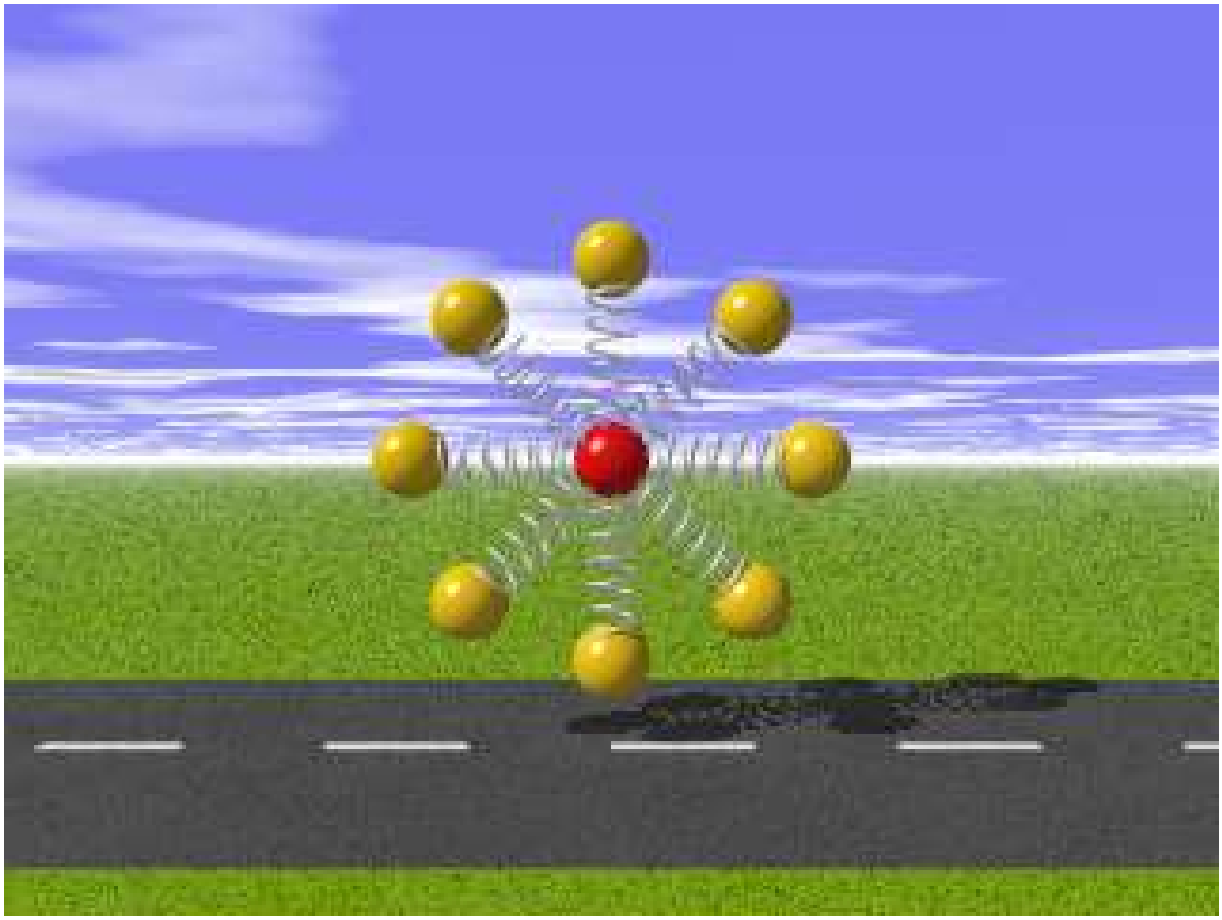
Scéna 3:

Scénou 3 jsem chtěl ukázat nabyté znalosti z výuky v druhém pololetí. Je zaměřená na ukázku animačních možností programu POV-RAY. V této scéně jsou zapracovány obě předchozí scény s úpravou objektů pro zajímavější pohyb v animaci.



Popis objektů:

Jak je vidět na obrázku výše, scéna obsahuje již 9 kuliček, kdy 8 z nich je s pružinovým efektem uspořádáno v kruhovém obrazci. Snahou je rozpohybovat kolo složené z několika pružin po silnici. Při tom ovšem i se zachovaným sinusovým pohybem jednotlivých spirál. Pozadí je využito z předešlé scény jako polojasná obloha se silnicí v travnatém podkladu.

Popis souboru Pruzinka_kolecko.pov:

```
#version 3.6;
global_settings{ assumed_gamma 1.0 }
#default{ finish{ ambient 0.1 diffuse 0.9 conserve_energy}}
#include "colors.inc"
#include "textures.inc"
#include "glass.inc"
#include "metals.inc"
#include "golds.inc"
#include "stones.inc"
#include "woods.inc"
#include "shapes.inc"
#include "shapes2.inc"
#include "functions.inc"
#include "math.inc"
#include "transforms.inc"
// include příkazem opět nahráváme knihovny s využitelnými balíčky pro naše scény
// všechny uvedené v naší scéně momentálně nevyužijeme, ale mám je nahrány pro
// případné další pokračování projektu
```

```
#declare Camera_Number = 2 ;
// v této scéně si deklarujeme více kamer, pro různou volbu záběru scény jednoduchou
// úpravou, a to změnou čísla vyžadované kamery
#declare Rapport = 3.00;
#declare Move = Rapport*clock;
// definujeme zde i pohyb, který využijeme až v následující scéně
#switch ( Camera_Number )
#case (0)
  #declare Camera_Position = < 0.00, 1.00, -5.00> ;
  #declare Camera_Look_At  = < 0.00, 1.00,  0.00> ;
  #declare Camera_Angle    =  65 ;
#break
// definice kamery číslo 0, definujeme pozici, pohled a úhel kamery. V tomto případě jde o
// čelní pohled na scénu
#case (1)
  #declare Camera_Position = < 4.00, 3.00, -6.00> ;
  #declare Camera_Look_At  = < 0.00, 0.00,  0.00> ;
  #declare Camera_Angle    =  45 ;
#break
// definice kamery číslo 1, definujeme pozici, pohled a úhel kamery. V tomto případě jde o
// pohled na scénu úhlopříčně
#case (2)
  #declare Camera_Position = < 10.00, 3.00,  0.00+Move> ;
  #declare Camera_Look_At  = < 0.00, 3.00,  0.00+Move> ;
  #declare Camera_Angle    =  65 ;
#break
// definice kamery číslo 2, definujeme pozici, pohled a úhel kamery. V tomto případě jde o
// pohled na scénu z pravé strany, lze vidět na obrázku výše, tato kamera je užita ve scéně
// POZOR, již zde využíváme dříve definovanou proměnnou pro pohyb
```

```
#case (3)
  #declare Camera_Position = < 0.00,10.00, -0.001> ;
  #declare Camera_Look_At  = < 0.00, 1.00,  0.00 > ;
  #declare Camera_Angle    =  65 ;
#break
// definice kamery číslo 3, definujeme pozici, pohled a úhel kamery. V tomto případě jde o
// horní pohled na scénu
#else
  #declare Camera_Position = < 0.00, 1.00, -5.00> ;
  #declare Camera_Look_At  = < 0.00, 1.00,  0.00> ;
  #declare Camera_Angle    =  65 ;
#break
// není-li uvedena požadovaná kamera. V tomto případě jde o čelní pohled na scénu
// v podstatě jde o totožnou kameru jako v případě ´´0´´
 #end
// konec těla definice kamer v podobě ´´SWITCH´´
camera{ location Camera_Position
        right    x*image_width/image_height
        angle    Camera_Angle
        look_at  Camera_Look_At}
// vyvolání objektu kamery
light_source{<1500,2500,-2500> color White*0.9}
// simulace slunce v podobě definice světelného toku
light_source{ Camera_Position  color rgb<0.9,0.9,1>*0.1 shadowless }
// další světelný tok vycházející z polohy kamery
```

```
plane{<0,1,0>,1 hollow
      texture{ pigment{ bozo turbulence 0.92
                  color_map { [0.00 rgb <0.20, 0.20, 1.0>*0.9]
                              [0.50 rgb <0.20, 0.20, 1.0>*0.9]
                              [0.70 rgb <1,1,1>]
                              [0.85 rgb <0.25,0.25,0.25>]
                              [1.0 rgb <0.5,0.5,0.5>]}
                  scale<1,1,1.5>*2.5  translate< 0,0,0>}
            finish {ambient 1 diffuse 0} }
      scale 10000}
// definice oblohy s pigmentem, barevností a měřítkem, původně byl plánován i pohyb
// oblohy, kde by společně s pohybem mraků prosvítalo slunce. Bohužel se mi nepodařilo
// tento pohyb vyladit, tak aby nebyla výsledná animace sekavá
fog { fog_type   2
     distance   500
     color      White
     fog_offset 0.1
     fog_alt    3.5
     turbulence 1.8}
// vytváříme tím mlžný opar v místě přechodu oblohy se zemí
plane { <0,1,0>, 0
       texture{ pigment{ color rgb<0.35,0.65,0.0>*0.72 }
                   normal { bumps 0.75 scale 0.015 }
              finish { phong 0.1 }}}
// zde definujeme podobu země samotné
union{
box{ <-3.00, 0.00,-500>,< 3.00, 0.0005, 500>
     texture{ pigment{ color rgb<1,1,1>*0.1}
              normal { bumps 0.5 scale 0.005}
              finish { phong 0.5}}}
 // definujeme podobu silnice-asfaltu
```

```
union{
#local Nr = -500;
#local EndNr = 500;
// počáteční a koncová pozice
#while (Nr< EndNr)
box{ <-0.1, 0.00, 0>,< 0.1, 0.0015, 1.50>
    texture{ pigment{ color rgb<1,1,1>*1.1}
          finish { phong 0.5}}
    translate<0,0,Nr*3.00>}
#local Nr = Nr + 1;
#end } // konec těla definující podobu asfaltové silnice s dělicími pruhy
rotate<0,0,0>
translate<0,0,0-Move>}
// POZOR, opět nově využíváme definovanou proměnnou pro pohyb silnice
#declare Amplitude = 0.60 ;
#declare Minimal_Length   = 0.80 ;
#declare Middle_Length    = Amplitude + Minimal_Length ;
// ve scéně máme 8 kuliček a pružin, pro každou proto nastavíme i časově sinusový průběh
// budeme-li chtít současný pohyb kuliček, tak bude stačit nastavit pouze jednu časovou
// proměnnou hodnotou za ´´Time_testX´´  nastavíme napnutí pružiny a sinusovou polohu
// kuličky v našem případě lze užít pouze hodnotu od 0.25 do 0.75
#declare Time_test1 = 0.25;
#declare Sp_Length1 = Middle_Length+Amplitude*sin((clock+Time_test1)*2*pi);
#declare Time_test2 = 0.25;
#declare Sp_Length2 = Middle_Length+Amplitude*sin((clock+Time_test2)*2*pi);
#declare Time_test3 = 0.25;
#declare Sp_Length3 = Middle_Length+Amplitude*sin((clock+Time_test3)*2*pi);
#declare Time_test4 = 0.25;
#declare Sp_Length4 = Middle_Length+Amplitude*sin((clock+Time_test4)*2*pi);
#declare Time_test5 = 0.25;
#declare Sp_Length5 = Middle_Length+Amplitude*sin((clock+Time_test5)*2*pi);
```

```
#declare Time_test6 = 0.25;
#declare Sp_Length6 = Middle_Length+Amplitude*sin((clock+Time_test6)*2*pi);
#declare Time_test7 = 0.25;
#declare Sp_Length7 = Middle_Length+Amplitude*sin((clock+Time_test7)*2*pi);
#declare Time_test8 = 0.25;
#declare Sp_Length8 = Middle_Length+Amplitude*sin((clock+Time_test8)*2*pi);
// máme sice 8 kuliček a jejich pružinek, ale využití najde pouze hodnota ´´0.25´´, ostatní
// jsem zde ponechal pro další případné variace
// následuje deklarace podoby a základní polohy samotných pružin a jejich kuliček
#declare Spiral1  =
union{
 #local N_per_Rev = 500;
 // k určení počtu prvků na jednu otáčku
 #local N_of_Rev  = 8.00;
// k určení počtu spirál, otáček na pružině
 #local H_per_Ref = Sp_Length1 / N_of_Rev;
// definice lokálního výpočtu délky jedné otáčky
 #local Nr = 0;
 // počáteční smyčka pružiny
 #local Wheel_Radius = 1.00;
// definice lokálního rádiusu pro kruhové rozestavění
 #while (Nr< N_per_Rev*N_of_Rev)
   sphere{ <0,0,0>,0.025
// poloha a průměr pružiny
        translate<0.25, -Nr*H_per_Ref/N_per_Rev, 0>
        rotate<0,  Nr * 360/N_per_Rev,0>
        scale Wheel_Radius rotate  0*x
// k určení pokračující polohy a POZOR rotace s využitím natočení pružiny na ose ´´X´´
```

```
 texture{ Chrome_Metal
                finish { phong 1}}}
// grafická podoba pružinky
 #local Nr = Nr + 2;
// k zajištění sinusového pohybu, výše máme uveden příkaz ´´WHILE´´, tedy dělej, dokud
// nenastane v našem případě se bude pružinka s kuličkou natahovat či smršťovat tak
// dlouho dokud nenarazí na svou maximální či minimální polohu definovanou
// ´´Time_testX´´
 #end
// konec ´´WHILE´´  první spirálové pružinky

sphere { <0,0,0>, 0.4
        translate<0,-Nr*H_per_Ref/N_per_Rev-0.2,0>
        scale Wheel_Radius rotate  0*x
// k určení pokračující polohy a POZOR rotace s využitím natočení pružiny na ose ´´X´´

 texture{ pigment{ color rgb<1,0.65,0>}
              finish { phong 1}}
// k určení umístění, následného pohybu a samotné podoby přidružené kuličky
      }}

// konec těla celé definice spirály s navazující pružinou
// jelikož se všechny zobrazené spirály se liší pouze nastavením sinusového pohybu, popis
// níže vypsaných deklarací je totožný s první deklarací spirály, změnou je pouze volání jiné
// deklarace sinusového pohybu
```

```
#declare Spiral2  =
union{
 #local N_per_Rev = 500;
 #local N_of_Rev  = 8.00;
 #local H_per_Ref = Sp_Length2 / N_of_Rev;
 #local Nr = 0;
 #local Wheel_Radius = 1.00;
 #while (Nr< N_per_Rev*N_of_Rev)
   sphere{ <0,0,0>,0.025
         translate<0.25, -Nr*H_per_Ref/N_per_Rev, 0>
         rotate<0,  Nr * 360/N_per_Rev,0>
         scale Wheel_Radius rotate  45*x
// zde již vidíme natočení na ose ´´X´´ o 45 stupňů
         texture{ Chrome_Metal
                  finish { phong 1}}}
 #local Nr = Nr + 2;
 #end

  sphere { <0,0,0>, 0.4
        translate<0,-Nr*H_per_Ref/N_per_Rev-0.2,0>
        scale Wheel_Radius rotate  45*x
// zde také vidíme natočení na ose ´´X´´ o 45 stupňů
        texture{ pigment{ color rgb<1,0.65,0>}
                 finish { phong 1}}
       }}
```

```
#declare Spiral3  =
union{
 #local N_per_Rev = 500;
 #local N_of_Rev  = 8.00;
 #local H_per_Ref = Sp_Length3 / N_of_Rev;
 #local Nr = 0;
 #local Wheel_Radius = 1.00;
 #while (Nr< N_per_Rev*N_of_Rev)
   sphere{ <0,0,0>,0.025
          translate<0.25, -Nr*H_per_Ref/N_per_Rev, 0>
          rotate<0,  Nr * 360/N_per_Rev,0>
          scale Wheel_Radius rotate  90*x
          texture{ Chrome_Metal
                 finish { phong 1}}}
 #local Nr = Nr + 2;
 #end
  sphere { <0,0,0>, 0.4
         translate<0,-Nr*H_per_Ref/N_per_Rev-0.2,0>
         scale Wheel_Radius rotate  90*x
         texture{ pigment{ color rgb<1,0.65,0>}
                finish { phong 1}}
       }}
#declare Spiral4  =
union{
 #local N_per_Rev = 500;
 #local N_of_Rev  = 8.00;
 #local H_per_Ref = Sp_Length4 / N_of_Rev;
 #local Nr = 0;
 #local Wheel_Radius = 1.00;
```

```
#while (Nr< N_per_Rev*N_of_Rev)
  sphere{ <0,0,0>,0.025
        translate<0.25, -Nr*H_per_Ref/N_per_Rev, 0>
        rotate<0,  Nr * 360/N_per_Rev,0>
        scale Wheel_Radius rotate  135*x
        texture{ Chrome_Metal
              finish { phong 1}}}
 #local Nr = Nr + 2;
 #end
  sphere { <0,0,0>, 0.4
       translate<0,-Nr*H_per_Ref/N_per_Rev-0.2,0>
       scale Wheel_Radius rotate  135*x
       texture{ pigment{ color rgb<1,0.65,0>}
             finish { phong 1}}
      }}
#declare Spiral5  =
union{
 #local N_per_Rev = 500;
 #local N_of_Rev  = 8.00;
 #local H_per_Ref = Sp_Length5 / N_of_Rev;
 #local Nr = 0;
 #local Wheel_Radius = 1.00;
 #while (Nr< N_per_Rev*N_of_Rev)
 sphere{ <0,0,0>,0.025
        translate<0.25, -Nr*H_per_Ref/N_per_Rev, 0>
        rotate<0,  Nr * 360/N_per_Rev,0>
        scale Wheel_Radius rotate  180*x
        texture{ Chrome_Metal
              finish { phong 1}}}
 #local Nr = Nr + 2;
 #end
```

```
 sphere { <0,0,0>, 0.4
        translate<0,-Nr*H_per_Ref/N_per_Rev-0.2,0>
        scale Wheel_Radius rotate  180*x
        texture{ pigment{ color rgb<1,0.65,0>}
                finish { phong 1}}
       }}
#declare Spiral6  =
union{
 #local N_per_Rev = 500;
 #local N_of_Rev  = 8.00;
 #local H_per_Ref = Sp_Length6 / N_of_Rev;
 #local Nr = 0;
 #local Wheel_Radius = 1.00;
 #while (Nr< N_per_Rev*N_of_Rev)
 sphere{ <0,0,0>,0.025
         translate<0.25, -Nr*H_per_Ref/N_per_Rev, 0>
         rotate<0,  Nr * 360/N_per_Rev,0>
         scale Wheel_Radius rotate  225*x
         texture{ Chrome_Metal
                 finish { phong 1}}}
 #local Nr = Nr + 2;
 #end
  sphere { <0,0,0>, 0.4
        translate<0,-Nr*H_per_Ref/N_per_Rev-0.2,0>
        scale Wheel_Radius rotate  225*x
        texture{ pigment{ color rgb<1,0.65,0>}
                finish { phong 1}}
       }}
```

```
#declare Spiral7  =
union{
 #local N_per_Rev = 500;
 #local N_of_Rev  = 8.00;
 #local H_per_Ref = Sp_Length7 / N_of_Rev;
 #local Nr = 0;
 #local Wheel_Radius = 1.00;
 #while (Nr< N_per_Rev*N_of_Rev)
 sphere{ <0,0,0>,0.025
         translate<0.25, -Nr*H_per_Ref/N_per_Rev, 0>
         rotate<0,  Nr * 360/N_per_Rev,0>
         scale Wheel_Radius rotate  270*x
         texture{ Chrome_Metal
                  finish { phong 1}}}
 #local Nr = Nr + 2;
 #end
  sphere { <0,0,0>, 0.4
        translate<0,-Nr*H_per_Ref/N_per_Rev-0.2,0>
        scale Wheel_Radius rotate  270*x
        texture{ pigment{ color rgb<1,0.65,0>}
                 finish { phong 1}}
       }}
#declare Spiral8  =
union{
 #local N_per_Rev = 500;
 #local N_of_Rev  = 8.00;
 #local H_per_Ref = Sp_Length8 / N_of_Rev;
 #local Nr = 0;
 #local Wheel_Radius = 1.00;
```

```
#while (Nr< N_per_Rev*N_of_Rev)
 sphere{ <0,0,0>,0.025
        translate<0.25, -Nr*H_per_Ref/N_per_Rev, 0>
        rotate<0,  Nr * 360/N_per_Rev,0>
        scale Wheel_Radius rotate  315*x
        texture{ Chrome_Metal
              finish { phong 1}}}
 #local Nr = Nr + 2;
 #end
  sphere { <0,0,0>, 0.4
       translate<0,-Nr*H_per_Ref/N_per_Rev-0.2,0>
       scale Wheel_Radius rotate  315*x
       texture{ pigment{ color rgb<1,0.65,0>}
             finish { phong 1}}
      }}
// konec popisu a definice objektů pro naši scénu
sphere { <0,3,0>, 0.4
translate <0,0,0+Move>
       texture{ pigment{ color rgb<1,0,0>}
        finish { phong 1}}}
// do místa, kde se nám střetávají pružinky umístíme červenou kuličku
object { Spiral1  rotate <45*clock,0,0>
// vyvoláním objektu říkáme, že se má pohybovat okolo osy ´´X´´ o 45 stupňů tedy o rozdíl
// nastavení polohy natočení pružin
      translate <0,3,0+Move> }
// celý kolotoč tímto nutíme pohybovat se po ose  ´´Z´´, tím vytváříme dojem, že se
// koukáme na kolo pohybující se společně s kamerou po silnici
```

```
object { Spiral2  rotate <45*clock,0,0>
      translate <0,3,0+Move> }
object { Spiral3  rotate <45*clock,0,0>
      translate <0,3,0+Move> }
object { Spiral4  rotate <45*clock,0,0>
      translate <0,3,0+Move> }
object { Spiral5 rotate <45*clock,0,0>
      translate <0,3,0+Move> }
object { Spiral6  rotate <45*clock,0,0>
      translate <0,3,0+Move> }
object { Spiral7  rotate <45*clock,0,0>
      translate <0,3,0+Move> }
object { Spiral8  rotate <45*clock,0,0>
      translate <0,3,0+Move> }
// nakonec si všechny objekty vyvoláme společně s určením jejich pozice na scéně a jejím
// pohybem po scéně
```

Popis Pruzinka_kolecko.ini:

```
Antialias=On
Antialias_Threshold=0.5
Antialias_Depth=5
// vyhlazovací technika snímkování
Input_File_Name=Pruzinka_kolecko.pov
// vstupní soubor k snímkování
Initial_Frame=1
Final_Frame=45
// zde dáváme programu vědět kolik si přejeme snímků, v našem případě 45 snímků
Initial_Clock=0
Final_Clock=1
// definice proměnné k ustanovení časového formátu
```

Cyclic_Animation=on

// požadavek na cyklickou animaci

Pause_When_Done=off

// ukončit máme-li vyrendrenováno