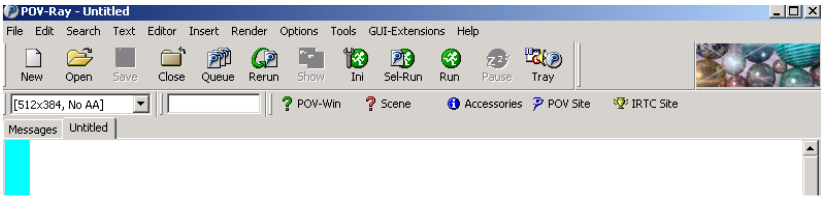
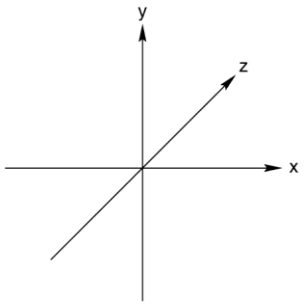
Persistence of Vision Raytracer – krótkie wprowadzenie dla WFiIS SPN

POV-Ray jest programem umożliwiającym programowe tworzenie grafiki fotorealistycznej, działającym w oparciu o metodę śledzenia promieni (*ray tracing*), rozwijanym w modelu otwartego oprogramowania (*open source*): http://www.povray.org/



Podstawowym zadaniem użytkownika jest opracowanie opisu sceny, umieszczonej w wirtualnej przestrzeni 3-wymiarowej. Na bazie tego modelu moduł renderujący programu POV-Ray tworzy plik lub sekwencję plików grafiki rastrowej, zawierające wizualizację przygotowanej kompozycji sceny.

Podstawą opisu modelu sceny jest lewoskrętny układ współrzędnych **<x, y, z>**:



W tak zdefiniowanym układzie współrzędnych należy umieścić podstawowy zestaw elementów:

- **kamerę** – punkt obserwacji, z którego wykonywana jest „fotografia" wirtualnego świata (z uwzględnieniem położenia, kierunku, rozmiaru kąta pola obserwacji, itp.);
- **źródła światła** – obiekty, przez które przechodzące promienie zapewniają oświetlenie tj. jasność i barwę pikseli obrazu (położenie, kształt, barwa, intensywność i kierunek oświetlania);
- **obiekty geometryczne** – elementy, z których skonstruowana jest wirtualna scena (kształt, położenie, właściwości powierzchni determinujące sposób odbijania promieni, a w konsekwencji fakturę i barwę, właściwości optyczne dla obiektów przezroczystych); pojęciem opisującym właściwości powierzchni za pomocą parametrów ilościowych jest tzw. **tekstura** – programowa symulacja rozpraszania światła na powierzchni.

Po uruchomieniu programu tworzymy plik definiujący scenę o nazwie z rozszerzeniem ***.POV**

1. Na początku pliku umieszczamy polecenia ładujące zdefiniowane, dostępne w programie zestawy tekstur i barw, za pomocą polecenia **#include "*plik_konfiguracyjny.inc*"** :

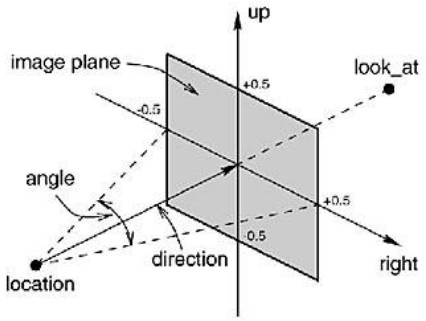
```
#include "colors.inc"
#include "textures.inc"
```

2. Następnie umieszczamy dyrektywy globalnych poleceń konfiguracyjnych
   global_settings { *parametr wartość*} :

```
global_settings {
 assumed_gamma 1.0
 max_intersections 64
 max_trace_level 10
}
```

3. Kolejnym elementem jest definicja parametrów kamery, czyli punktu obserwacji:

```
camera {
 location  <30.0, 90, -120.0>          // położenie <x, y, z> kamery
 look_at   <0.0, 0.0,  0.0>            // punkt na  centralnej osi obserwacji kamery
 angle 20                              // kąt rozwarcia obserwacji kamery w stopniach
}
```



The perspective camera.

4. Każda scena wymaga zdefiniowania źródeł oświetlenia:

```
light_source { <0, 0, 0> color rgb <1, 1, 1>  translate <-30, 30, 0>}
light_source { <0, 0, 0> color rgb <1, 1, 1>  translate <0, 30, -30>}

//  gdzie:          <x, y, z>  - współrzędne początkowego położenia źródła światła
//          color rgb<r, g, b>  - intensywność składowych barwy światła od 0 (czerń) do 1 (biel)
//          translate <dx, dy, dz>  - wektor przesunięcia położenia początkowego
```

5. Konstruujemy kompozycję sceny, ustawiając w niej elementy geometryczne. Kolejność ich umieszczenia w pliku ma wpływ na hierarchię ich wzajemnego położenia. Poszczególne obiekty składają się z deklaracji nazwy i typu geometrii obiektu:
   #declare nazwa_obiektu=typ_geometrii
   oraz ciała (ograniczonego nawiasami klamrowymi **{ }**), zawierającego definicję atrybutów obiektu (położenie, właściwości tekstury powierzchni albo barwy), lub jego elementów składowych. Deklaracje obiektów nie powodują jeszcze ich umieszczenia w modelu sceny.

```
#declare slonce=sphere
{
    <0, 0, 0>, 1          // <x, y, z> - położenie początkowe środka, r – promień sfery
    texture { pigment { color Yellow} finish { ambient 1} }      // atrybuty tekstury
}

#declare ziemia=sphere
{
    <0, 0, 0>, 5
    pigment { checker Blue, Yellow scale 5 }    // atrybuty barwy: szachownica z 2  barw
}

#declare ksiezyc=sphere
{
    <0, 0, 0>, 1
    pigment { color Red }  // prosta barwa powierzchni, nazwa predefiniowana (z dużej litery)
}
```

6. Z wcześniej zdefiniowanych obiektów można tworzyć obiekty złożone, a następnie nadawać im nowe właściwości. Przykładem może być parametryczne definiowanie położenia obiektu. Jeżeli w dalszej części projektu parametr np. **clock** zostanie powiązany z numerem obrazu, możliwe będzie wygenerowanie sekwencji obrazów stanowiących poszczególne klatki animacji.

```
#declare ziemia_ksiezyc=union
{
   object { ziemia                    rotate y*360*365*clock } // przemieszczenia zależne od czasu
   object { ksiezyc  translate x*10  rotate y*360*365*-clock } // parametr clock jest obliczany
}
```

7. Na koniec, umieszczamy wcześniej zdefiniowane elementy, jako obiekty w modelu sceny. Służy temu dyrektywa **object { *nazwa_obiektu dodatkowe_atrybuty*}** :

```
object { slonce }
object { ziemia_ksiezyc   translate x*20     rotate y*360*clock }
```

8. Dodatkowe atrybuty sceny np. wygląd domyślnego tła (zamiast czarnej czeluści):

```
background { color Gray50 }
```

Aby zdefiniować parametry określające zakres i przebieg procesu tworzenia klatek animacji, tworzymy pomocniczy plik konfiguracyjny o nazwie z rozszerzeniem *.INI

```
; Persistence Of Vision raytracer version sample file.
+W480 +H360
Antialias=off
Antialias_Threshold=0.3
```

Antialias_Depth=3
Input_File_Name=Planety.pov
Output_File_Type=C
Output_File_Name= Planety
Initial_Frame=1
Final_Frame=12
Initial_Clock=0
Final_Clock=1
Cyclic_Animation=on
Pause_when_Done=off

Najważniejsze użyte atrybuty to:

- **+W**XXX **+H**YYY – rozdzielczość generowanego obrazu w pikselach XXX, YYY (domyślny stosunek boków /*aspect ratio*/ wynosi 4:3);
- Input_File_Name – nazwa pliku z opisem modelu, do którego odnosi się plik konfiguracyjny;
- Output_File_Type – format plików rastrowych, w których umieszczone zostaną obrazy wygenerowanych klatek animacji (C – ozn. format TARGA);
- Output_File_Name – prefiks nazwy plików rastrowych, automatycznie uzupełniany numerem kolejnej klatki;
- Initial_Frame, Final_Frame – odpowiednio numer pierwszej i ostatniej klatki animacji;
- Initial_Clock , Final_Clock – odpowiednio początkowa i końcowa wartość parametru **clock**, występująca w pliku definicji sceny; wartości pośrednie są interpolowane wg ilości klatek.

Uruchomienie procesu renderingu (generowania klatek) realizujemy za pomocą przycisku **„Run”** na pasku ikon programu POV-Ray. W zależności od tego czy bieżącym aktywnym dokumentem jest plik typu POV, czy typu INI, generowana jest początkowa klatka, lub cała sekwencja klatek animacji.

PLANETY.POV

```
/* Plik opisu sceny programu POV-Ray */
#include "colors.inc"
#include "textures.inc"

global_settings {
  assumed_gamma 1.0
  ambient_light <1.0,1.0,1.0>
  max_intersections 64
  max_trace_level 10
}

camera {
  location  <30.0, 90, -120.0>
  look_at   <0.0, 0.0,  0.0>
  angle 20

}
```

```
light_source { <0, 0, 0> color rgb <1, 1, 1>  translate <-30, 30, 0>}
light_source { <0, 0, 0> color rgb <1, 1, 1>  translate <0, 30, -30>}

background { color Gray50 }

#declare slonce=sphere
{
      <0,0,0>,1
      texture { pigment { color Yellow} finish { ambient 1} }
}

#declare ziemia=sphere
{
      <0,0,0>,5
      pigment { checker Blue, Yellow scale 5 }

}
#declare ksiezyc=sphere
{
      <0,0,0>,1
      pigment { color Red }
}

#declare ziemia_ksiezyc=union
{
      object { ziemia                          rotate y*360*365*clock }
      object { ksiezyc translate x*10       rotate y*360*365*-clock }
}

object { slonce }
object { ziemia_ksiezyc   translate x*20   rotate y*360*clock }

// dodatkowy obiekt – prostopadłościan, służy jako tło dla cieni (należy usunąć komentarze /* i */ )
/*
box {
   <-20, -40, 30>,                 // Near lower left corner
   < 20, 40,  31>                  // Far upper right corner
   pigment { color Green }
}
*/
```

PLANETY.INI

```
; Plik konfiguracji animacji programu POV-Ray
+W480 +H360
Antialias=off
Antialias_Threshold=0.3
Antialias_Depth=3
Input_File_Name=Planety.pov
Output_File_Type = C
Output_File_Name= Planety
```

Initial_Frame=1
Final_Frame=12
Initial_Clock=0
Final_Clock=1
Cyclic_Animation=on
Pause_when_Done=off