

# ЗАФАРБОВУВАННЯ

Слайди до лекцій з дисципліни  
«Математичні та алгоритмічні основи комп'ютерної графіки»

Лектор: к.т.н., доцент Сулема Є.С.

Каф. ПЗКС, ФПМ, КПІ ім. Ігоря Сікорського

2019/2020 навч. рік

# Що розуміють під зафарбовуванням?

- **2D** ⇒ заповнення кольором (*filling*)
- **3D** ⇒ розрахунок кольору (*shading* + *lightening* → *rendering*)

# Алгоритми заповнення кольором



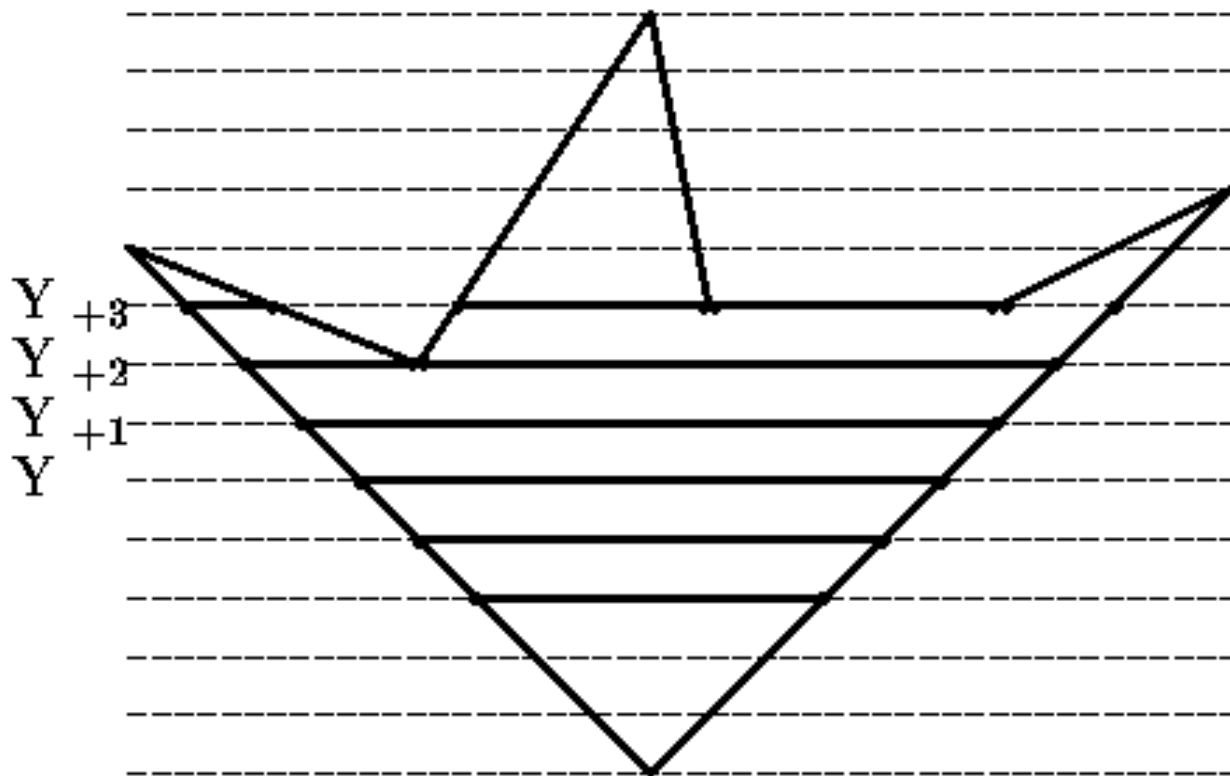
# Варіанти заповнення

- Заповнення кольором внутрішньої частини багатокутника, що заданий координатами вершин (алгоритм порядкового заповнення);
- зафарбовування області, яка або окреслена границею іншого кольору, або зафарбована іншим кольором (алгоритм заповнення з початкової точки).

# Алгоритм порядкового заповнення

- Алгоритм базується на припущенні, що сусідні піксели у рядку однакові та змінюються лише там, де рядок перетинається з ребром багатокутника. Це називається когерентністю растрових рядків (рядки сканування  $Y_i$ ,  $Y_{i+1}$ ,  $Y_{i+2}$  на рисунку). При цьому достатньо визначити X-координати перетинів рядків сканування з ребрами багатокутника. Пари відсортованих точок перетину задають інтервали зафарбування.

- Крім того, якщо які-небудь ребра перетинались  $i$ -м рядком, то вони скоріше за все будуть перетинатися також і рядком  $i+1$  (рядки сканування  $Y_i$  та  $Y_{i+1}$  на рисунку). Це називається когерентністю ребер.



При переході до нового рядка легко обчислити нову X-координату точки перетину ребра, використовуючи X-координату старої точки перетину і тангенс кута нахилу ребра:

$$X_{i+1} = X_i + 1/k,$$

де  $k$  – тангенс кута нахилу ребра.

$$k = dy/dx,$$

оскільки  $dy = 1$ , то  $1/k = dx$ .


Зміна ж кількості інтервалів зафарбовування виникає лише тоді, коли в рядку сканування з'являється вершина.


- Для кожного рядка сканування розглядаються лише ті ребра, які перетинають рядок. Вони задаються списком активних ребер (САР).
- При переході до наступного рядка для ребер, що перетинаються, переобчислюються X-координати перетинів.
- При появі в рядку сканування вершин виконується перебудова САР.
- Ребра, які перестали перетинатися, видаляються з САР, а всі нові ребра, що перетинаються рядком, заносяться у нього.



# Кроки алгоритму

1. Підготувати службові цілочисельні масиви  $Y$ -координат вершин та номерів вершин.
2. Одночасно відсортувати  $Y$ -координати по зростанню та масив номерів вершин для того, щоб можна було визначити вихідний номер вершини.
3. Визначити границі зафарбовування по осі  $Y$  –  $Y_{\min}$  та  $Y_{\max}$ . Починаючи з поточного значення  $Y_c = Y_{\min}$ , виконувати пункти 4-9 до завершення зафарбовування.
4. Визначити число вершин, що розташовані у поточному рядку сканування (рядку з  $Y_c$ ).

- 
5. Якщо вершини є, то для кожної з вершин доповнити САР, використовуючи інформацію про сусідні вершини. Для кожного ребра в САР заносяться:
    - максимальне значення Y-координати ребра,
    - приріст X-координати при збільшенні Y на 1,
    - початкове значення X-координати.
  6. Якщо знаходяться горизонтальні ребра, то вони просто зафарбовуються і інформація про них в САР не заносяться. Якщо після цього з'ясовується, що САР пустий, то зафарбовування закінчене.

- 
7. Відповідно до САР визначається наступна  $Y$ -координата найближчої вершини  $Y_n$ .
    - В циклі від  $Y_c$  до  $Y_n$ :
    - вибрати з САР та відсортувати  $X$ -координати перетинів активних ребер з рядком сканування;
    - визначити інтервали та виконати зафарбування;
    - переобчислити координати перетинів для наступного рядка сканування.
  8. Перевірити чи не досягли максимальну  $Y$ -координату. Якщо досягли, то зафарбування закінчене.
  9. Очистити САР від ребер, що закінчилися у рядку  $Y_n$  та перейти до пункту 4.

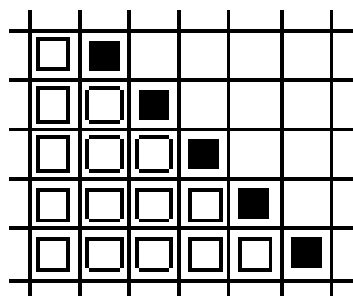
# Алгоритм заповнення з початкової точки

- В алгоритмі заповнення з початкової точки задається область, що зафарбовується, колір, яким буде зафарбовуватися ця область, та початкова точка в області, починаючи з якої виконується зафарбовування.

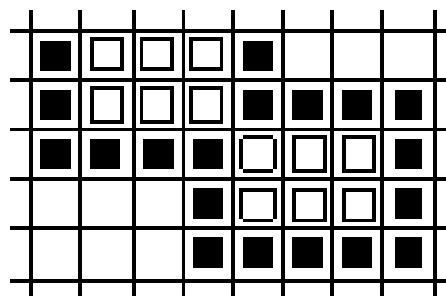
По способу завдання області поділяються на два типи:

- 1) гранично-визначені, що задаються своєю замкненою границею, причому такою, що колір пікселів границі відрізняється від кольору внутрішньої (тобто тієї, що перефарбовується) частини області. На колір пікселів внутрішньої частини області накладаються дві умови - він має відрізнятися від кольору пікселів границі та кольору перефарбування. Якщо всередині гранично-визначеної області є ще одна границя, що намальована пікселями того ж кольору, що і зовнішня границя, то відповідна частина області не повинна перефарбовуватися;
- 2) внутрішньо-визначені, що намальовані одним визначеним кольором. При зафарбовуванні цей колір замінюється на новий колір – колір зафарбовування.

- Область, що зафарбовується, або її границя – це певна зв'язна множина пікселів.
- По способу доступу до сусідніх пікселів області розділяються на 4-х та 8-ми зв'язні:



а)




б)

а) 4-х зв'язна внутрішньо-визначена область

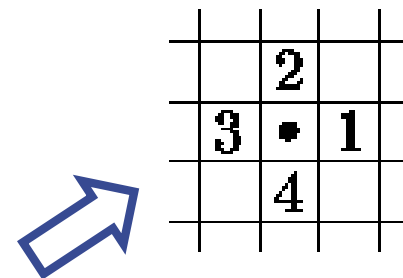
б) 8-ми зв'язна внутрішньо-визначена область

(білим кольором показані піксели області, а чорним – піксели границі)

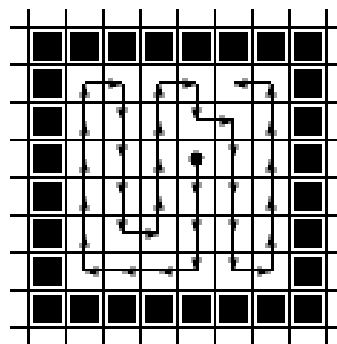


Важливо: для 4-х зв'язної прямокутної області границя 8-ми зв'язна (рис. а) і навпаки – у 8-ми зв'язної області границя 4-х зв'язна (рис. б). Отже, заповнення 4-х зв'язної області 8-ми зв'язним алгоритмом може призвести до "просочування" кольору через границю та зафарбовування пікселів в прилеглий області. Загалом, 4-х зв'язну область можна зафарбовувати як 4-х, так і 8-ми зв'язним алгоритмом. Зворотне ж невірно.

- Кроки ітеративного алгоритму заповнення 4-х зв'язної гранично-визначеної області:
  1. Помістити координати початкової точки в стек.
  2. Для кожного з 4-х сусідніх пікселів перевірити чи є він граничним чи вже перефарбований. Якщо ні, то занести його координати в стек.
  3. Поки стек не пустий:
    - а) отримати координати пікселя з стека,
    - б) перефарбувати піксел.



- Приклад порядку перебору сусідніх пікселів
- Порядок зафарбування простої гранично-визначеної області:





# Алгоритми розрахунку кольору



# Складові процесу розрахунку кольору

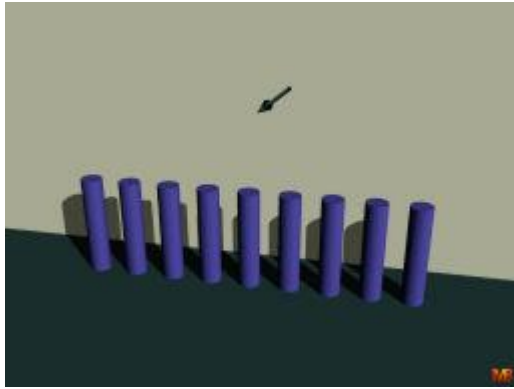
- Розрахунок освітлення (*lightening*) → модель освітлення
  - Розрахунок затінення (*shading*) → метод зафарбовування
- 
- Візуалізація (*rendering*) → накладання текстур, освітлення, затінення, застосування ефектів (наприклад, туман)

# Види освітлення

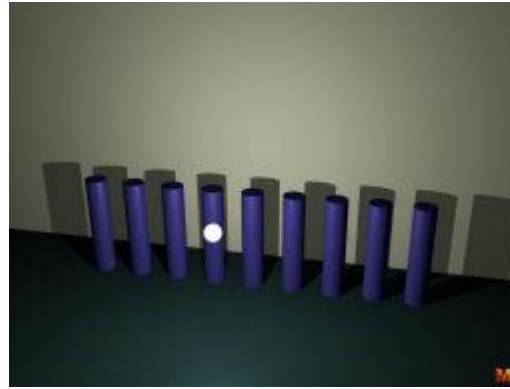
- Глобальне освітлення – беруться до уваги властивості всієї сцени,
- локальне освітлення – береться до уваги матеріал, геометрія поверхні та характеристики світла (колір, відстань від джерела світла, тип джерела світла).

# Типи джерел освітлення

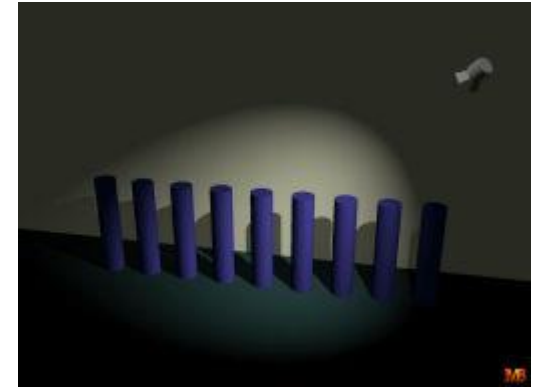
- Навколишнє світло (*ambient*) – непряме та не спрямоване всебічне освітлення;
- нескінченно віддалене джерело (*infinite light*) – спрямоване всебічне освітлення (вектор освітлення вважається однаковим для всіх об'єктів в сцені);
- точкове джерело (*point light*) – світло рівномірно випромінюється з певної точки в усі боки;
- прожектор (*spotlight*) – спрямоване світло, що випромінюється з певної точки в межах деякого кута.



- Нескінченно віддалене джерело: напрямок та інтенсивність світла однакові для всіх точок.



- Точкове джерело: інтенсивність світла обернено пропорційна відстані до джерела.



- Прожектор: область світла – конус, інтенсивність зменшується зі збільшенням кута.

# Типи матеріалів

- Матеріали, що розсіюють світло (*diffuse*) – наприклад, тканина, камінь, дерево;
- матеріали, що віддзеркалюють світло (*specular*) – наприклад, дзеркальні та металеві поверхні;
- матеріали, пропускають світло (*translucent*) – наприклад, вода, скло (тобто прозорі та напівпрозорі матеріали).

# Моделі освітлення

- Інтенсивність навколишнього світла в будь-якій точці поверхні:

$$I_{amb} = K_a \cdot I_a,$$

де  $K_a$  – коефіцієнт відбиття поверхні,

$I_a$  – інтенсивність навколишнього світла.

- Дифузійне відбиття світла точкового джерела від ідеального розсіювача визначається за законом Ламберта:

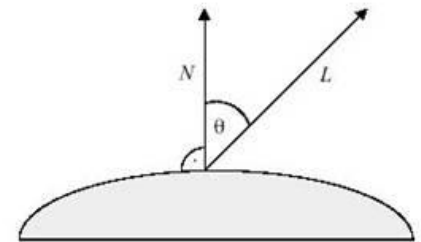
$$I_r = I_p \cdot P_d \cdot \cos(\theta),$$

де  $I_r$  – інтенсивність відбитого світла,

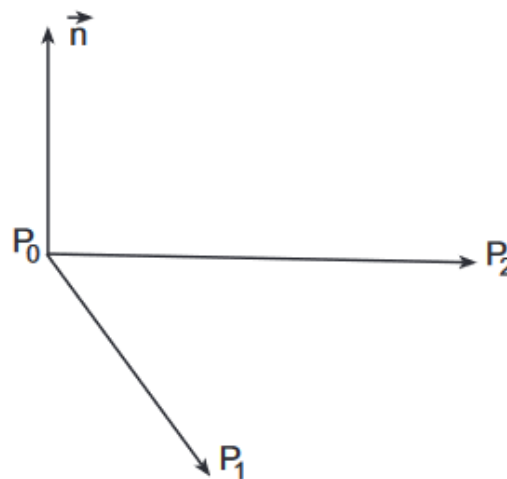
$I_p$  – інтенсивність точкового джерела,

$0 \leq P_d \leq 1$  – коефіцієнт дифузійного відбиття,

$0 \leq \theta \leq \pi/2$  – кут між напрямком світла та нормалю до поверхні.



Обчислення нормалі до поверхні ґрунтується на тому, що поверхня – це полігон, що задається принаймні 3 точками:



$$\text{Тоді } \vec{n} = \overrightarrow{P_0P_1} \times \overrightarrow{P_0P_2}.$$



В реальних сценах, окрім світла від точкових джерел, присутнє й розсіяне світло, яке спрощено враховується за допомогою коефіцієнту розсіювання:

$$I = I_r \cdot P_r + I_p \cdot Pd \cdot \cos(\theta),$$

де  $I_r$  – інтенсивність розсіяного світла,

$0 \leq P_r \leq 1$  – коефіцієнт відбиття розсіяного світла.

Інтенсивність світла зменшується залежно від відстані та середовища поширення:

$$I = I_r \cdot P_r + \frac{I_p \cdot Pd \cdot \cos(\theta)}{d + K},$$

де  $d$  – відстань від центра проекції до об'єкта,

$K > 0$  – довільна константа.

Обчислення інтенсивності відбитого світла від дзеркальної поверхні (дзеркальне відбиття) визначається емпіричною моделлю Фонга:

$$I_s = I_p \cdot W(\lambda, \theta) \cdot \cos^n(\varphi),$$

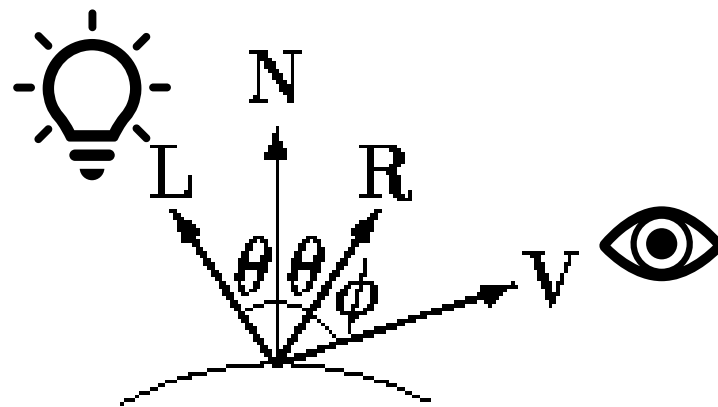
де  $W(\lambda, \theta)$  – крива відбиття;

$$-\pi/2 \leq \varphi \leq \pi/2,$$

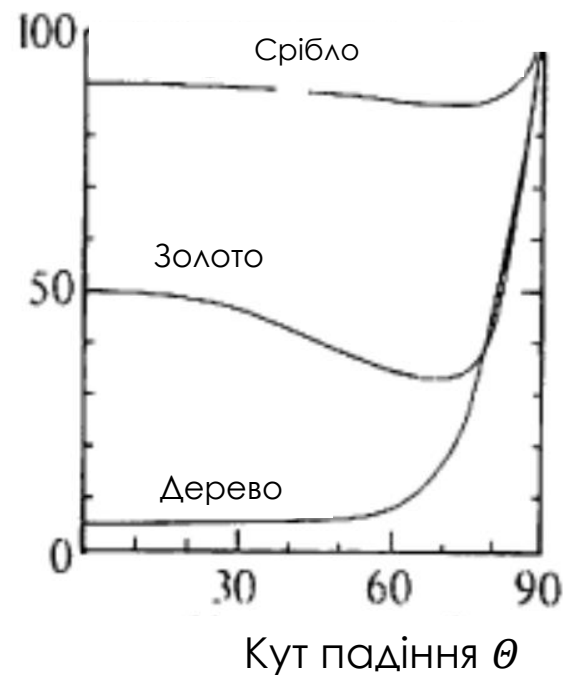
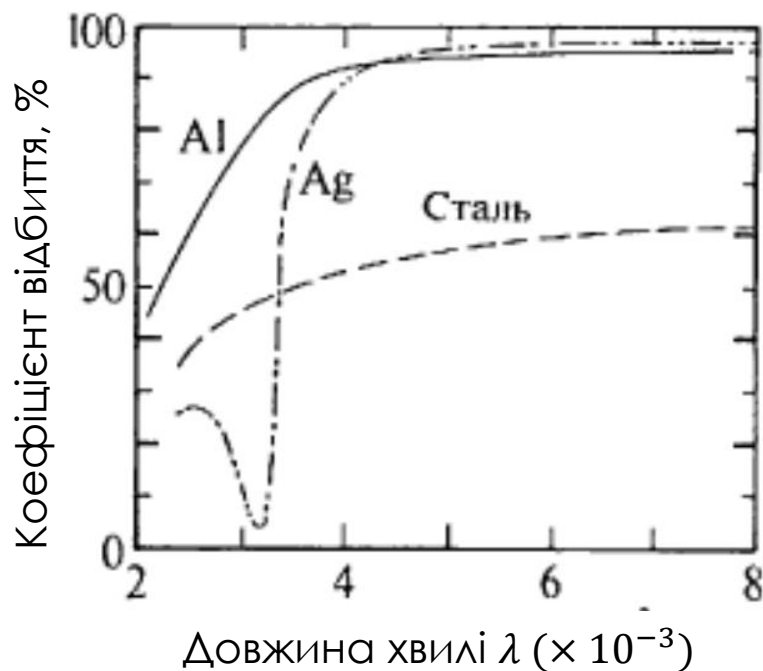
$$1 \leq n \leq 200.$$

Для ідеального відбивача  $n = \infty$ .

Для матових, негладких поверхонь типу крейди або сажі  $n \approx 1$ .



$W(\lambda, \theta)$  – крива відбиття, що визначає відношення дзеркально відбитого світла до світла, що падає, як функцію кута падіння  $\theta$  та довжини хвилі  $\lambda$ :



Для спрощення замість знаходження  $W(\lambda, \theta)$  використовують емпірично визначену константу ( $K_s$ ).

Сумарна модель освітлення:

$$I = I_r P_r + \frac{I_p}{d + K} (P_d \cos(\theta) + K_s \cos^n(\varphi))$$

- Якщо джерел світла декілька:

$$I = I_r P_r + \sum_{j=1}^m \frac{I_{p_j}}{d + K} (P_d \cos(\theta_j) + K_s \cos^n(\varphi_j))$$

де  $m$  – кількість джерел світла.

$$\cos \theta = \frac{\bar{N} \cdot \bar{L}}{|\bar{N}| \cdot |\bar{L}|} = \bar{N}_e \cdot \bar{L}_e,$$

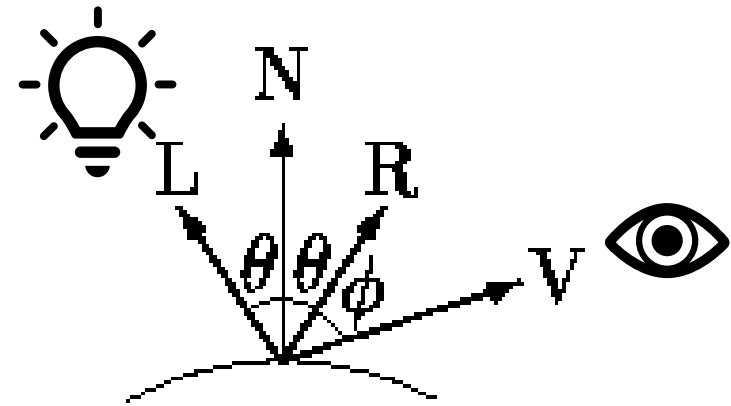
де  $\bar{N}_e$  та  $\bar{L}_e$  – одиничні вектори нормалі та світла відповідно.

$$\cos \varphi = \frac{\bar{R} \cdot \bar{V}}{|\bar{R}| \cdot |\bar{V}|} = \bar{R}_e \cdot \bar{V}_e,$$

де  $\bar{R}_e$  та  $\bar{V}_e$  – одиничні вектори нормалі та світла відповідно.

Розглянемо приклад розрахунку для металевого об'єкта за умови, що:

- $I_r = 1, I_p = 10,$
- $K_s = 0.8, P_r = P_d = 0.15, n = 5,$
- $d = 0, K = 1.$



Нехай  $\bar{N} = \bar{j}, \bar{L} = -\bar{i} + 2\bar{j} - \bar{k}, \bar{V} = \bar{i} + 1.5\bar{j} + 0.5\bar{k}.$

Тоді вектор відбиття:  $\bar{R} = \bar{i} + 2\bar{j} + \bar{k}.$

Знайдемо кути:

$$\cos \theta = \frac{\bar{j} \cdot (-\bar{i} + 2\bar{j} - \bar{k})}{1 \cdot \sqrt{(-1)^2 + 2^2 + (-1)^2}} = \frac{2}{\sqrt{6}} \Rightarrow \theta = \arccos \frac{2}{\sqrt{6}} \approx 35^\circ;$$

$$\cos \varphi = \frac{(\bar{i} + 2\bar{j} + \bar{k}) \cdot (\bar{i} + 1.5\bar{j} + 0.5\bar{k})}{\sqrt{1^2 + 2^2 + 1^2} \cdot \sqrt{1^2 + 1.5^2 + 0.5^2}} = \frac{9}{2\sqrt{21}} \Rightarrow \varphi \approx 11^\circ.$$

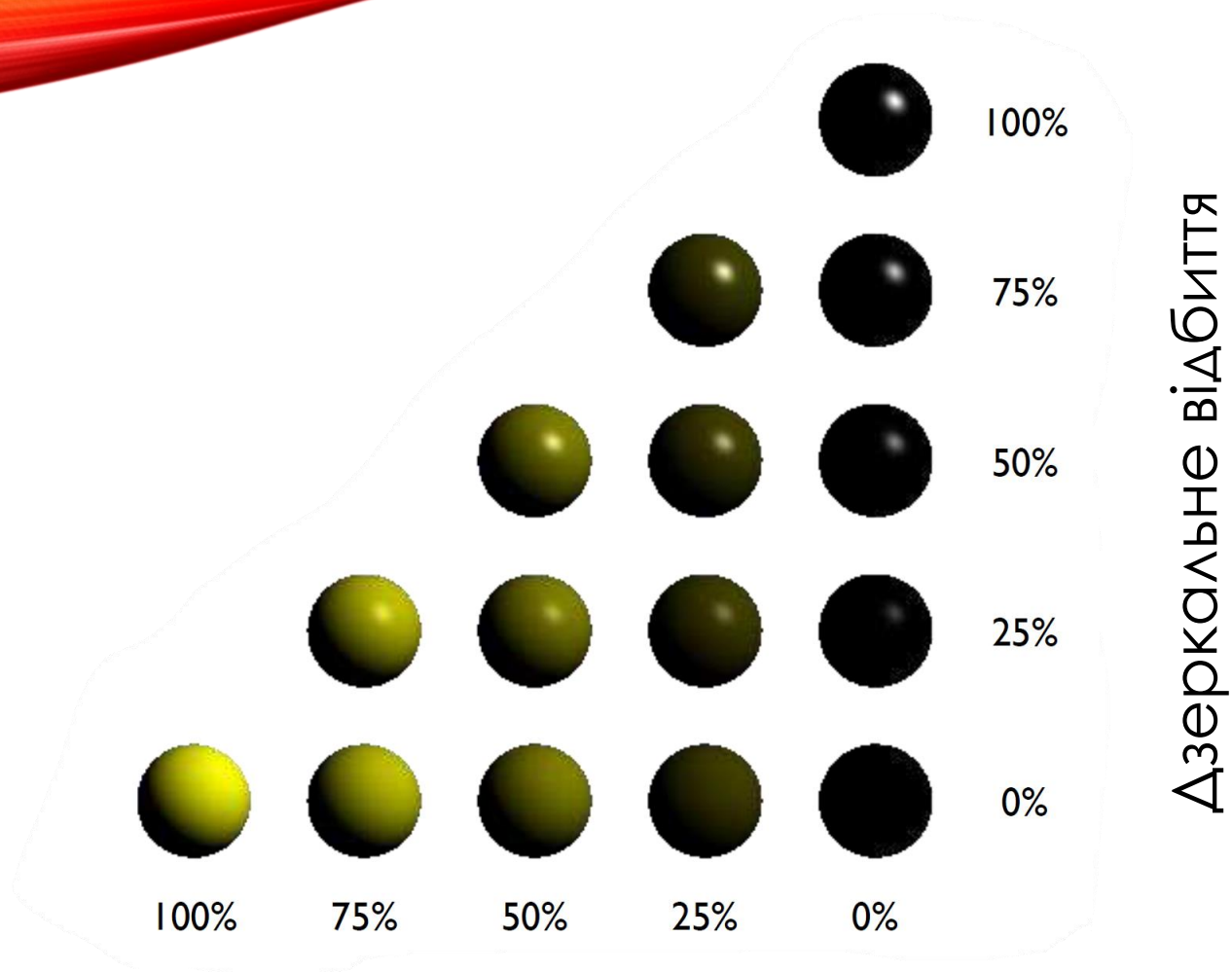
Знайдемо інтенсивність освітлення при таких кутах:

$$I = 1 \cdot 0.15 + \frac{10}{0 + 1} \left( 0.15 \cdot \frac{2}{\sqrt{6}} + 0.8 \cdot \left( \frac{9}{2\sqrt{21}} \right)^5 \right) = 8.67.$$

Якщо збільшити кут  $\varphi \approx 45^\circ$ , то інтенсивність зменшиться:  
 $I = 2.8$ .

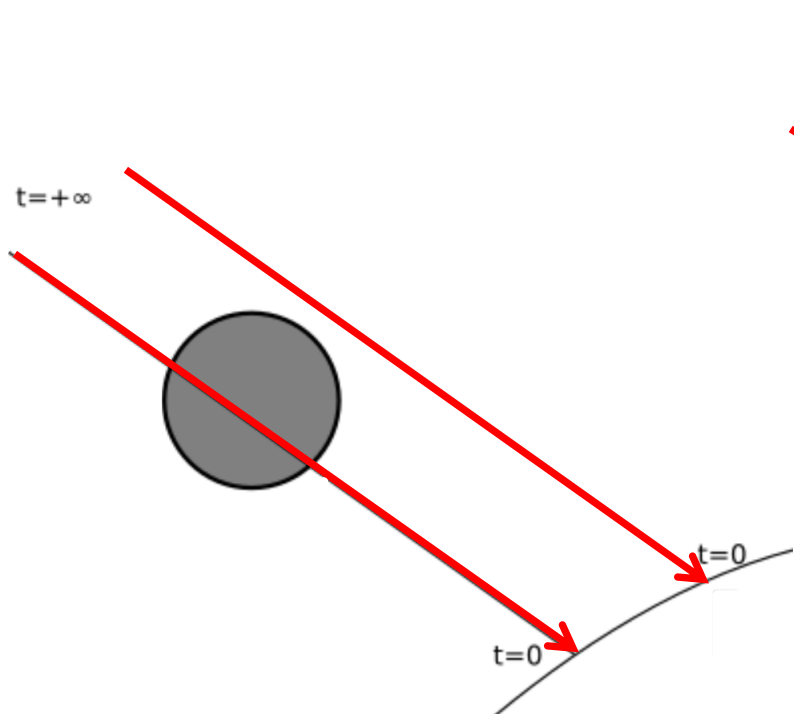
$\theta$	$\varphi$	$I$
10	11	8.92
35		8.67
60		8.19
10	45	3.04
35		2.80
60		2.32
10	85	1.63
35		1.38
60		0.90

**Запитання:** як це  
допоможе нам у  
визначенні кольору?

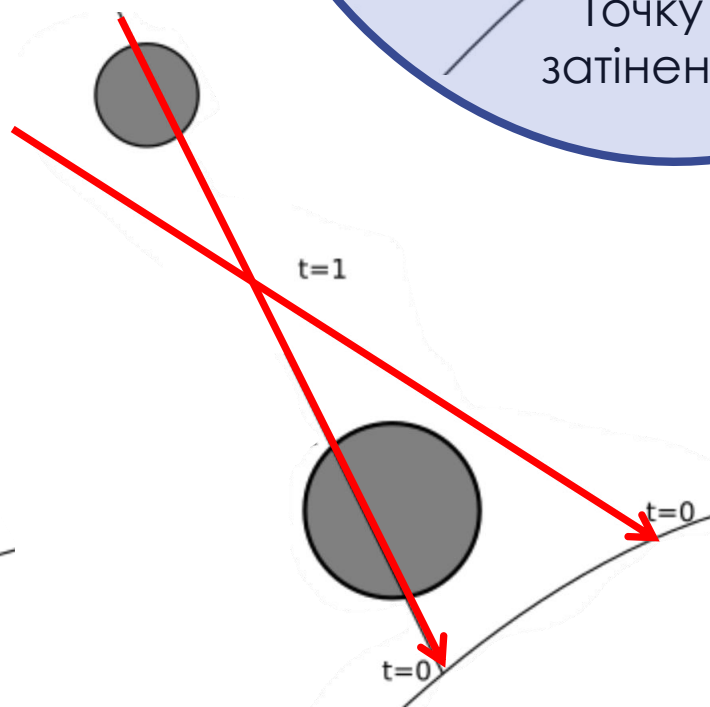


Дифузне відбиття

# Обчислення тіней



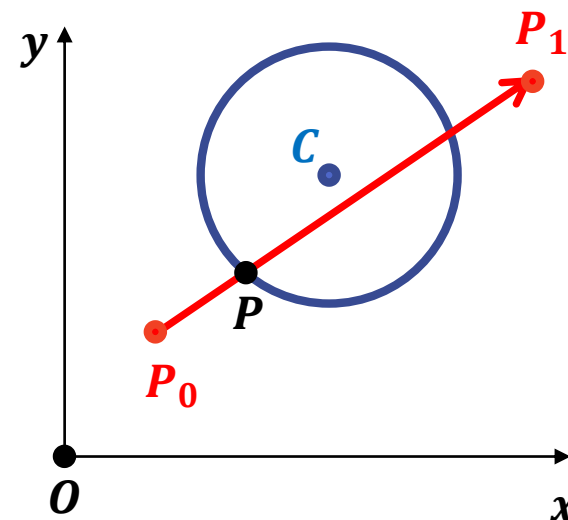
(а) нескінченно віддалене джерело



(б) точкове джерело



Розглянемо проекцію на площину екрану сфери з радіусом  $R$  та центром у точці  $C(x_C, y_C, z_C)$ , яка можливо розташована на шляху світла від джерела до інших об'єктів у сцені, та окремого променя світла, що проходить через точки  $P_0(x_0, y_0, z_0)$  та  $P_1(x_1, y_1, z_1)$ .



Розглянемо промінь світла як вектор. Тоді його рівняння:

$$\bar{p} = \bar{p}_0 + (\bar{p}_1 - \bar{p}_0) \cdot t.$$

Розглянемо коло з центром  $C(x_C, y_C)$ . Його рівняння:

$$(x - x_C)^2 + (y - y_C)^2 = R^2.$$

Це рівняння можна розглянути як скалярний добуток:

$$(\bar{p} - \bar{c}) \cdot (\bar{p} - \bar{c}) = R^2,$$

де  $\bar{p}(x, y)$ ,  $\bar{c}(x_C, y_C)$ .

Тоді:

$$(\bar{p}_0 + (\bar{p}_1 - \bar{p}_0) \cdot t - \bar{c}) \cdot (\bar{p}_0 + (\bar{p}_1 - \bar{p}_0) \cdot t - \bar{c}) = R^2.$$

Виконаємо заміну векторів:

$$(\overline{CP_0} + \overline{P_0P_1} t) \cdot (\overline{CP_0} + \overline{P_0P_1} t) = R^2$$

та скористаємось дистрибутивністю скалярного добутку:

$$\overline{P_0P_1} \cdot \overline{P_0P_1} t^2 + 2 \overline{CP_0} \cdot \overline{P_0P_1} t + \overline{CP_0} \cdot \overline{CP_0} - R^2 = 0$$

Введемо нові позначення:

$$k_1 = \overline{P_0P_1} \cdot \overline{P_0P_1}, \quad k_2 = 2 \overline{CP_0} \cdot \overline{P_0P_1}, \quad k_3 = \overline{CP_0} \cdot \overline{CP_0} - R^2.$$

Отже, отримуємо квадратне рівняння:

$$k_1 t^2 + k_2 t + k_3 = 0.$$

Його рішення

$$t_{1,2} = \frac{-k_2 \pm \sqrt{k_2^2 - 4k_1k_3}}{2k_1}$$

(якщо вони є) визначають точку/точки перетину кола з вектором, а отже й відповідають на питання, чи відкидає тінь сфера на інші об'єкти у сцені.

**Запитання:** який висновок щодо затінення можна зробити для випадків (а) та (б)?

Розглянемо приклад. Нехай задано наступні значення:

$$C(2,2), R = 1, P_0(0,0), P_1(4,3).$$

Тоді:

$$k_1 = \overline{P_0P_1} \cdot \overline{P_0P_1} = (x_1 - x_0)^2 + (y_1 - y_0)^2 = 4^2 + 3^2 = 25,$$

$$k_2 = 2 \overline{CP_0} \cdot \overline{P_0P_1} = 2(x_0 - x_C)(x_1 - x_0) + 2(y_0 - y_C)(y_1 - y_0) = -28,$$

$$k_3 = \overline{CP_0} \cdot \overline{CP_0} - R^2 = (x_0 - x_C)^2 + (y_0 - y_C)^2 - R^2 = (-2)^2 + (-2)^2 + 1 = 7.$$

Отримуємо квадратне рівняння:

$$25t^2 - 28t + 7 = 0.$$

Його рішення:

$$t_1 = \frac{28 - \sqrt{84}}{50} \approx 0.38 \text{ та } t_2 = \frac{28 + \sqrt{84}}{50} \approx 0.74.$$

Рівняння вектору:

$$\begin{cases} x = x_0 + (x_1 - x_0)t = 4t \\ y = y_0 + (y_1 - y_0)t = 3t \end{cases}$$

Отже, точки перетину кола та вектора:  $P_A(1.52, 1.14), P_B(2.96, 2.22)$ .

# Рендеринг

## Спрямоване освітлення

### Модель освітлення

Методи  
«растрового»  
тонування полігону

Метод кидання  
променів

Пласке  
тонування

Метод  
Гуро

Метод  
Фонга

## Глобальне освітлення

Дзеркальні  
поверхні

Метод трасування  
променів

Дифузні  
поверхні

Метод  
ВИПРОМІНЮВАНOSTI

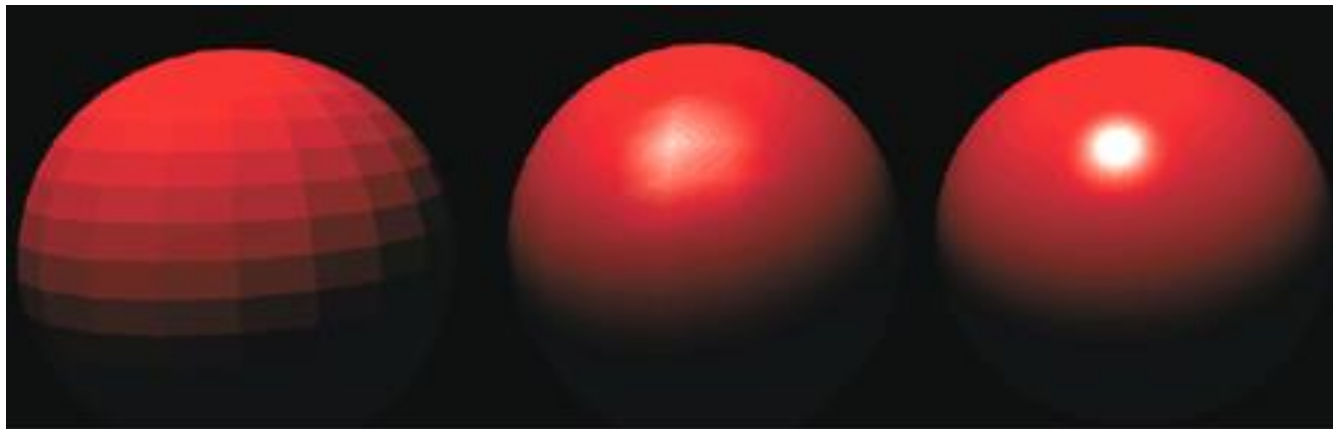
Метод фотонних  
мап

# Методи тонування (зафарбовування)

- Пласке тонування (*flat shading*)
- Метод Гуро (*Gouraud shading*)
- Метод Фонга (*Phong shading*)

гладке тонування  
(*smooth shading*)

«растрові» методи



Flat

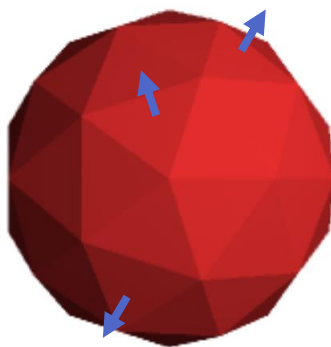
Gouraud

Phong

- 
- Метод кидання променів (*ray casting*)

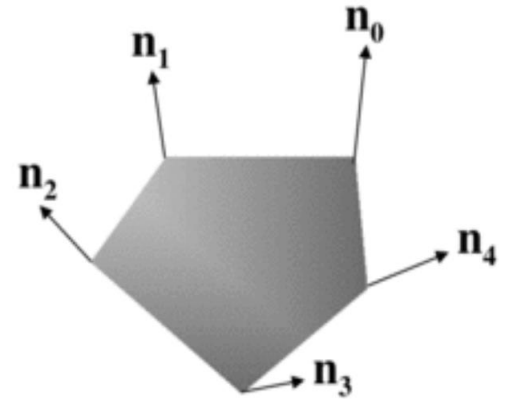
# Пласке тонування

- Розрахунок освітлення виконується для однієї точки полігону (встановлюється одна нормаль). Визначене значення кольору розповсюджується на всю поверхню полігону.
- Недолік: об'єкт має вигляд гранованого:



# Метод Гуро

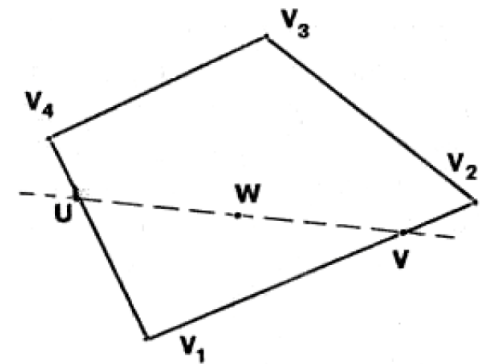
- Обчислюються нормалі до вершин полігону (усереднення для всіх сусідніх граней).
- Обчислюється інтенсивність кольору у вершинах.
- Відбувається інтерполяція значення кольору по лініях сканування.



$$I_U = (1 - u)I_{V_4} + uI_{V_1},$$

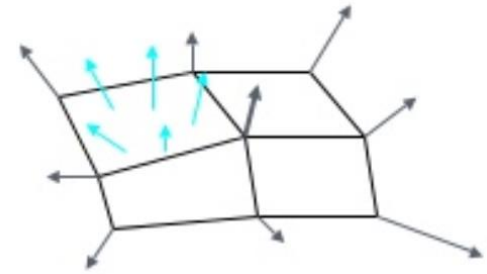
$$I_V = (1 - v)I_{V_1} + vI_{V_2},$$

$$\text{де } u = \frac{|V_4U|}{|V_4V_1|}, 0 \leq u \leq 1, v = \frac{|V_1V|}{|V_1V_2|}, 0 \leq v \leq 1.$$



# Метод Фонга

- Обчислюються нормалі до вершин полігону.
- Відбувається інтерполяція нормалей по лініях сканування.
- Обчислюється інтенсивність кольору у кожній точці.



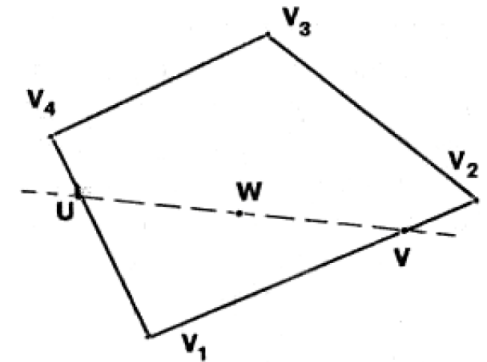
$$n_W = \frac{(1-t)n_U + tn_V}{|(1-t)n_U + tn_V|},$$

$$\text{де } t = \frac{|UW|}{|UV|},$$

$$n_U = (1-u)n_{V_4} + un_{V_1},$$

$$n_V = (1-v)n_{V_1} + vn_{V_2},$$

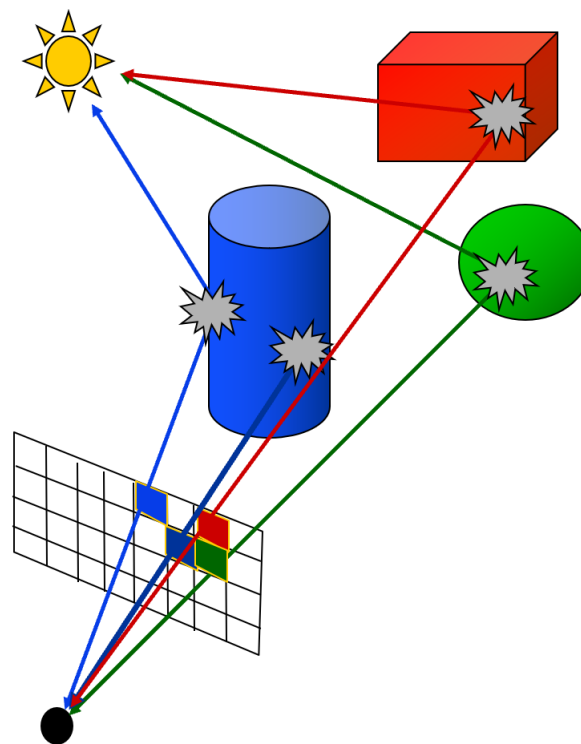
$$\text{де } u = \frac{|V_4U|}{|V_4V_1|}, v = \frac{|V_1V|}{|V_1V_2|}.$$





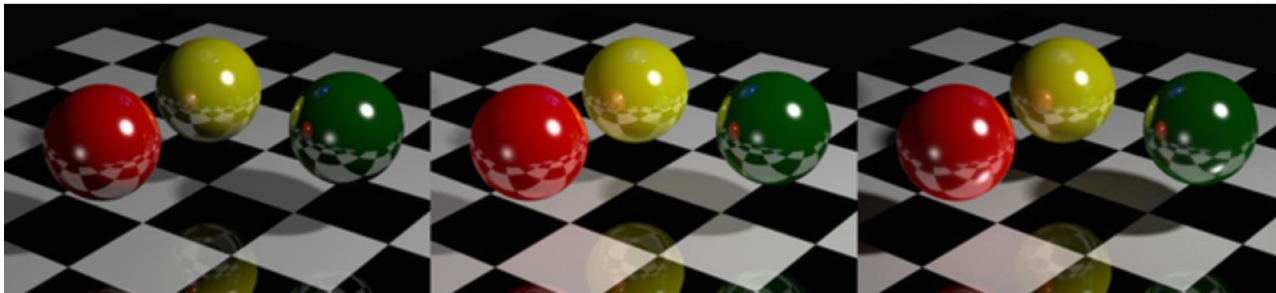
# Метод кидання променів

- Основні кроки:
  1. Генерація променя, який проходить через поточний піксель.
  2. Знаходження 1-ї поверхні, що перетинається з цим променем.
  3. Обчислення кольору на основі обраної моделі освітлення.



# Методи тонування

- Метод трасування променів (*ray tracing*)
- Метод випромінюваності (*radiosity method*)
- Метод фотонних мап (*photon mapping*)



# Трасування променів

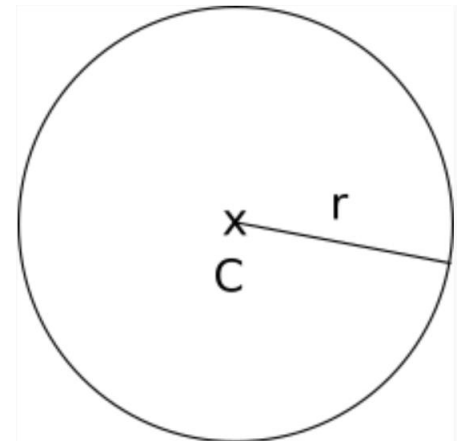
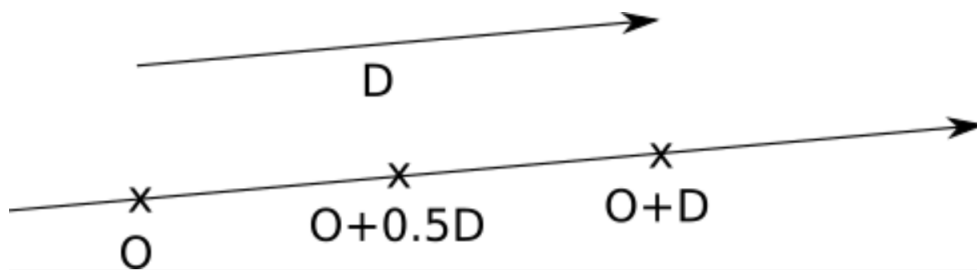
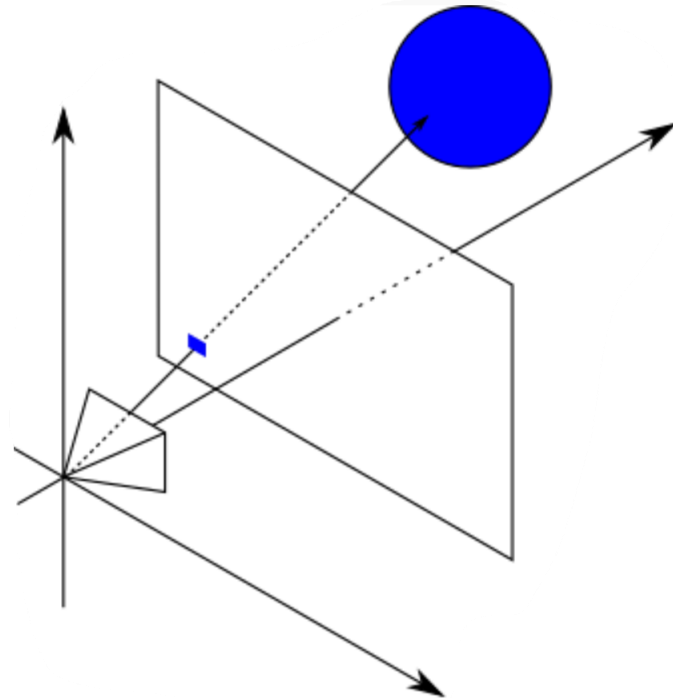
Два види трасування:

- пряме,
- зворотне.

$$\bar{P} = \bar{O} + (\bar{V} - \bar{O})t$$

$$\bar{D} = (\bar{V} - \bar{O})$$

$$\bar{P} = \bar{O} + \bar{D}t$$



$$\langle P - C, P - C \rangle = r^2$$

$$P = O + t\vec{D}$$

$$\langle O + t\vec{D} - C, O + t\vec{D} - C \rangle = r^2$$

$$\vec{OC} = O - C$$

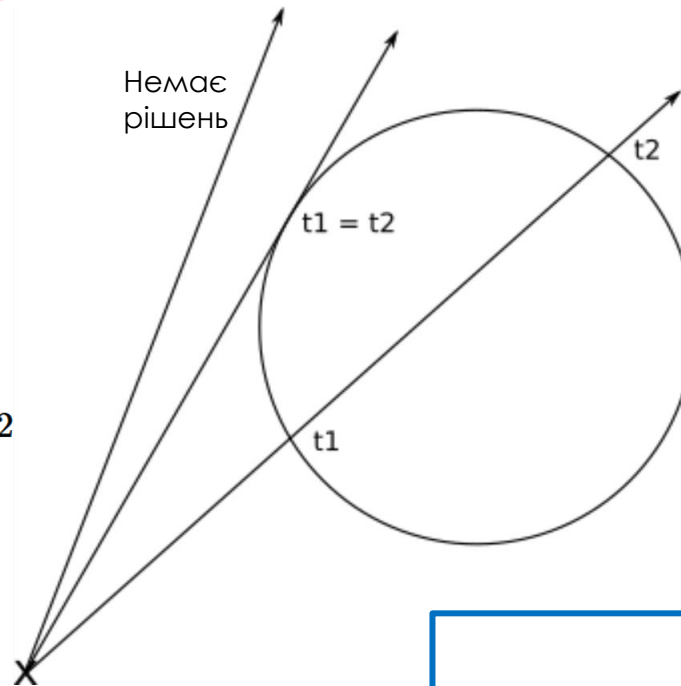
$$\langle \vec{OC} + t\vec{D}, \vec{OC} + t\vec{D} \rangle = r^2$$

$$\langle \vec{OC} + t\vec{D}, \vec{OC} \rangle + \langle \vec{OC} + t\vec{D}, t\vec{D} \rangle = r^2$$

$$\langle \vec{OC}, \vec{OC} \rangle + \langle t\vec{D}, \vec{OC} \rangle + \langle \vec{OC}, t\vec{D} \rangle + \langle t\vec{D}, t\vec{D} \rangle = r^2$$

$$\langle t\vec{D}, t\vec{D} \rangle + 2\langle \vec{OC}, t\vec{D} \rangle + \langle \vec{OC}, \vec{OC} \rangle = r^2$$

$$t^2 \langle \vec{D}, \vec{D} \rangle + t(2\langle \vec{OC}, \vec{D} \rangle) + \langle \vec{OC}, \vec{OC} \rangle - r^2 = 0$$



$$k_1 = \langle \vec{D}, \vec{D} \rangle$$

$$k_2 = 2\langle \vec{OC}, \vec{D} \rangle$$

$$k_3 = \langle \vec{OC}, \vec{OC} \rangle - r^2$$

$$k_1 t^2 + k_2 t + k_3 = 0$$

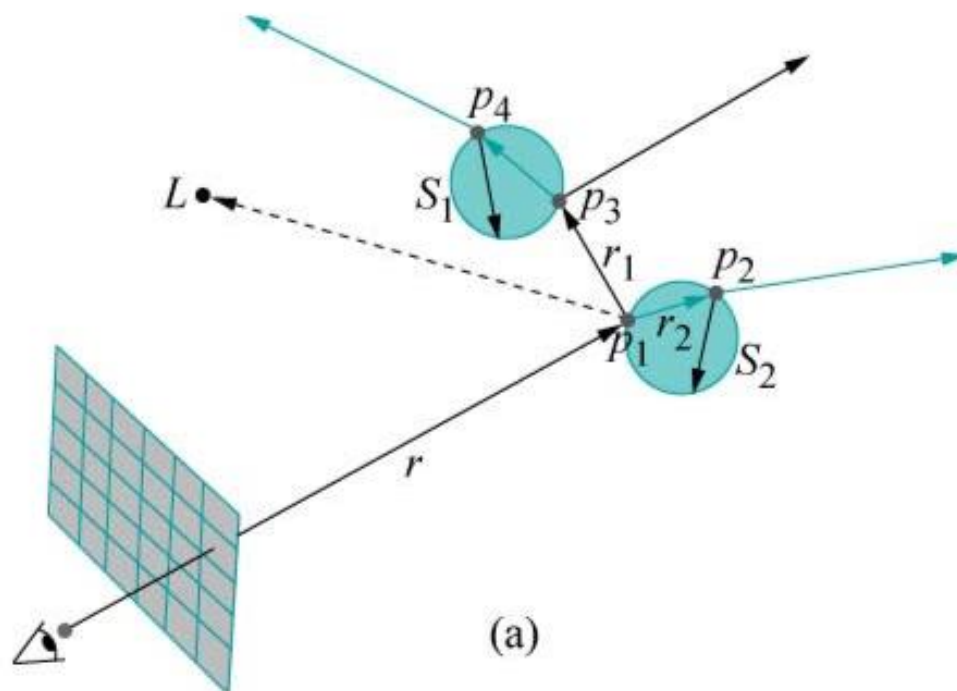
$$\{t_1, t_2\} = \frac{-k_2 \pm \sqrt{k_2^2 - 4k_1 k_3}}{2k_1}$$

$t > 1$  – сцена

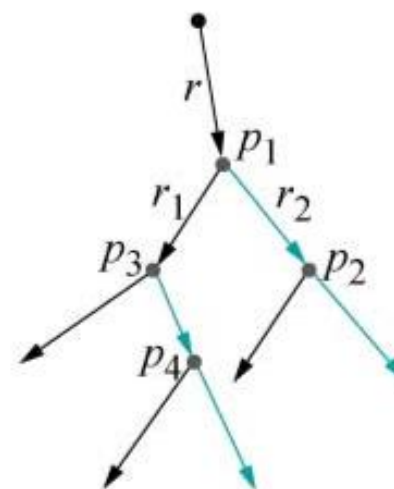
$t < 0$  – простір за камерою

$0 \leq t \leq 1$  – простір між камерою та сценою

- Рекурсивный процесс:



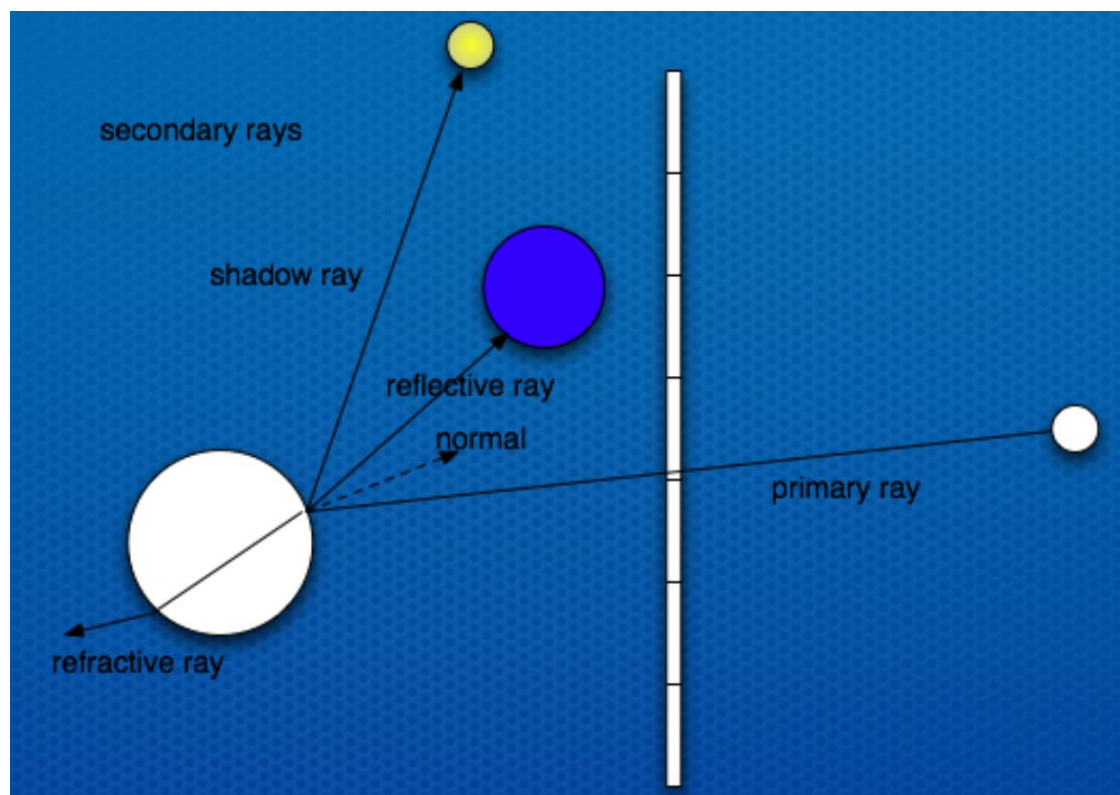
(a)



(b)

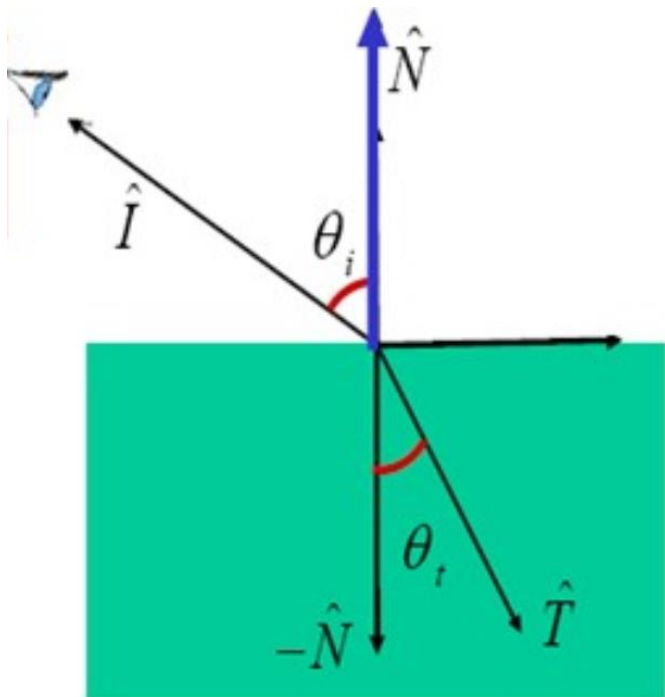
Типи променів:

- первинні,
- вторинні (відбиття, заломлення, тінь).



Обчислення вторинних променів є головною відмінністю методу трасування променів від методу кидання променів, де використовуються лише первинні промені.

- Врахування заломлення – закон Снела:



$$\frac{\sin \theta_i}{\sin \theta_t} = \frac{\eta_t}{\eta_i} = \eta_r$$







Основні кроки алгоритму трасування променів:

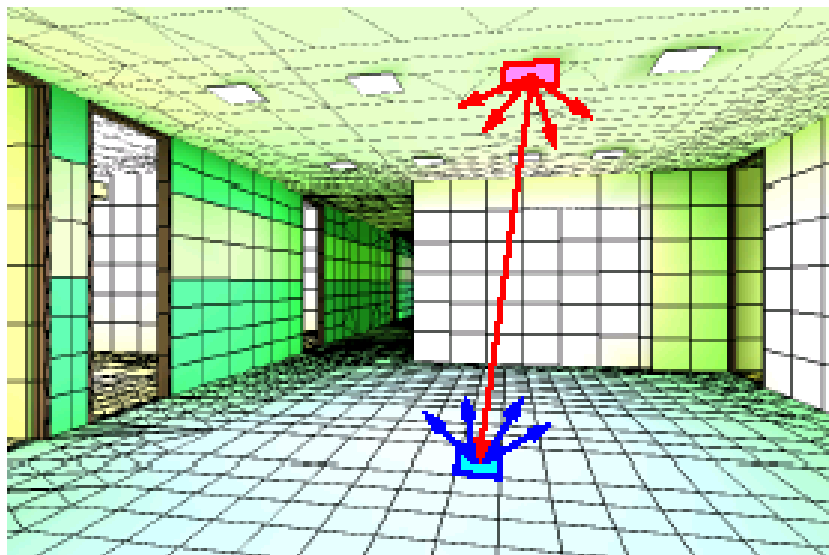
- Для кожного пікселя:

1. Згенерувати первинний промінь.
2. Перевірити, чи перетинається промінь з поверхнею.
3. Якщо є перетин, згенерувати вторинні промені:
  - промінь, що відбивається – для всіх поверхонь,
  - промінь, що заломлюється – для прозорих поверхонь.
4. Для кожного вторинного променю – перейти на п.2.



# Метод випромінюваності

- Поверхня кожного об'єкту у сцені складається з фрагментів (*patch*), кожен з яких характеризується однаковим константним значенням світлової енергії;
- Кожен фрагмент / полігон є джерелом світла;
- «Взаємодія» між дифузними поверхнями;
- Закон збереження енергії.



## Відмінність від трасування променів:

- не залежить від точки перегляду;
- працює у просторі об'єкту (замість простору зображення);
- глобальне освітлення;
- дифузне відбиття.

## Рівняння випромінюваності:

$$B_i = E_i + \rho_i \sum_j B_j F_{ij} ,$$

де  $B_i$  – світлова енергія, що «належить»  $i$ -ому фрагменту,

$B_j$  – світлова енергія, що «належить»  $j$ -ому фрагменту,

$E_i$  – енергія випромінювання  $i$ -го фрагмента (дорівнює нулю, якщо ця поверхня не є поверхнею джерела освітлення),

$\rho_i$  – коефіцієнт дифузного відбиття,

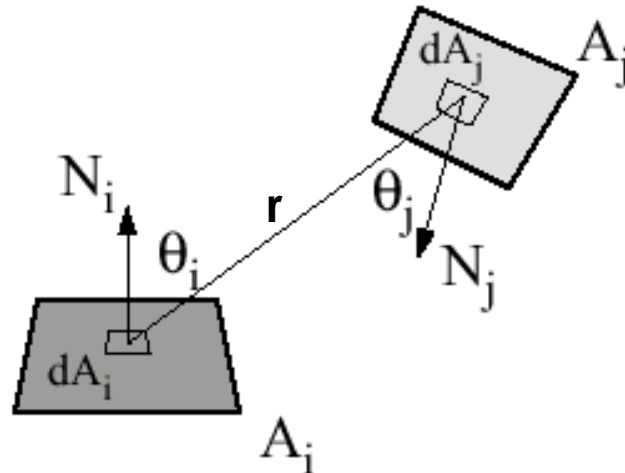
$F_{ij}$  – форм-фактор.

Форм-фактор визначає частку світлової енергії  $j$ -го фрагменту, яка досягла  $i$ -го фрагменту.

Обчислення форм-фактору:

$$F_{ij} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos \theta_i \cos \theta_j}{\pi r^2} V_{ij} dA_j dA_i$$

$$\text{де } V_{ij} = \begin{cases} 1 & \text{якщо } dA_j \text{ є видимим з } dA_i \\ 0 & \text{якщо } dA_j \text{ є невидимим з } dA_i \end{cases}$$



Рівняння у матричній формі:

$$\begin{bmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \cdot & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \cdot & -\rho_2 F_{2n} \\ -\rho_3 F_{31} & -\rho_3 F_{32} & \cdot & -\rho_3 F_{3n} \\ \cdot & \cdot & \cdot & \cdot \\ -\rho_n F_{n1} & -\rho_n F_{n2} & \cdot & 1-\rho_n F_{nn} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ \cdot \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ E_3 \\ \cdot \\ E_n \end{bmatrix}$$

**A**

**X**

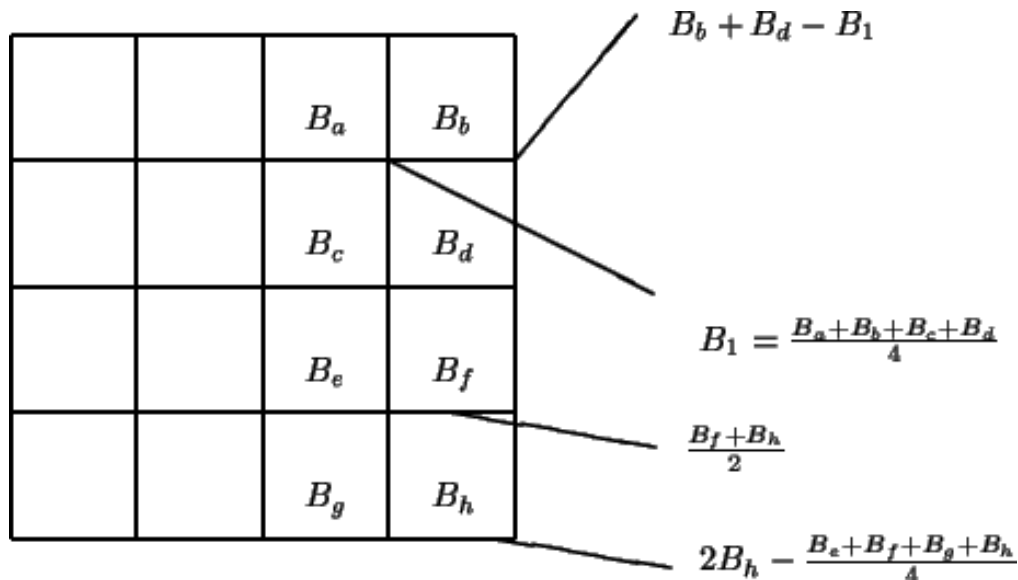
**=**

**B**

лінійна система

## Основні кроки алгоритму випромінюваності:

1. Розділити сцену на фрагменти (полігони).
2. Обчислити форм-фактори (тобто коефіцієнти передавання світлової енергії між фрагментами).
3. Розв'язати лінійну систему (рівняння випромінюваності).



Обчислення  
інтенсивності  
освітлення  
вершин та  
ребер: (білінійна)  
інтерполяція



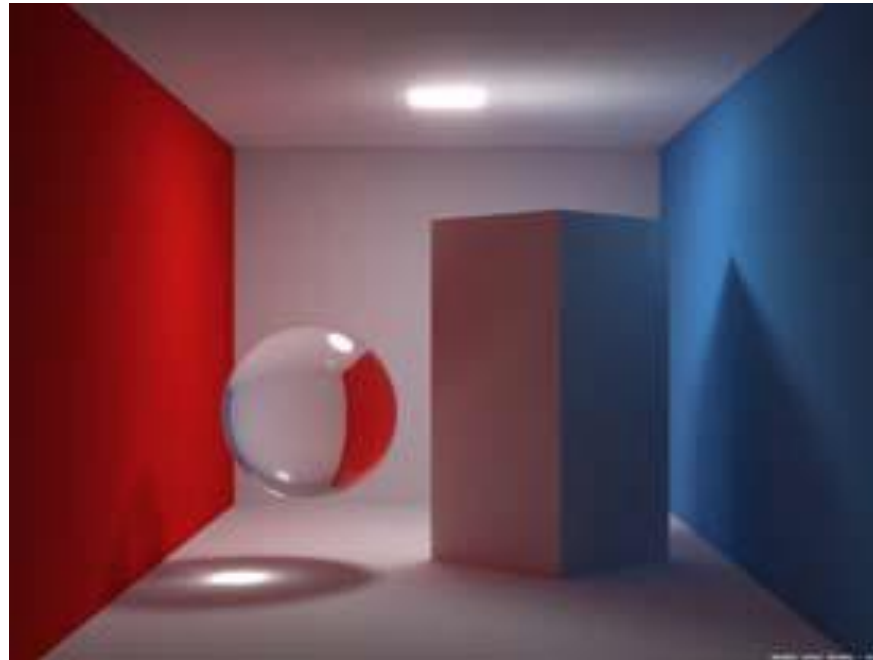
Результат застосування  
методу випромінюваності



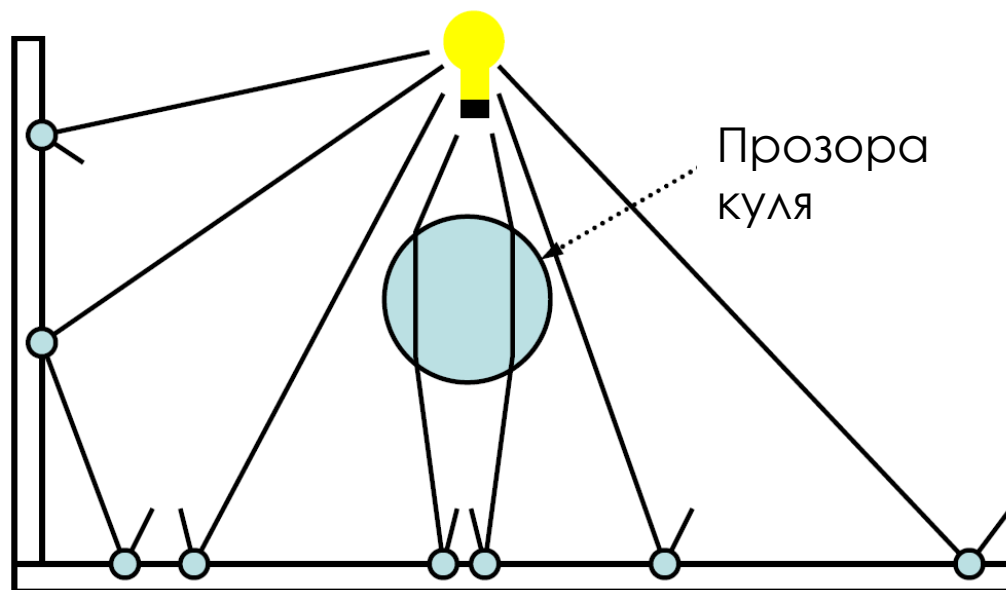
Результат  
комбінування методів  
випромінюваності та  
трасування променів

# Метод фотонних мап

- Дозволяє оброблювати не лише дифузні, а й дзеркальні та прозорі поверхні.



Нехай потужність джерела світла складає 100 Вт, тоді при генерації 100 фотонів потужність кожного фотону складе 1 Вт.



Моделювання поширення світлового потоку → метод Монте-Карло:

- 1) для кожного фотону генерується псевдовипадкове значення  $p$ ;
- 2) якщо  $0 < p \leq p_d$ , то відбиття фотону є «дифузним»;
- 3) якщо  $p_d < p \leq p_s$ , то відбиття фотону є «дзеркальним»;
- 4) якщо  $p > p_s$ , то фотон абсорбується (далі не поширюється).

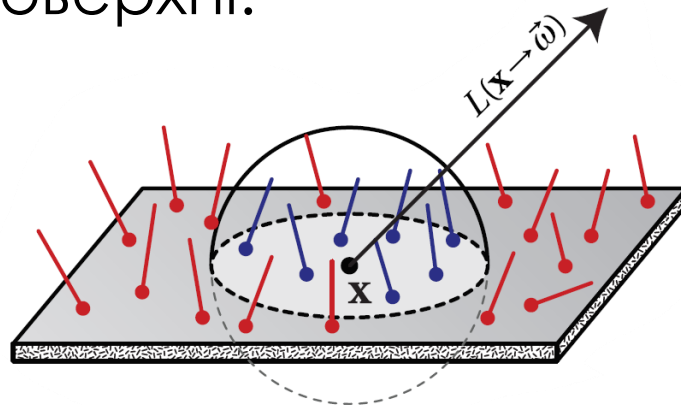


Для кожного фотона зберігається:

- 1) позиція,
- 2) потужність (колір та яскравість),
- 3) напрям руху.

Інформація про всі фотони зберігається у вигляді kd-дерева (бінарного дерева, що розділяє простір площинами, які є паралельними координатним осям) → «фотонна мапа».

Для візуалізації об'єктів сцени визначається сумарне освітлення в околі точки поверхні:



## Основні кроки алгоритму фотонних мап:

- *Перший прохід*: трасування фотонів (моделювання розповсюдження фотонів з джерела світла до поверхні).
- *Другий прохід*: рендеринг (візуалізація об'єктів на основі інформації про освітлення з фотонної мапи).



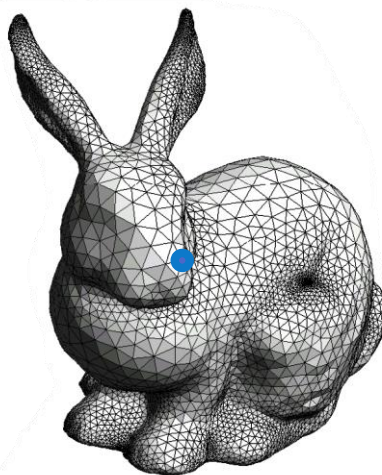
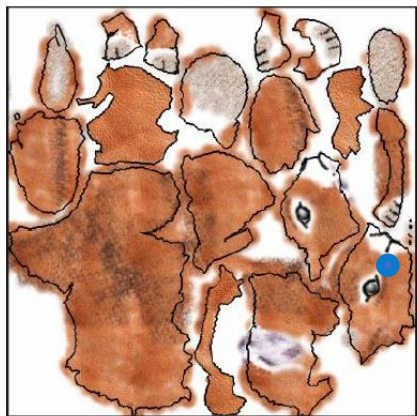
# **Алгоритми текстурування**

Накладання (відображення) текстур та  
процедурне текстурування



# Що таке текстура?

- Текстура – це зображення, що застосовується для визначення (завдання) характеристик поверхні об'єкту у 3D сцені.
- Накладання текстури (*texture mapping*) – це процес співставлення зображення, що визначає текстуру, та геометричної моделі 3D об'єкту.





# Процес відображення текстури:

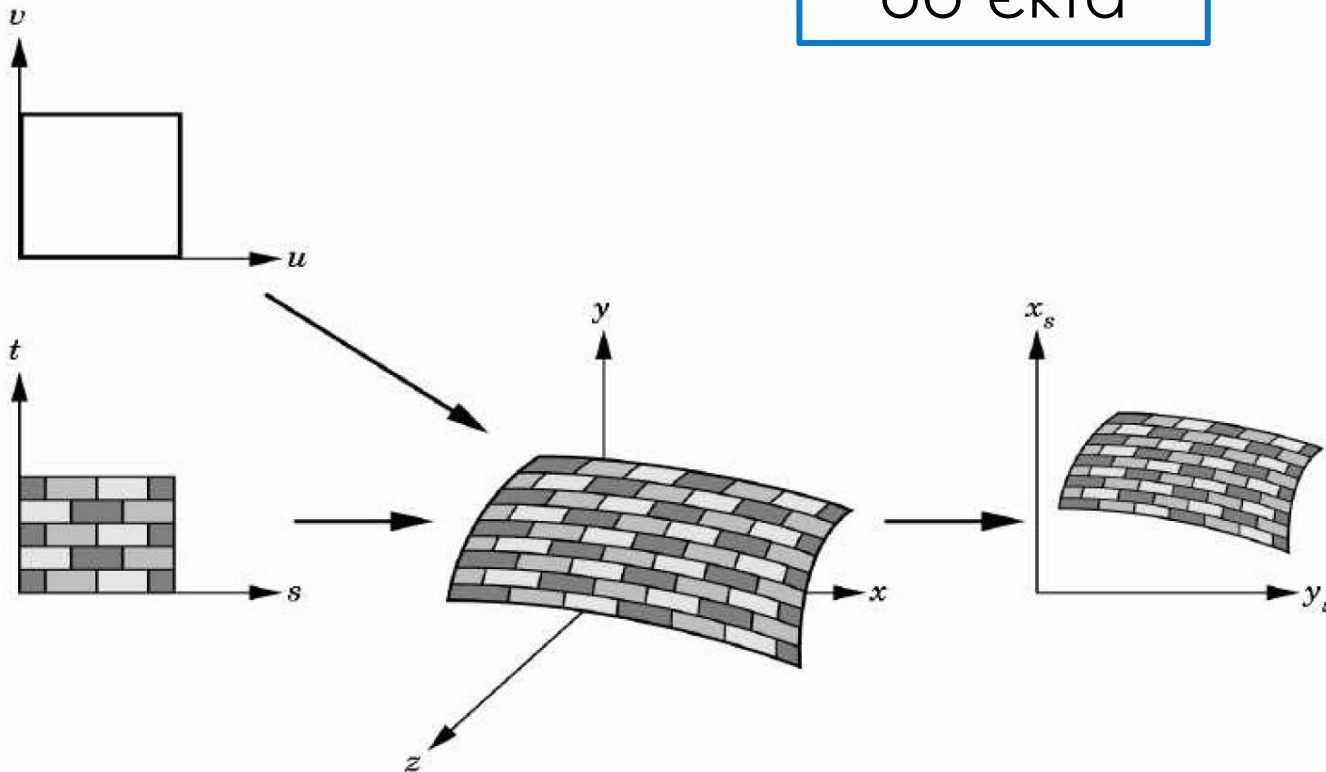
2D простір  
текстури

Параметризація

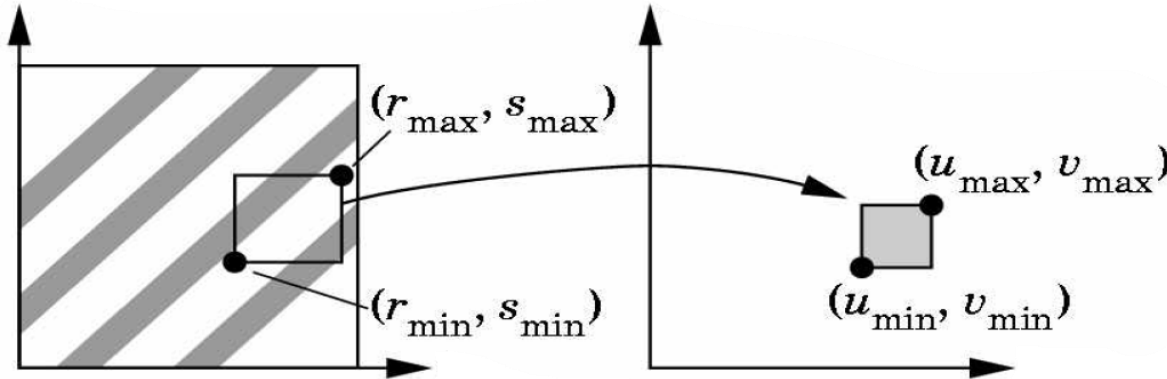
3D простір  
об'єкта

Проеціювання

2D простір  
екрана

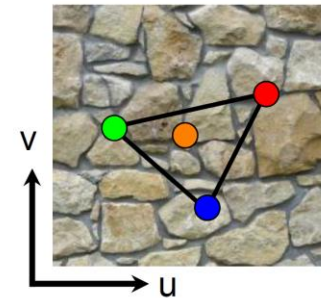
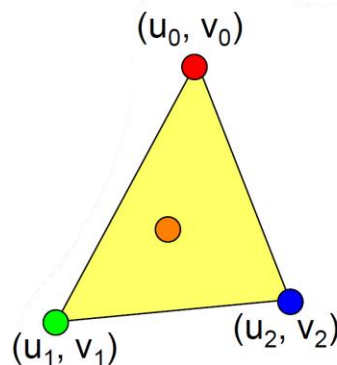


# Текстурування на площині

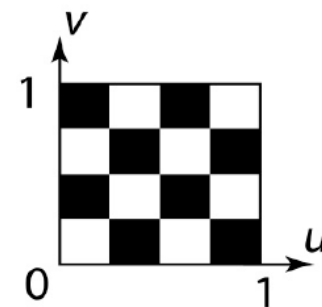
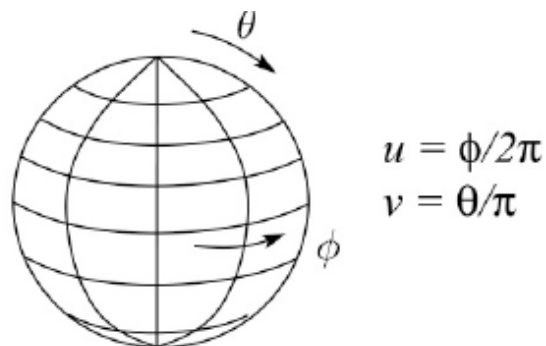
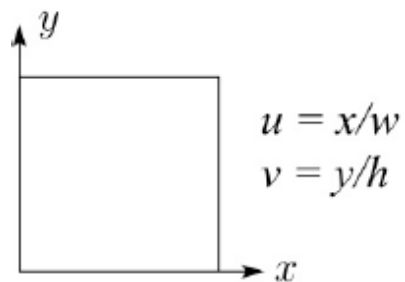


$$u = u_{\min} + \frac{r - r_{\min}}{r_{\max} - r_{\min}} (u_{\max} - u_{\min})$$

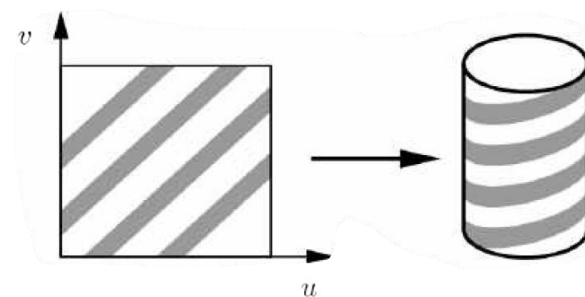
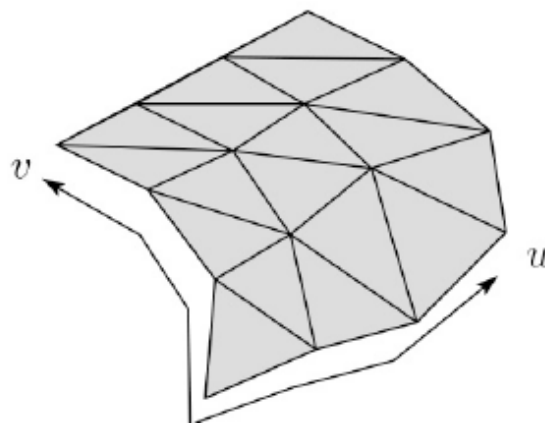
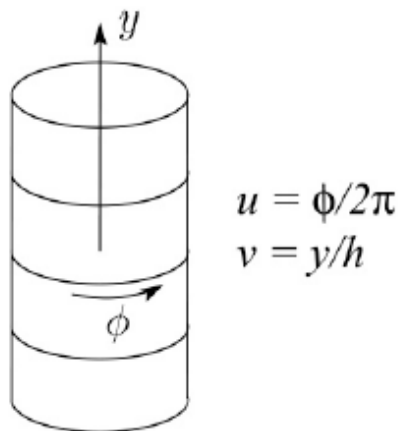
$$v = v_{\min} + \frac{s - s_{\min}}{s_{\max} - s_{\min}} (v_{\max} - v_{\min})$$



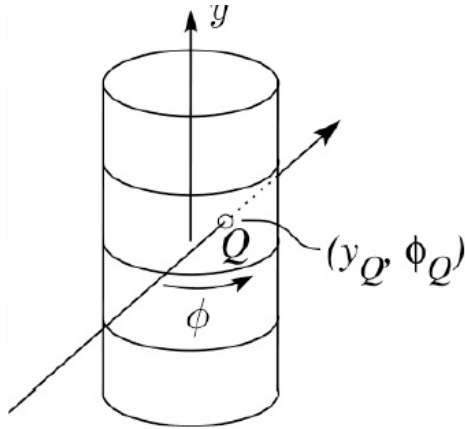
# Текстурування на поверхні



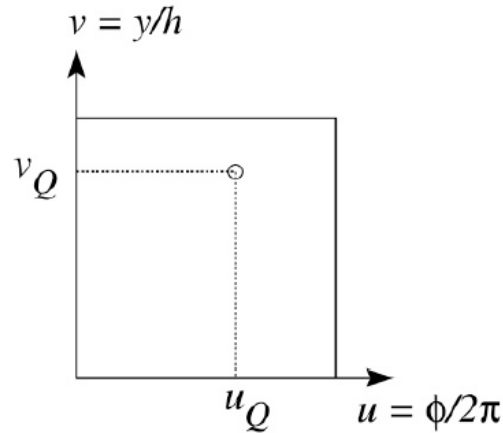
Текстурні  
координати



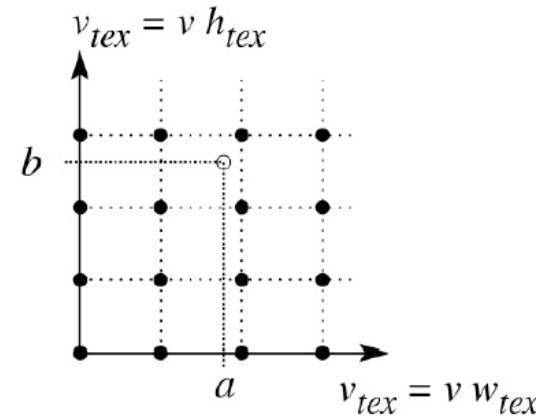
# Передискретизація текстур



Кидання променю



Абстрактні текстурні координати



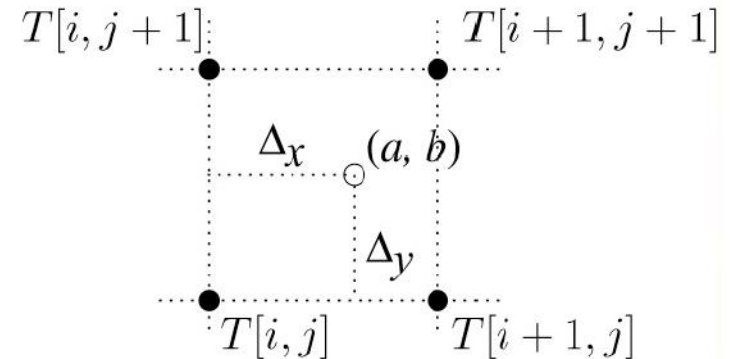
Піксельні текстурні координати

Білінійна інтерполяція:

$$T(a, b) = T[i + \Delta_x, j + \Delta_y] =$$

$$= (1 - \Delta_x)(1 - \Delta_y) T[i, j] + \Delta_x (1 - \Delta_y) T[i + 1, j] +$$

$$+ (1 - \Delta_x) \Delta_y T[i, j + 1] + \Delta_x \Delta_y T[i + 1, j + 1]$$





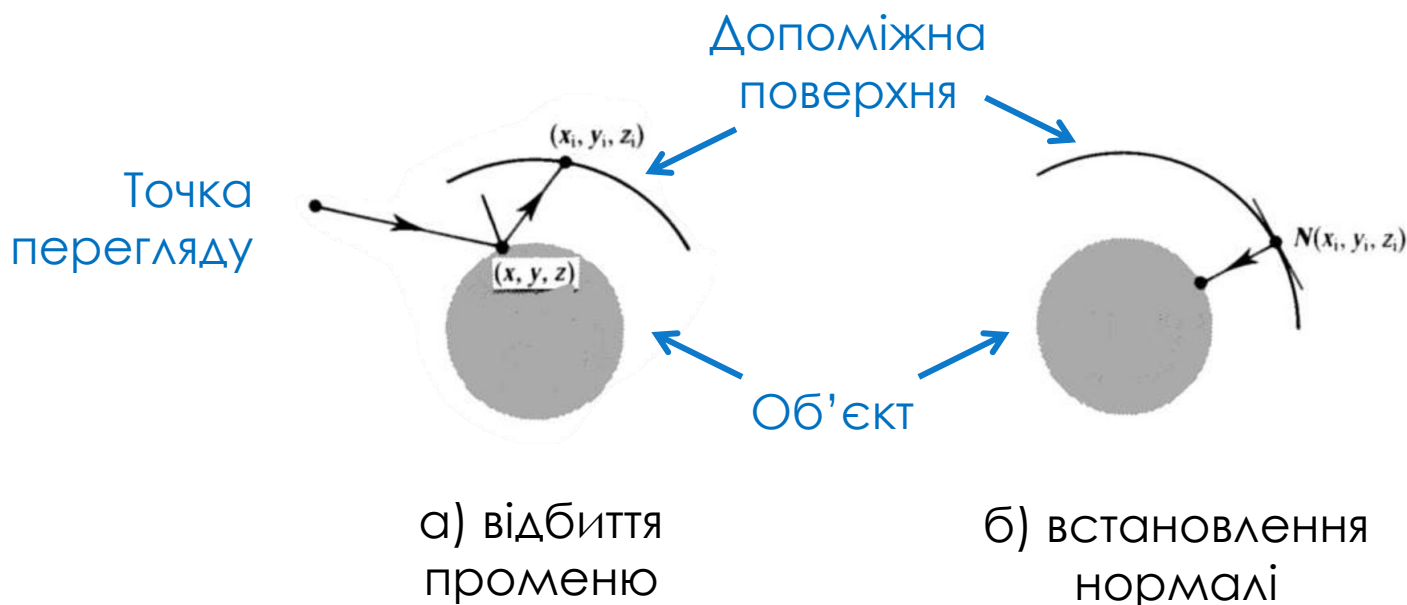
Базовий алгоритм накладання текстур ґрунтується на наступних перетвореннях:

- Для кожного графічного примітиву визначається перетворення  $M$ , яке відображає точку  $(x, y, z)$  поверхні у текстурні координати:  $M(x, y, z) \rightarrow (u, v)$ .
- Зображення текстури  $T$  є відображенням текстурних координат  $u, v \in [0, 1]$  у координати колірного простору:  $T(u, v) \rightarrow (r, g, b)$ .
- Визначення кольору пікселя, що відповідає точці з координатами  $(x, y, z)$  на поверхні, на яку накладено текстуру  $T$  визначається за формулою:

$$(r, g, b) = T(M(x, y, z)).$$

Накладання текстур на складні об'єкти відбувається за двоетапним алгоритмом:

1. Відображення текстури на допоміжну поверхню (площина, циліндр, сфера, куб).
2. Проекціювання допоміжної поверхні на площину об'єкта:



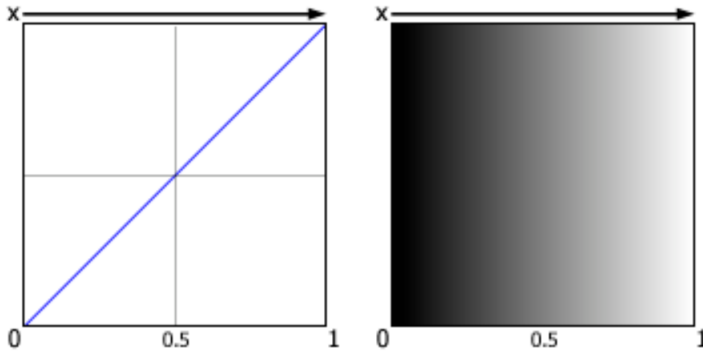
# Процедурне текстуровання

Процедурна текстура є результатом таких методів та їх комбінацій:

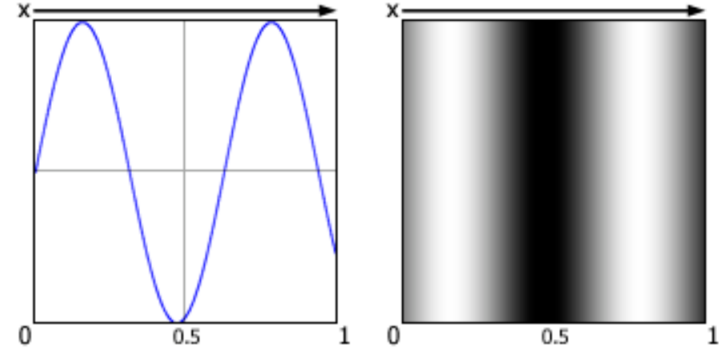
- застосування математичних функцій (наприклад,  $\sin$ ),
- фільтрація,
- застосування діаграм Вороного,
- застосування шуму.

# Застосування математичних функцій

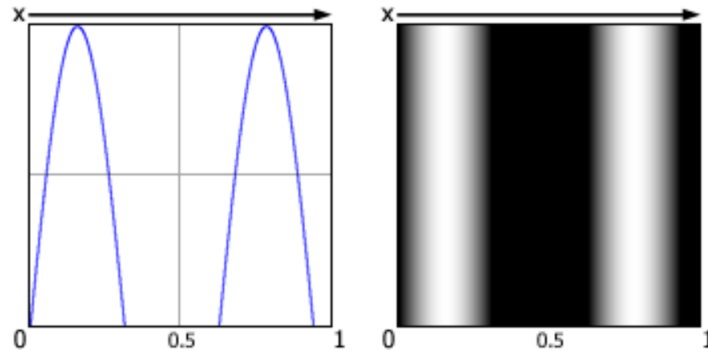
$$f(x) = x$$



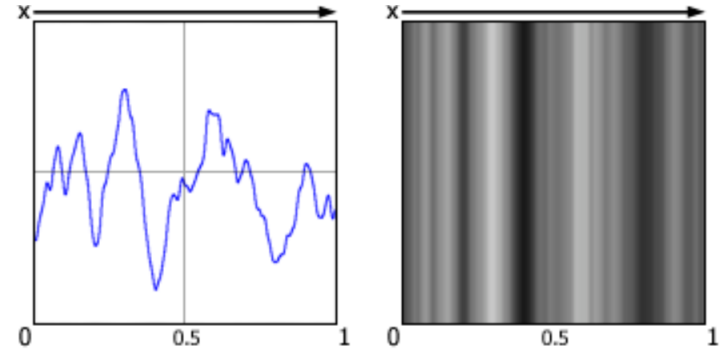
$$f(x) = \frac{1 + \sin(10x)}{2}$$



$$f(x) = \sin(10x)$$



$$f(x) = \text{noise}(10x)$$



# Фільтрація

- Згладжування
- Розмиття
- Виділення границь
- Усереднення



# Застосування діаграм Вороного

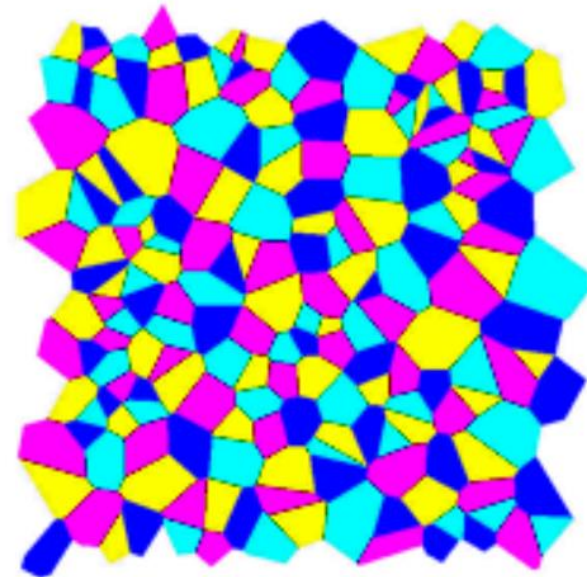
Діаграма Вороного – це розбиття площини на  $n$  областей виду:

$$V_i = \left\{ (x, y) : \rho((x, y), P_i) = \min_{k=1 \dots N} \rho((x, y), P_k) \right\},$$

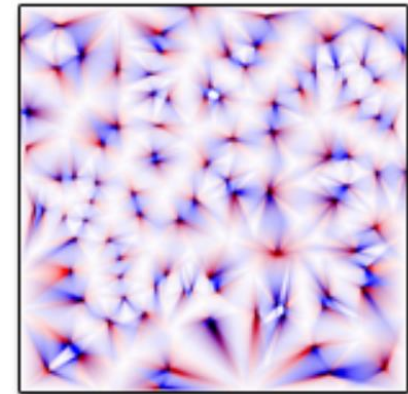
де  $P_i(x_i, y_i)$  – точка на площині,  $i = [1 \dots N]$ ;

$\rho$  – метрика, наприклад, Евклідова:

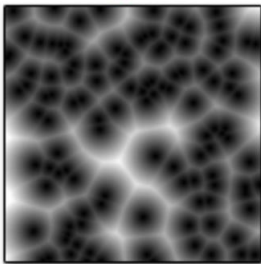
$$\rho(P_1, P_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$



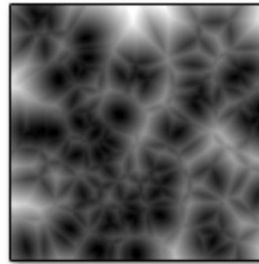
- Приклади застосування діаграм Вороного для процедурного створення текстур:



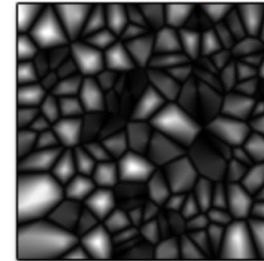
f1



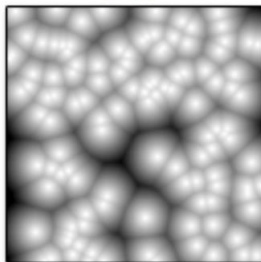
f2



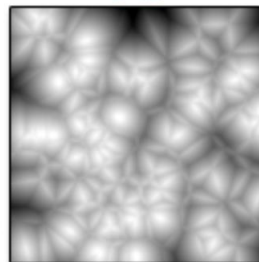
f2 - f1 (Worley)



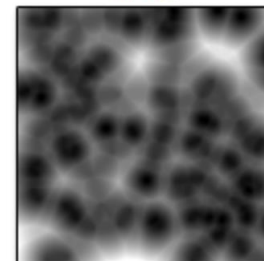
1 - f1



1 - f2



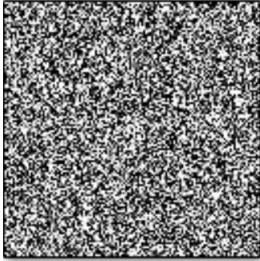
f1 + f2



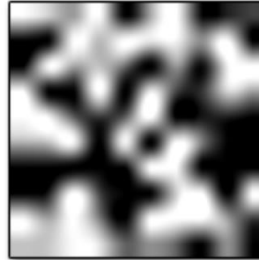


# Застосування шуму

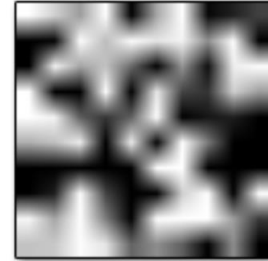
Pure Noise



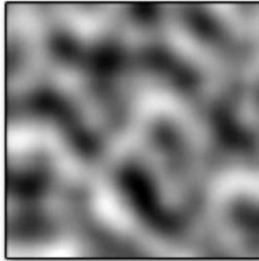
10x10 Bicubic



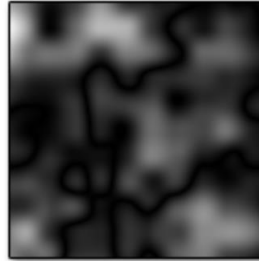
10x10 Bilinear



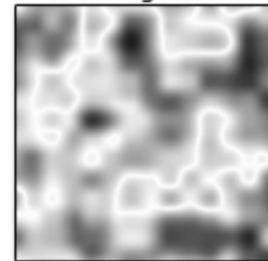
10x10 Gradient (Perlin)



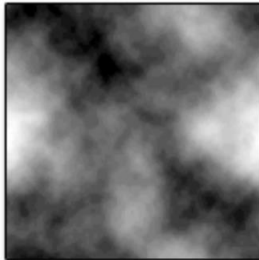
10x10 Inflected Perlin



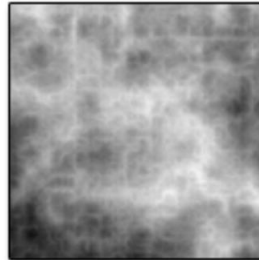
10x10 Ridged Perlin



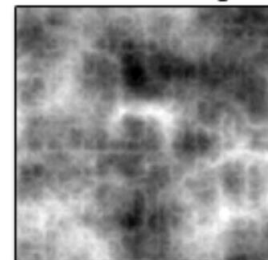
Multi-Octave Perlin



Multi-Octave Inflected

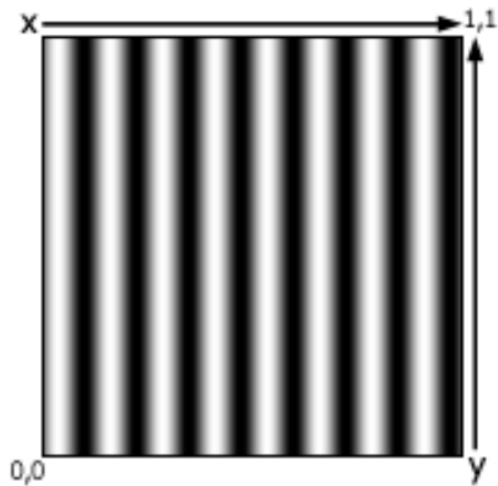


Multi-Octave Ridged

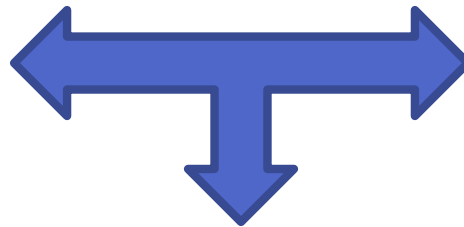
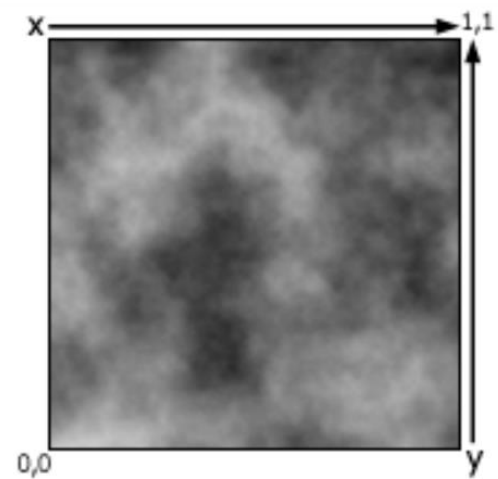




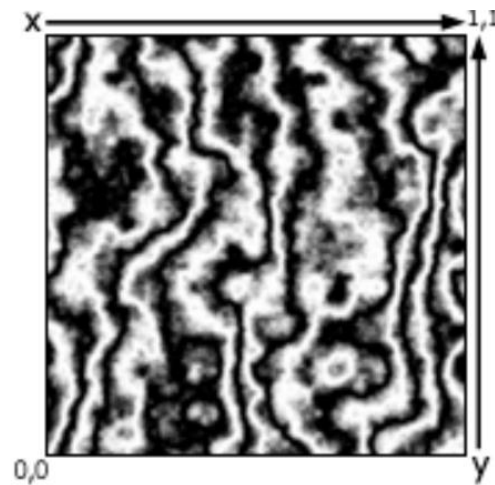
$$f(x, y) = \frac{1 + \sin(50x)}{2}$$



$$f(x, y) = \text{noise}(5x, 5y)$$



$$f(x, y) = \frac{1 + \sin\left(50\left(\frac{x + \text{noise}(5x, 5y)}{2}\right)\right)}{2}$$



# Питання?

## ЗАФАРБОВУВАННЯ

Слайди до лекцій з дисципліни  
«Математичні та алгоритмічні основи комп'ютерної графіки»

Лектор: к.т.н., доцент Сулема Є.С.

Каф. ПЗКС, ФПМ, КПІ ім. Ігоря Сікорського  
2019/2020 навч. рік