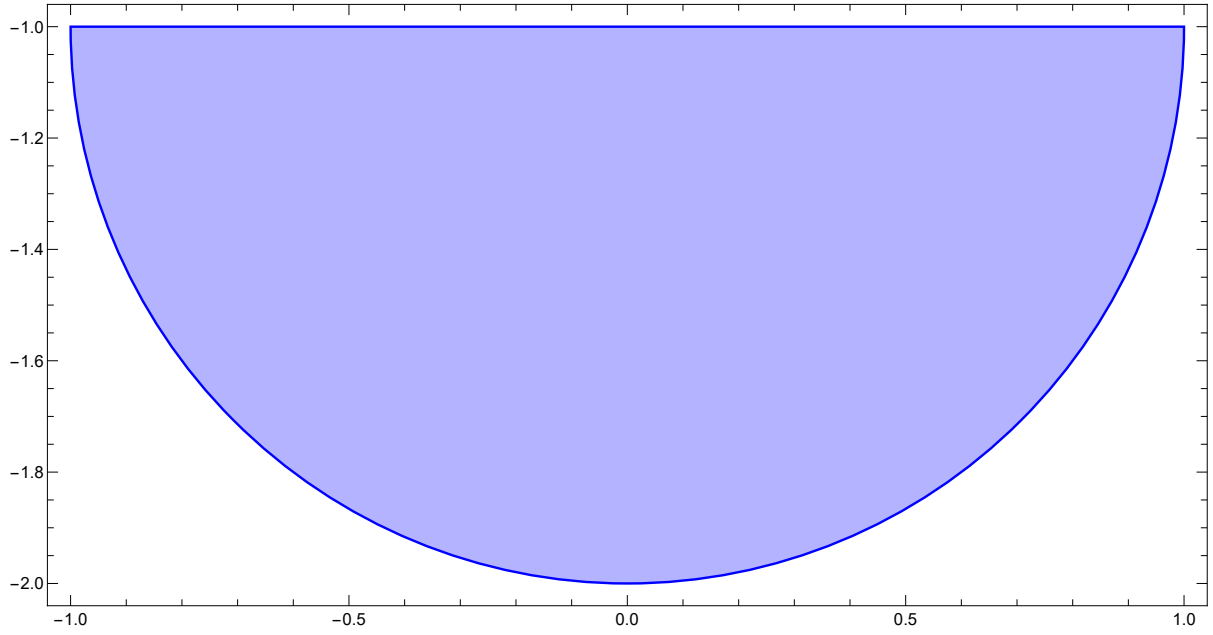


Угол отсчитывается от горизонтальной оси против часовой стрелки

Тело 1 механизма

In[•]:=
```
Graphics[{
  Lighter[Blue, 0.7],
  EdgeForm[{Thick, Blue}],
  Disk[{0, -1}, 1, {-180 °, 0 °}]},
 Frame → True, ImageSize → 1000]
```

Out[•]=

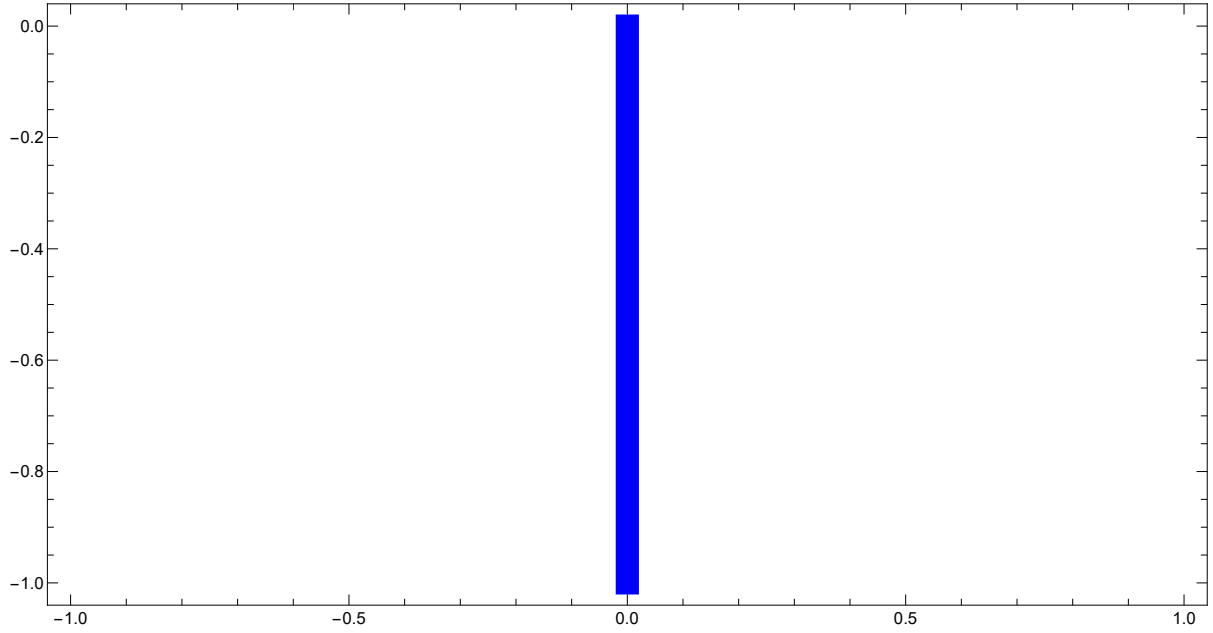


Опора

Функция Line[{{x1,y1},{x2,y2},{x3,y3}}]

In[•]:=
```
Graphics[
 {
  Blue,
  Thickness[0.02],
  Line[{{0, 0}, {0, -1}}] (* Line[{{x1,y1},{x2,y2},{x3,y3},...}] *)
 },
 Frame → True, ImageSize → 1000]
```

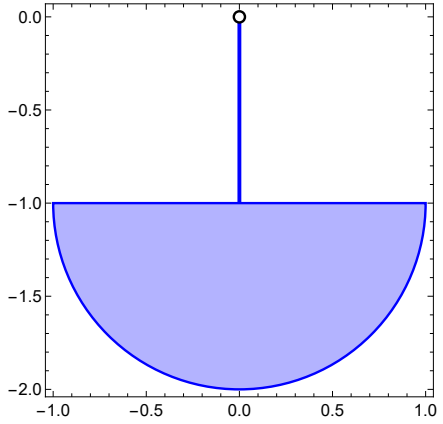
Out[•]=



Стойка, стержень и диск

In[•]:=
```
Graphics[{
   (* Стержень *)
  Blue,
  Thickness[0.01],
  Line[{{0, 0}, {0, -1}}],
  (* Стойка *)
  White,
  EdgeForm[{Thick, Black}],
  Disk[{0, 0}, 0.03],
  (* Диск *)
  Lighter[Blue, 0.7],
  EdgeForm[{Thick, Blue}],
  Disk[{0, -1}, 1, {-180 °, 0 °}]
 },
 Frame → True]
```

Out[•]=



Стойка, стержень и диск

Объединяем в список геометрических объектов, с которым в дальнейшем будем работать, как с единым объектом

```
In[•]:= body1 = {
          Blue, Thickness[0.01], Line[{{0, 0}, {0, -1}}],

          Lighter[Blue, 0.7], EdgeForm[{Thick, Blue}],
          Disk[{0, -1}, 1, {-180 °, 0 °}],

          White, EdgeForm[{Thick, Black}],
          Disk[{0, 0}, 0.03]
         }
```

```
Out[•]= {■, Thickness[0.01], Line[{{0, 0}, {0, -1}}], ■, EdgeForm[{Thickness[Large], ■}], Disk[{0, -1}, 1, {-180 °, 0}], □,
         EdgeForm[{Thickness[Large], ■}], Disk[{0, 0}, 0.03]}
```

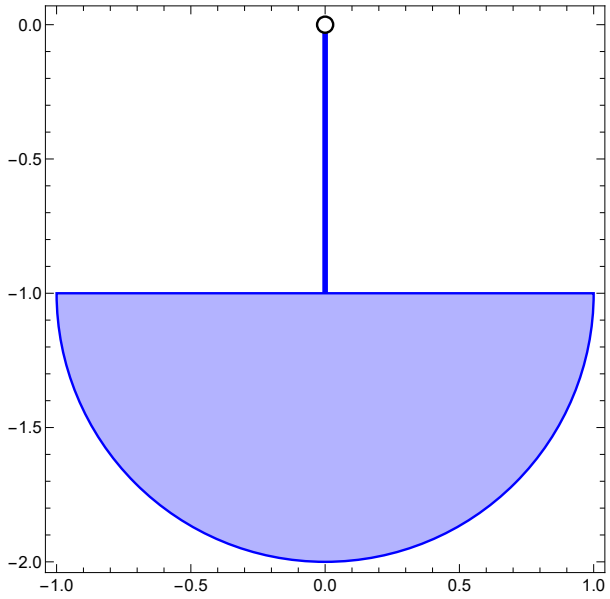
Чтобы показать эти объекты необходимо использовать функцию **Graphics[]**

Перемещение и поворот

Графический комплекс, состоящий из стойки и пластины

In[•]:=
```
Graphics[body1, Frame → True, ImageSize → 500]
```

Out[•]=

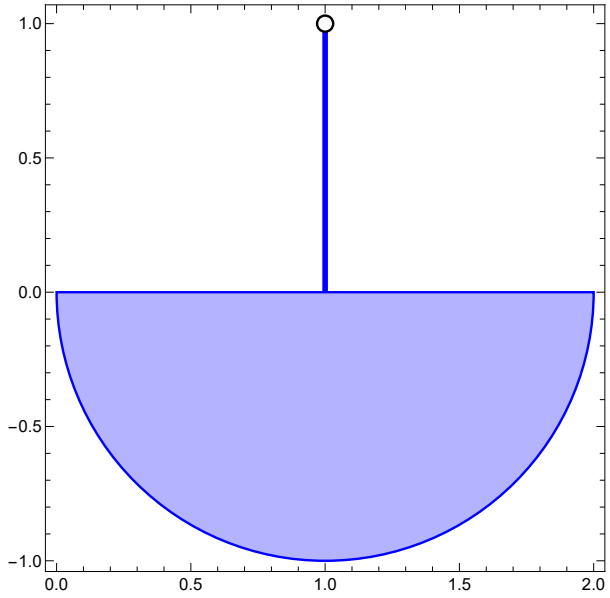


Перемещение и поворот

Перенесём объект на *1* единицу вдоль осей *x* и *y*. Используем функцию GeometricTransformation.
Первый аргумент функции - геометрический объект, второй - список, состоящий из двух элементов: матрица поворота и вектор смещения. Таким образом, функция GeometricTransformation определяет поворот и смещение объекта.

In[•]:=
```
Graphics[
 GeometricTransformation[body1, {RotationMatrix[0], {1, 1}}],
 Frame → True, ImageSize → 500]
```

Out[•]=



GeometricTransformation[ОБЪЕКТ, {МатрицаПовотора, {Смещение}}]

Матрица поворота

Матрица поворота в плоскости (вокруг оси z, перпендикулярной плоскости рисунка)

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \cos\varphi & \cos\varphi \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

Точка с координатами $\begin{pmatrix} x \\ y \end{pmatrix}$ поворачивается на угол φ вокруг начала координат

```
In[•]:= RotationMatrix[45.0 °] // MatrixForm
```

```
Out[•]//MatrixForm=
( 0.707107  -0.707107 )
( 0.707107   0.707107 )
```

Поворачиваем вектор {1, 0} на 20 градусов против часовой стрелки

```
In[•]:= RotationMatrix[20.0 °].{1, 0}
```

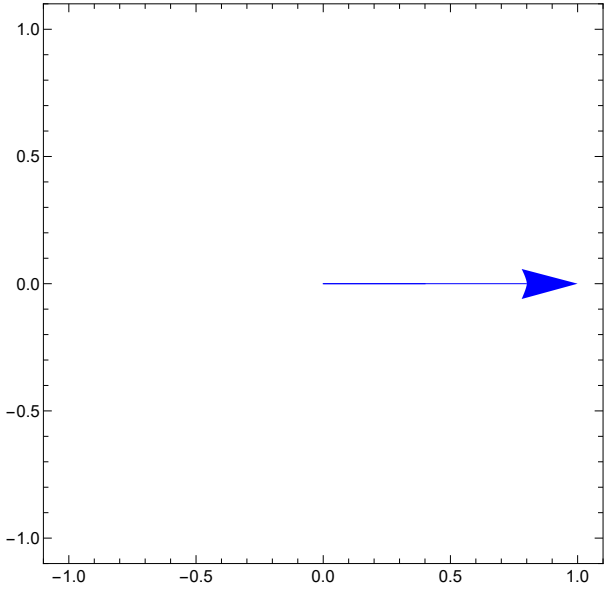
```
Out[•]= {0.939693, 0.34202}
```

Поворот вектора

Исходный вектор

In[•]:=
```
r1 = {1.0, 0.0};
Graphics[{
  Blue,
  Arrowheads[0.1], (* Размер стрелки *)
  Arrow[{{0, 0}, r1}]
 },
 PlotRange → {{-1.1, 1.1}, {-1.1, 1.1}}, Frame → True, ImageSize → 500]
```

Out[•]=

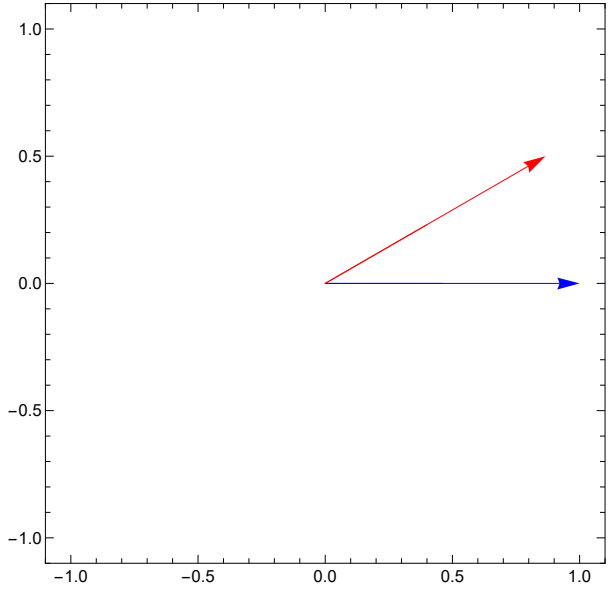


Поворот вектора

Повернутый вектор

In[•]:=
```
Graphics[{
  Blue, Arrow[{{0, 0}, r1}],
  Red, Arrow[{{0, 0}, RotationMatrix[30 °]. r1}]
 },
PlotRange → {{-1.1, 1.1}, {-1.1, 1.1}}, Frame → True, ImageSize → 500]
```

Out[•]=



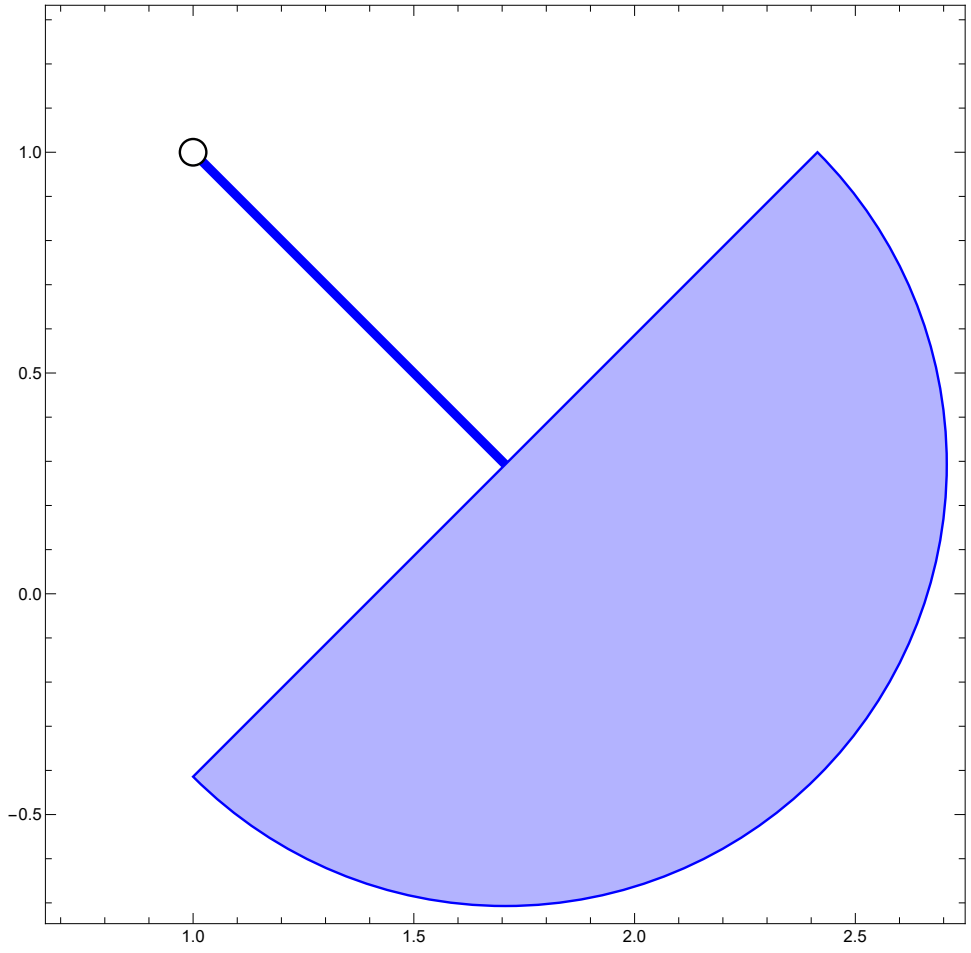
Это “активная” точка зрения на поворот - поворачивается объект.

GeometricTransformation

Использование функции **GeometricTransformation** для поворота и перемещения

In[•]:=
```
Graphics[
 GeometricTransformation[body1, {RotationMatrix[45 °], {1, 1}}],
 Frame → True, ImageSize → 800]
```

Out[•]=

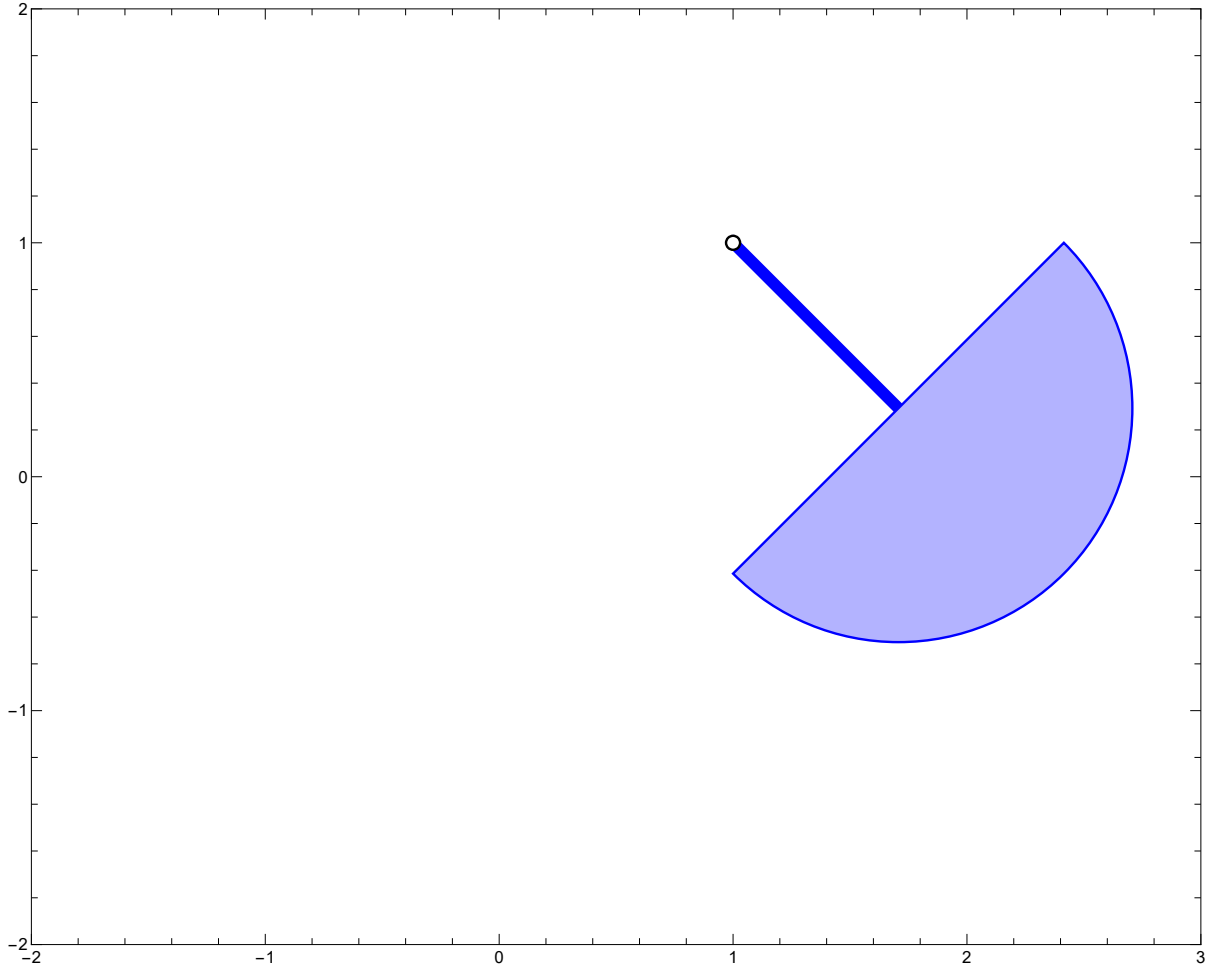


Держимся в рамках

Опция **PlotRange** позволяет определить границы области рисунка, чтобы при перемещении объекта Mathematica не выполняла автоматический выбор границ по осям координат (для построения анимации это не нужно).

In[•]:=
```
Graphics[
 GeometricTransformation[body1, {RotationMatrix[45 °], {1, 1}}],
 Frame → True, PlotRange → {{-2, 3}, {-2, 2}}, ImageSize → 1000]
```

Out[•]=



Объект как функция положения

Создадим функцию, для формирования геометрического объекта в заданном положении. Удалим все определения, связанные с именем **body1:**

In[188]:=
```
ClearAll[body1];
```

Определяем функцию от угла поворота с именем body

In[189]:=
```
body1[φ_] := Module[{b},
  b = {
    (* Стержень *)
    Blue, Thickness[0.01], Line[{{0, 0}, {0, -1}}],
    (* Сектор, пластина *)
    Lighter[Blue, 0.7], EdgeForm[Directive[Thick, Blue]],
    Disk[{0, -1}, 1, {-180 °, 0 °}],
    (* Стойка *)
    White, EdgeForm[Directive[Thick, Black]],
    Disk[{0, 0}, 0.03]
   };
  (* Результат работы функции – повернутый и смещенный объект *)
  GeometricTransformation[b, {RotationMatrix[φ], {0, 0}}]
 ];
```

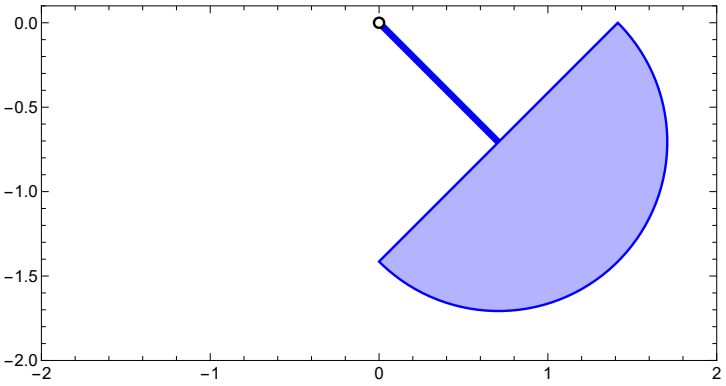
Функция **Module** позволяет создавать многострочные функции с локальными переменными, как в традиционных языках программирования.

Поворот

Рисуем, вызывая новую функцию

In[•]:=
```
Graphics[body1[45 °], Frame → True, PlotRange → {{-2, 2}, {-2, 0.1}}, ImageSize → 600]
```

Out[•]=



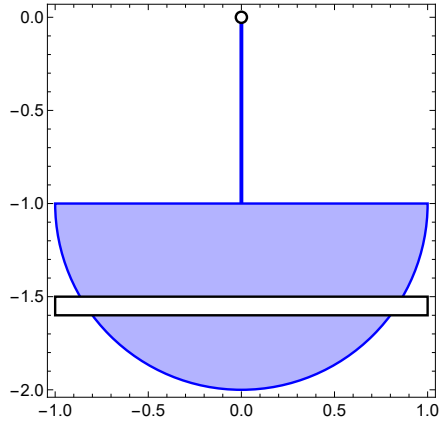
Добавляем канал для шарика

Нарисуем поверх пластины прямоугольник, который будет изображать канал для шарика.

In[•]:=
```
body1[φ_] := Module[{b},
  b = {
    (* Стержень *)
    Blue, Thickness[0.01], Line[{{0, 0}, {0, -1}}],
    (* Сектор, пластина *)
    Lighter[Blue, 0.7], EdgeForm[Directive[Thick, Blue]],
    Disk[{0, -1}, 1, {-180 °, 0 °}],
    (* Стойка *)
    White, EdgeForm[Directive[Thick, Black]],
    Disk[{0, 0}, 0.03],
    (* Канал *)
    Rectangle[{-1, -1.6}, {1, -1.5}]
   };
  GeometricTransformation[b, {RotationMatrix[φ], {0, 0}}]
 ];
```

```
Graphics[body1[0 °], Frame → True]
```

Out[•]=

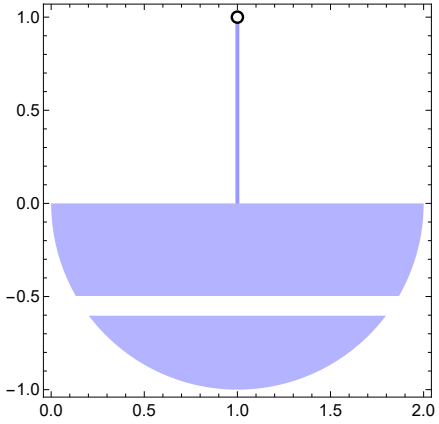


Канал для шарика

Маскируем (убираем границу у всех объектов), чтобы казалось, что пластина с каналом это один объект.

In[•]:=
```
body1[φ_] := Module[{b},
 b = {
  Lighter[Blue, 0.6], Thickness[0.01], Line[{{0, 0}, {0, -1}}],

  Lighter[Blue, 0.7],
  Disk[{0, -1}, 1, {-180°, 0°}],

  White, EdgeForm[Directive[Thick, Black]],
  Disk[{0, 0}, 0.03],

  EdgeForm[Directive[Thin, White]],
  Rectangle[{-1, -1.6}, {1, -1.5}]
  };
 GeometricTransformation[b, {RotationMatrix[φ], {1, 1}}]
 ];
Graphics[body1[0°], Frame → True]
```

Out[•]=



Пружина

Функция **Table** генерирует списки
Table[выражение, {i, начальное значение, конечное значение, шаг}]

```
In[•]:= Table[i^2, {i, 0, 5}]
```

```
Out[•]= {0, 1, 4, 9, 16, 25}
```

Пружина в виде ломаной линии

```
In[232]:= Spring[L_, d_, n_] := Table[{i * L / n, (-1)^i d / 2}, {i, 0, n}];
          Spring[0.5, 0.01, 5]
```

```
Out[233]= {{0., 0.005}, {0.1, -0.005}, {0.2, 0.005}, {0.3, -0.005}, {0.4, 0.005}, {0.5, -0.005}}
```

Чтобы нарисовать пружину, используем функцию **Line[{p1, p2, p3,..., pn}]**

```
In[•]:= Graphics[
         Line[Spring[0.5, 0.01, 20]]
        ]
```

```
Out[•]=
```

Больше “витков”, больше диаметр

```
In[•]:= Graphics[
         Line[Spring[0.5, 0.03, 50]]
         ]
```

```
Out[•]=
```

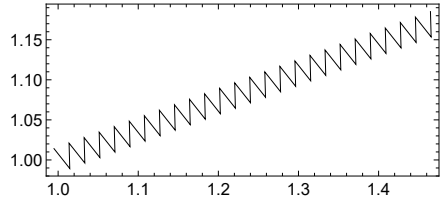
Перемещение и поворот пружины

Линию, нарисованную по точкам, тоже можно поворачивать и перемещать, используя функцию **Geometric-Transformation**

Поворачиваем на 20 градусов и перемещаем на *1* и *1* по *x* и *y* соответственно

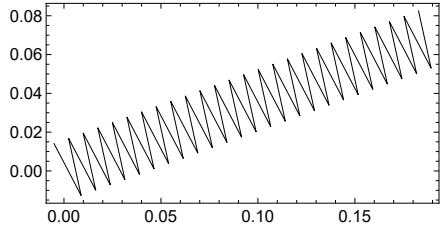
In[•]:=
```
Graphics[
 GeometricTransformation[Line[Spring[0.5, 0.03, 50]], {RotationMatrix[20 °], {1, 1}}],
 Frame → True
 ]
```

Out[•]=



In[•]:=
```
Graphics[
 GeometricTransformation[Line[Spring[0.2, 0.03, 50]], {RotationMatrix[20 °], {0, 0}}],
 Frame → True
]
```

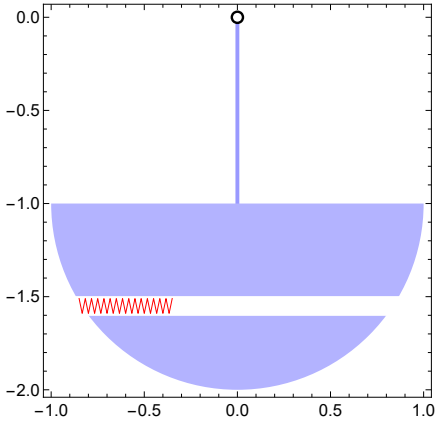
Out[•]=



Стержень, пластина и пружина

In[•]:=
```
body1[φ_] := Module[{b},
   b =
    {
     Lighter[Blue, 0.6], Thickness[0.01], Line[{{0, 0}, {0, -1}}],

     Lighter[Blue, 0.7], Disk[{0, -1}, 1, {-180°, 0°}],

     White, EdgeForm[{Thick, Black}],
     Disk[{0, 0}, 0.03],

     EdgeForm[{Thin, White}],
     Rectangle[{-1, -1.6}, {1, -1.5}],

     Red, Thin, Translate[Line[Spring[0.5, 0.08, 30]], {-0.85, -1.55}]}
   ;
  GeometricTransformation[b, {RotationMatrix[φ], {0, 0}}]
  ];
Graphics[body1[0°], Frame → True]
```

Out[•]=

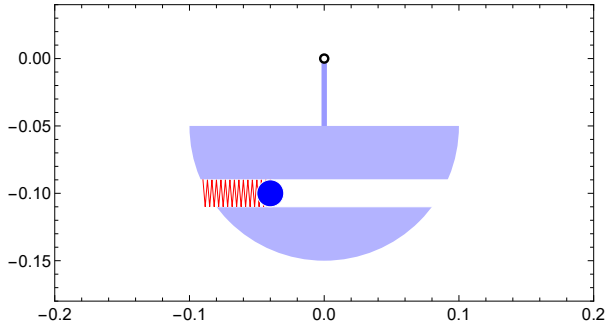


Шарик

Добавляем в систему шарик (функция **Disk**)

In[•]:=
```
body1[φ_, x_] := Module[{b},
   b =
    {
     (* СТЕРЖЕНЬ *)
     Lighter[Blue, 0.6], Thickness[0.01], Line[{{0, 0}, {0, -0.05}}],
     (* ПЛАСТИНА *)
     Lighter[Blue, 0.7], Disk[{0, -0.05}, 0.1, {-180°, 0°}],
     (* ОПОРА *)
     White, EdgeForm[{Thick, Black}], Disk[{0, 0}, 0.003],
     (* КАНАЛ *)
     EdgeForm[{Thin, White}], Rectangle[{-1, -0.09}, {1, -0.11}],
     (* ПРУЖИНА *)
     Red, Thin, Translate[Line[Spring[x, 0.02, 30]], {-0.09, -0.1}],
     (* ШАРИК *)
     Blue, Disk[{-0.09 + x, -0.1}, 0.01]
    };
  GeometricTransformation[b, {RotationMatrix[φ], {0, 0}}]
  ];

Graphics[body1[0, 0.05], Frame → True, PlotRange → {{-0.2, 0.2}, {-0.18, 0.04}}, ImageSize → 500]
```

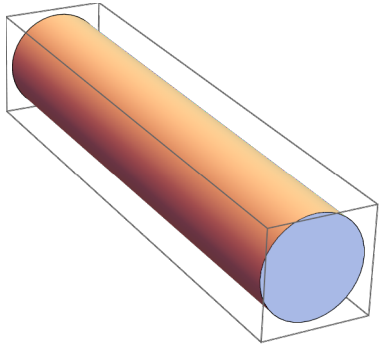
Out[•]=



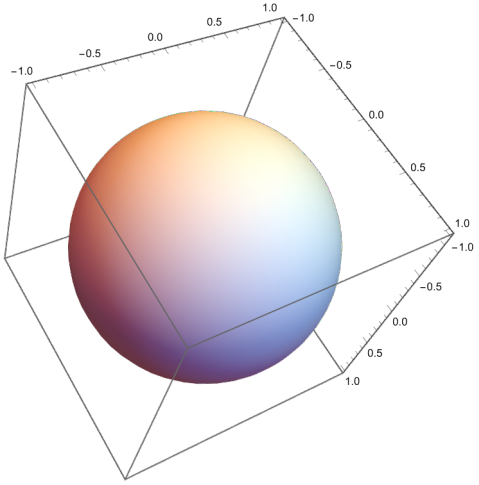
Переходим в 3D

In[•]:=
```
Graphics3D[{Cylinder[{{0, 0, 0}, {0, 0, 1}}, 0.1]}]
Graphics3D[{Sphere[{0, 0, 0}, 1]}, Axes → True]
```

Out[•]=



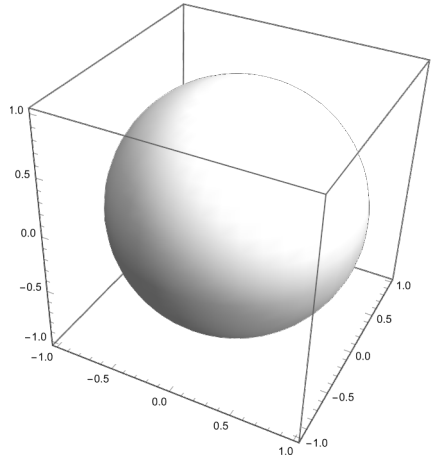
Out[•]=



Освещение

```
Graphics3D[{Sphere[{0, 0, 0}, 1]}, Axes → True, Lighting → "Neutral"]
```

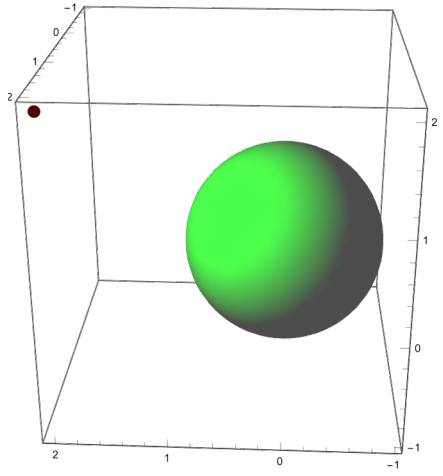
Out[•]=



Освещение

In[•]:=
```
lp = 2 {1, 1, 1};
lightColor = Red;
Graphics3D[{
   {White, Sphere[{0, 0, 0}, 1]},
   {Specularity[0], lightColor, Sphere[lp, 0.05]}
 }, Axes → True, Lighting → {{"Point", Green, lp}, {"Ambient", GrayLevel[0.3]}}]
```

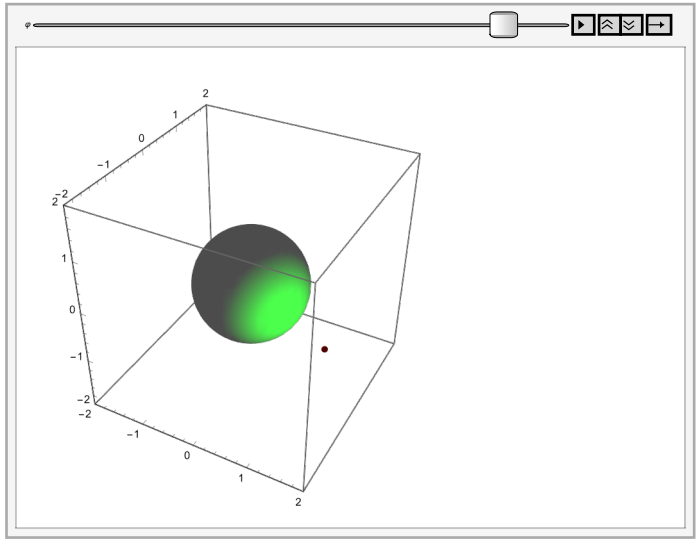
Out[•]=



Освещение

In[190]:=
```
lightColor = Red;
Animate[
 Graphics3D[{
   {White, Sphere[{0, 0, 0}, 1]},
   {lightColor, Sphere[2 {Cos[φ], Sin[φ], 0}, 0.05]}
  }, Axes → True, Lighting → {{"Point", Green, 2 {Cos[φ], Sin[φ], 0}}, {"Ambient", GrayLevel[0.3]}}, PlotRange → {{-2, 2}, {-2, 2}, {-2, 2}}]
 , {φ, 0, 2 π}]
```

Out[191]=



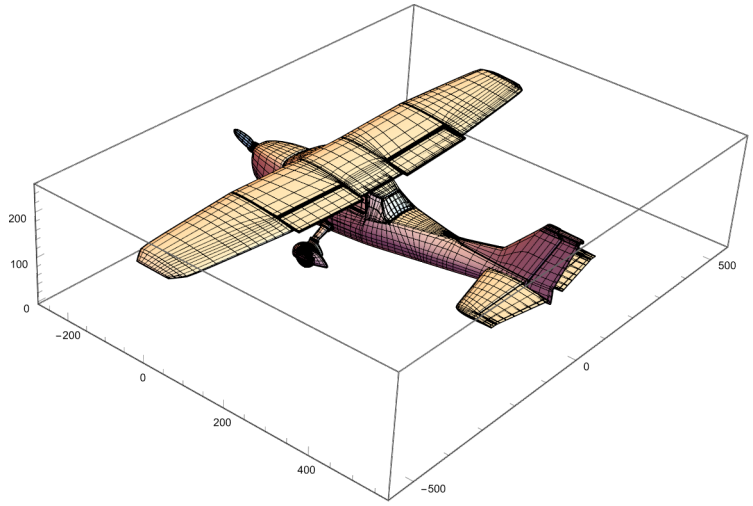
Импорт 3D объекта

Импорт 3D объекта из файла (.obj)

In[192]:=
```
SetDirectory[NotebookDirectory[]];
object = Import["11804_Airplane_v2_l2.obj", {"GraphicsComplex"}];
```

In[194]:=
```
Graphics3D[{object}, Axes → True]
```

Out[194]=



Импорт 3D объекта

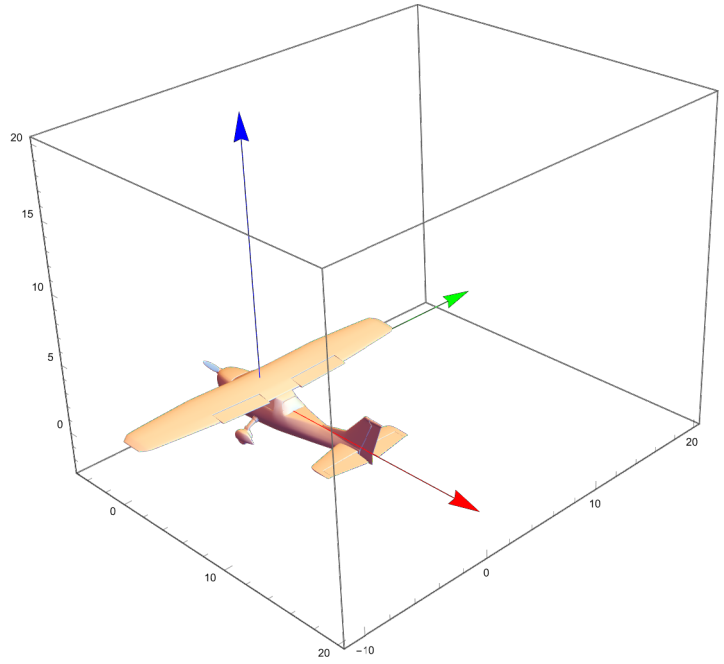
Связанная с объектом система координат (начало координат определяется в файле с моделью)

In[195]:=
```
SK = {Red, Arrow[{{0, 0, 0}, {20, 0, 0}}], Green, Arrow[{{0, 0, 0}, {0, 20, 0}}], Blue, Arrow[{{0, 0, 0}, {0, 0, 20}}]};
```

Масштабирование и перемещение (функции Translate и Scale)

In[196]:=
```
Graphics3D[{SK, EdgeForm[None], Translate[Scale[object, 0.02], {-145, 0, -128}]}, Axes → True]
```

Out[196]=

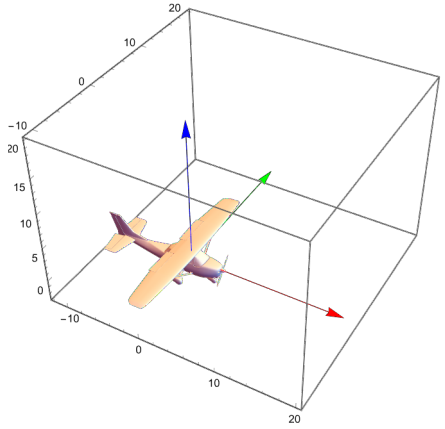


Импорт 3D объекта

Поворот

In[197]:=
```
Graphics3D[{EdgeForm[None], SK,
  Rotate[
   Translate[Scale[object, 0.02], {-145, 0, -128}],
   180 °, {0, 0, 1}
  ]
 }, Axes → True]
```

Out[197]=



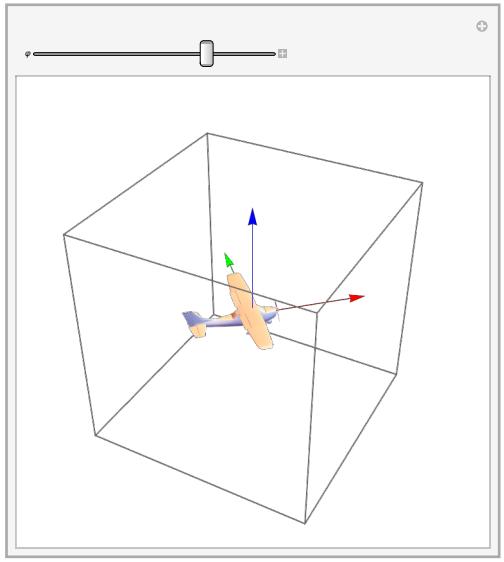
In[198]:=
```
plane = {EdgeForm[None], SK, Rotate[Translate[Scale[object, 0.02], {-145, 0, -128}], 180 °, {0, 0, 1}]};
```

Поворот

Поворот вокруг оси z на угол φ (рыскание) - RotationMatrix[φ,{0,0,1}]

In[•]:=
```
Manipulate[
 Graphics3D[GeometricTransformation[plane, {RotationMatrix[φ, {0, 0, 1}], {0, 0, 0}}], PlotRange → {{-20, 20}, {-20, 20}, {-20, 20}}],
 {φ, -90 °, 90 °}
]
```

Out[•]=



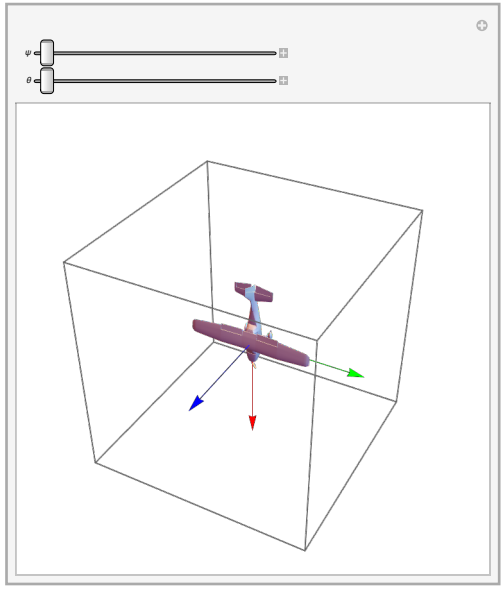
Два поворота

Перемножив матрицы получим сложный поворот по рысканию и крену

In[199]:=
```
A[ψ_, θ_] := RotationMatrix[ψ, {0, 0, 1}].RotationMatrix[-θ, {0, 1, 0}];
```

In[200]:=
```
Manipulate[
 Graphics3D[GeometricTransformation[plane, {A[ψ, θ], {0, 0, 0}}], PlotRange → {{-20, 20}, {-20, 20}, {-20, 20}}],
 {ψ, -90 °, 90 °},
 {θ, -90 °, 90 °}
]
```

Out[200]=



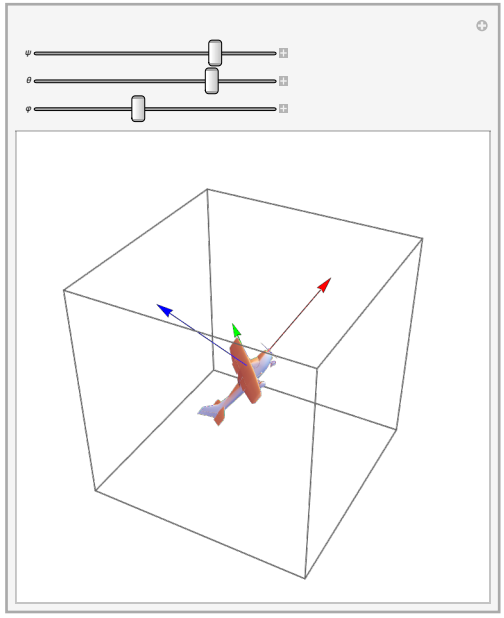
Три поворота

Рыскание, тангаж и крен

In[201]:=
```
A[ψ_, θ_, φ_] := RotationMatrix[ψ, {0, 0, 1}].RotationMatrix[-θ, {0, 1, 0}].RotationMatrix[φ, {1, 0, 0}];
```

In[202]:=
```
Manipulate[
 Graphics3D[GeometricTransformation[plane, {A[ψ, θ, φ], {0, 0, 0}}], PlotRange → {{-20, 20}, {-20, 20}, {-20, 20}}],
 {ψ, -90 °, 90 °},
 {θ, -90 °, 90 °},
 {φ, -90 °, 90 °}
]
```

Out[202]=



Интегрирование дифференциальных уравнений

Положение шарика относительно системы координат O x_1 y_1 z_1, связанной с первым телом

Радиус – вектор точки A относительно системы Ox_1 y_1 z_1

In[203]:=
```
ρA = {-Sqrt[R^2 - h^2], -2 * h};
```

Радиус вектор точки M относительно точки A в системе координат Ox_1 y_1 z_1

In[204]:=
```
ρM = ρA + {l0 + x[t], 0};
```

Положение шарика относительно неподвижной системы координат Oxyz

In[205]:=
```
rM = RotationMatrix[φ[t]].ρM
```

Out[205]=
$$\left\{2\,h\,\text{Sin}[\varphi[t]] + \text{Cos}[\varphi[t]]\left(-\sqrt{-h^2+R^2} + l_0 + x[t]\right),\ -2\,h\,\text{Cos}[\varphi[t]] + \text{Sin}[\varphi[t]]\left(-\sqrt{-h^2+R^2} + l_0 + x[t]\right)\right\}$$

Вектор абсолютной скорости шарика

In[206]:=
```
vM = D[rM, t] // FullSimplify
```

Out[206]=
$$\left\{\text{Cos}[\varphi[t]]\,x'[t] + \left(2\,h\,\text{Cos}[\varphi[t]] + \sqrt{-h^2+R^2}\,\text{Sin}[\varphi[t]] - \text{Sin}[\varphi[t]]\,(l_0 + x[t])\right)\varphi'[t],\right.$$
$$\left.\text{Sin}[\varphi[t]]\,x'[t] + 2\,h\,\text{Sin}[\varphi[t]]\,\varphi'[t] + \text{Cos}[\varphi[t]]\left(-\sqrt{-h^2+R^2} + l_0 + x[t]\right)\varphi'[t]\right\}$$

Кинетическая энергия

Кинетическая энергия шарика

In[207]:=
```
T2 = FullSimplify[m2 vM.vM / 2]
```

Out[207]=
$$\frac{1}{2} m_2 \left(x'[t]^2 + 4\,h\,x'[t]\,\varphi'[t] + \left(3h^2 + R^2 - \left(2\sqrt{-h^2+R^2} - l_0 - x[t]\right)(l_0 + x[t])\right)\varphi'[t]^2\right)$$

Кинетическая энергия первого тела (стержень + сектор)

Стержень длиной h, массой m_{11}, вращающийся вокруг оси O. Момент инерции $m_{11}\,h^2/3$

In[208]:=
```
T11 = (m11 h^2 / 3) * φ'[t]^2 / 2;
```

Сектор, вращающийся вокруг оси O

In[209]:=
```
J12 = 1/2 * (m12 R^2)/2 + m12 h^2;
T12 = (J12 * φ'[t]^2)/2
```

Out[210]=
$$\frac{1}{2}\left(h^2 m_{12} + \frac{R^2 m_{12}}{4}\right)\varphi'[t]^2$$

Кинетическая энергия системы

In[211]:=
```
Tsys = FullSimplify[T11 + T12 + T2]
```

Out[211]=
$$\frac{1}{24}\left(\left(4h^2 m_{11} + 3\left(4h^2 + R^2\right)m_{12}\right)\varphi'[t]^2 + 12\,m_2\left(x'[t]^2 + 4\,h\,x'[t]\,\varphi'[t] + \left(3h^2 + R^2 - \left(2\sqrt{-h^2+R^2} - l_0 - x[t]\right)(l_0 + x[t])\right)\varphi'[t]^2\right)\right)$$

Потенциальная энергия шарика

Нулевой уровень - горизонтальная плоскость, проходящая через ось вращения *O*.

Потенциальная энергия шарика (координата y в неподвижной системе, умноженная на массу шарика и ускорение свободного падения.

In[212]:=
```
Π_g2 = m_2 * g * r_M[[2]]
```

Out[212]=
$$g\, m_2 \left(-2\, h \operatorname{Cos}[\varphi[t]] + \operatorname{Sin}[\varphi[t]] \left(-\sqrt{-h^2 + R^2} + l_0 + x[t]\right)\right)$$

Потенциальная энергия пружины

In[213]:=
```
Π_c2 = c (x[t] - l_0)^2 / 2;
```

Потенциальная энергия тела 1 (пластина)

Положение центра масс стержня и сектора относительно системы координат $O\,x_1\,y_1\,z_1$

In[214]:=
```
ρ1 = (m11 {0, -h/2} + m12 {0, -h - 4 R/(3 π)}) / (m11 + m12);
```

Положение центра масс стержня и сектора относительно системы координат *Oxyz*

In[215]:=
```
r1 = RotationMatrix[φ[t]].ρ1
```

Out[215]=
$$\left\{-\frac{\text{Sin}[\varphi[t]]\left(-\frac{h\,m_{11}}{2}+\left(-h-\frac{4R}{3\pi}\right)m_{12}\right)}{m_{11}+m_{12}},\ \frac{\text{Cos}[\varphi[t]]\left(-\frac{h\,m_{11}}{2}+\left(-h-\frac{4R}{3\pi}\right)m_{12}\right)}{m_{11}+m_{12}}\right\}$$

Координата y этого вектора будет *высотой* центра масс системы пластина + стержень относительно нулевого уровня энергии

In[216]:=
```
Πg1 = (m11 + m12) * g * r1[[2]]
```

Out[216]=
$$g\,\text{Cos}[\varphi[t]]\left(-\frac{h\,m_{11}}{2}+\left(-h-\frac{4R}{3\pi}\right)m_{12}\right)$$

Потенциальная энергия системы

In[217]:=
```
Πsys = Πg2 + Πc2 + Πg1
```

Out[217]=
$$g\,\text{Cos}[\varphi[t]]\left(-\frac{h\,m_{11}}{2}+\left(-h-\frac{4R}{3\pi}\right)m_{12}\right)+\frac{1}{2}c\,(-l_0+x[t])^2+g\,m_2\left(-2h\,\text{Cos}[\varphi[t]]+\text{Sin}[\varphi[t]]\left(-\sqrt{-h^2+R^2}+l_0+x[t]\right)\right)$$

Уравнения движения

Уравнения движения

In[218]:=
```
eq1 = FullSimplify[D[D[T_sys, φ'[t]], t] - D[T_sys, φ[t]] == -D[Π_sys, φ[t]]]
```

Out[218]=
$$m_{12}\left(-4\,g\,(3\,h\,\pi+4\,R)\,\text{Sin}[\varphi[t]]-3\,\pi\left(4\,h^2+R^2\right)\varphi''[t]\right)+12\,\pi\,m_2\left(g\sqrt{-h^2+R^2}\,\text{Cos}[\varphi[t]]-2\,g\,h\,\text{Sin}[\varphi[t]]+2\sqrt{-h^2+R^2}\,x'[t]\,\varphi'[t]-2\,h\,x''[t]-\left(3\,h^2+R^2\right)\varphi''[t]-(l_0+x[t])\left(g\,\text{Cos}[\varphi[t]]+2\,x'[t]\,\varphi'[t]+\left(-2\sqrt{-h^2+R^2}+l_0+x[t]\right)\varphi''[t]\right)\right)==2\,h\,\pi\,m_{11}\,(3\,g\,\text{Sin}[\varphi[t]]+2\,h\,\varphi''[t])$$

In[219]:=
```
eq2 = FullSimplify[D[D[T_sys, x'[t]], t] - D[T_sys, x[t]] == -D[Π_sys, x[t]]]
```

Out[219]=
$$c\,x[t]+m_2\left(g\,\text{Sin}[\varphi[t]]+\left(\sqrt{-h^2+R^2}-l_0-x[t]\right)\varphi'[t]^2+x''[t]+2\,h\,\varphi''[t]\right)==c\,l_0$$

Численное интегрирование

Параметры системы

In[220]:=
```
params = {m11 → 1, m12 → 1, m2 → 0.3, R → 1, l0 → 0.4, g → 9.807, c → 9.0, h → R / 2};
```

Интегрирование

In[221]:=
```
numEq = {eq1, eq2, φ[0] == 0.3, φ'[0] == 0.3, x[0] == 0, x'[0] == 0} //. params
```

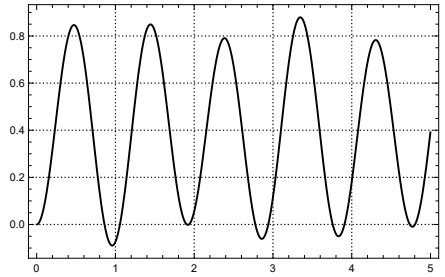
Out[221]=
$$\Big\{-341.77\,\text{Sin}[\varphi[t]] - 6\,\pi\,\varphi''[t] + 11.3097\left(8.49311\,\text{Cos}[\varphi[t]] - 9.807\,\text{Sin}[\varphi[t]] + \sqrt{3}\,x'[t]\,\varphi'[t] - x''[t] - \frac{7\,\varphi''[t]}{4} - (0.4 + x[t])\,(9.807\,\text{Cos}[\varphi[t]] + 2\,x'[t]\,\varphi'[t] + (-1.33205 + x[t])\,\varphi''[t])\right) == \pi\,(29.421\,\text{Sin}[\varphi[t]] + \varphi''[t]),$$
$$9.\,x[t] + 0.3\left(9.807\,\text{Sin}[\varphi[t]] + (0.466025 - x[t])\,\varphi'[t]^2 + x''[t] + \varphi''[t]\right) == 3.6,\ \varphi[0] == 0.3,\ \varphi'[0] == 0.3,\ x[0] == 0,\ x'[0] == 0\Big\}$$

In[222]:=
```
sol = NDSolve[numEq, {φ[t], φ'[t], x[t], x'[t]}, {t, 0, 5}] // Flatten
```

Out[222]=
$$\{\varphi[t] \to \text{InterpolatingFunction}[\langle\text{Domain: }\{\{0.,\,5.\}\}\ \text{Output: scalar}\rangle][t],\ \varphi'[t] \to \text{InterpolatingFunction}[\langle\text{Domain: }\{\{0.,\,5.\}\}\ \text{Output: scalar}\rangle][t],$$
$$x[t] \to \text{InterpolatingFunction}[\langle\text{Domain: }\{\{0.,\,5.\}\}\ \text{Output: scalar}\rangle][t],\ x'[t] \to \text{InterpolatingFunction}[\langle\text{Domain: }\{\{0.,\,5.\}\}\ \text{Output: scalar}\rangle][t]\}$$

In[223]:=
```
Plot[x[t] /. sol, {t, 0, 5}, PlotTheme → {"Monochrome", "FrameGrid"}]
```

Out[223]=

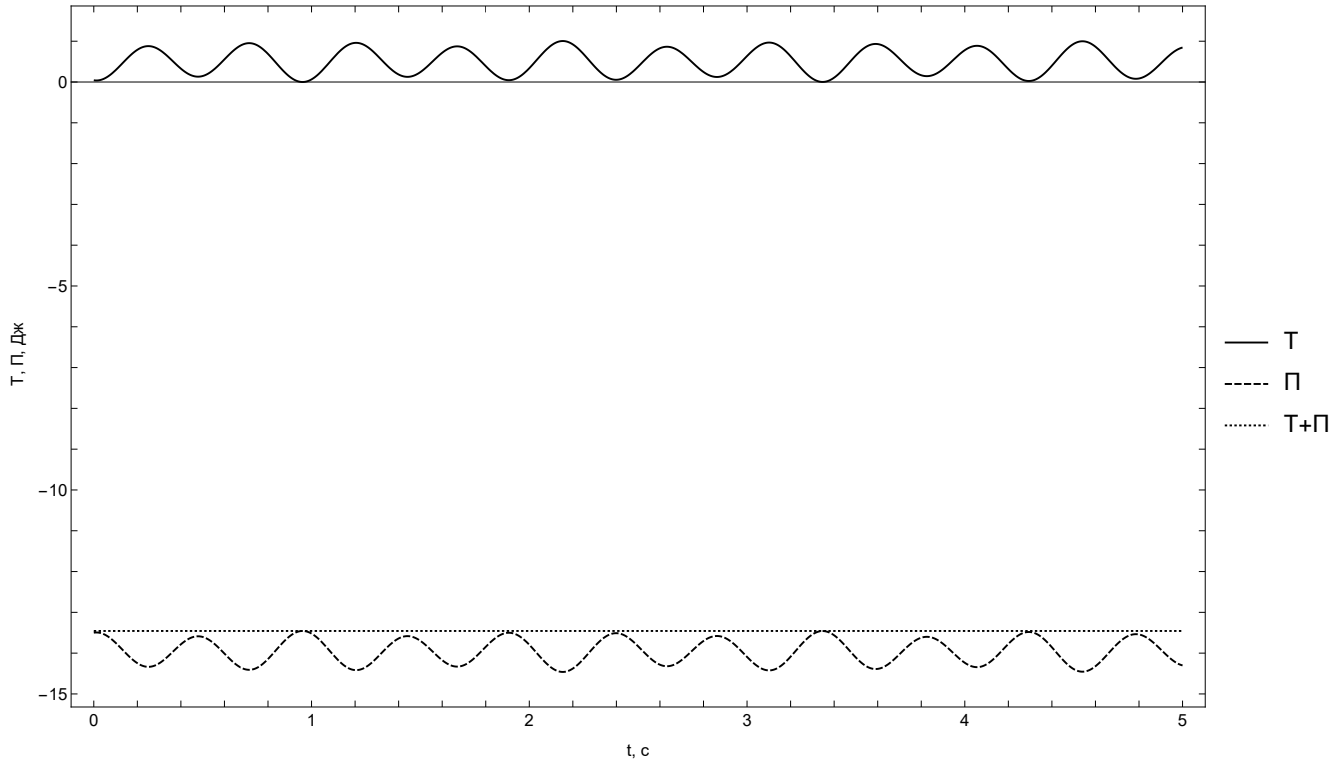


Интеграл энергии

Проверка интеграла энергии

In[224]:=
```
Plot[{T_sys, П_sys, T_sys + П_sys} //. params /. sol // Evaluate, {t, 0.0, 5.0}, Frame → True, BaseStyle → 14,
 FrameLabel → {"t, с", " T, П, Дж"}, PlotLegends → {"T", "П", "T+П"}, PlotTheme → "Monochrome", ImageSize → 1000]
```

Out[224]=

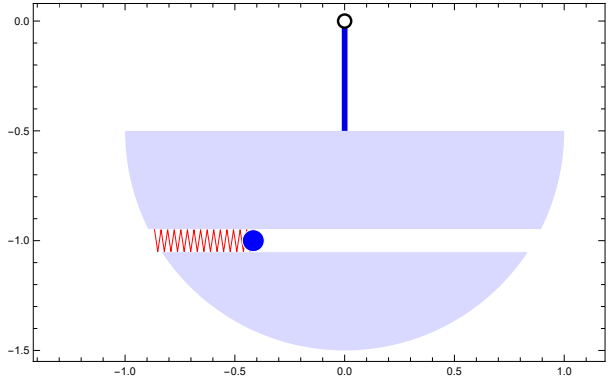


Рисуем систему

Рисуем систему в соответствии с параметрами в списке params

In[234]:=
```
ClearAll[body1]
Spring[L_, d_, n_] := Table[{i * L / n, (-1)^i d / 2}, {i, 0, n}];
body1[φi_, xi_] := Module[{b, rshar = 0.05, dpru = 0.1},
   b = {(* СТЕРЖЕНЬ *)
        Darker[Blue, 0.1], Thickness[0.01], Line[{{0, 0}, {0, -h}}],
        (* ПЛАСТИНА *)
        Lighter[Blue, 0.85], Disk[{0, -h}, R, {-180°, 0°}],
        (* ОПОРА *)
        White, EdgeForm[{Thick, Black}], Disk[{0, 0}, 0.03],
        (* КАНАЛ *)
        EdgeForm[{Thin, White}], Rectangle[ρA + {-0.5, -dpru / 2}, ρA + {2 R, dpru / 2}],
        (* ПРУЖИНА *)
        Red, Thin, Translate[Line[Spring[x[t] + l0, dpru, 30]], ρA],
        (* ШАРИК *)
        Blue, Disk[ρM, rshar]
       } /. {φ[t] → φi, x[t] → xi} //. params;
 GeometricTransformation[b, {RotationMatrix[φi], {0, 0}}]
 ];

Graphics[body1[0, 0.05], Frame → True, ImageSize → 500]
```

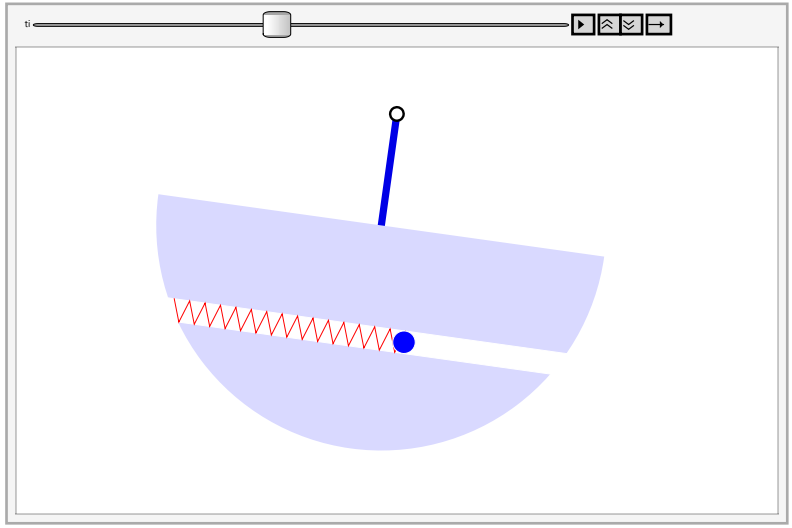
Out[237]=



Анимация

In[228]:=
```
Animate[
 Graphics[
  body1[φ[t], x[t]] /. sol /. t → ti, PlotRange → {{-1.6, 1.6}, {-1.7, 0.2}}, ImageSize → 600
 ],
 {ti, 0, 5, 0.05},
 DisplayAllSteps → True]
```

Out[228]=



Экспорт анимации

Анимацию можно сохранить в файл типа **анимированный gif**

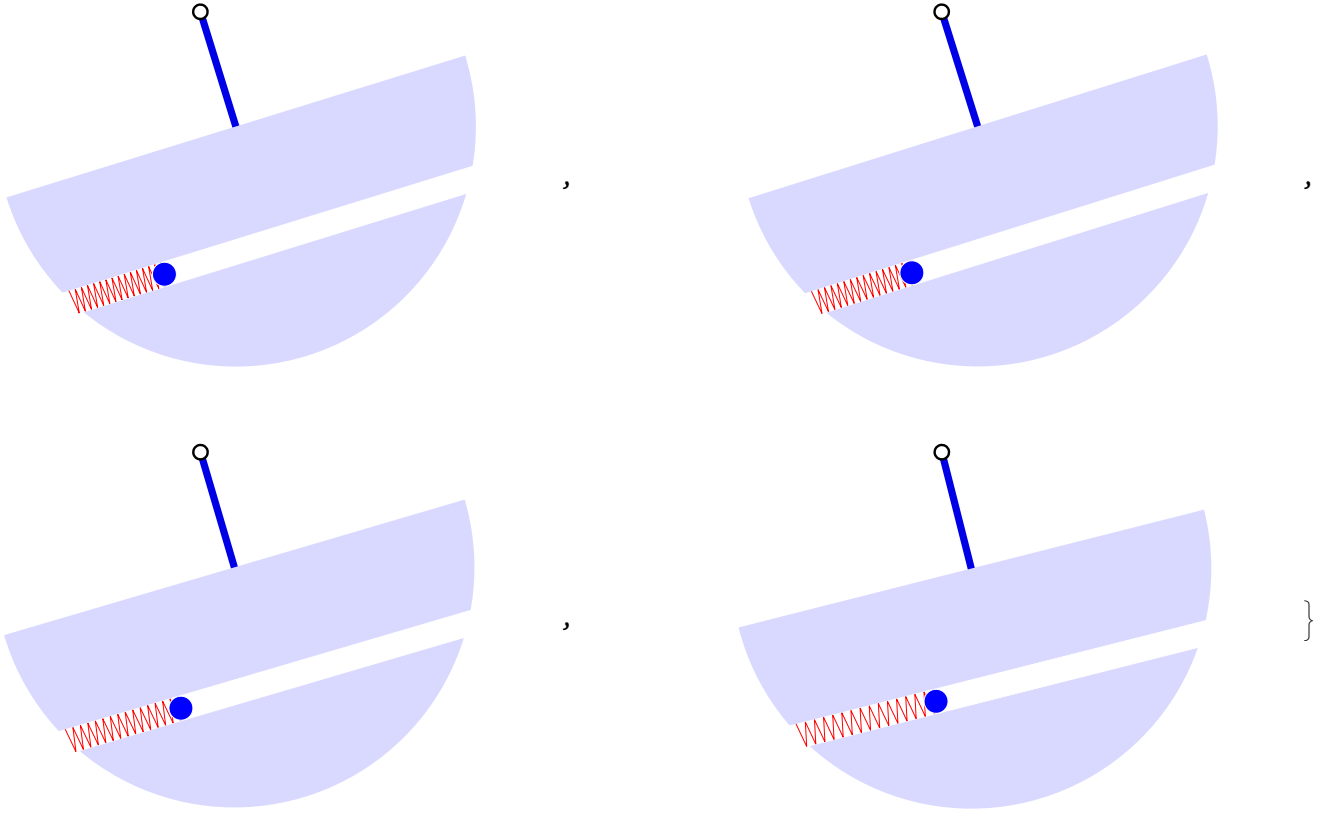
Генерируем кадры: t от 0 до $t_{\max}$ с шагом 0.05 (100 кадров для 5 секунд) при помощи функции **Table**

```
In[•]:= aniTable = Table[
          Graphics[
              body1[φ[t], x[t]] /. sol /. t → ti, PlotRange → {{-1.5, 1.5}, {-1.5, 0.1}}, ImageSize → 600
            ],
          {ti, 0, 5, 0.05}];
```

Первые пять кадров

```
In[•]:= aniTable[[1 ;; 4]]
```

Out[•]=



Экспорт

Для записи кадров в файл используем функцию **Export**

In[•]:=
```
Export["animation.gif", aniTable]
```

Out[•]=
```
$Aborted
```

Расширение имени файла (gif, avi) определит формат файла

In[•]:=
```
Export["animation.avi", aniTable]
```

Out[•]=
```
animation.avi
```

Файл получится большим, т.к. видео не сжато

Экспорт

Чтобы создался файл поменьше попытаемся использовать сжатие (формат MPEG-4)

```
In[•]:= Export["animation.avi", aniTable, "VideoEncoding" → "MPEG-4 Video"]
```

Export: The value MPEG-4 Video specified for the option "VideoEncoding" is invalid.

```
Out[•]= $Failed
```

Список форматов, поддерживаемых системой (Windows)

```
In[•]:= Internal`$VideoEncodings
```

```
Out[•]= {H264, JPEG, HEVC}
```

В некоторых операционных системах сжатие не поддерживается -- только Uncompressed (несжатый).