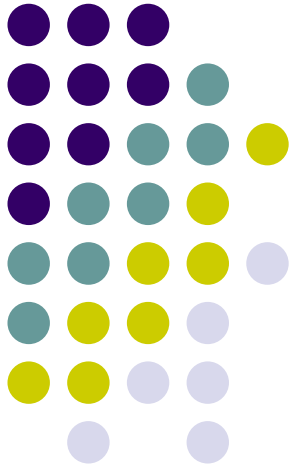
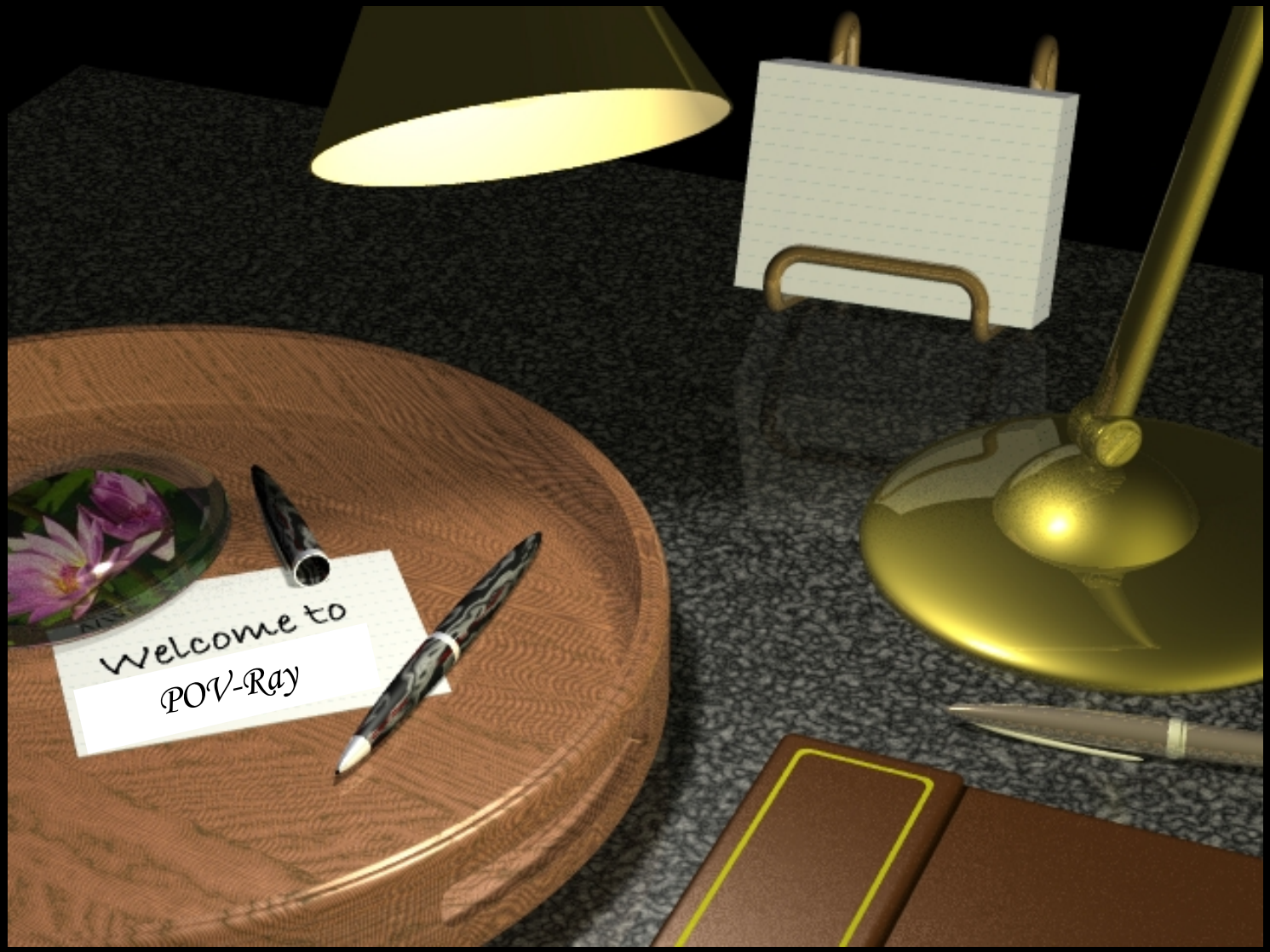
Комп’ютерна графіка

Лекція :

Візуальний аналіз і

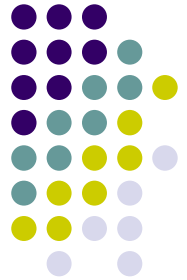
Введення в POV-Ray





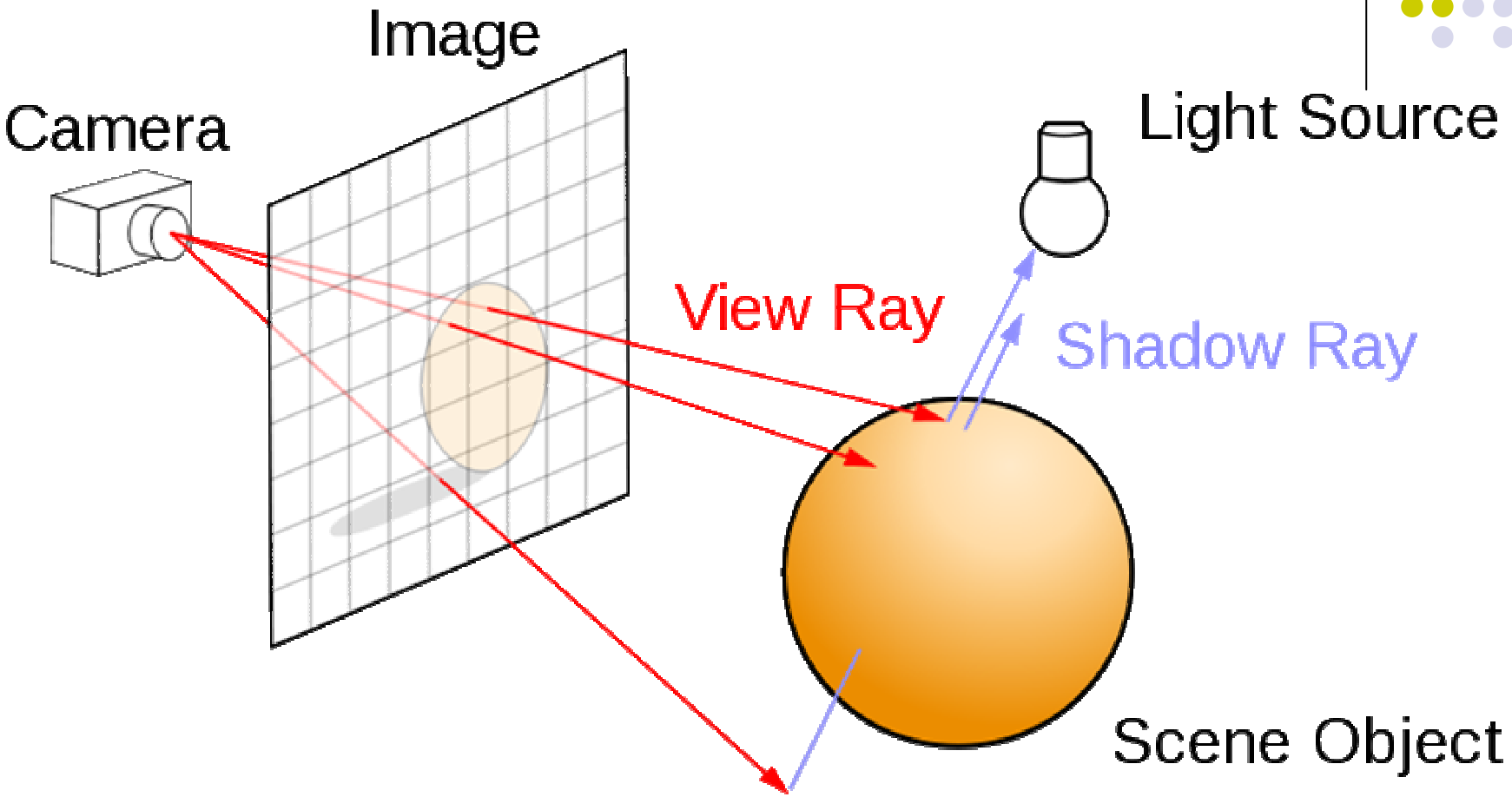
Болбот Ігор Михайлович

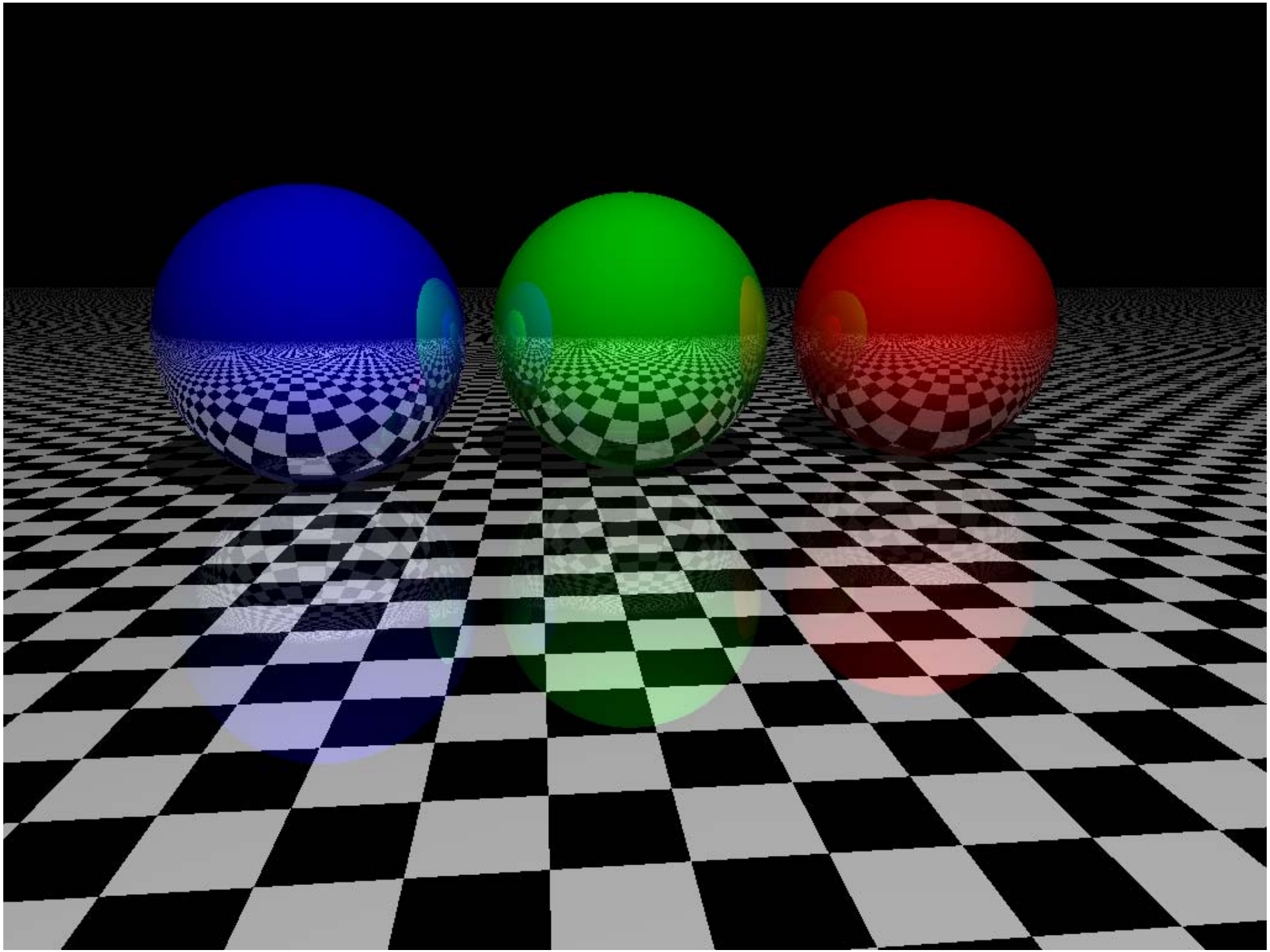
POV-Ray



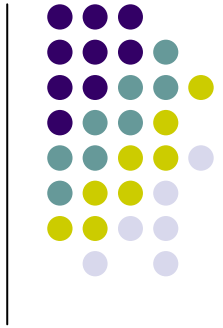
- Програма створює тривимірні, фото-реалістичниє зображення, використовуючи техніку, яка називається "трасування променів ". Вона читає текстовий файл, що містить інформацію з описом об'єктів і освітлення в сцені, і створює образ цією сценою за допомогою камери, також описуючи в текстовому файлі. Трасування променів не дуже швидкий процес в порівнянні з іншими засобами, але він дає дуже високу якість зображення з реалістичними віддзеркаленнями, штрихуванням, перспективою і іншими ефекти.

POV-Ray



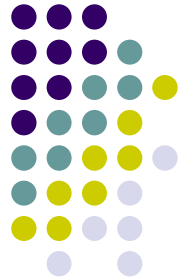


Мудрі слова

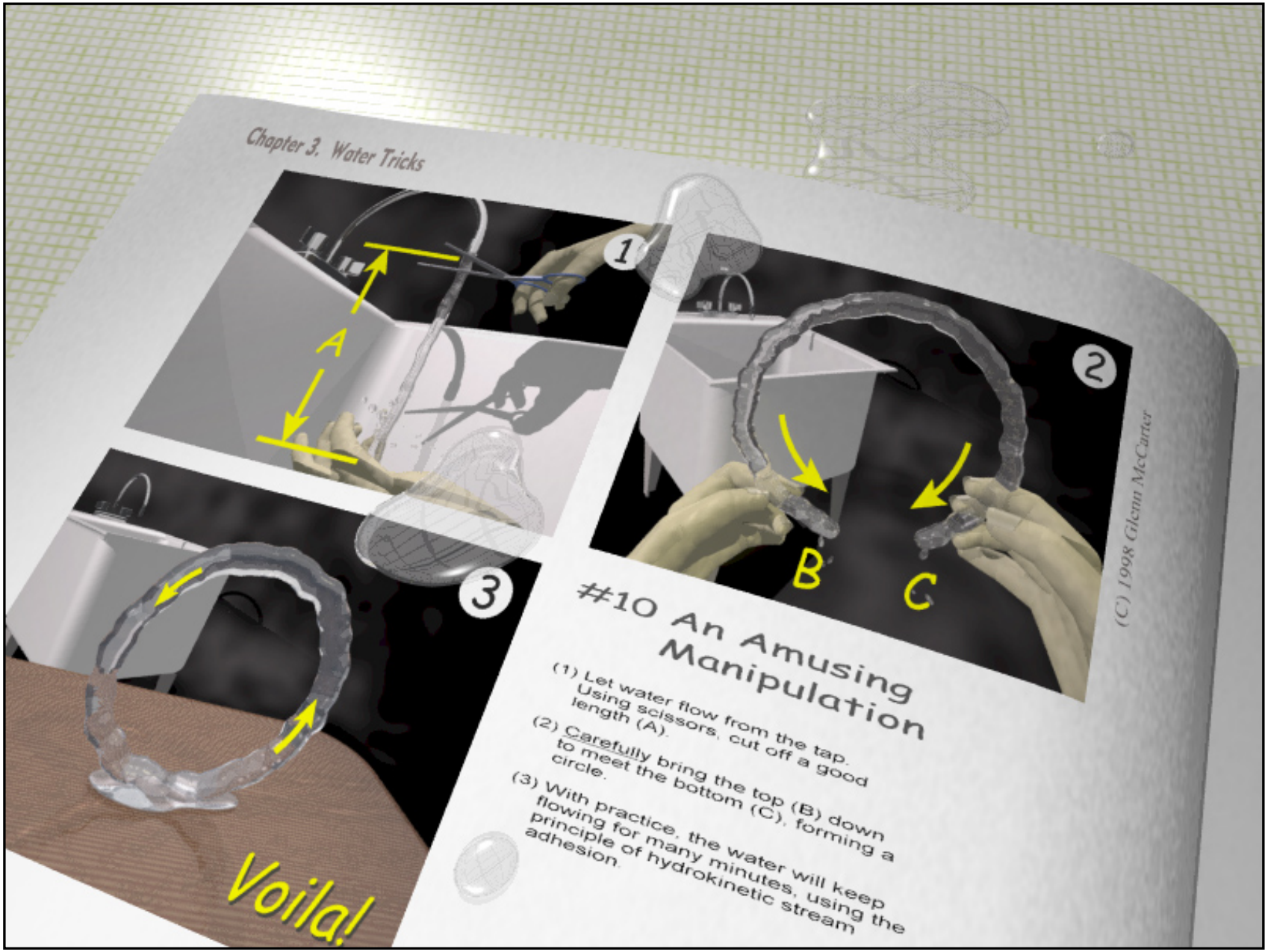


- **Кевін Грегер сказав: „Уолт Дісней це майбутнє анімації”.**
- **На Вашу думку цей вислів це результат появи промислової комп'ютерної графік?**
  - Відсутність компетентного таланту, наявність великої кількості претендентів, багато хто володіє знаннями для роботи з спеціальним програмним забезпеченням, але відносно небагато, хто володіє естетичними і концептуальними навичками, відтворювати реалістичні об'єкти. Навколо цього твердження кружляє непорозуміння, склалася така думка, що все, що вам потрібно зробити, це вивчити головні пакети програм і ви готові працювати в галузі **комп'ютерної графіки**.

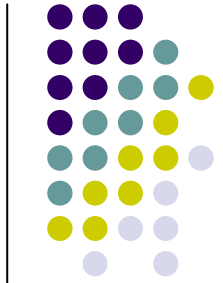
Перші кроки



- Етапи створення малюнку:
  - ☑ **Історія**
  - ☑ **Визначити, скільки візуальних ефектів можемо використати (інформативність зображення)**
  - ☑ **Визначення, який візуальний ефект потрібен**
  - ☑ **Написання програми, використовуючи інструментальні програмні засоби.**


Chapter 3. Water Tricks
1
A
2
B
C
3
Voila!
#10 An Amusing Manipulation
(1) Let water flow from the tap. Using scissors, cut off a good length (A).
(2) Carefully bring the top (B) down to meet the bottom (C), forming a circle.
(3) With practice, the water will keep flowing for many minutes, using the principle of hydrokinetic stream adhesion
(C) 1998 Glenn McCarter

Перші кроки



- Етапи створення малюнку:
  - ☑ **Історія**
  - ☑ **Визначити, скільки візуальних ефектів можемо використати (інформативність зображення)**
  - ☑ **Визначити, який візуальний ефект потрібен**
  - ☑ **Написати програму, використовуючи інструментальні програмні засоби.**

- Створення цього ланцюга не повинне залежати від специфічних інструментів, які ми маємо.
- Необхідно ознайомитись з загальними методами, що використовуються, щоб створити, компонувати і намалювати малюнок.

Це дозволяє вивчити курс “Комп’ютерна графіка”

Візуальна граматика

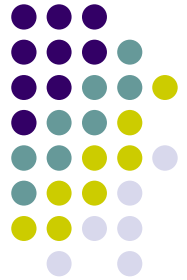
- **Ідентифікація і аналіз алгоритмів візуалізації, що використовують при малюванні:**
  - **Зовнішні алгоритми**
  - **Побудова тіней (синтизоване графічне зображення)**
  - **Картографії (текстура)**
  - **Інші.**
- **Переваги та недоліки того чи іншого алгоритму**
  - **Наприклад: вплив якості малюка на швидкість його рисування**

Візуальний Аналіз

- **Метод ідентифікації графічних ефектів в зображеннях або анімація (мультиплікація), визначення специфічних візуальних ефектів.**
- **Чому важливо пам‘ятати?**
  - **Ланцюг: *Історія → Візуальне відтворення → Візуальні ефекти → Програмне забезпечення***
  - **Ідея зв’язку деталей це не тільки композиція картинки, але це і ефект вибраний художником і програмістом, людиною хто створює картину.**

Приклад: Осмислена поверхня (картина) – більш розкішніша для рисування. Наскільки важливе - зображення, що відтворене з використанням історії? Чи можемо підробити зображення без візуального аналізу?

Загальні непорозуміння



- **КГ є знанням про те, як використовувати зв'язки графічних пакетів**
- **КГ** не **вимагає програмування**
- **КГ** не **вимагає математичних основ**
- **КГ** не **вимагає знання про малюнок, його склад, або редагування**

Але існують програми...

- В деяких випадках, в комп'ютерній графіці для відтворення реалістичного зображення використовуються мови програмування
  - 3DMax – написання сценаріїв
  - Pixar - засоби опису процесу тривимірної візуалізації для отримання реалістичного відображення
  - Інші

Правда є…

- Позаду анімації використовуються візуальні ефекти і математичне моделювання…
  - Алгоритми, що використовуються генераторами візуальних ефектів, які засновані на геометрії, алгебрі і в деяких випадках фізиці

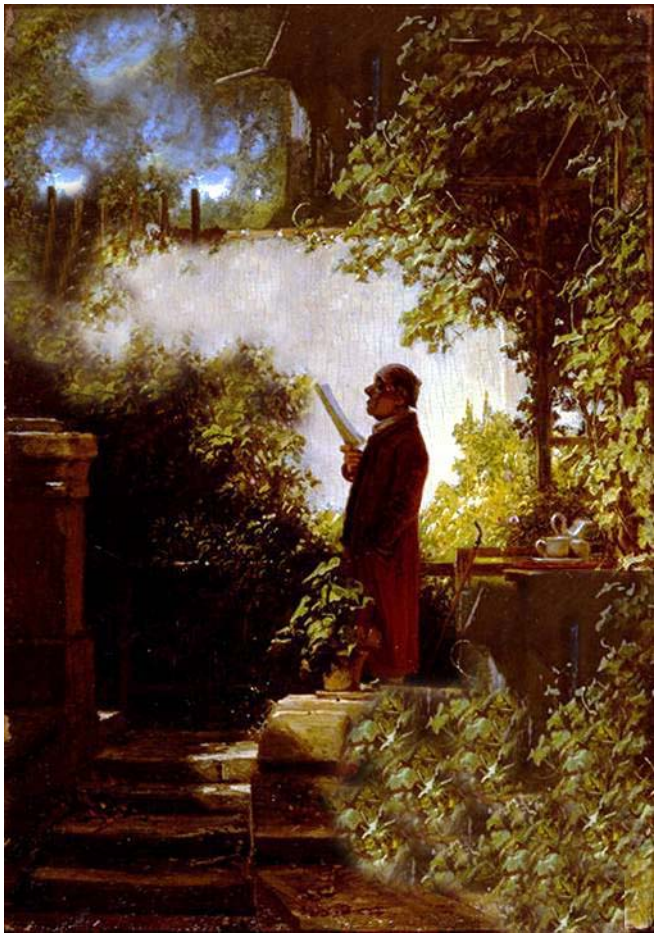
Правда є…

- Наповненість місця події вимагає артистичну чутливість

  ➢Монтаж моделей, щоб супроводжувати візуальне відтворення
  ➢Освітлення
  ➢Візуальна гармонія

Правда є…

- Порівняйте…



Правда є…

- Інші приклади:

Це зображення є по суті поглиблене

Правда є…

- Інші приклади...

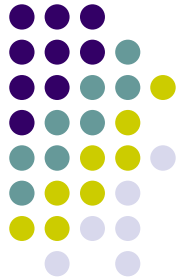
Збільшивши малюнок та відкинувши зайве ми бачимо предмети, що знаходяться в глибині малюнка

Правда є…

- Інші приклади...

Корекція освітленості додає ефект реальності

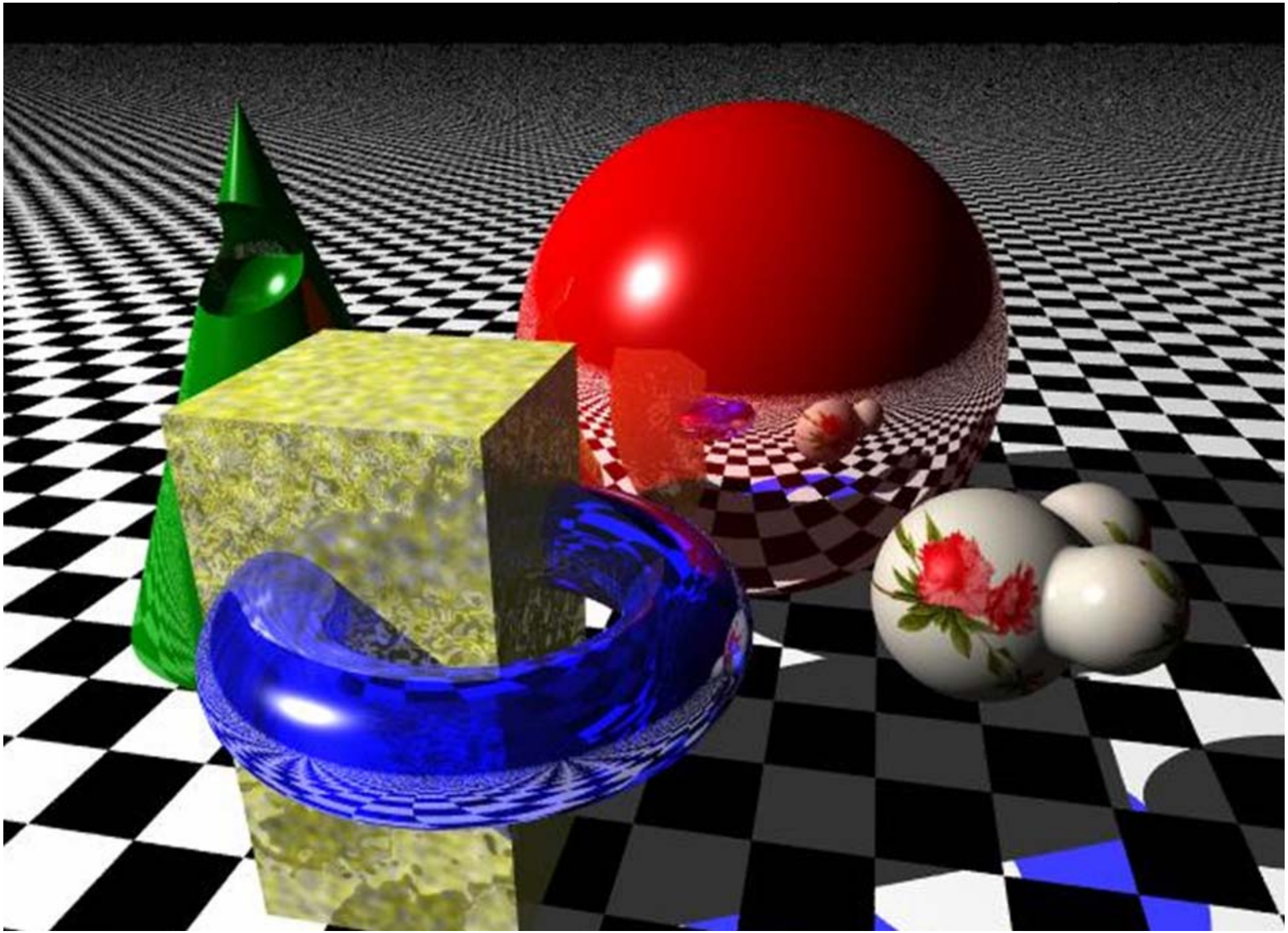
Добра новина…



- Пакети програм на сьогодні вже мають багато математичних інструментів.
- Проте, це не означає, що ви не повинні знати яка функція команди та будова.
- Пакети програм забезпечують вас лише інструментами, а не ідеями, знанням та досвідом.

При використанні POV-Ray

- Ви розвиватимете свою візуальну письмовість.
- Ви будете займатися візуалізацією і частково анімацією, через написання коду сценарію (програми).
- Ви будете вивчати і застосовувати деяку основну математику, що використовується в комп'ютерній графіці.



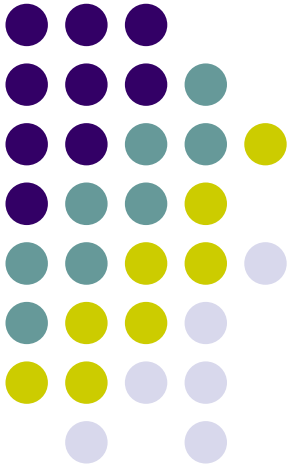
Kara Salganik
DePaul University

Pat Motoike
DePaul University

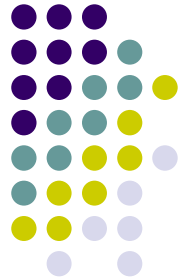
Daiva Skuodyte
DePaul University


A Practical Guide to Linux
UNIX
THE Linux Bible

Моделювання місця дії та його Візуалізація



Моделювання місця дії та його Візуалізація



- Створення зображення відбувається двома етапами
  - Моделювання місця дії
    - Створення представлення об'єкту, місця дії або віртуального світу.
  - Створення місця дії
    - Представлення починається від архітектури
    - Будується малюнок зображення моделі
- Для кожного зображення:
  - Звідки спостерігати за місцем дії?
  - Як розгорнуто можемо глянути з місця спостерігача?
  - Яке обрати освітлення?

Розгляд Місця дії

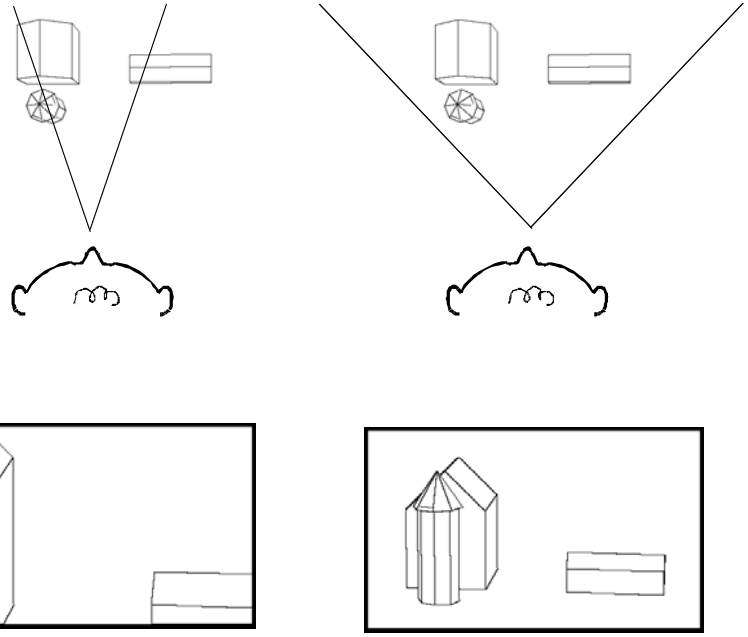
Звідки спостерігати за місцем дії?

- Нам потрібно розмістити себе де завгодно і дивитися на місце дії під різним кутом

• В КГ ми досягаємо цього змінюючи Камеру

• Як розгорнуто можемо глянути з місця спостерігача?

• В КГ ми досягаємо цього за допомогою Області зору



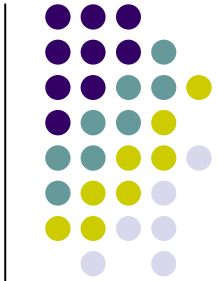
Розгляд Місця дії

- Область зору
  - Як розгорнуто можемо глянути з місця спостерігача?

Яке обрати освітлення?

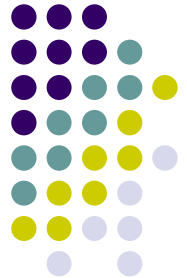
- Як багато ми можемо побачити без освітлення?
  - Сонячне світло
  - Штучне світло
- Нам природно потрібне освітлення для того, щоб побачити
- В КГ місце дії без освітлення виглядає похмуро, темно...

Моделювання + Візуалізація =



Зображення високої якості

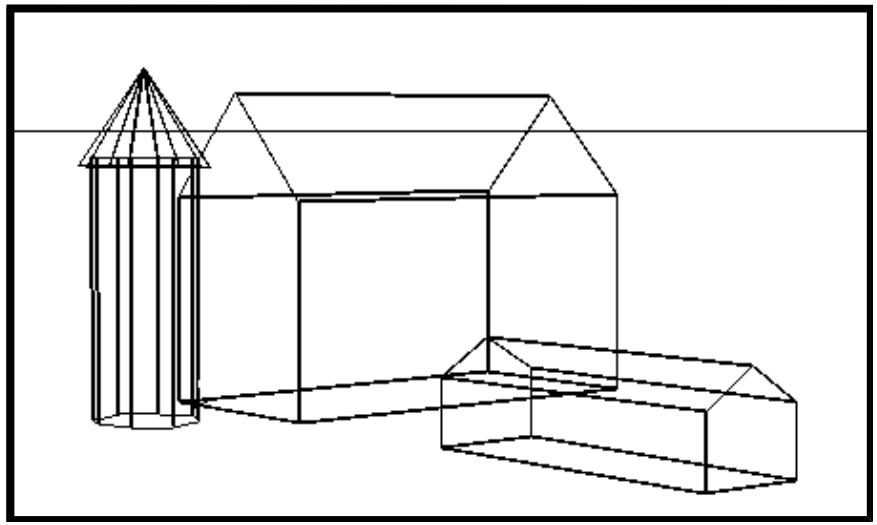
Зорові (візуальні) ефекти

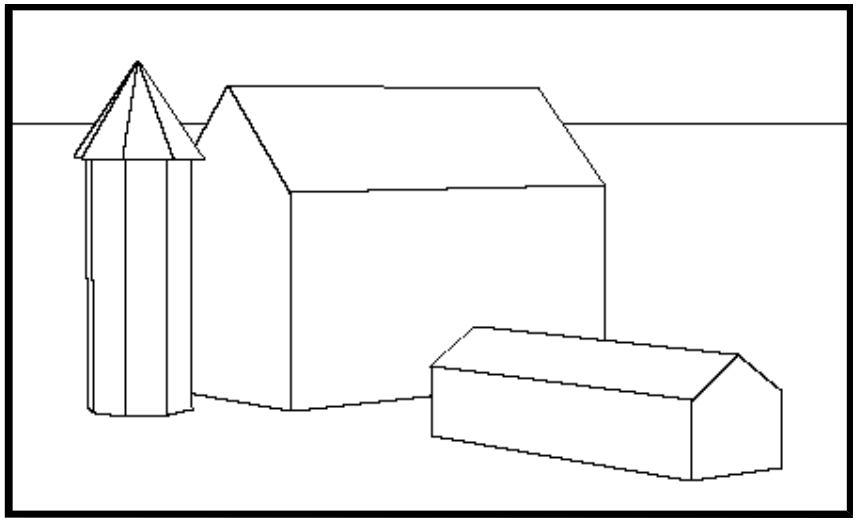


Аналізуючи місце дії, існують деякі візуальні ефекти, які ви можете розглядати

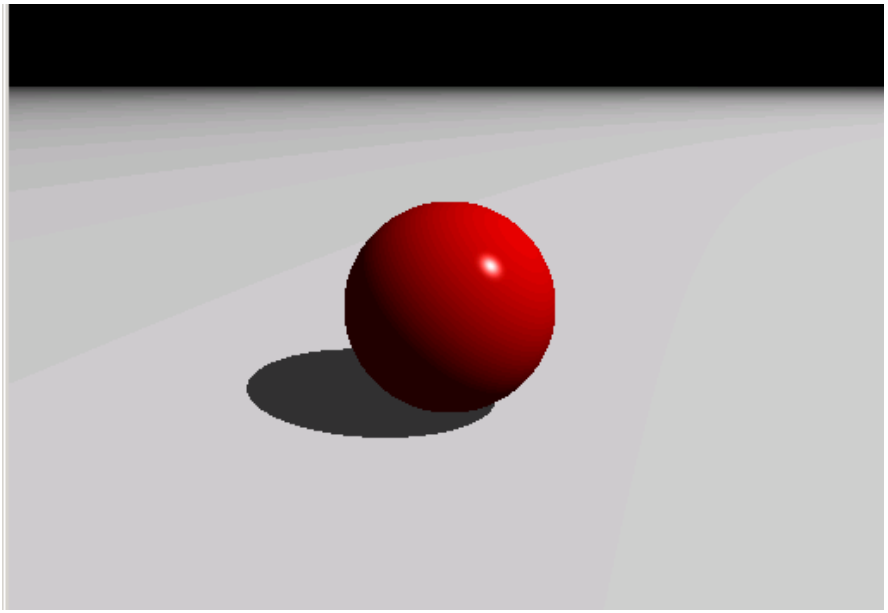
- Видимі поверхні
- Тінь
- Заломлення
- Відображення
- Прозорість
- Освітлення

Видимість фону (задній план)

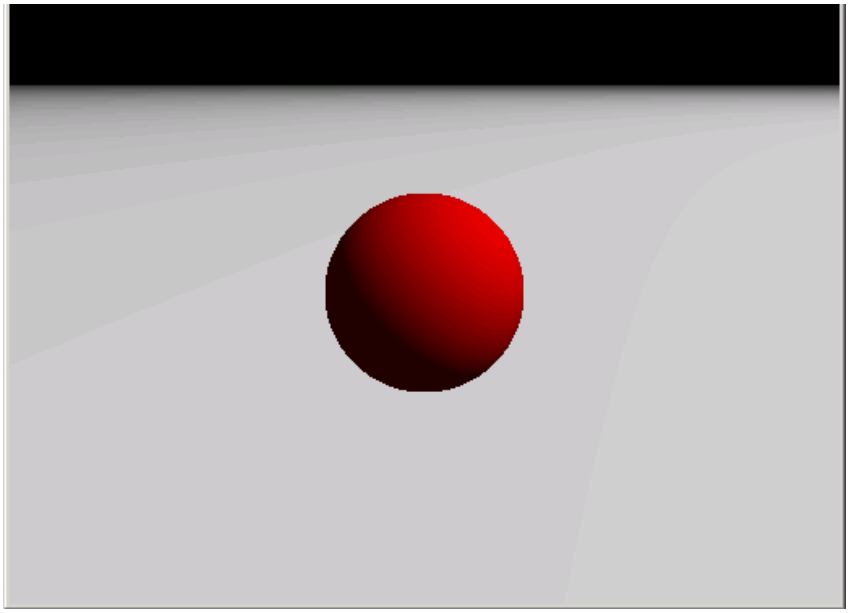




Тінь та затінення

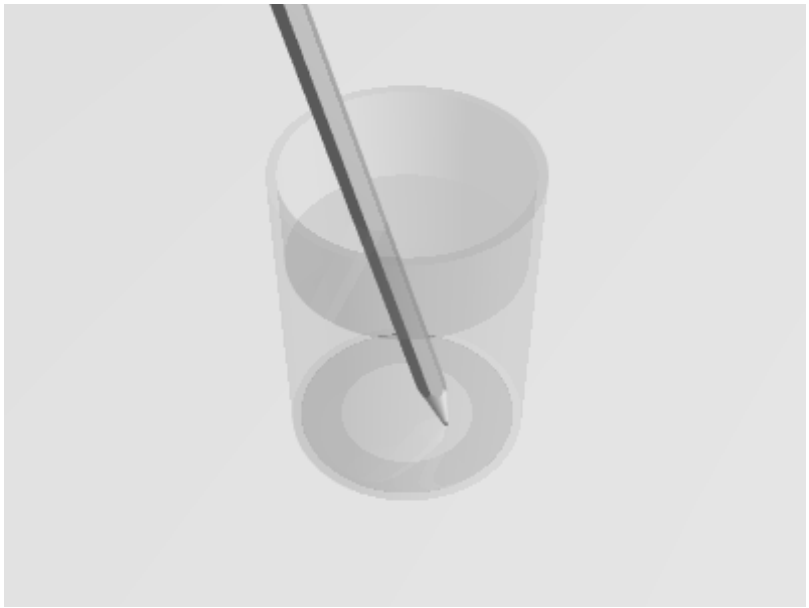


тінь + затінення (маємо гарне освітлення)

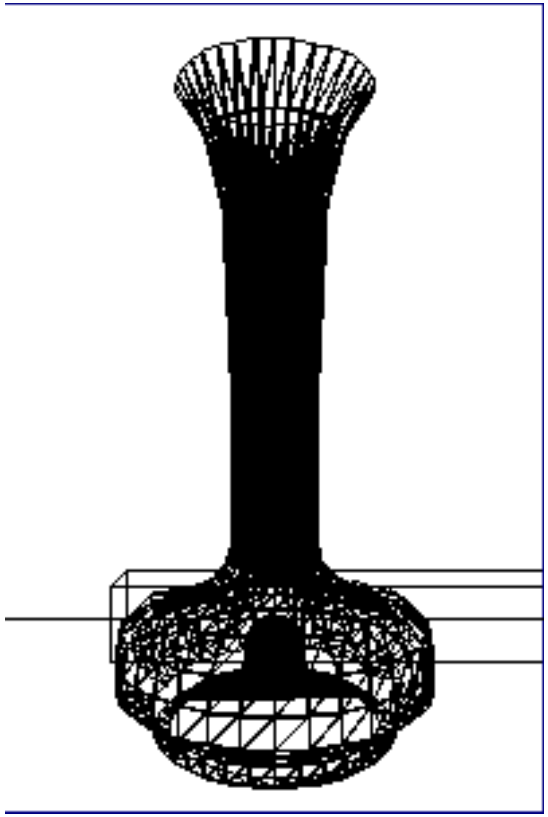


Тільки затінення

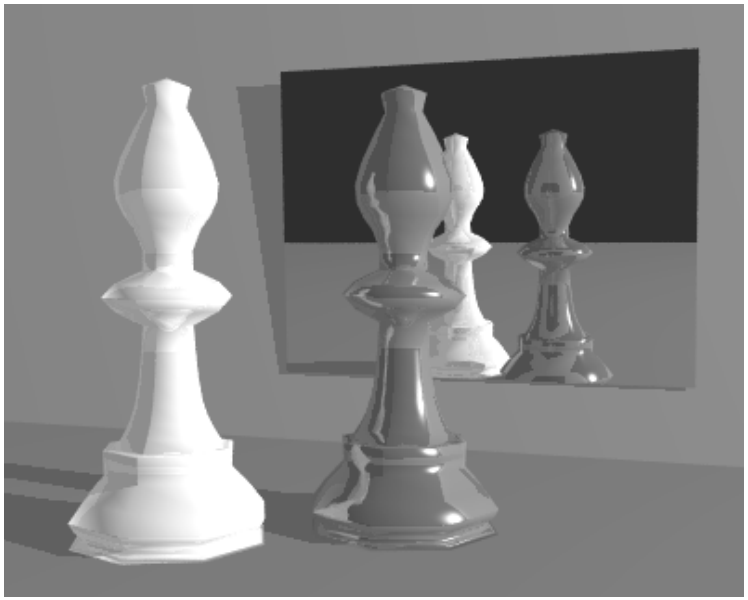
Заломлення

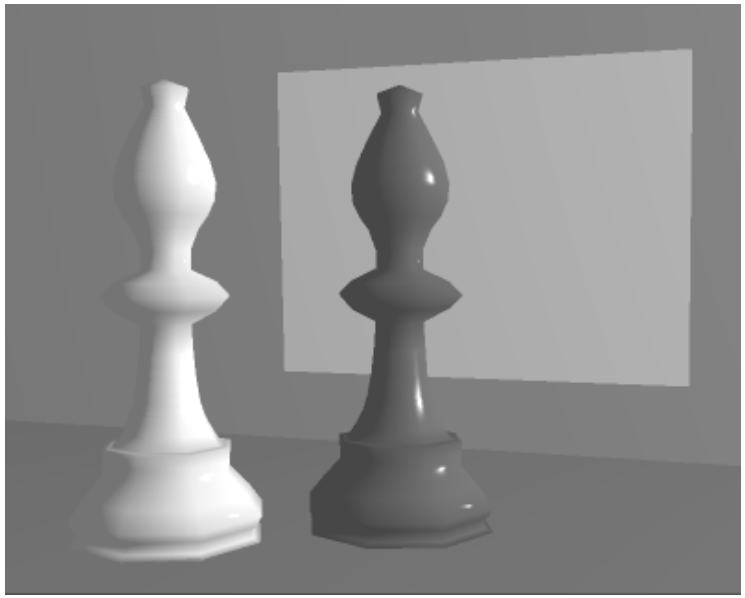


Прозорість



Відображення

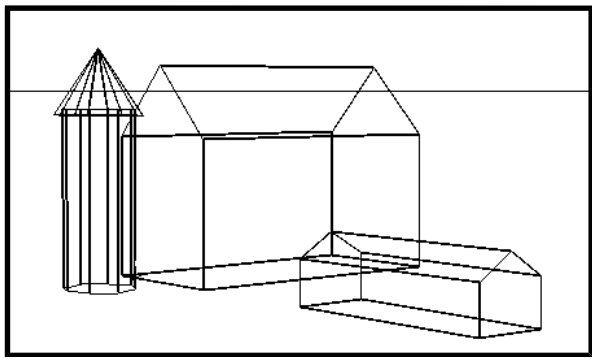




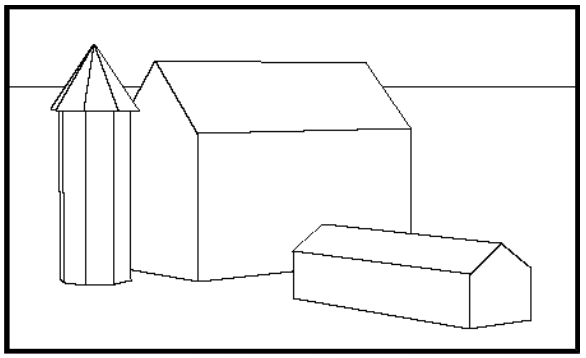
Відзначимо багаторазові рівні відображень між стінним дзеркалом і між центрами фігур

Зовнішні алгоритми

- Використовується, щоб визначити, які поверхні видимі

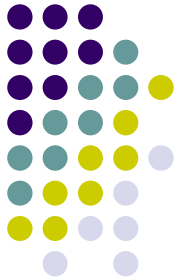


Модель

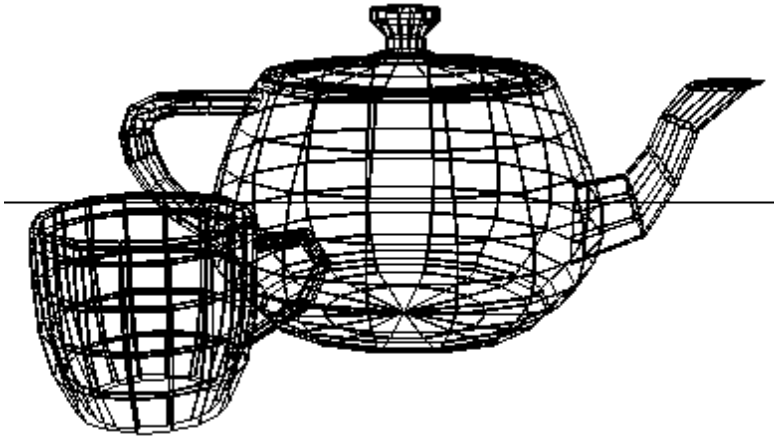


Результат візуалізації

- Існує чотири головні зовнішні алгоритми:
  - Дротяна рамка
  - Невидима лінія
  - Променеве зображення
  - Z-буфер

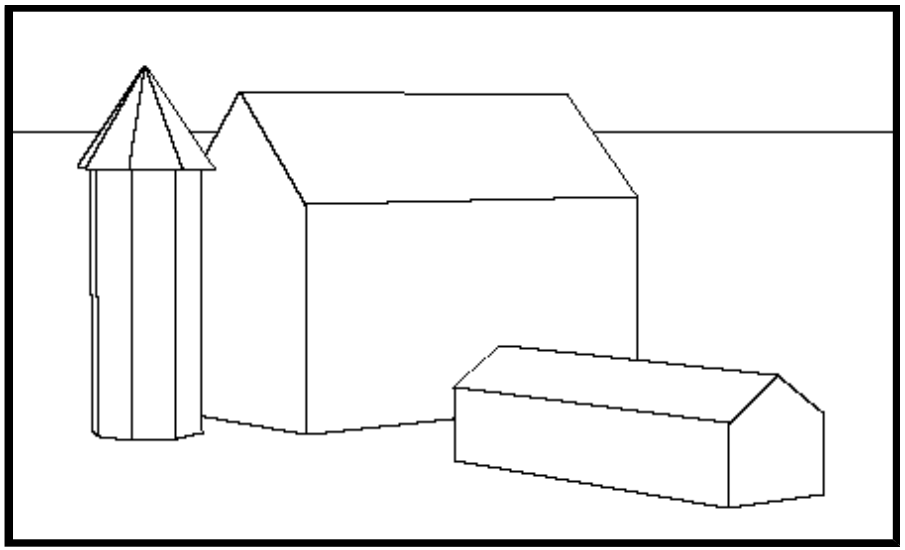


- Ескіз в багатокутниках
- Горизонт та другорядні об'єкти цілком видимі.

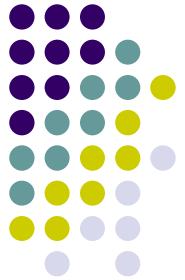


Дротяна рамка

- Ескіз в багатокутниках
- Об'єкти, що закриті стають невидимими



Невидима лінія



- Заповнені багатокутники
- немає заломлення відображення чи тіні

Z-буфер

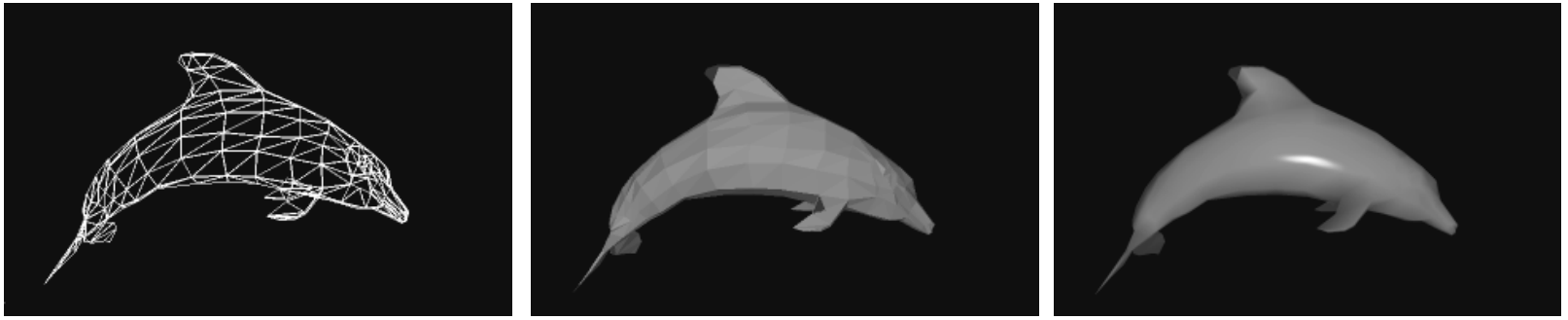
- Об'ємні багатокутники
- Заломлення, Відображення, Тіні

Променеве зображення

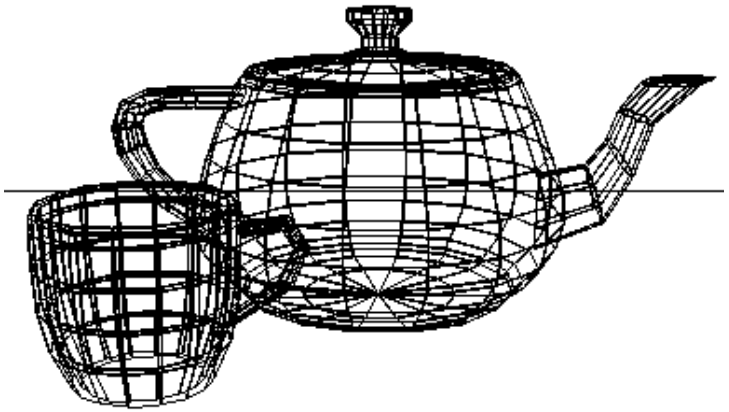
Візуальний аналіз

- Метод ідентифікації графічних ефектів в зображеннях або анімації, покриває п’ятнами віддалені візуальні частинки.
- Алгоритм ідентифікації частинок.
  - Спочатку визначаються частинки.
  - Потім пропонується алгоритм.
- Головні зовнішні алгоритми:
  - Багатокутниковий вигляд
  - Видимість окремих сторін, діапазонів
  - Заломлення, відображення, тінь

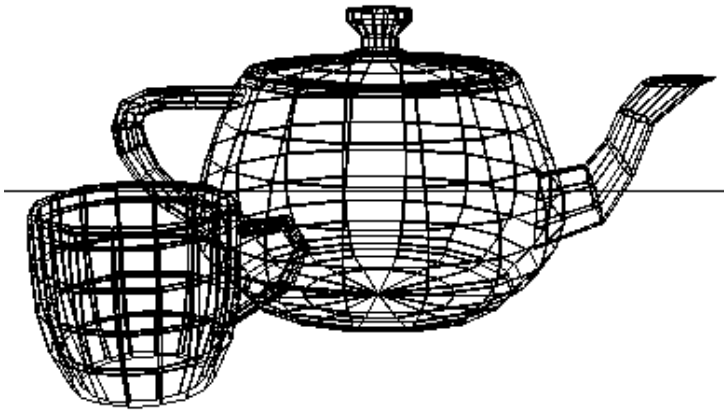
Багатокутниковий вигляд



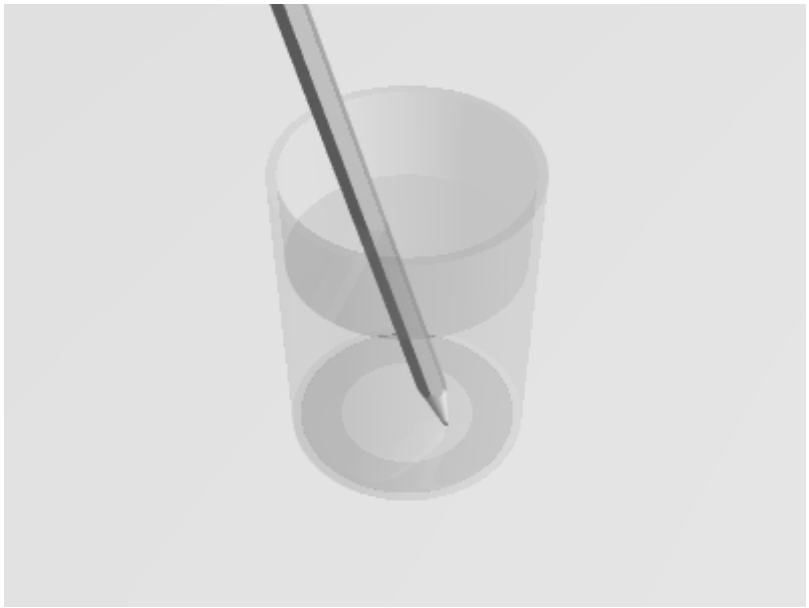
Видимість окремих сторін, діапазонів



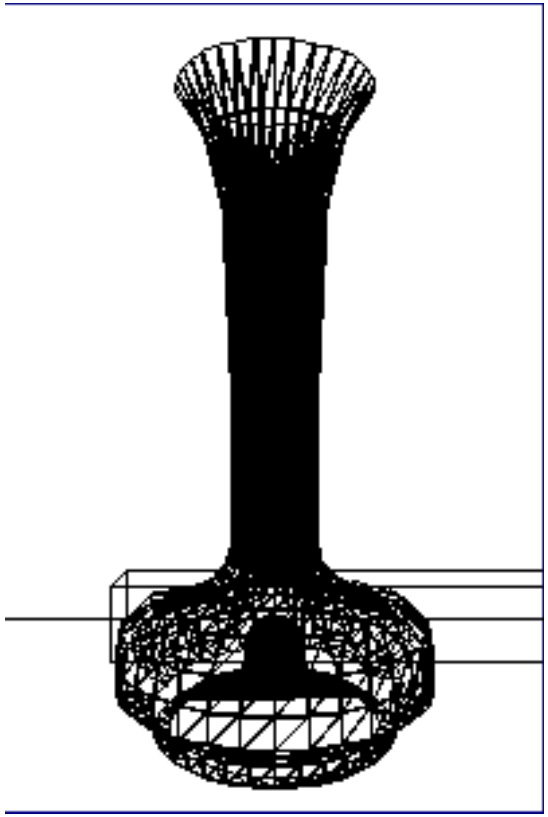
Заломлення, відображення, тінь



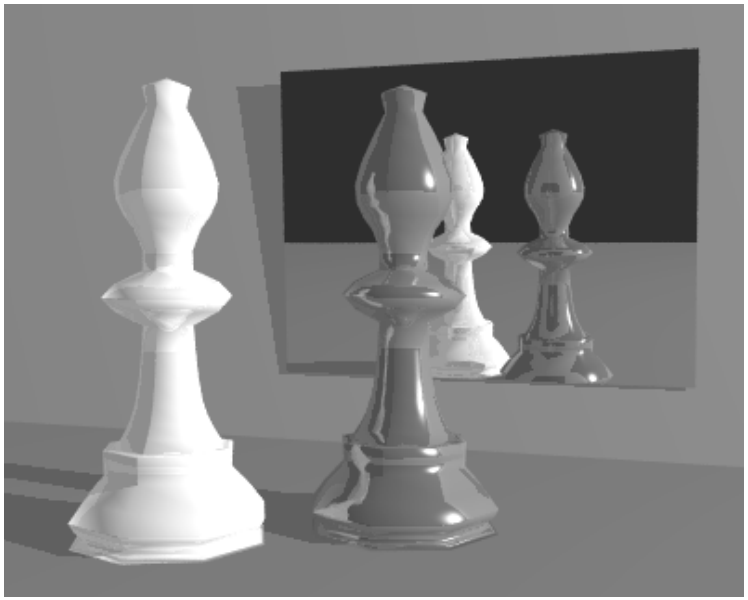
Заломлення

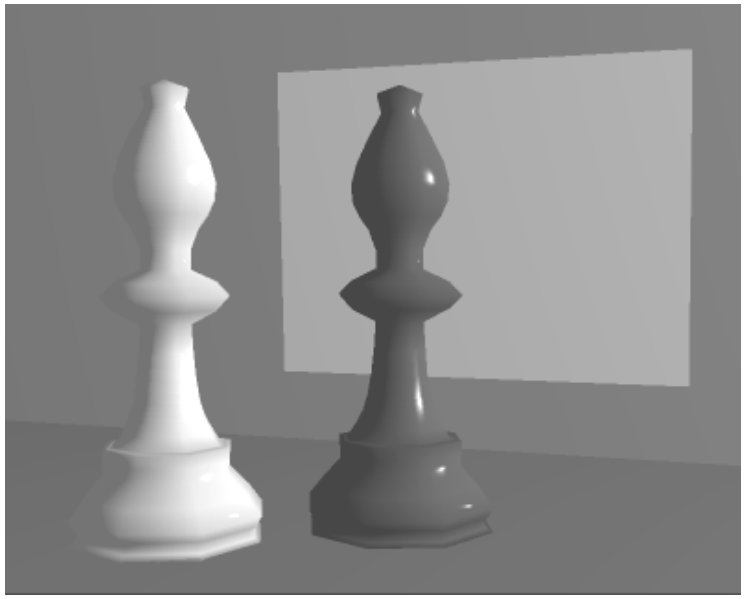


Прозорість

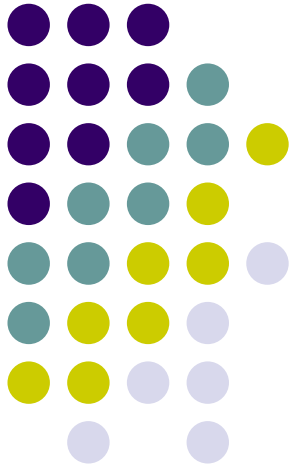


Відображення



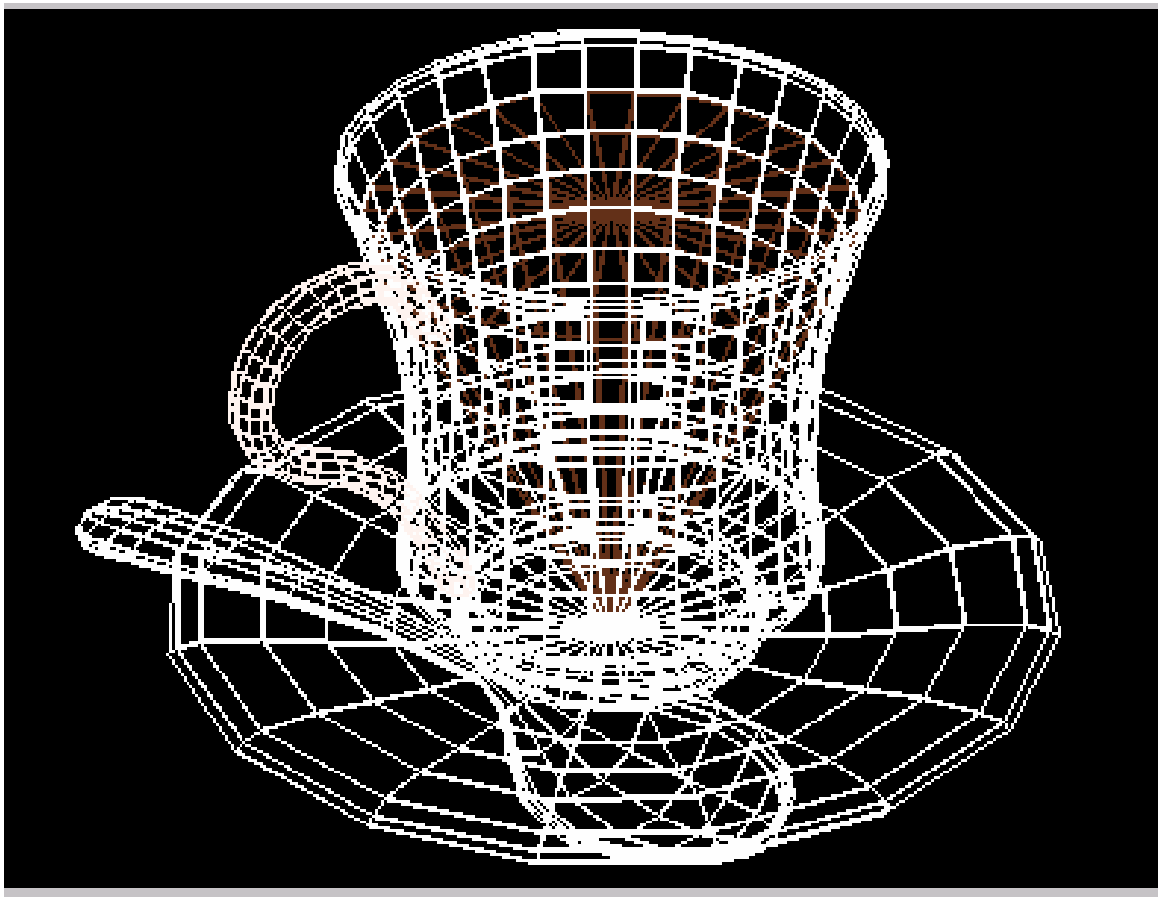


Практичні приклади

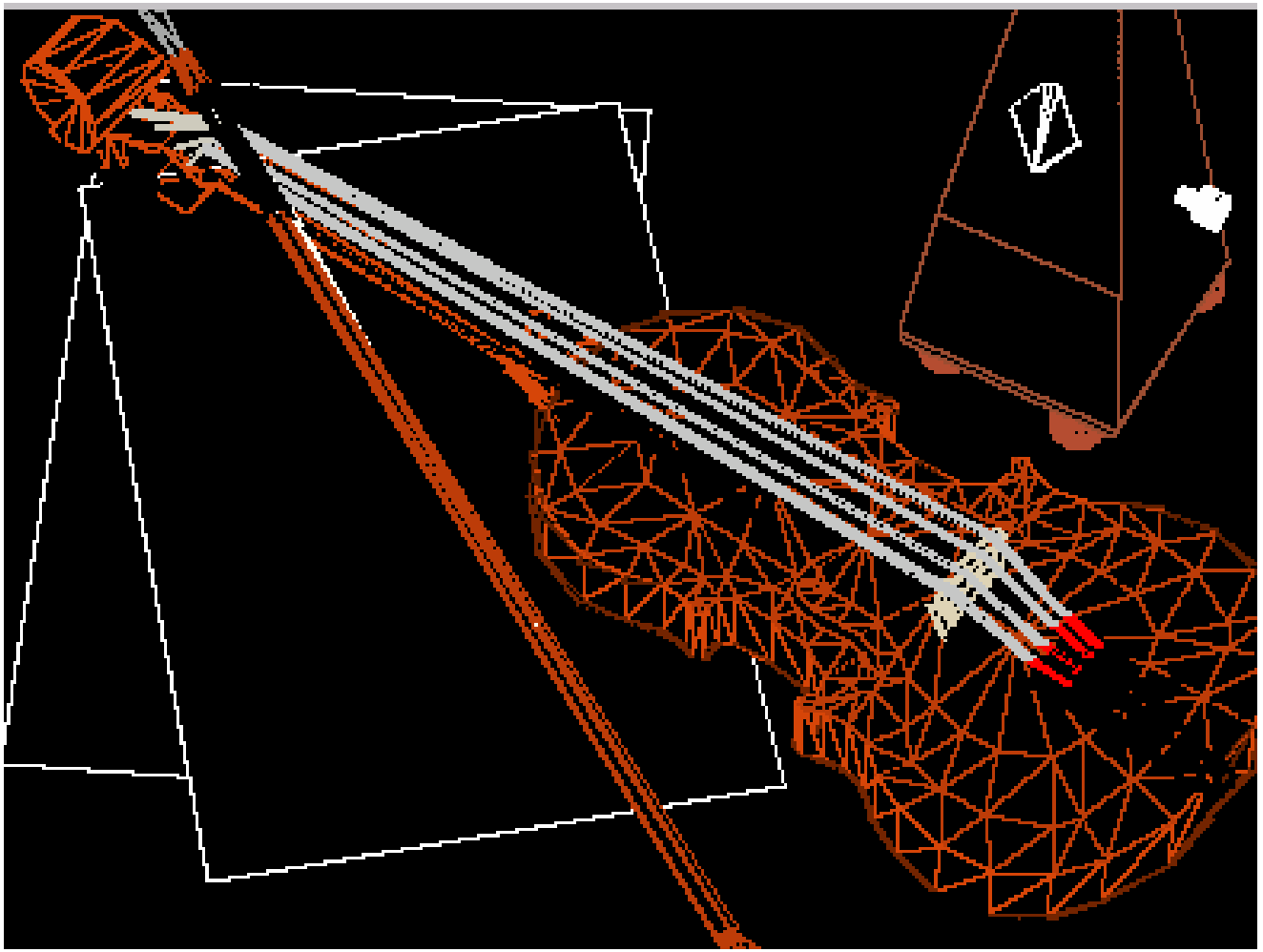


Зовнішній алгоритм?

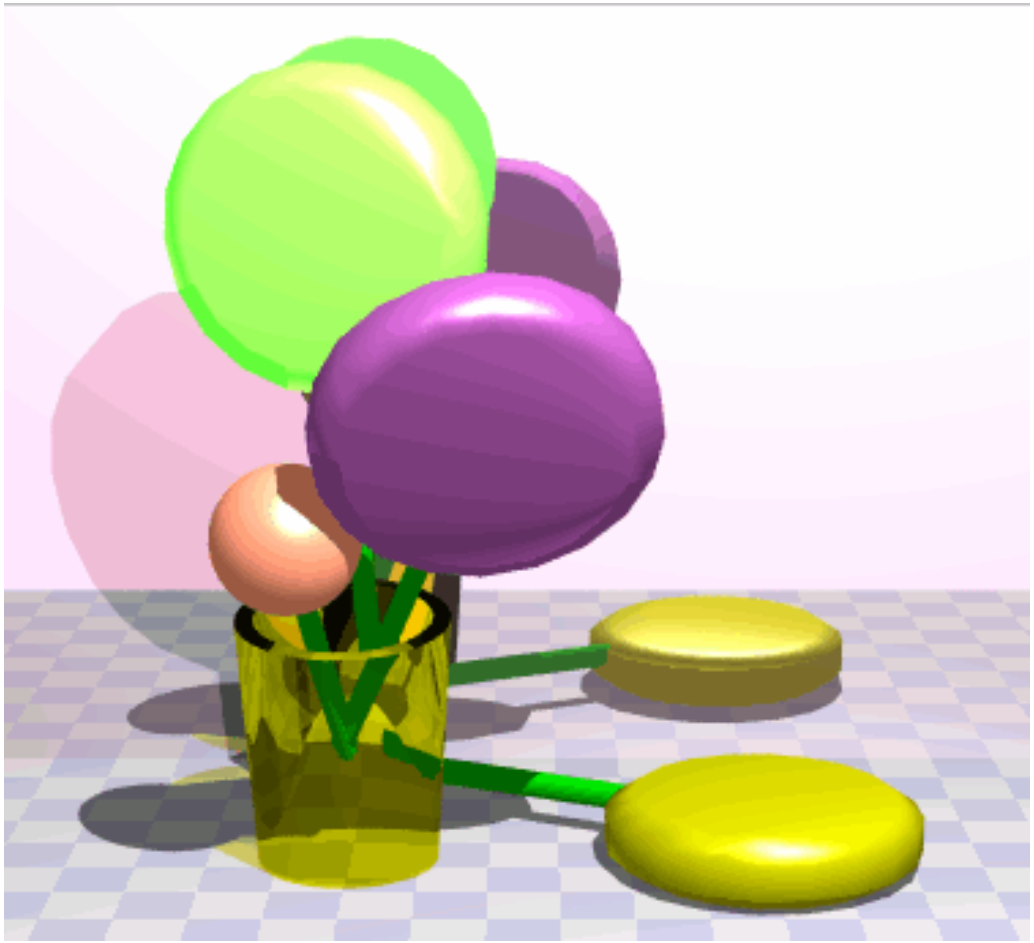
Зовнішній алгоритм?



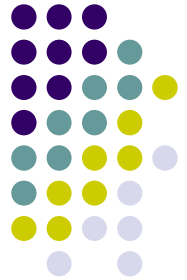
Зовнішній алгоритм?



Зовнішній алгоритм?



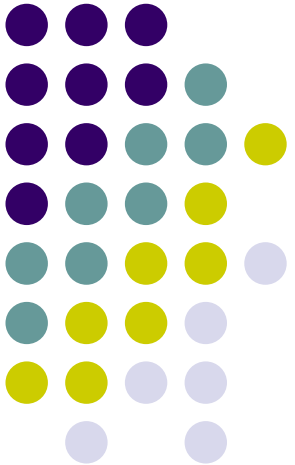
Отримання або освоєння візуальної письменності



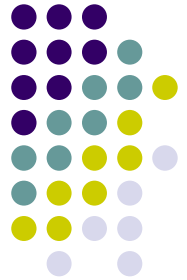
- Практика в програмуванні.
- Візуалізація, візуалізація і візуалізація….
  - Проблема найкращої візуалізації полягає в недостатку найціннішого (час).

POV-Ray

Постійне відслідковування променя

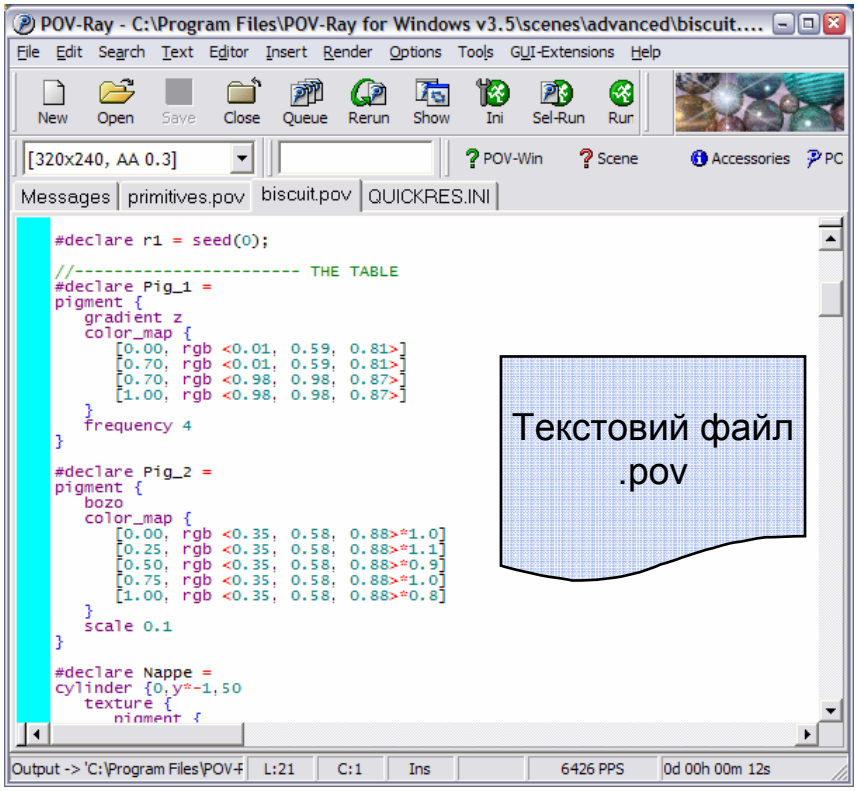


POV-Ray



- Створює фото-реалістичне зображення, яке використовує відслідковування променя (візуальна техніка).
- Читає текстовий файл, що описує об'єкти, освітлення, розташування камери в місці дії і відтворює зображення від точки спостереження камери
- Це не моделююче програмне забезпечення

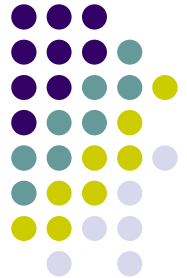
POV-Ray



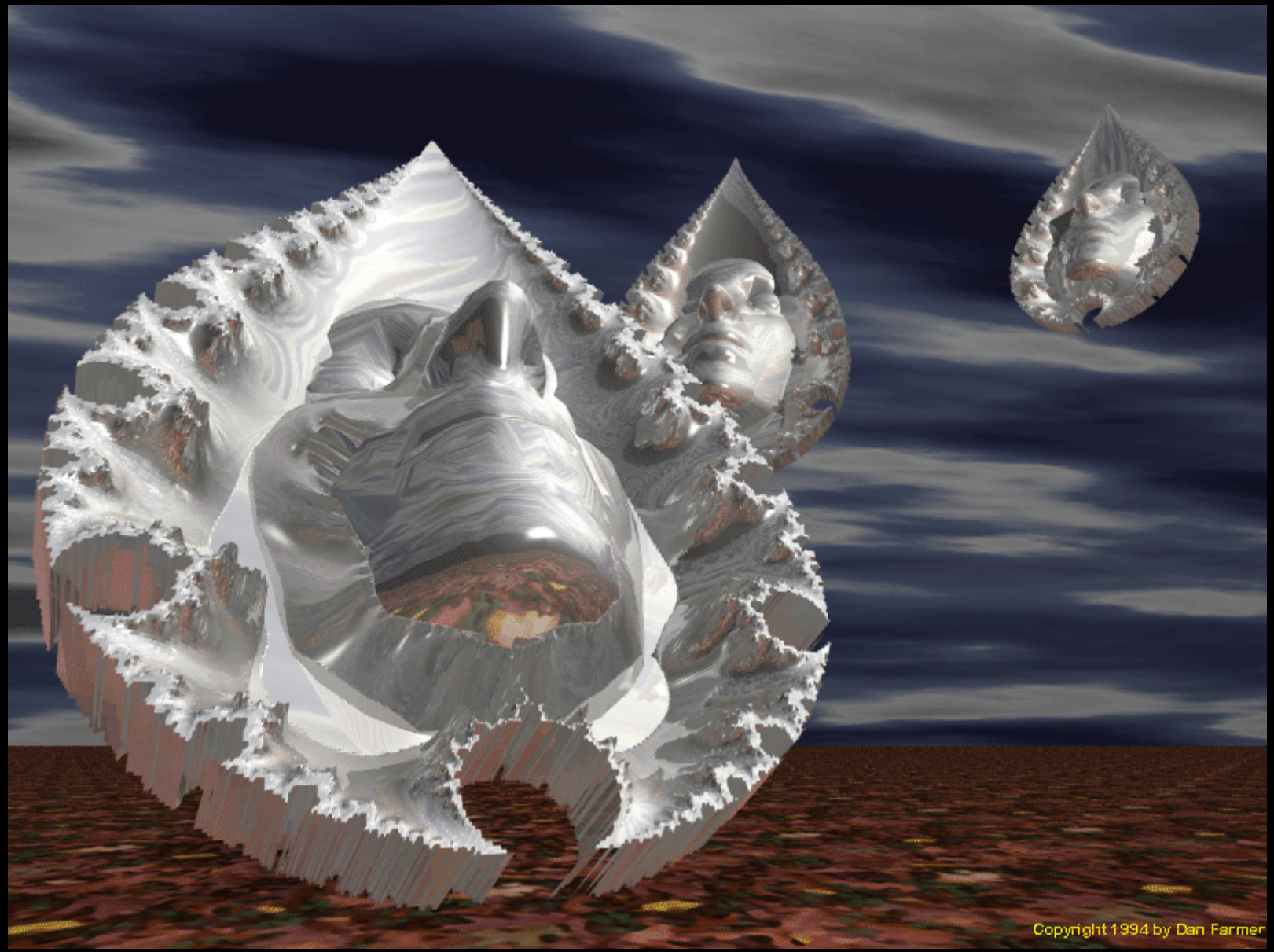
Текстовий файл .pov

Візуалізація

POV-Ray

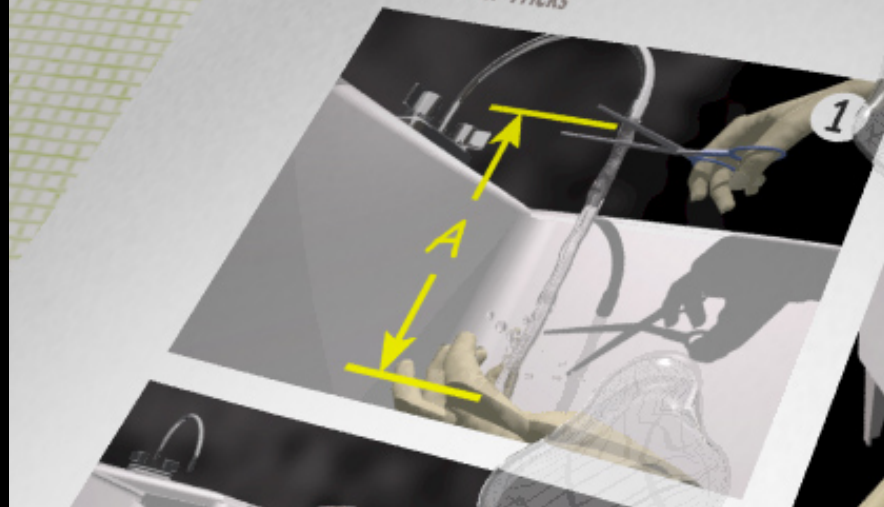


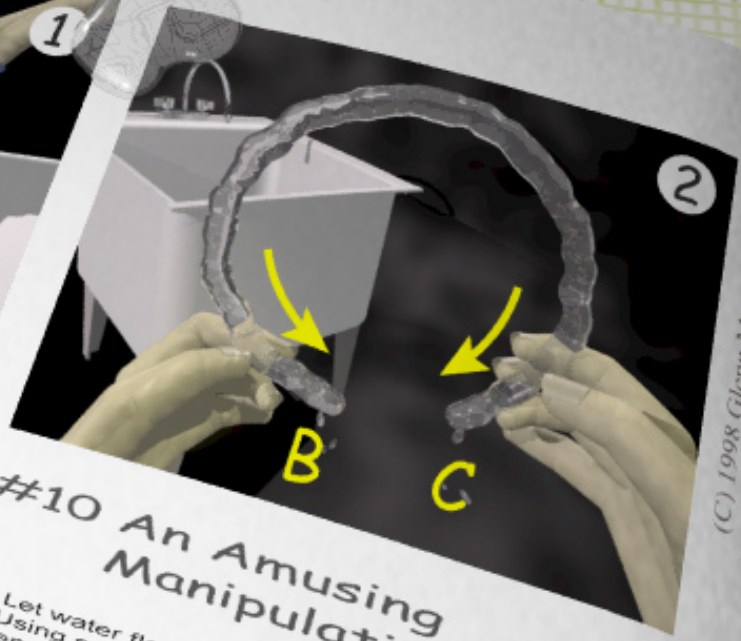
- Ви моделюєте місце дії через написання сценарію
- Ви можете перенести моделі, створені в інших пакетах програм
- Вільне завантаження
  - http://www.povray.org
- 9,8 Mb об'єм.
- Оперативна документація (англійською)


Copyright 1994 by Dan Farmer


1971


Hamilton
To compute the
the following int
(2)
(3)





#10 An Amusing Manipulation

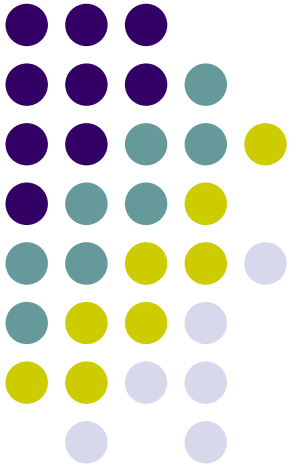
(1) Let water flow from the tap. Using scissors, cut off a good length (A).

(2) Carefully bring the top (B) down to meet the bottom (C), forming a circle.

(3) With practice, the water will keep flowing for many minutes, using the principle of hydrokinetic stream adhesion

POV-Ray

Перше місце дії (сцена)



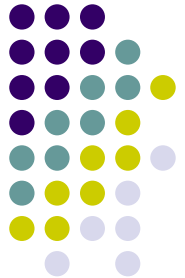
Робота з POV-Ray

- Чотири головних компонента
  - .pov файл Сцена моделювання в POV-Ray сценарії (c)
  - .inc файл Допоміжні файли (також в коді)
  - Файли зображення Зображення малюнку (зазвичай *.bmp формат)

Також

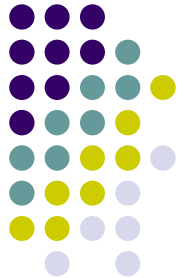
- .ini файл Візуалізація параметрів але поки ми не будемо їх розглядати

Проста сцена

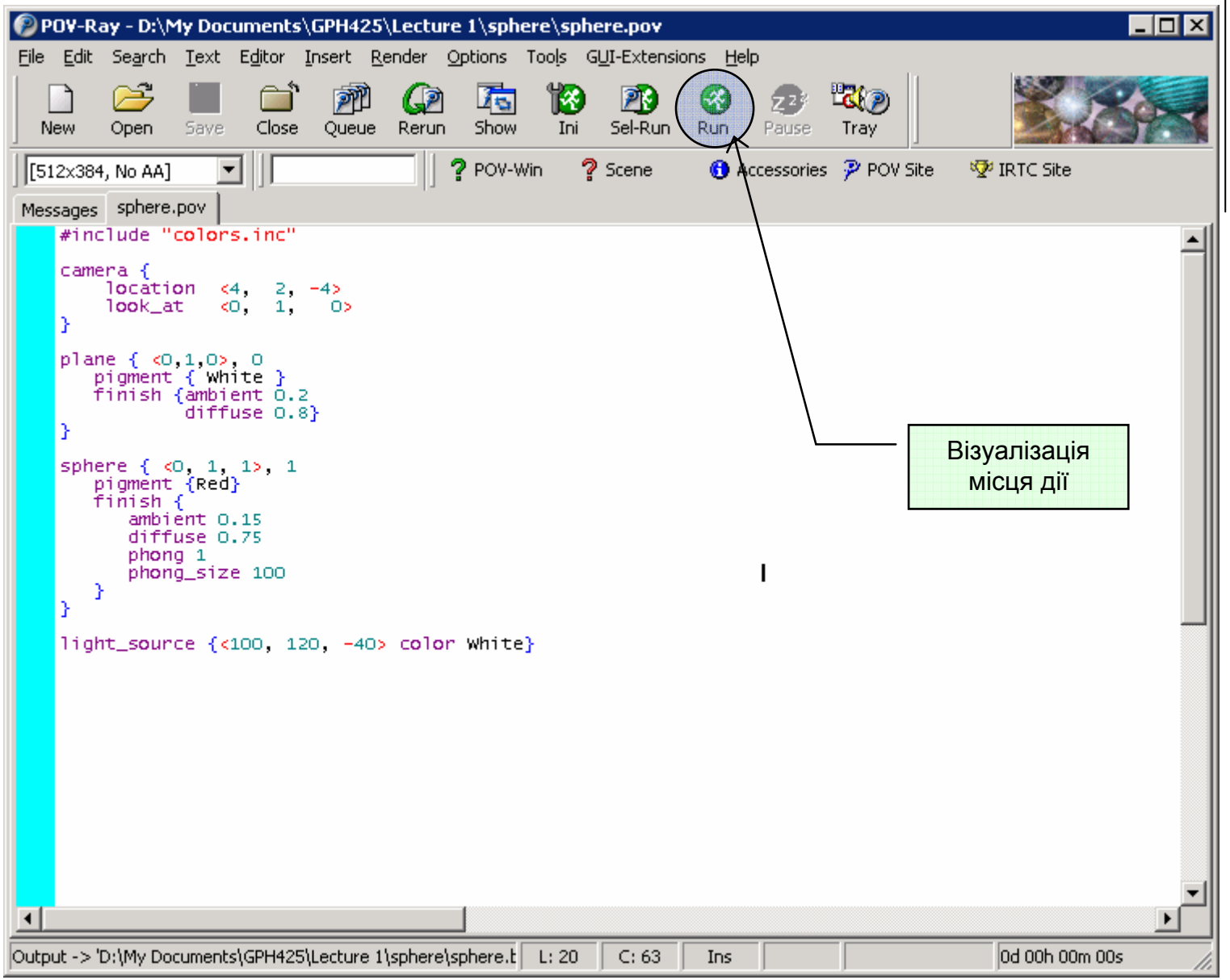


- Створить об'єкт.
- Створить камеру.
- Створить джерело світла.

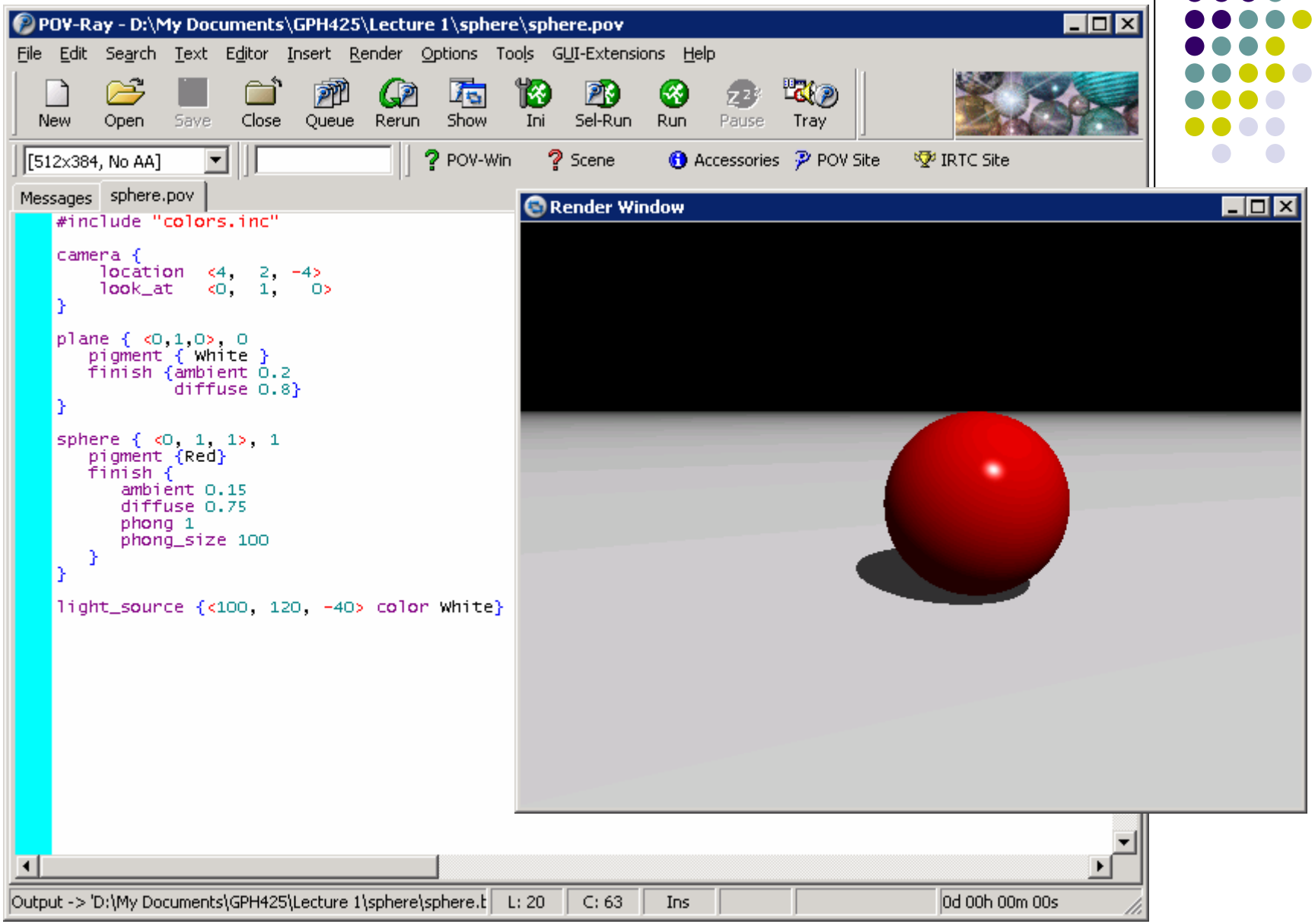
Приклад: *sphere.pov*



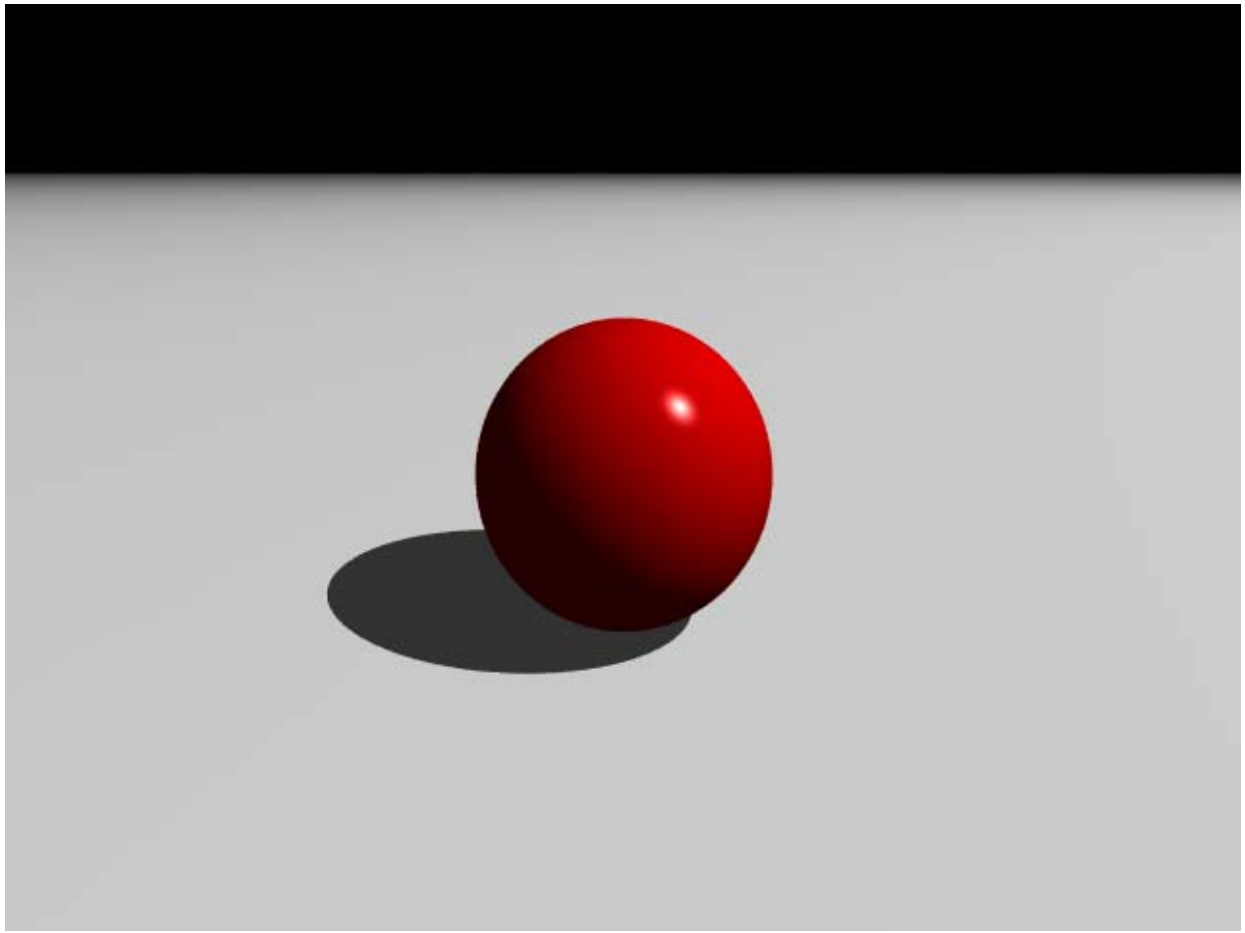
```
#include "colors.inc"

camera {
   location  <0,  3, -6>
   look_at   <0,  1,   0>
}

plane { <0,1,0>, 0
   pigment {White}
   finish {ambient 0.2
           diffuse 0.8}
}

sphere { <0, 1, 0>, 1
   pigment {Red}
   finish {
      ambient 0.15
      diffuse 0.75
      phong 1
      phong_size 100
   }
}

light_source {<100, 120, -40>
                color White}
```

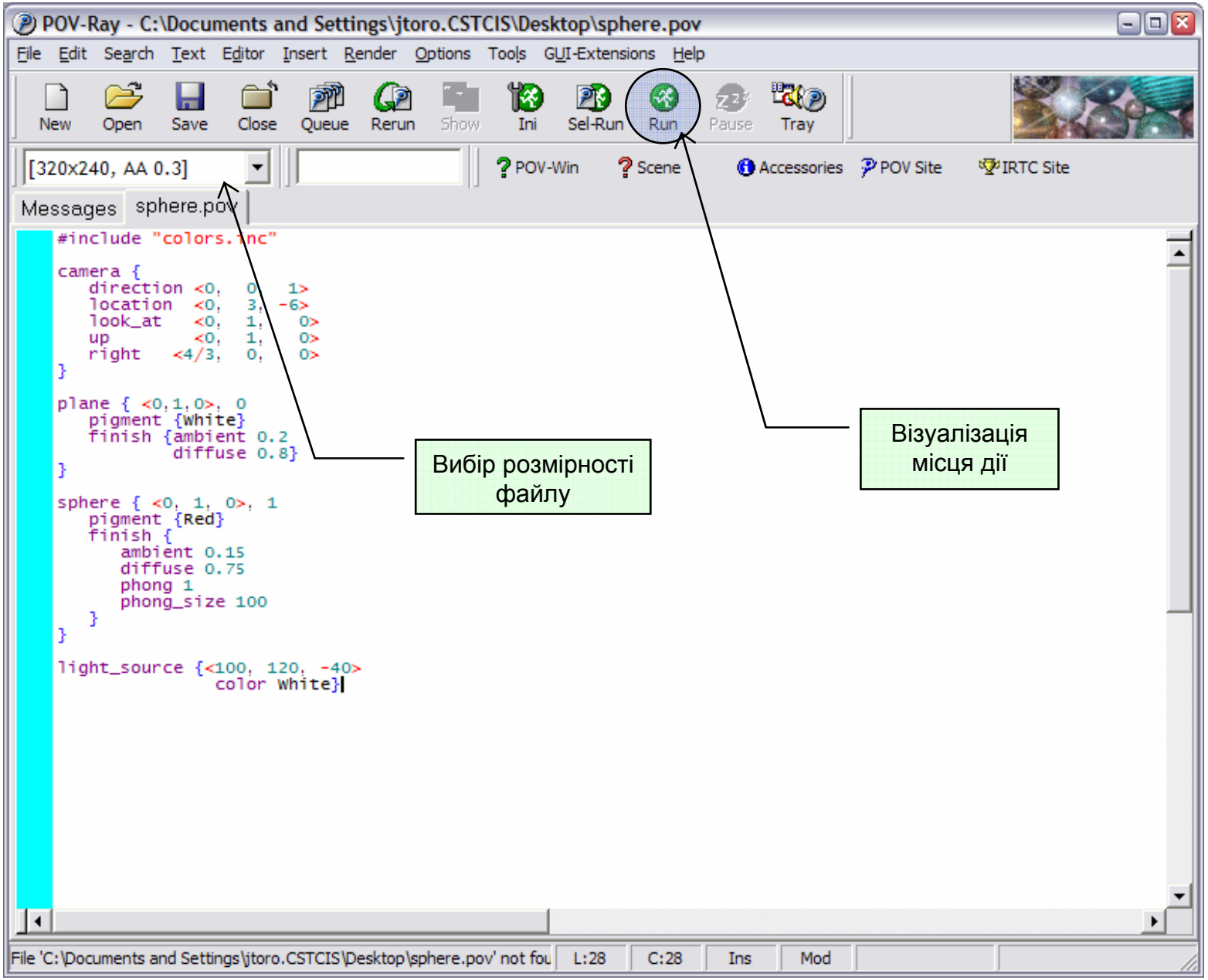


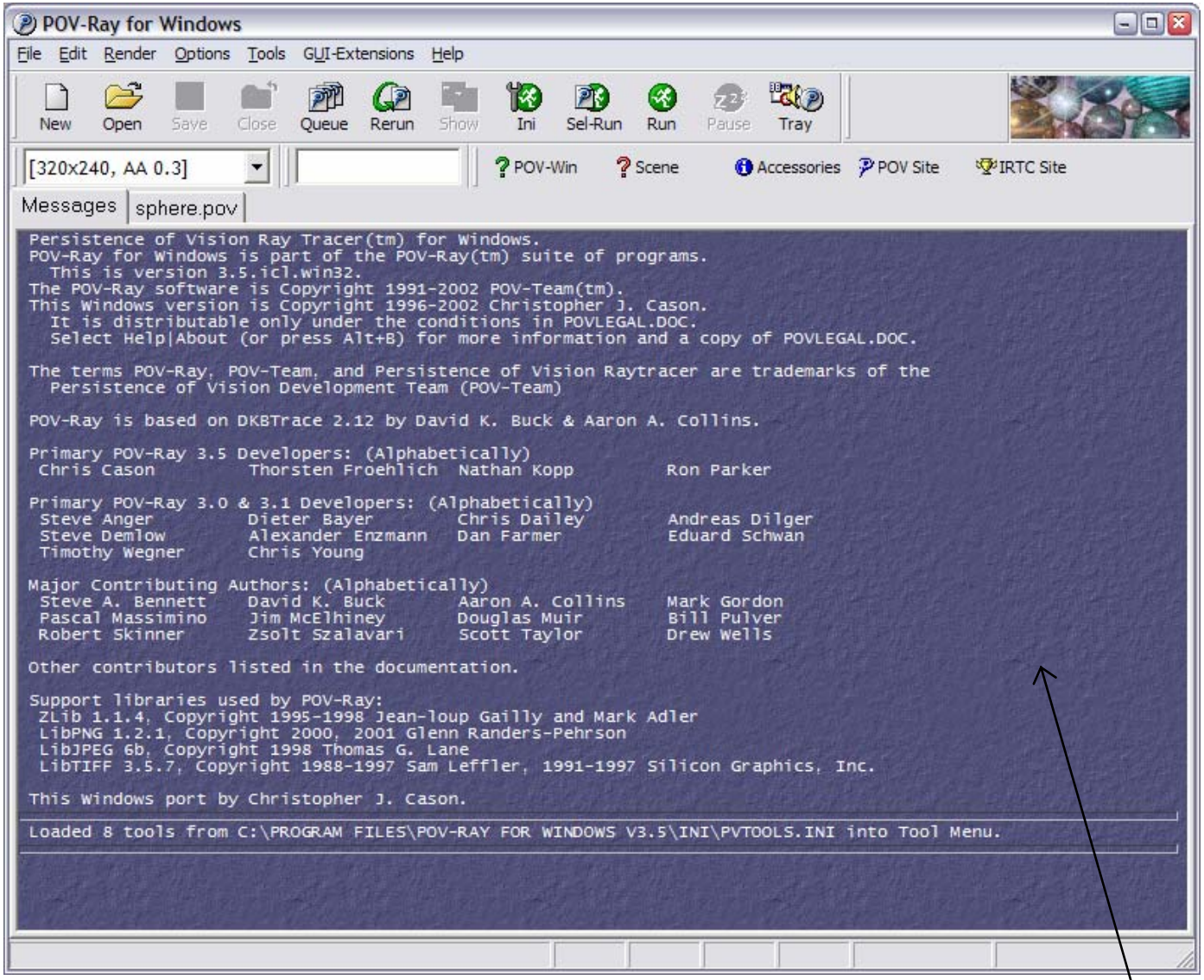
Візуалізація місця дії


POV-Ray - D:\My Documents\GPH425\Lecture 1\sphere\sphere.pov
File Edit Search Text Editor Insert Render Options Tools GUI-Extensions Help
New Open Save Close Queue Rerun Show Ini Sel-Run Run Pause Tray
[512x384, No AA]
POV-Win Scene Accessories POV Site IRTC Site
Messages sphere.pov
#include "colors.inc"
camera {
location <4, 2, -4>
look_at <0, 1, 0>
}
plane { <0,1,0>, 0
pigment { White }
finish {ambient 0.2
diffuse 0.8}
}
sphere { <0, 1, 1>, 1
pigment {Red}
finish {
ambient 0.15
diffuse 0.75
phong 1
phong_size 100
}
}
light_source {<100, 120, -40> color White}
Render Window
Output -> 'D:\My Documents\GPH425\Lecture 1\sphere\sphere.t
L: 20
C: 63
Ins
0d 00h 00m 00s

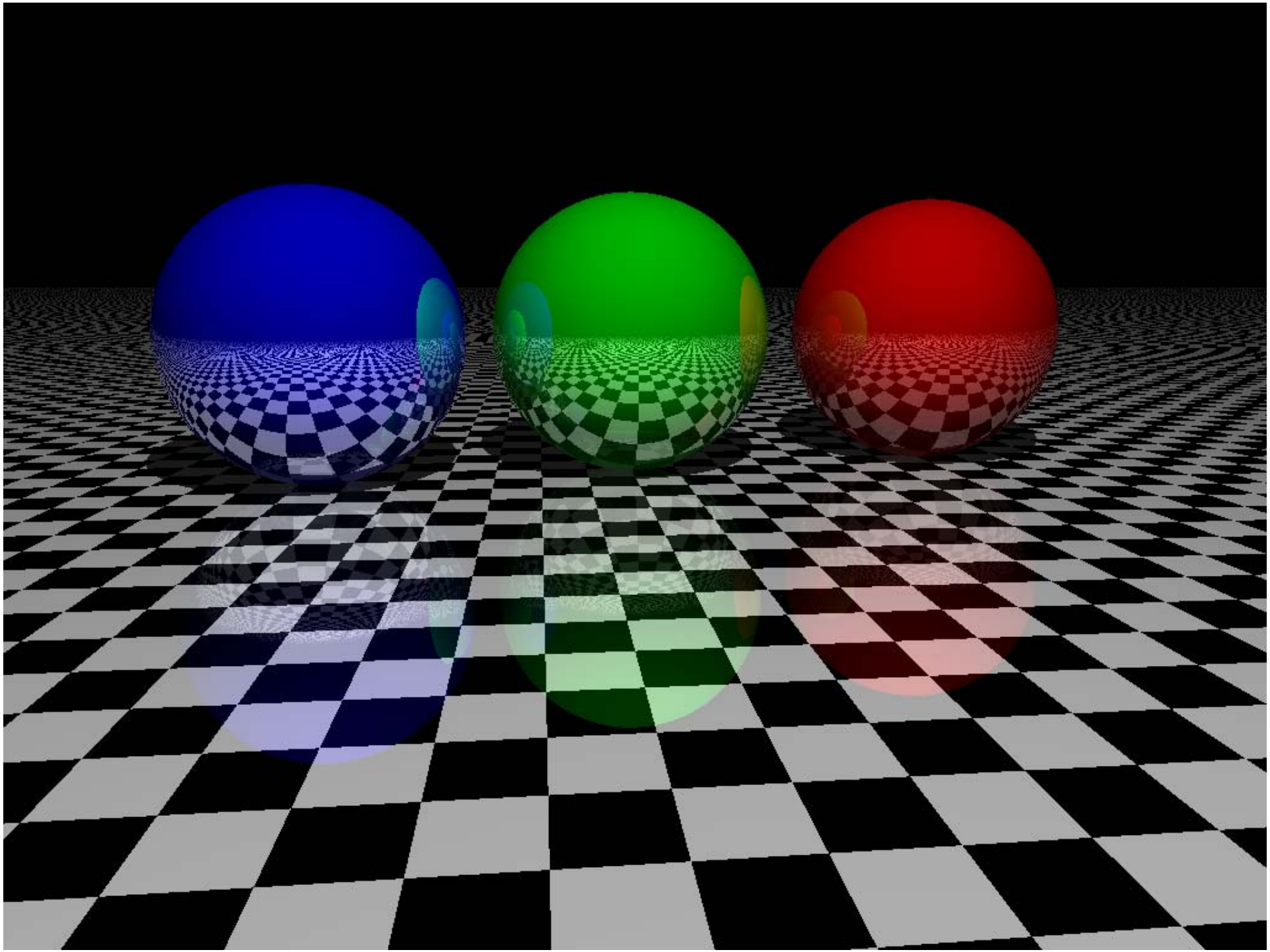
Результат: *sphere.bmp*




POV-Ray - C:\Documents and Settings\jtoro.CSTCIS\Desktop\sphere.pov
File Edit Search Text Editor Insert Render Options Tools GUI-Extensions Help
New Open Save Close Queue Rerun Show Ini Sel-Run Run Pause Tray
[320x240, AA 0.3]
POV-Win Scene Accessories POV Site IRTC Site
Messages sphere.pov
#include "colors.inc"
camera {
direction <0, 0, 1>
location <0, 3, -6>
look_at <0, 1, 0>
up <0, 1, 0>
right <4/3, 0, 0>
}
plane { <0,1,0>, 0
pigment {White}
finish {ambient 0.2
diffuse 0.8}
}
sphere { <0, 1, 0>, 1
pigment {Red}
finish {
ambient 0.15
diffuse 0.75
phong 1
phong_size 100
}
}
light_source {<100, 120, -40>
color White}
File 'C:\Documents and Settings\jtoro.CSTCIS\Desktop\sphere.pov' not fou
L:28
C:28
Ins
Mod
Вибір розмірності файлу
Візуалізація місця дії



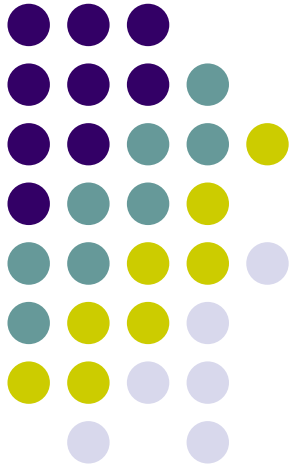
Вікно повідомлень



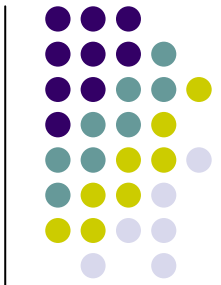
```
#include "colors.inc"

plane {  y, 0
  pigment { checker color Black color White }
  finish  {reflection 0.3                  }
}

sphere {  <-7, 3, -2>, 3
  pigment { color rgb <1, 0, 0> }
  finish  {reflection 0.4       }
}

sphere {  <0, 3, -1>, 3
  pigment { color rgb <0, 1, 0> }
  finish  {reflection 0.6       }
}

sphere {  <6.5, 3, 0>, 3
  pigment { color rgb <0, 0, 1> }
  finish  {reflection 0.8       }
}

camera {
  location <2, 4, 20>
  look_at  <0, 0, 0>
}

light_source {           <0, 20, 20>
  color rgb <1, 1, 1>
}
```

Просте Моделювання

Попрацюємо з файлом sphere.pov



Об'єкт (команда) *sphere (сфера)*



Формат:

sphere { <центр>, радіус [МОДИФІКАТОРИ_ОБ'ЄКТУ...] }

де

- <центр> в 3D координатах
- Радіус - значення
- МОДИФІКАТОРИ_ОБ'ЄКТУ такі ефекти як колір, текстура, прозорість, інші.

Зокрема, два загальні модифікатори це *pigment*{} та *finish*{}.

- *pigment*{ }: встановлює властивий колір об'єкту (колір, який залежить від освітленості, позиції інших об'єктів, і т.д.).

  Примітка: Пошук ‘colors.inc’ в меню допомоги в списку існуючих кольорів.
- *finish*{ }: встановлює зовнішні ефекти як наприклад відображення, високе освітлення, і т.д.

Властивість *finish (закінчення)*

Властивість *finish* має чисельні параметри. Найзагальніший це:

- *ambient (розсіяння)* <*величина*> де <*величина*> змінюється на проміжку від 0 до 1.

  Вказує наскільки розсіяне світло сприяє освітленню об'єкта

- *diffuse (розпливчастість)*<*величина*> де <*величина*> змінюється на проміжку від 0 до 1.

  Вказує наскільки світло відбивається від поверхні

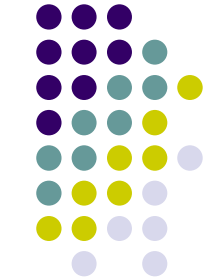
- *phong* та *phong_size*: одна частина основного освітлення (яскраві плями)

- *reflection (відображення)* <*величина*>: задається дзеркальність поверхні.

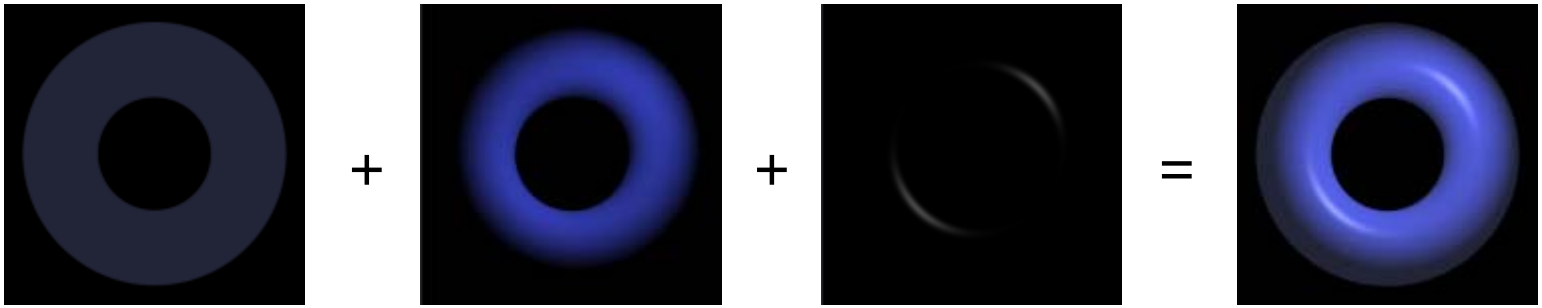
Отже створимо декілька сфер… Наприклад: одна маленька жовта та одна маленька зелена**.**

Проблема: нам потрібно мати систему координат! Особливо, нам потрібний знати де початок цієї системи

Модель віддзеркалення Фонга



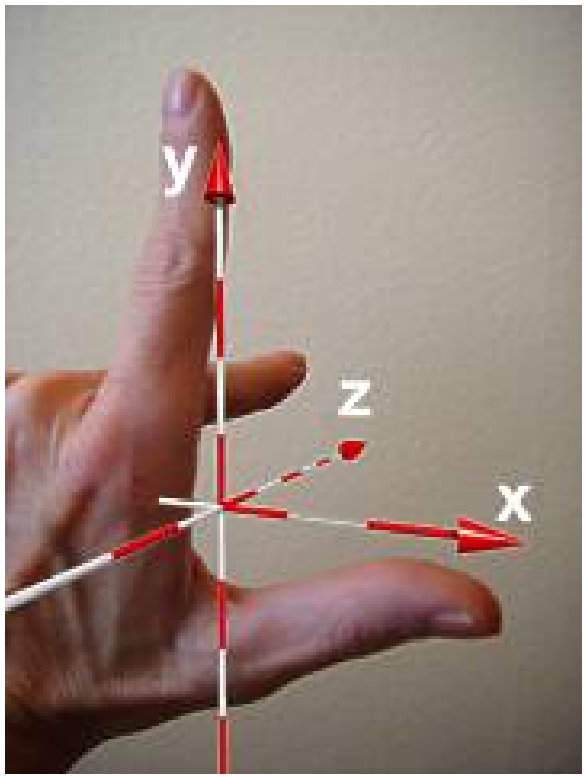
Задано джерело світла, поверхню, яка буде освітлюватися і спостерігач.Кожна точка поверхні має свої координати і в ній визначена нормаль до поверхні. Її освітленість складається з трьох компонент: фонове освітлення (ambient), розсіяне світло (diffuse) і складова відблиску (specular). Властивості джерела визначають потужність випромінювання для кожної з цих компонент, а властивості матеріалу поверхні визначають її здатність сприймати кожен вид освітлення.

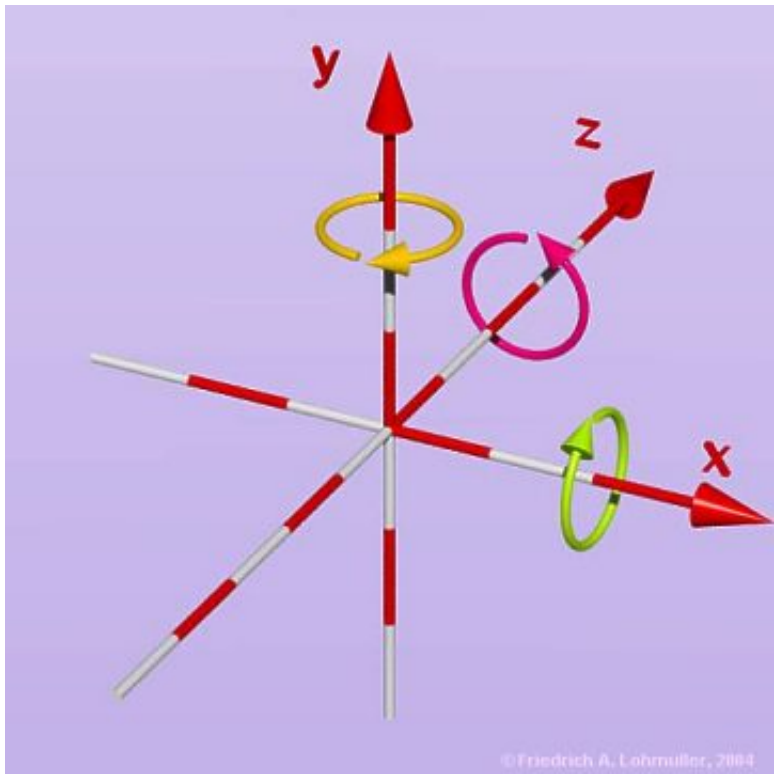


Фонова складова | Розсіяна складова | Дзеркальна складова | Сумарне освітлення

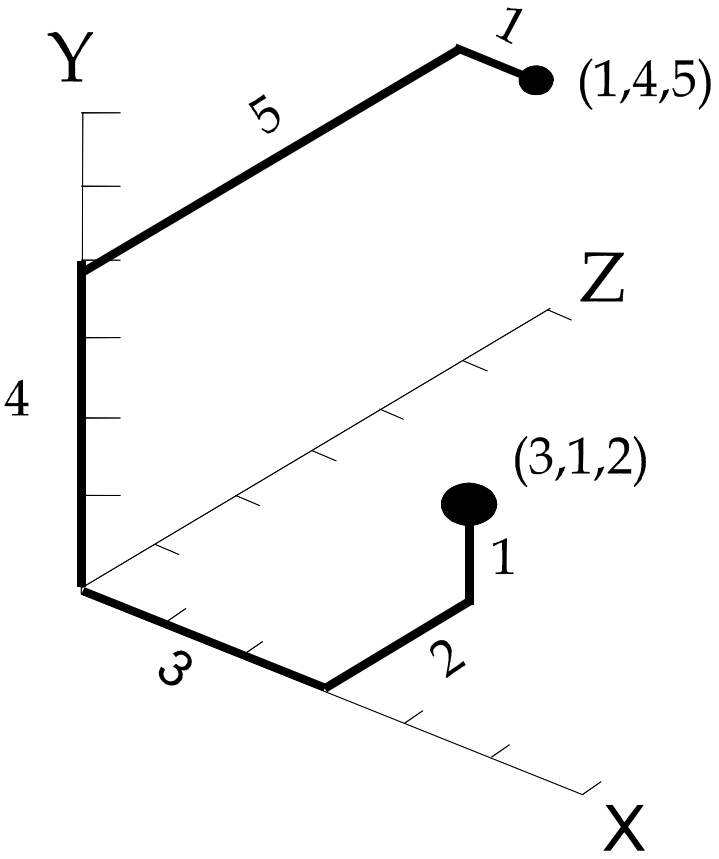
Основи візуалізації: Координатна система

- Ліва рука

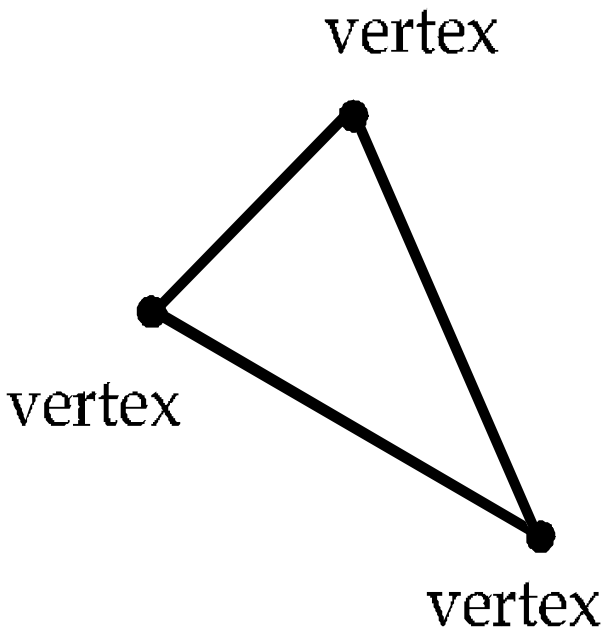




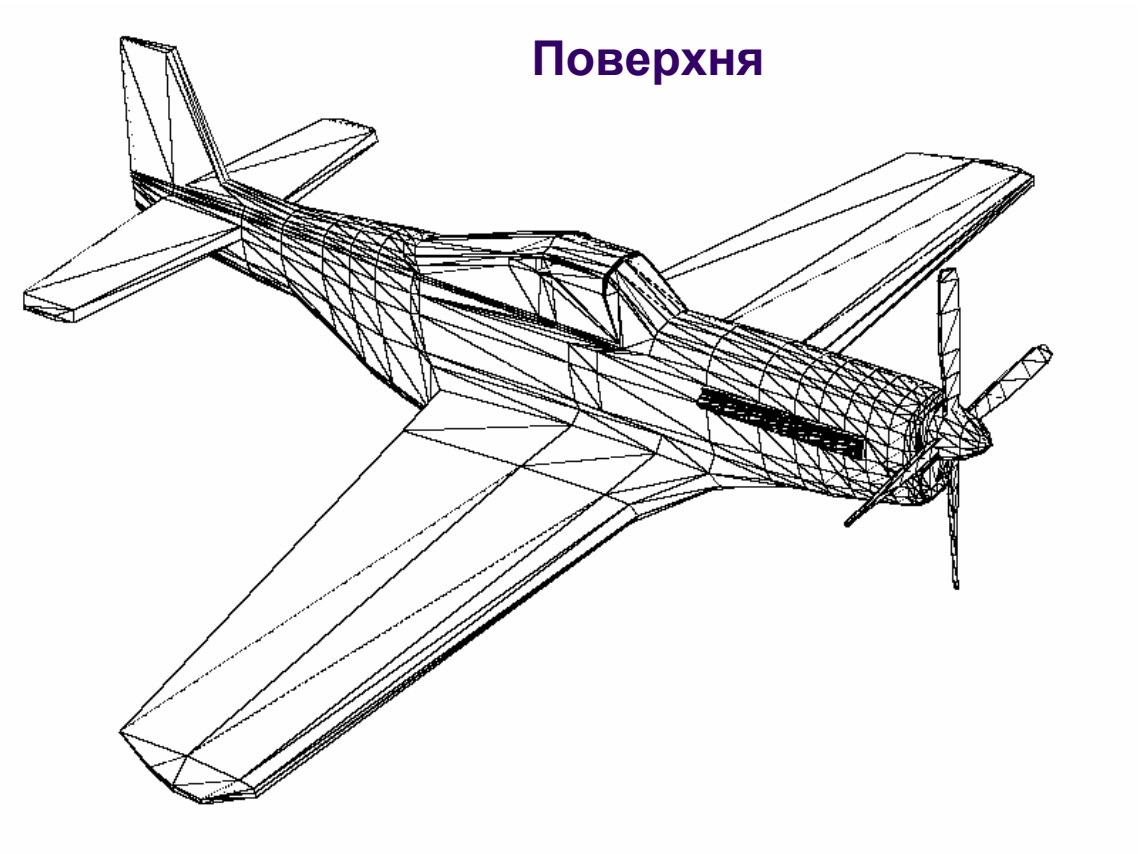
Основи візуалізації: Точка / Загальна точка



Основи візуалізації: Трикутник



Основи візуалізації: Багато трикутників =

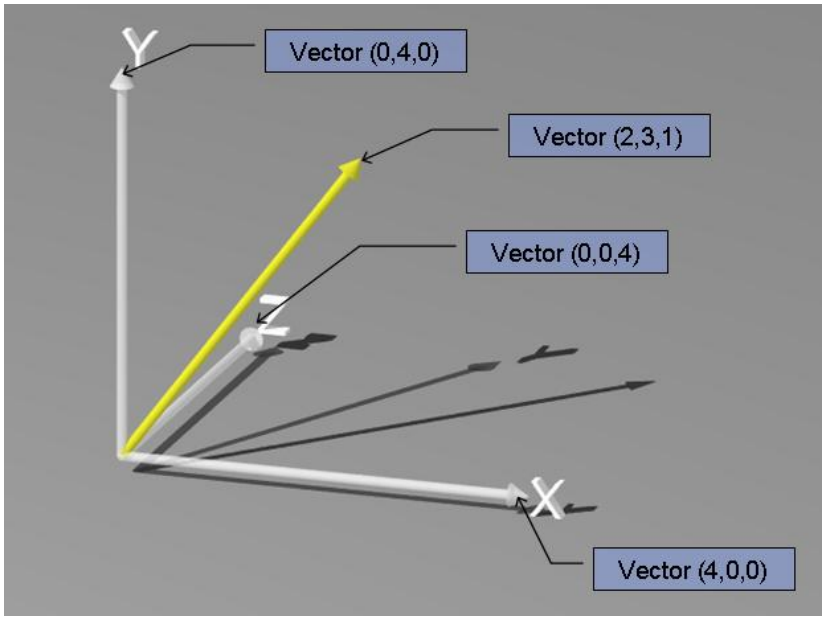


Основи візуалізації: Вектор

- Вектор має напрям і довжину
- Зазвичай, вектор вказується, використовуючи координати точок <x,y,z>

з початком від центру системи координат:

  - Основа вектора знаходиться в координаті <0,0,0>
  - Вершина вектора знаходиться в координаті <x,y,z>
  - Довжина вектора визначається як: $\sqrt{x^2 + y^2 + z^2}$



Приєднання файлів

В POV-Ray, .inc файли містять заздалегідь записані коди і інструменти, які можуть використовуватися в .pov файлі. В файлах .inc …

- **Використовується мова опису місця дії (*scene description language* - SDL) - це код, що використовується в .pov файлах**
- **В цих файлах побудовані певні об'єкти без специфічних ефектів, освітлення ...**
- **Використовується як: #include “ *name of file* ”**
- **POV-Ray шукає цей файл**
  1. **Спершу в директорії типової установки для POV-Ray**
  2. **Потім в робочому каталозі (де .pov файл знаходиться)**

Приклад:

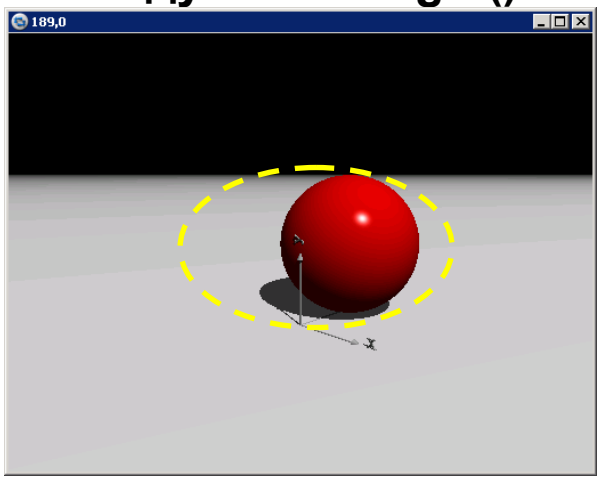
• Файл *colors.inc* у частині типової установки для POV-Ray. Цей файл описує ряд кольорів і їх властивостей. Пошук ‘colors.inc’ в меню допомоги, допоможе вам переглянути список вбудованих кольорів.

• В*и можете використовувати colors.inc в POV-RAY, АЛЕ НЕ МОЖЕТЕ МОДИФІКУВАТИ ЙОГО!!!!!*

Origin.inc файл

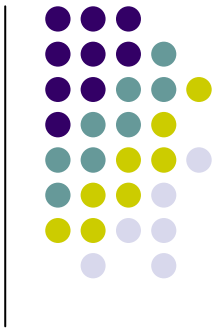
Я створив Origin.inc файл, який дозволить мені показати точку відліку та напрями координатних осей, коли ми будемо розвивати місце дії.

- **розмістіть копію *Origin.inc* файлу у вашому поточному робочому каталозі**
- **пропишіть: #include “Origin.inc” у вашому .pov файлі**
- **для того щоб показати точку відліку та напрями координатних осей вам необхідно написати в .pov файлі команду ShowOrigin()**



- **як альтернатива, ви можете використати ShowOrigin_color(колір), для того щоб змінити колір осей з сірого на кольоровий.**

Об'єкт *plane*



Формат:

plane { <координати>, відстань [МОДИФІКАТОРИ_ОБ'ЄКТУ...] }

де

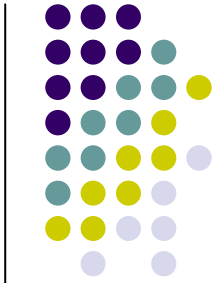
- <координати> вказуються координати вектора перпендикулярно якого розташовується план
- відстань - нульова мітка плану.
- МОДИФІКАТОРИ_ОБ'ЄКТУ такі ефекти як колір, текстура, прозорість, інші.

На перший погляд об'єкт *plane* може здатися дещо складним. Коли ми перейдемо до розгляду більш складних геометричних об'єктів, то ви зрозумієте на скільки даний об'єкт простий.

Як порада, для початку надавайте окраску (*pigment*) плану у вигляді шахової дошки, приклад:

```
plane { <0,1,0>, 0
    pigment { checker White Gray80 }
    finish {ambient 0.2   diffuse 0.8}
}
```

Об'єкт *camera*



Об'єкт *camera* має багато параметрів, які необхідно настроювати, але самі головні параметри це:

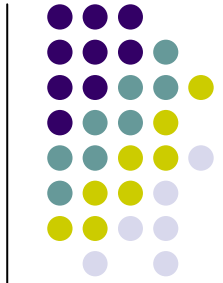
camera{
location *координата*
look_at *координата*
}

де:

- location **координата точки з якої ми дивимось**
- look_at **координата точки на яку направлений погляд**

Додатково, ми можемо конкретизувати кут (*angle*) зору, який дозволяє обширно поглянути на місце дії.

Об'єкт *light_source*



Об'єкт light_source також має багато параметрів, які необхідно настроювати, але самі головні параметри це:

light_source {
    < *координата* >, *КОЛІР*
    [МОДИФІКАТОРИ_ОСВІТЛЕННЯ...]
}

де:

- < *координата* > **координата точки розташування джерела освітленості**
- *КОЛІР* **– колір джерела освітленості**
- МОДИФІКАТОРИ_ОСВІТЛЕННЯ **дану властивість ми розглянемо пізніше**

Примітка: джерела освітленості випромінюють світло, АЛЕ САМОГО ДЖЕРЕЛА ОСВІТЛЕНОСТІ НЕ ВИДНО!!!