Київський національний університет імені Тараса Шевченка
Факультет комп'ютерних наук та кібернетики
Кафедра дослідження операцій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА
на тему:

Алгоритми трасування променів для 3D сцени в реальному часі

студента 4 курсу
Мячкова Владислава Вікторовича

Науковий керівник:
асистент
Денисов Сергій Вікторович

Робота заслухана на засіданні кафедри дослідження операцій та рекомендована до захисту в ЕК, протокол № ..... від ..... 2020 р.

Завідувач кафедри ДО проф. Іксанов О.М.

Київ - 2020

Зміст

Визначення основних понять

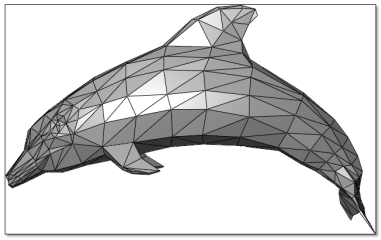


Рис. 1: Полігон

Полігон - це набір вершин, ребер та граней, які визначають форму багатогранного об'єкта в тривимірній комп'ютерній графіці. (Рис. 1)
Камера - це точка, з якої "знімається"сцена та відображається на екрані.
AABB (англ. axis-aligned bounding box — «паралельний осям обмежуючий паралелепіпед») — це паралелепіпед зі сторонами, паралельними осям координат, що обмежує деякий геометричний об'єкт в просторі. При обертанні об'єкта паралелепіпед зберігає свою орієнтацію, однак може змінювати свої розміри.

Вступ

В наші дні важливість розробки комп'ютерної графіки складно переоцінити. Світ не стоїть на місці, швидкі темпи розвитку зробили комп'ютерну графіку обов'язковою у багатьох напрямках промислово-побутової сфери. 3D графіка є невід'ємним супутником архітекторів, дизайнерів, діячів культури, рекламних фахівців, фахівців в області ігрової індустрії і важкого машинобудування.

Головною ціллю комп'ютерної графіки завжди було і буде створення найбільш реалістичних відображень. Проте моделювання деяких фізичних або математичних моделей потребує дуже багато часу. Адже кожен комп'ютер або обчислювальна техніка мають обмежені обчислювальні потужності – що і підводить до цілі комп'ютерної графіки в наш час: передати найбільш реалістичну картинку користувачу використавши найменшу кількість ресурсів.

Взагалом цей процес називається растерізацією - перетворення всіх об'єктів в змодельованому світі на кінцеве зображення, а саме використання простих елементів під назвою полігони, що являють собою трикутники з яких складаються всі елементи 3D світу, які потім перетворюються на зображення на екрані монітору. Мета растерізації - визначити, яку частину дисплею охоплює кожен полігон.

Один і той же трикунтик може охоплювати як і весь екран, так і декілька пікселів, в залежності від положення в просторі та кута, під яким розгядається трикутник. Після визначення показуваних пікселів переходять до таких речей, як текстури та освітлення. Виконання цих оперцій для кожного полігону для кожного кадру видається досить затратним процесом, оскільки багато полігонів виявляються прихованими (тобто знаходяться за іншими полігонами) й їх відображення є явна трата ресурсів комп'ютеру. Протягом багатьох років підходи до растеризації та обладнання поліпшилися, щоб зробити її набагато швидше, а сучасне програмне забеспечення може прийняти мільйони потенційно видимих полігонів і обробити їх за долі секунди.

Проте трасування променів – це дещо інший підхід до відобрабраження об'єктів на сцені. Головна його мета - симуляція світлових променів та їх трасування допоки вони не перетнуться з камерою, через яку на віртуальний світ дивиться користувач. Але мінусом такого підходу є його надзвичайна ресурсозатратність. Проектування одного єдиного променю включає в себе знаходження полігону, в який промінь влучив та його колір після того, як його освітили джерела світла. Тільки беручи до увагу даний факт можна уявити наскільки

вимогливий є алгоритм. Насправді ж для отримання справедливого зображення, що схоже на реальність потрібно набагато більше променів за один на піксель, оскільки треба брати до уваги усі джерела світла та їх промені. Не слід забувати й матеріал з якого зроблена поверхня об'єктів, який може варіюватися від матового до дзеркального. Щоб визначити кількість світла, яка потрапляє на один піксель з одного джерела світла, формула трасування променів повинна знати, наскільки далеко знаходиться джерело світла, наскільки воно яскраве, якого воно кольору і який кут відбивної поверхні відносно кута джерела світла. Цей процес повторюється для усіх джерел світла, включаючи непряме освітлення від променів, які відбилися від інших об'єктів сцени. Розрахунки застосовуються до матеріалів, що визначаються рівнем їх розсіяності чи дзеркальної відбивної здатності або обох. Прозорі або напівпрозорі поверхні, такі як скло або вода, заломлюють промені, а не відбивають, додаючи складності розрахункам. Тож всім променям задають деякий параметр відбивання, що обмежує кількість раз, що промінь може відбитися від поверхні.

Завдяки новітнім технологіям у сфері розробки графічних процесорів, з'явилася можливість використовувати алгоритм трасування променів у реальному часі.

Дипломна робота присвячений розробці та оптимізації стабільного трасувальника світлових променів.

1 Алгоритм трасування променів

1.1 Основи трасування променів

Уявімо, що ви перебуваєте в якомусь екзотичному місці і насолоджуєтеся приголомшливим видом, настільки приголомшливим, що ви просто зобов'язані зберегти його на картині. У вас є папір і маркери, але абсолютно немає художнього таланту. Невже все втрачено? Не обов'язково. Може у вас і немає таланту, але є методичність.

Можна зробити найбільш очевидну річ: взяти сітку від комах і помістити її в прямокутну раму, приєднавши раму до палиці. Потім подивитися крізь цю сітку, вибрати найкращий ракурс і поставити ще одну палицю точно там, де повинна бути голова, щоб отримати таку саму точку огляду:

Рис. 2: Факультет кібернетики

Ви ще не почали малювати, але принаймні, у вас є фіксована точка огляду і фіксована рама, крізь яку ви бачите. Більш того, ця фіксована рама розділена на дрібні квадрати. І тепер ми приступаємо до методичної частини. Намалюємо на папері сітку з тією ж кількістю квадратів, що і в сітці від комах. Тепер подивимося на правий верхній квадрат сітки. Який колір є в ньому домінуючим? Небесно-синій. Тому ми малюємо в правому верхньому квадраті паперу небесно-

синім кольором. Повторюємо те ж саме для кожного квадрата і досить скоро ми отримаємо хорошу картину ландшафту, як ніби видиму з вікна:

Рис. 3: Факультет кібернетики

Якщо задуматися, то комп'ютер, по суті, є дуже методичної машиною, у якій зовсім відсутні художні таланти. Якщо ми замінимо квадрати на папері пікселями на екрані, то зможемо описати процес рендеринга сцени наступним чином:

```
Розмістити точку огляду і рамку в потрібних місцях
Для кожного пікселя полотна
    Визначити квадрат сітки, що відповідає цьому пікселю
    Визначити колір, видимий крізь цей квадрат
    Зафарбувати піксель цим кольором
```

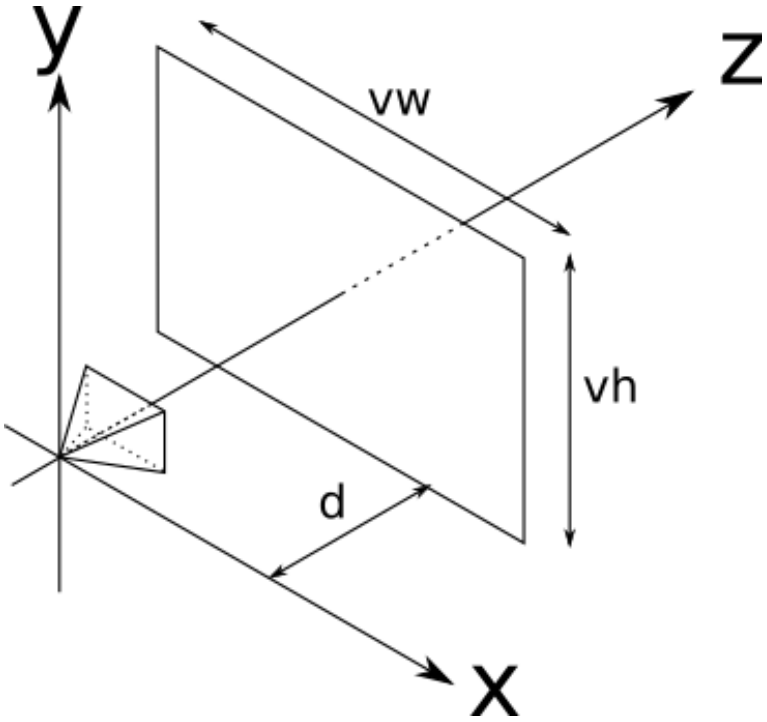


Рис. 4: Вікно перегляду

Цей прямокутник, який буде нашим вікном у світ, називається вікном перегляду (viewport). По суті, ми будемо малювати на полотні все те, що бачимо через вікно перегляду. Важливо, що розміри вікна перегляду і відстань до камери визначають кут видимості з камери, званий областю видимості (field of view) або для стислості FOV.

1.2 Трасування променя

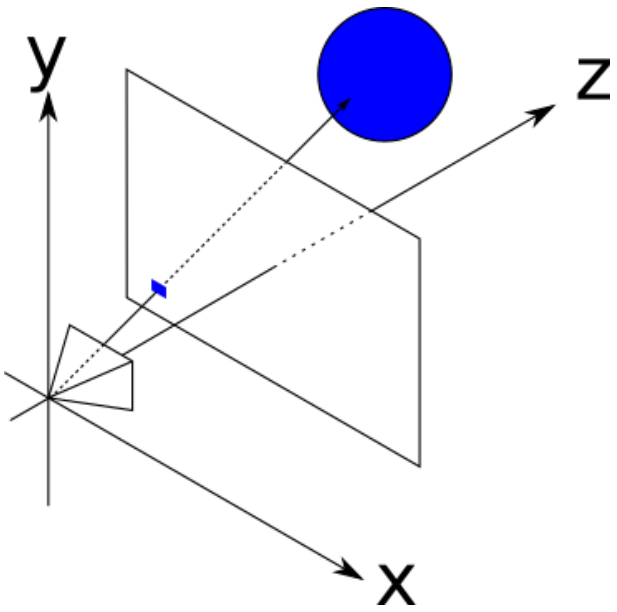


Рис. 5: Демонстарція трасування променя

У реальному світі світло виходить з джерела світла (сонця, лампочки і т.д.), відбивається від декількох об'єктів і нарешті досягає наших очей. Ми можемо спробувати симулювати шлях кожного фотона, випущеного з симульованих джерел світла, але це буде неймовірно затратно по часу. Замість цього ми будемо трасувати промені «в зворотному напрямку» - ми почнемо з променя, що знаходиться на камері, що проходить через точку в вікні перегляду і рухаючись, поки він не зіткнеться з яким-небудь об'єктом в сцені. Цей об'єкт буде «видно» з камери через цю точку вікна перегляду.

1.3 Рівняння променя

Рівняння променя задається як

$$P = O + t(V - O)$$

$O = (O_x, O_y, O_z)$ - положення камери в просторі,
$V = (V_x, V_y, V_z)$ - точка вікна перегляду (viewport),
t - довільне дійсне число.
Позначимо $(V - O)$, тобто напрямок променя, як $\overrightarrow{D}$; тоді рівняння прийме простий вигляд:

$$P = O + t\overrightarrow{D} \tag{1}$$

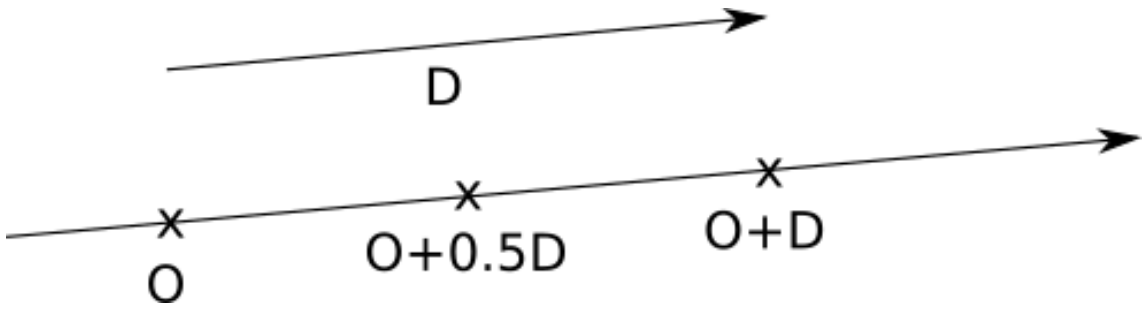


Рис. 6: Рівняння променя

1.4 Перетин променя зі сферою

Що таке сфера? Сфера - це множина точок, що лежить на постійній відстані (радіус сфери) від фіксованої точки (центр сфери):

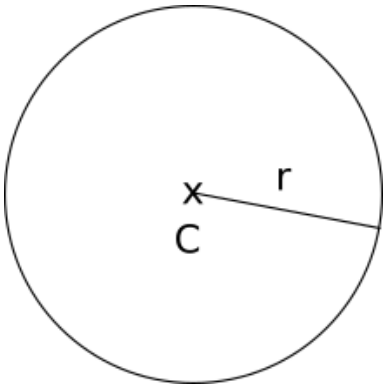


Рис. 7: Сфера

Якщо C - центр сфери, а r - радіус сфери, то точки P на поверхні сфери задовольняють наступним рівнянням:

$$|P - C| = r$$

Довжина вектора - це квадратний корінь його скалярного добутку на себе:

$$\sqrt{\langle P - C, P - C \rangle} = r$$

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

Точка P, в якій промінь падає на сферу, є одночасно і точкою променя, і точкою на поверхні сфери, тому вона повинна задовольняти обом рівнянням одночасно:

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

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

Оскільки P - це одна і та ж точка в обох рівняннях, ми можемо замінити P в першому на вираз для P у другому. Це дає нам

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

Позначимо $\overrightarrow{OC} = O - C$

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

Розкладемо скалярний добуток на його компоненти, скориставшись його дистрибутивністю:

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

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

Перемістивши параметр t з скалярних добутків, а r^2 в іншу частину рівняння, отримаємо:

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

Позначимо

$$k_1 = \langle \overrightarrow{D}, \overrightarrow{D} \rangle$$
$$k_2 = 2\langle \overrightarrow{OC}, \overrightarrow{D} \rangle$$
$$k_3 = \langle \overrightarrow{OC}, \overrightarrow{OC} \rangle - r^2$$
$$k_1 t^2 + k_2 t + k_3 = 0$$

Розв'язки квадратного рівняння є значення параметра t, при яких промінь перетинається зі сферою:

$$\{t_1, t_2\} = \frac{-k_2 \pm \sqrt{k_2^2 - 4k_1k_3}}{2k1}$$

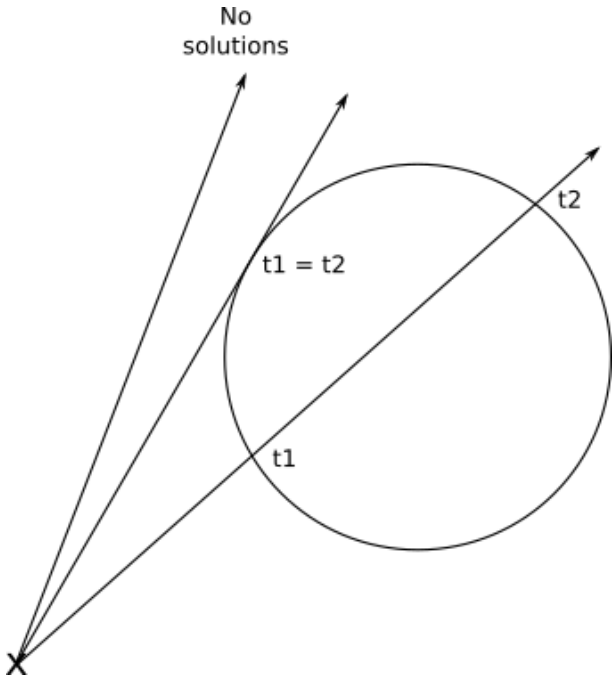


Рис. 8: Розв'язки рівняння

Вектор нормалі сфери у точці перетину з променем:

$$\overrightarrow{N} = \frac{P - C}{|P - C|}$$

1.5 Перетин променя з площиною

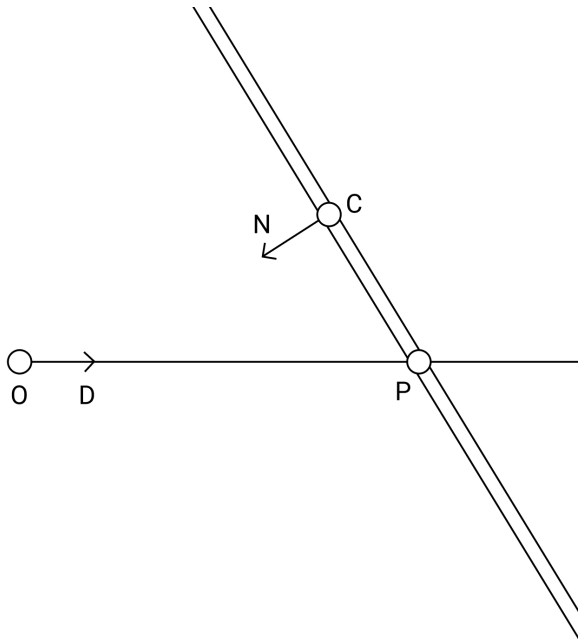


Рис. 9: Перетин променя з площиною

C - точка, що лежить на площині
$\overrightarrow{N}$ - вектор нормалі площини
$P = O + t\overrightarrow{D}$ - рівняння променя
Щоб промінь влучив у площину, ми зазначим, що:

$$\langle (P - C), \overrightarrow{N} \rangle = 0$$

Це означає, що вектор $P - C$ перпендикулярний нормалі, що справедливо, коли точка P лежить на площині.

$$\langle (O + \overrightarrow{D}t - C), \overrightarrow{N} \rangle = 0$$
$$\langle (\overrightarrow{D}t + \overrightarrow{OC}), \overrightarrow{N} \rangle = 0$$
$$\langle \overrightarrow{D}t, \overrightarrow{N} \rangle + \langle \overrightarrow{OC}, \overrightarrow{N} \rangle = 0$$
$$t\langle \overrightarrow{D}, \overrightarrow{N} \rangle = -\langle \overrightarrow{OC}, \overrightarrow{N} \rangle$$

$$t = -\frac{\langle \overrightarrow{OC}, \overrightarrow{N} \rangle}{\langle \overrightarrow{D}, \overrightarrow{N} \rangle}, \langle \overrightarrow{D}, \overrightarrow{N} \rangle \neq 0$$

Вектор нормалі в точці P дорівнює нормалі площині, але якщо $\langle \overrightarrow{D}, \overrightarrow{N} \rangle < 0$, то вектор нормалі в точці P дорівнює $-\overrightarrow{N}$.

1.6 Перетин променя з циліндром

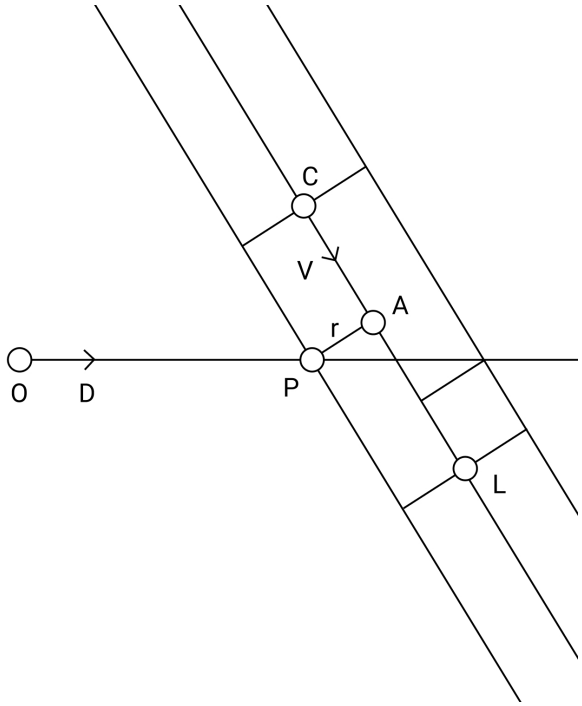


Рис. 10: Перетин променя з циліндром

C - точка старту основи циліндра
$\overrightarrow{V}$ - одиничний вектор, який визначає вісь циліндра
r - радіус основи циліндра
L - визначає кінцеву точку основи циліндра
$P = O + t\overrightarrow{D}$ - рівняння променя
Щоб промінь влучив у циліндр, ми зазначим, що:

$$A = C + \overrightarrow{V}m$$
$$\langle P - A, \overrightarrow{V} \rangle = 0$$
$$|P - A| = r$$

де m - скаляр, який визначає найближчу точку на осі до точки влучення ($m = |\overrightarrow{CA}|$). Вектор $P - A$ перпендикулярний $\overrightarrow{V}$, що гарантує найближчу відстань

до осі.

Розглянемо $\langle P - A, \overrightarrow{V} \rangle = 0$

$$\begin{aligned}
&\langle P - C - \overrightarrow{V}m, \overrightarrow{V} \rangle = 0 \\
&\langle P - C, \overrightarrow{V} \rangle - \langle \overrightarrow{V}m, \overrightarrow{V} \rangle = 0 \\
&\langle P - C, \overrightarrow{V} \rangle = \langle \overrightarrow{V}, \overrightarrow{V} \rangle m \\
&\langle P - C, \overrightarrow{V} \rangle = m \\
&m = \langle P - C, \overrightarrow{V} \rangle \\
&m = \langle \overrightarrow{D}t + O - C, \overrightarrow{V} \rangle
\end{aligned}$$

Позначимо $\overrightarrow{OC} = O - C$

$$\begin{aligned}
&m = \langle \overrightarrow{D}t + \overrightarrow{OC}, \overrightarrow{V} \rangle \\
&m = t\langle \overrightarrow{D}, \overrightarrow{V} \rangle + \langle \overrightarrow{OC}, \overrightarrow{V} \rangle
\end{aligned}$$

Розглянемо $|P - A| = r$

$$\begin{aligned}
&\langle P - C - \overrightarrow{V}m, P - C - \overrightarrow{V}m \rangle = r^2 \\
&\langle \overrightarrow{D}t + O - C - \overrightarrow{V}m, \overrightarrow{D}t + O - C - \overrightarrow{V}m \rangle = r^2 \\
&\langle \overrightarrow{D}t + \overrightarrow{OC} - \overrightarrow{V}m, \overrightarrow{D}t + \overrightarrow{OC} - \overrightarrow{V}m \rangle = r^2
\end{aligned}$$

Розкладемо скалярний добуток на його компоненти, скориставшись його дистрибутивністю, підставимо m та отримаємо квадратне рівняння:

$$\begin{aligned}
t^2(\langle D, D \rangle - \langle D, \overrightarrow{V} \rangle^2) - 2t(\langle \overrightarrow{OC}, \overrightarrow{V} \rangle \cdot \langle D, \overrightarrow{V} \rangle + 2\langle D, \overrightarrow{OC} \rangle) \\
-\langle \overrightarrow{OC}, \overrightarrow{V} \rangle^2 + \langle \overrightarrow{OC}, \overrightarrow{OC} \rangle - r^2 = 0
\end{aligned}$$

Розв'язками рівняння і будуть точки, в яких промінь перетне циліндр. Вектор нормалі циліндра у точці перетину з променем:

$$\begin{aligned}
&\overrightarrow{N} = \frac{P - A}{|P - A|} \\
&\overrightarrow{N} = \frac{P - C - \overrightarrow{V}m}{|P - C - \overrightarrow{V}m|}
\end{aligned}$$

1.7 Перетин променя з конусом

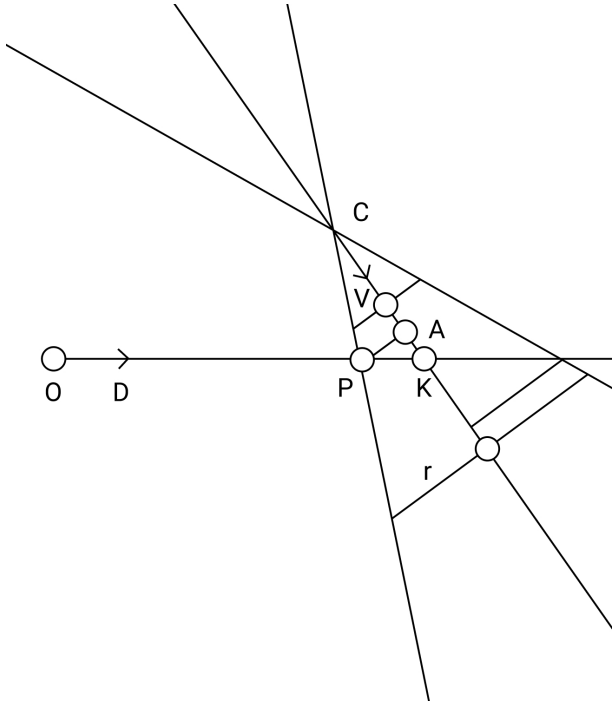


Рис. 11: Перетин променя з конусом

C - вершина конуса
$\overrightarrow{V}$ - одиничний вектор, який визначає вісь конуса
α - половина кута у вершини конуса
r - радіус основи конуса
$P = O + t\overrightarrow{D}$ - рівняння променя
Щоб промінь влучив у конус, ми зазначим, що:

$$A = C + \overrightarrow{V}m$$
$$\langle P - A, \overrightarrow{V}\rangle = 0$$
$$\frac{|P - A|}{m} = \tan\alpha$$

де m - скаляр, який визначає найближчу точку на осі до точки влучення ($m = |\overrightarrow{CA}|$) m виражається аналогічно до випадку перетину променя з циліндром.

$$m = t\langle\overrightarrow{D}, \overrightarrow{V}\rangle + \langle\overrightarrow{OC}, \overrightarrow{V}\rangle$$

Розглянемо $\frac{|P-A|}{m} = \tan\alpha$:

$$\langle P - A, P - A\rangle = m^2 \tan^2\alpha$$
$$\langle \overrightarrow{D}t + \overrightarrow{OC} - \overrightarrow{V}m, \overrightarrow{D}t + \overrightarrow{OC} - \overrightarrow{V}m\rangle = m^2 \tan^2\alpha$$

Розкладемо скалярний добуток на його компоненти, скориставшись його дистрибутивністю, підставимо m та отримаємо квадратне рівняння:

$$\begin{aligned}
&t^2(\langle \overrightarrow{D}, \overrightarrow{D}\rangle - \langle \overrightarrow{D}, \overrightarrow{V}\rangle^2 - \langle \overrightarrow{D}, \overrightarrow{V}\rangle^2 \tan^2\alpha) \\
&+ 2t(\langle \overrightarrow{D}, \overrightarrow{OC}\rangle - \langle \overrightarrow{D}, \overrightarrow{V}\rangle\langle \overrightarrow{OC}, \overrightarrow{V}\rangle - \langle \overrightarrow{D}, \overrightarrow{V}\rangle\langle \overrightarrow{OC}, \overrightarrow{V}\rangle \tan^2\alpha) \\
&- \langle \overrightarrow{OC}, \overrightarrow{V}\rangle^2 \tan^2\alpha + \langle \overrightarrow{OC}, \overrightarrow{OC}\rangle - \langle \overrightarrow{V}, \overrightarrow{OC}\rangle^2 = 0
\end{aligned}$$

Розв'язками рівняння і будуть точки, в яких промінь перетне конус.
Для обчислення нормалі в точці достатньо помітити, що необхідно нормалізувати вектор $(P - C - \overrightarrow{V}m) - \overrightarrow{V}a$. Де $a = |\overrightarrow{AK}|$. Як обчислити a? Кут між нормаллю і $P - C - \overrightarrow{V}m$ такий же, як і α. Звідси:

$$\begin{aligned}
&\frac{a}{|\overrightarrow{AP}|} = \tan\alpha \\
&\frac{|\overrightarrow{AP}|}{m} = \tan\alpha \\
&a = m\tan^2\alpha \\
&\overrightarrow{N} = \frac{P - C - \overrightarrow{V}m - \overrightarrow{V}m\tan^2\alpha}{|P - C - \overrightarrow{V}m - \overrightarrow{V}m\tan^2\alpha|} \\
&\overrightarrow{N} = \frac{P - C - (1 + \tan^2\alpha)\overrightarrow{V}m}{|P - C - (1 + \tan^2\alpha)\overrightarrow{V}m|}
\end{aligned}$$

1.8 Перетин променя з полігоном

Не кожний об'єкт можна задати аналітично, але, навіть, якщо і вийде, то інколи це буває досить складно. Одним з неаналітичних способів представлення поверхні є її апроксимація набором трикутників різних розмірів. Для роботи з полігонами найпоширенішими є файли формату OBJ. Формат файлів OBJ - це простий формат даних, який містить тільки 3D геометрію, а саме, позицію кожної вершини, зв'язок координат текстури з вершиною, нормаль для кожної вершини, а також параметри, які створюють полігони. Тобто, якщо полігон

складається з множини трикутників, то для того, щоб промінь перетинав полігон, достатньо, щоб промінь перетнув хоча б один трикуник.

1.8.1 Алгоритм Моллера-Трумбора

Алгоритм Моллера-Трумбора використовується для перетину променя з трикутником.

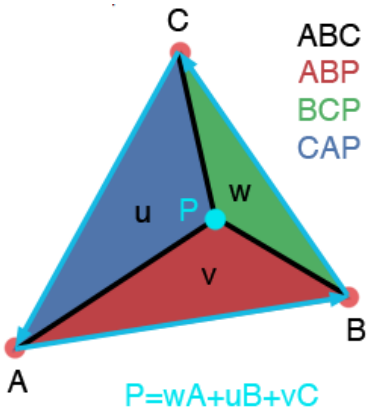


Рис. 12: Барицентричні координати точки P

Точка P задається співвідношенням $P = wA + uB + vC$, де

$$w = \frac{S_{BPC}}{S_{ABC}}$$
$$u = \frac{S_{CPA}}{S_{ABC}}$$
$$v = \frac{S_{APB}}{S_{ABC}}$$
$$w + u + v = 1$$

Виразимо w через u та v:

$$P = (1 - u - v)A + uB + vC$$
$$P = A - uA - vA + uB + vC$$
$$P = A + u(B - A) + v(C - A)$$

Розглянемо рівняння променя $P = O + t\overrightarrow{D}$, підставимо P у рівняння вище:

$$O + t\overrightarrow{D} = A + u(B - A) + v(C - A)$$
$$O - A = -t\overrightarrow{D} + u(B - A) + v(C - A)$$

Запишемо у матричному вигляді:

$$\begin{pmatrix} -\overrightarrow{D} & (B-A) & (C-A) \end{pmatrix} \begin{pmatrix} t \\ u \\ v \end{pmatrix} = (O-A)$$

Позначимо $\overrightarrow{T} = O - A$, $\overrightarrow{E_1} = B - A$, $\overrightarrow{E_2} = C - A$

$$\begin{pmatrix} -\overrightarrow{D} & \overrightarrow{E_1} & \overrightarrow{E_2} \end{pmatrix} \begin{pmatrix} t \\ u \\ v \end{pmatrix} = \overrightarrow{T}$$

Скористаємось правилом Крамера та отримаємо:

$$\begin{pmatrix} t \\ u \\ v \end{pmatrix} = \frac{1}{\begin{vmatrix} -\overrightarrow{D} & \overrightarrow{E_1} & \overrightarrow{E_2} \end{vmatrix}} \begin{pmatrix} \begin{vmatrix} \overrightarrow{T} & \overrightarrow{E_1} & \overrightarrow{E_2} \end{vmatrix} \\ \begin{vmatrix} -\overrightarrow{D} & \overrightarrow{T} & \overrightarrow{E_2} \end{vmatrix} \\ \begin{vmatrix} -\overrightarrow{D} & \overrightarrow{E_1} & \overrightarrow{T} \end{vmatrix} \end{pmatrix}$$

Скористаємося властивістю мішаного добутку векторів:

$$\begin{vmatrix} \overrightarrow{A} & \overrightarrow{B} & \overrightarrow{C} \end{vmatrix} = -(\overrightarrow{A} \times \overrightarrow{C}) \cdot \overrightarrow{B} = -(\overrightarrow{C} \times \overrightarrow{B}) \cdot \overrightarrow{A}$$

і отримаємо:

$$\begin{pmatrix} t \\ u \\ v \end{pmatrix} = \frac{1}{(\overrightarrow{D} \times \overrightarrow{E_2}) \cdot \overrightarrow{E_1}} \begin{pmatrix} (\overrightarrow{T} \times \overrightarrow{E_1}) \cdot \overrightarrow{E_2} \\ (\overrightarrow{D} \times \overrightarrow{E_2}) \cdot \overrightarrow{T} \\ (\overrightarrow{T} \times \overrightarrow{E_1}) \cdot \overrightarrow{D} \end{pmatrix}$$

Бачимо, що зараз легко знайти значення t, u та v. Ми можемо отримати їх скалярними та векторними добутками між відомими змінними: вершинами трикутника, початком та напрямком променя.

Зрозуміло, якщо шукати перетин променя з кожним трикутником, то складність алгоритму рендерингу буде дуже великою. Мова про оптимізацію буде далі.

1.9 Рендеринг об'єктів на сцені

Підіб'ємо підсумок - для кожного пікселя на полотні ми можемо обчислити відповідну точку в вікні перегляду. Знаючи положення камери, ми можемо виразити рівняння променя, який виходить з камери і проходить через задану точку вікна перегляду. Маючи об'єкт, ми можемо обчислити точку, в якій промінь його перетинає. Тобто нам досить тільки обчислити перетини променів з об'єктами, зберегти найближчі до камери точки і зафарбувати пікселі на полотні відповідним кольором. Однак варто приділити особливу увагу параметру t. Повернемося до рівняння променя:

$$P = O + t(V - O)$$

Оскільки вихідна точка і напрямок променя постійні, змінюючи t в множині дійсних чисел, ми отримаємо кожну точку P на цьому промені. Зауважимо, що при $t = 0$ ми отримаємо $P = O$, а при $t = 1$ ми отримаємо $P = V$. При від'ємних числах ми отримаємо точки в протилежному напрямку, тобто за камерою. Ми можемо розділити область параметрів на три частини:

$t < 0$	За камерою
$0 \leq t \leq 1$	Між камерою та площиною проекції
$t > 0$	Сцена

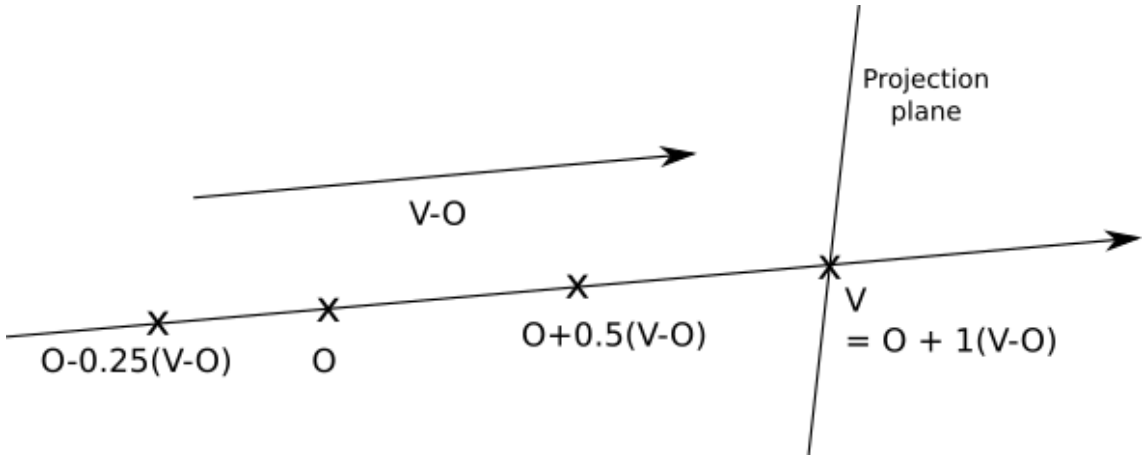


Рис. 13: Рівняння променя

Зауважимо, що нічого в рівнянні перетину не говорить, що об'єкт повинен бути перед камерою; рівняння абсолютно без проблем дає рішення і для перетинів за камерою. Очевидно, що нам це не потрібно, тому необхідно ігнорувати всі рішення при $t < 0$. Щоб уникнути додаткових математичних складнощів,

ми обмежимо рішення $t > 1$, тобто ми будемо рендерити все, що знаходиться за площиною проекції.

З іншого боку, нам не потрібно встановлювати верхню межу значення t; ми хочемо бачити об'єкти перед камерою, незалежно від того, наскільки вони далеко. Отже, $t \in [1, +\infty)$.

2 Модель освітлення

2.1 Джерела освітлення

2.1.1 Точкові джерела

Точкове джерело випромінює світло з фіксованою точки в просторі. Світло випромінюється рівномірно у всіх напрямках; саме тому його також називають всеспрямованим освітленням. Отже, точкове джерело повністю характеризується його позицією і яскравістю.

Лампа розжарювання - гарний приклад з реального світу того, наближенням чого є точкове джерело освітлення. Хоча лампа розжарювання не випускає світло з однієї точки і він не є абсолютно всеспрямованим, але наближення досить гарне.

Задамо вектор $\overrightarrow{L}$ як напрямок з точки P в сцені до джерела освітлення Q. Цей вектор, званий світловим вектором, дорівнює $Q - P$. Зауважимо, що оскільки Q фіксована, а P може бути будь-якою точкою сцени, то в загальному випадку $\overrightarrow{L}$ буде різним для кожної точки сцени.

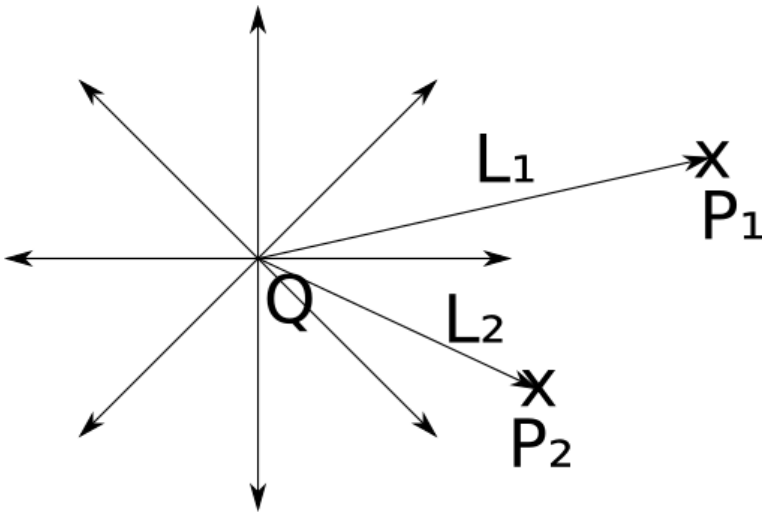


Рис. 14: Точкове джерело світла

2.1.2 Спрямовані джерела

Якщо точкове джерело є гарною аппроксимацией лампи розжарювання, то що може служити аппроксимацией Сонця?

У масштабах Сонячної системи Сонце можна приблизно вважати точковим джерелом. Зрештою, воно випромінює світло з точки (хоча і досить великої) і випускає його у всіх напрямках, тобто підходить під обидві вимоги.

Однак якщо на сцені дія відбувається на Землі, то це не дуже добре наближення. Сонце знаходиться так далеко, що кожен промінь світла буде насправді мати однаковий напрямок. Для таких випадків використовуються спрямовані джерела освітлення. На відміну від точкового джерела світла, у спрямованого немає позиції. Замість позиції у нього є напрямок. Можна сприймати його як нескінченно віддалений точковий джерело, що світить в певному напрямку.

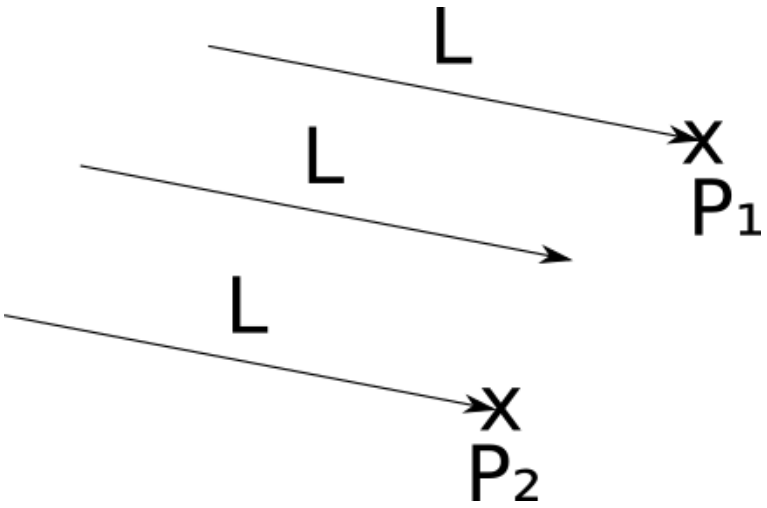


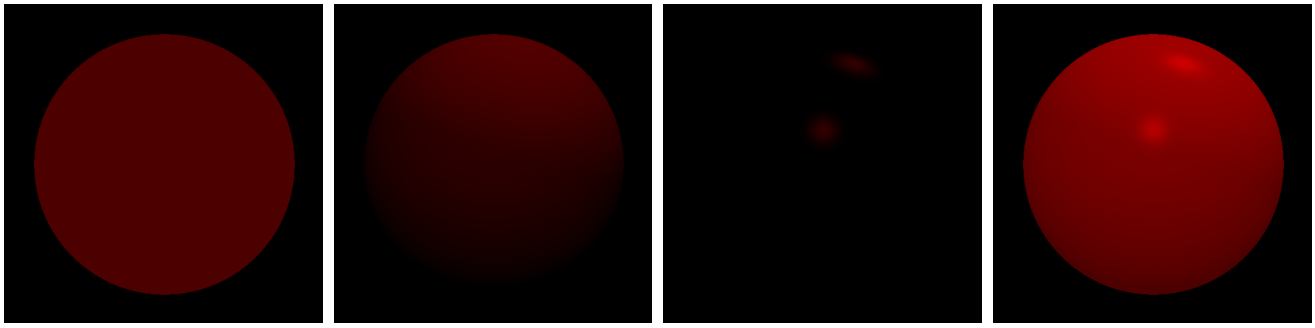
Рис. 15: Спрямоване джерело світла

2.1.3 Навколишнє освітлення

Коли світло падає на об'єкт, частина його поглинається, але інша частина розсіюється в сцені. Це означає, що світло може надходити не тільки від джерел освітлення, але і від інших об'єктів, які отримують його від джерел освітлення і розсіюють його назад. Але навіщо зупинятися на цьому? Розсіяне освітлення в свою чергу падає на який-небудь інший об'єкт, частина його поглинається, а частина знову розсіюється в сцені. Це означає, що потрібно вважати джерелом освітлення кожен об'єкт. Як можна уявити, це сильно збільшує складність моделі, тому ми не підемо таким шляхом. Але ми все одно не хочемо, щоб кожен об'єкт був чи освітлений безпосередньо, або був повністю темним. Щоб подолати цю перешкоду, ми поставимо третій тип джерел освітлення, званий навколишнім освітленням, яке характеризується тільки яскравістю. Вважається, що воно має незаперечний внесок освітлення в кожну точку сцени. Це дуже сильне спрощення надзвичайно складної взаємодії між джерелами освітлення і поверхнями сцени, але воно працює.

2.2 Модель Фонга

У загальному випадку, на сцені буде одне джерело навколишнього освітлення і довільна кількість точкових і спрямованих джерел. Для обчислення освітленості точки нам необхідно обчислити кількість світла, що вноситься кожним джерелом і скласти їх, щоб отримати одне число, яке представляє загальна кількість отриманого точкою освітлення. Потім колір поверхні в точці множиться на це число, щоб отримати правильно освітлений колір. Кожна точка поверхні має свої координати і в ній визначена нормаль до поверхні. Її освітленість складається з трьох компонент: навколишнє освітлення (ambient), розсіяне світло (diffuse) і відблискова складова (specular). Властивості джерела визначають потужність випромінювання для кожної з цих компонент, а властивості матеріалу поверхні визначають її здатність сприймати кожен вид освітлення.



(а) (б) (в) (г)

Рис. 16: (а) Фонова складова (б) Розсіяна складова (в) Відблискова складова (г) Сумарне освітлення

$$I = I_a + I_d + I_s$$

I_a - фонова складова (ambient);
I_d - розсіяна складова (diffuse);
I_s - відблискова складова (specular).

2.2.1 Дифузне відбиття світла

Коли промінь світла падає на матовий об'єкт, то через нерівності його поверхні на мікроскопічному рівні, він відбиває промінь на сцену рівномірно у всіх напрямках, тобто виходить «розсіяне» («дифузне») відбивання світла.

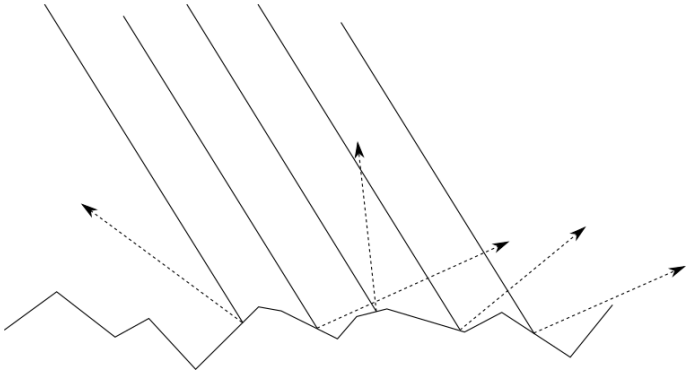


Рис. 17: Дифузне відбиття

Промінь світла з напрямком $\overrightarrow{L}$ і яскравістю i_d падає на поверхню з нормаллю $\overrightarrow{N}$. Яка частина I_d відбивається назад на сцену як функція від K_d, i_d, $\overrightarrow{N}$ і $\overrightarrow{L}$?

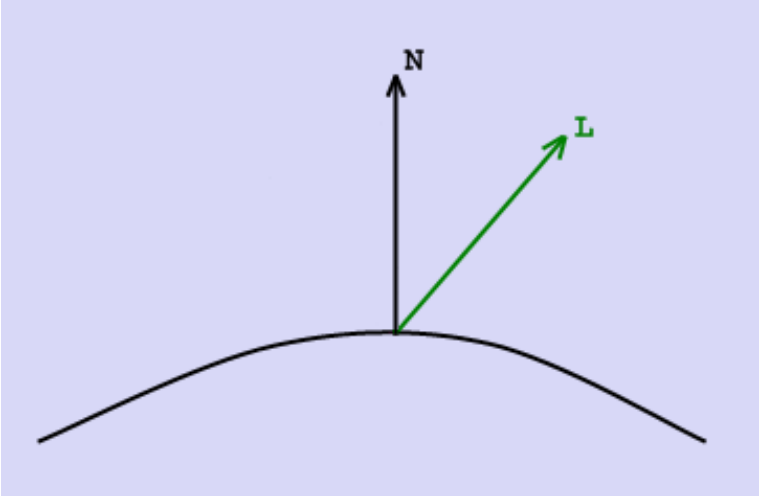


Рис. 18: Промінь і нормаль

Кут між вектором нормалі $\overrightarrow{N}$ і вектором напрямком світла $\overrightarrow{L}$ і буде мірою освітлення точки об'єкта.

$$I_d = K_d \cos(\overrightarrow{L}, \overrightarrow{N}) i_d = K_d \frac{\langle \overrightarrow{L}, \overrightarrow{N} \rangle}{|\overrightarrow{L}||\overrightarrow{N}|} i_d$$

I_d - розсіяна складова освітленості в точці;

K_d - властивість матеріалу сприймати розсіяне освітлення;

i_d - потужність розсіяного освітлення;

$\overrightarrow{L}$ - напрямок з точки на джерело;

$\overrightarrow{N}$ - вектор нормалі в точці.

Зауважимо, що при кутах більше $90°$, значення $\cos \alpha$ стає від'ємним. Якщо ми не замислюючись використовуємо це значення, то в результаті отримаємо джерела

світла, які віднімають світло. Це не має ніякого фізичного сенсу; кут більше 90° просто означає, що світло насправді досягає задньої частини поверхні, і не вносить свій внесок в освітленні точки. Тобто якщо $\cos\alpha$ стає від'ємним, то ми його ігноруємо.

2.2.2 Відбиття свяітла від гладкої поверхні

Коли промінь світла падає на дзеркало, він відбивається в єдиному напрямку, яке симетрично куту падіння відносно нормалі дзеркала. Якщо ми назвемо напрям відбитого світла $\overrightarrow{R}$ і домовимося, що $\overrightarrow{L}$ вказує на джерело світла, то отримаємо таку ситуацію:

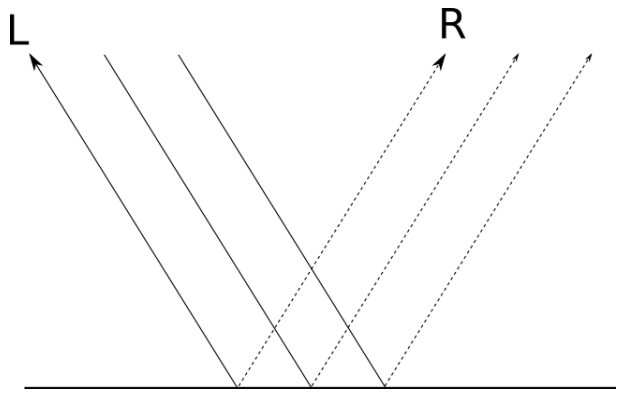


Рис. 19: Відбиття свяітла від дзеральної поверхні

Нас цікавить те, як з'ясувати, яка кількість світла від $\overrightarrow{L}$ відбивається назад в напрямку нашої точки огляду (бо це світло, яке ми використовуємо для визначення кольору кожної точки). Якщо $\overrightarrow{V}$ - це «вектор огляду», який вказує з P в камеру, а α - кут між $\overrightarrow{R}$ і $\overrightarrow{V}$, то ось, що ми маємо:

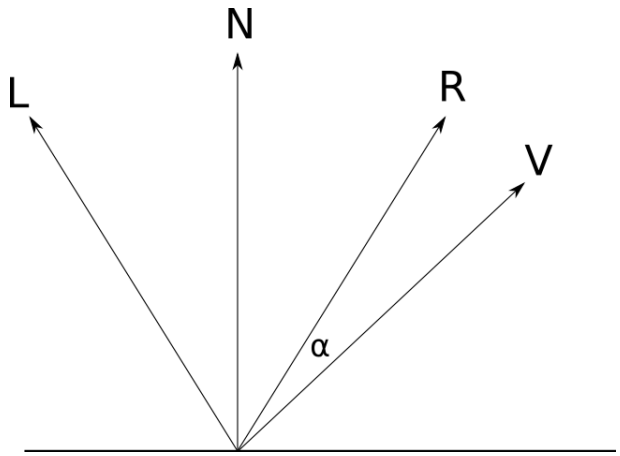


Рис. 20: Відбиття свяітла в напрямку точки огляду

Відблиск - міра того, наскільки швидко функція відбиття зменшується при

збільшенні α. Дуже простий спосіб отримання різних кривих блиску полягає в обчисленні степеня $\cos(\alpha)$.

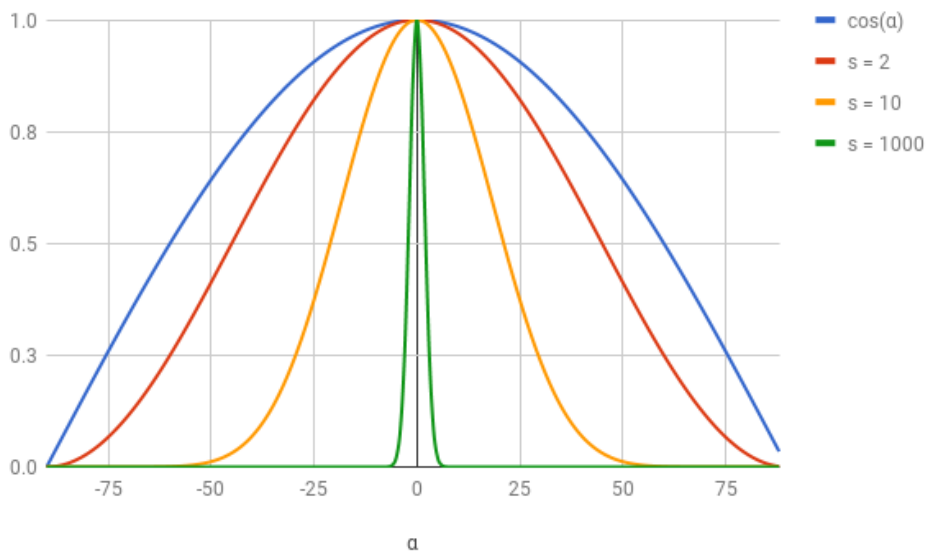


Рис. 21: Піднесення $\cos(\alpha)$ до степеня

Оскільки $0 \leq \cos(\alpha) \leq 1$, то очевидно, що $0 \leq \cos^s \alpha \leq 1$ Чим більше значення s, тим «вужче» стає функція в околі 0, і тим більш блискучим виглядає об'єкт. s зазвичай називають показником відбиття, і він є властивістю поверхні. Оскільки модель не заснована на фізичній реальності, значення можна визначити тільки методом проб і помилок, тобто налаштовуючи значення до тих пір, поки вони не почнуть виглядати «природно».

Промінь $\overrightarrow{L}$ падає на поверхню в точці P, де нормаль дорівнює $\overrightarrow{N}$, а показник відбиття - s. Яка кількість світла відіб'ється в напрямку огляду $\overrightarrow{V}$? Ми вже вирішили, що це значення дорівнює $\cos^s \alpha$, де α - це кут між $\overrightarrow{V}$ і $\overrightarrow{R}$, який в свою чергу є $\overrightarrow{L}$, відбитим відносно $\overrightarrow{N}$. Тобто першим кроком буде обчислення $\overrightarrow{R}$ з $\overrightarrow{N}$ і $\overrightarrow{L}$.

Ми можемо розкласти $\overrightarrow{L}$ на два вектора $\overrightarrow{L_P}$ і $\overrightarrow{L_N}$, таких, що, $\overrightarrow{L} = \overrightarrow{L_P} + \overrightarrow{L_N}$, де $\overrightarrow{L_N}$ паралельний $\overrightarrow{N}$, а $\overrightarrow{L_P}$ перпендикулярний $\overrightarrow{N}$:

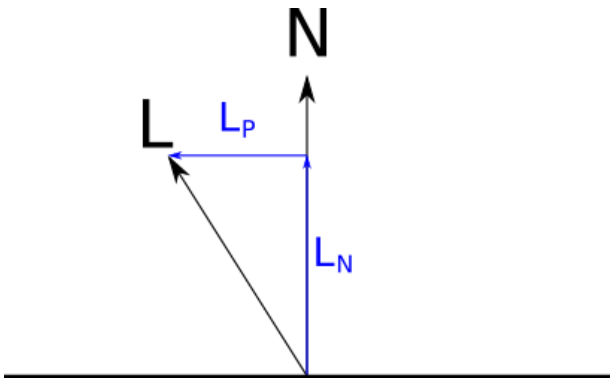


Рис. 22: Розкладання вектора

$\overrightarrow{L_N}$ - це проекція $\overrightarrow{L}$ на $\overrightarrow{N}$; за властивостями скалярного добутку і виходячи з того, що $|\overrightarrow{N}| = 1$, довжина цієї проекції дорівнює $\langle \overrightarrow{N}, \overrightarrow{L} \rangle$. Ми визначили, що $\overrightarrow{L_N}$ буде паралельний $\overrightarrow{N}$, тому $\overrightarrow{L_N} = \overrightarrow{N}\langle \overrightarrow{N}, \overrightarrow{L} \rangle$. Оскільки $\overrightarrow{L} = \overrightarrow{L_P} + \overrightarrow{L_N}$, відразу можна отримати $\overrightarrow{L_P} = \overrightarrow{L} - \overrightarrow{L_N} = \overrightarrow{L} - \overrightarrow{N}\langle \overrightarrow{N}, \overrightarrow{L} \rangle$.

Тепер розглянемо $\overrightarrow{R}$; оскільки він симетричний $\overrightarrow{L}$ відносно $\overrightarrow{N}$, його компонент, паралельний $\overrightarrow{N}$, той же, що і у $\overrightarrow{L}$, а перпендикулярний компонент протилежний компоненту $\overrightarrow{L}$; тобто $\overrightarrow{R} = \overrightarrow{L_N} - \overrightarrow{L_P}$:

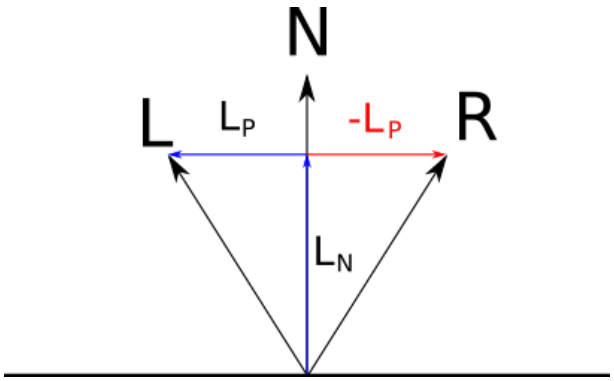


Рис. 23: Розкладання векторів

Підставляючи отримані раніше вирази, ми отримаємо

$$\overrightarrow{R} = \overrightarrow{N}\langle \overrightarrow{N}, \overrightarrow{L} \rangle - \overrightarrow{L} + \overrightarrow{N}\langle \overrightarrow{N}, \overrightarrow{L} \rangle$$

і трохи спростивши, отримуємо

$$\overrightarrow{R} = 2\overrightarrow{N}\langle \overrightarrow{N}, \overrightarrow{L} \rangle - \overrightarrow{L}$$

$$I_s = K_s \cos^s(\overrightarrow{R}, \overrightarrow{V}) i_s = K_s \left(\frac{\langle \overrightarrow{R}, \overrightarrow{V} \rangle}{|\overrightarrow{R}||\overrightarrow{V}|} \right)^s i_s$$

I_s - відблискова складова освітленості в точці;

K_s - коефіцієнт дзеркальності;

i_s - потужність дзеркального освітлення;

$\overrightarrow{R}$ - напрямок відбитого променя;

$\overrightarrow{V}$ - напрямок на спостерігача;

s - коефіцієнт блиску, властивість матеріалу.

Як і в разі дифузного освітлення, $\cos\alpha$ може бути від'ємним, і ми знову повинні це ігнорувати. Крім того, не кожен об'єкт повинен бути блискучим; для таких об'єктів (які ми будемо представляти через $s \leq 0$) значення «дзеркальності» взагалі не буде обчислюватися.

2.3 Тіні

Чому повинні бути тіні? Тіні з'являються там, де є світло, але його промені не можуть досягти об'єкта, тому що на їх шляху є інший об'єкт. Виділомо два наступних випадка:

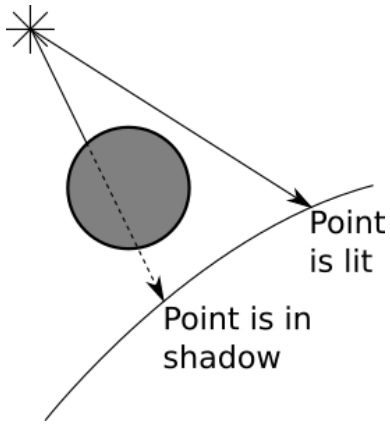


Рис. 24: Умова освітленості точки

Почнемо з направленого джерела світла. Ми знаємо P; це точка, яка нас цікавить. Ми знаємо $\overrightarrow{L}$; це частина визначення джерела освітлення. Маючи P та $\overrightarrow{L}$, можемо задати промінь, а саме $P + t\overrightarrow{L}$, який проходить з точки до нескінченно віддаленого джерела освітлення. Перетинає цей промінь інший об'єкт? Якщо ні, то між точкою і джерелом нічого немає, тобто ми можемо обчислити освітленість від цього джерела і додати його до загальної освітленості. Якщо перетинає, то ми ігноруємо це джерело.

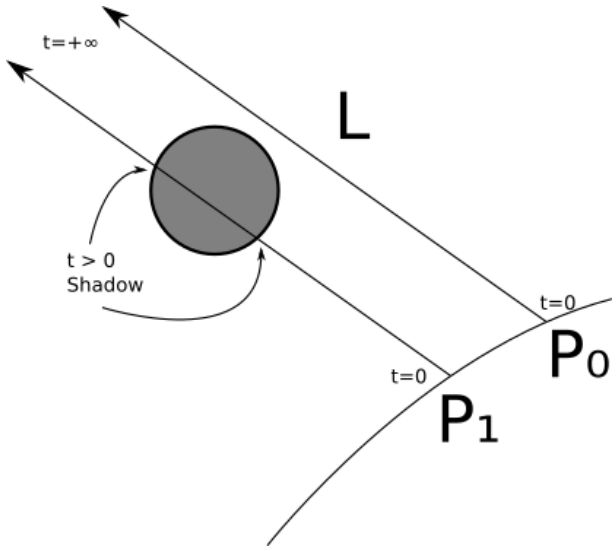


Рис. 25: Спрямоване джерело і перешкода

Нас цікавлять перетини з усіма об'єктами після P на нескінченній відстані; це означає, що $t \in [0, +\infty)$.

Ми можемо обробляти точкові джерела дуже схожим чином, але з двома винятками. По-перше, не заданий $\overrightarrow{L}$, але його дуже просто обчислити з позиції джерела і P. По-друге, нас цікавлять будь-які перетини, починаючи з P, але тільки до L (в іншому випадку, об'єкти за джерелом освітлення могли б створювати тіні!); тобто в цьому випадку $t \in [0, 1]$.

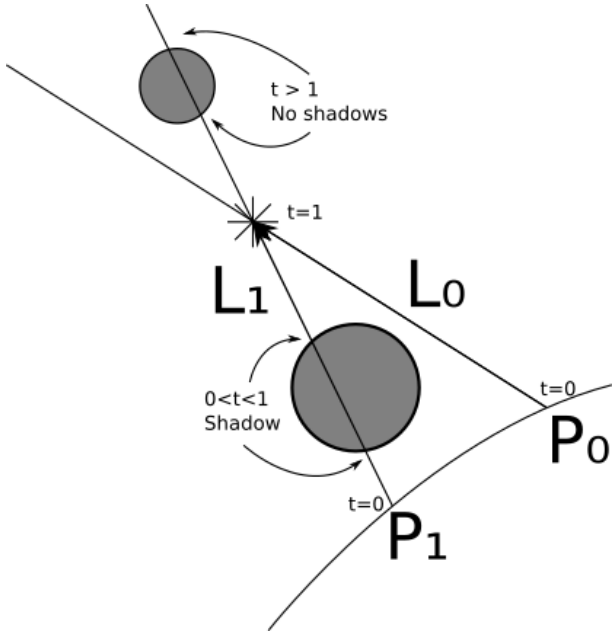


Рис. 26: Точкове джерело і перешкода

Існує один крайовий випадок, який нам потрібно розглянути. Візьмемо промінь $P + t\overrightarrow{L}$. Якщо ми будемо шукати перетини, починаючи з $t = 0$, швидше за все, знайдемо саму P при $t = 0$, тому що P дійсно знаходиться на об'єкті,

і $P + 0\overrightarrow{L} = P$; іншими словами, кожен об'єкт буде відкидати тіні сам на себе. Найпростіший спосіб справитися з цим - використовувати в якості нижньої межі значень t замість 0 мале значення ϵ.
Тобто для спрямованих джерел $t \in [0, +\infty)$, а для точкових - $t \in [\epsilon, 1]$.

3 Довільна камера в просторі

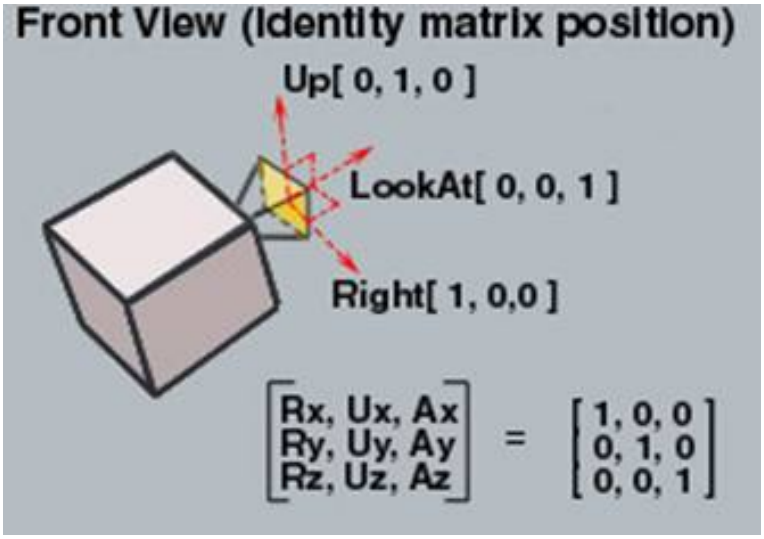


Рис. 27: Вектори камери

Для визначення камери нам потрібно її положення у світовому просторі, напрямок, куди вона дивиться, вектор, що вказує праворуч і вектор, спрямований вгору від камери. Такими векторами є LookAt, Right, Up позначимо їх $\overrightarrow{L}, \overrightarrow{R}, \overrightarrow{U}$ відповідно, а позиція задається точкою $O = (O_x, O_y, O_z)$. Початкове положення камери є $O = (0, 0, 0)$, а початковими значеннями векторів є $\overrightarrow{L} = (0, 0, 1)$(вздовж Z^+), $\overrightarrow{U} = (0, 1, 0)$(вздовж Y^+), а вектор $\overrightarrow{R} = \overrightarrow{U} \times \overrightarrow{L}$, тобто є векторним добутком двох векторів. Усі вектори нормовані.

Для переміщення камери вздовж векторів $\overrightarrow{R}, \overrightarrow{U}, \overrightarrow{L}$ достатньо просто помножити вектор на скаляр та додати до позиції камери. Тобто якщо необхідно перемістити камеру вперед з початковим положенням $O = (O_x, O_y, O_z)$, нове положеня матиме наступні координати $O^* = (O_x + cL_x, O_y + cL_y, O_z + cL_z)$, де c довільний скаляр.

Для обертання навколо осей використовуються матриці повороту:

$$M_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}$$

$$M_y(\alpha) = \begin{pmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{pmatrix}$$

$$M_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Загальна матриця повороту має вигляд:

$$M(\alpha, \beta, \gamma) = M_z(\alpha) M_y(\beta) M_x(\gamma)$$

Початкова матриця повороту має вигляд:

$$R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Тобто для повороту необхідно помножити початкову матрицю R на матрицю повороту $M(\alpha, \beta, \gamma)$.

$$R^* = R \cdot M(\alpha, \beta, \gamma)$$

і після цього обновити вектори камери:

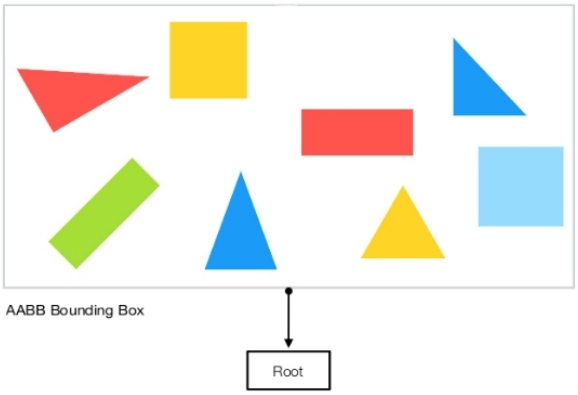
$$\overrightarrow{L}^* = R^* \cdot \overrightarrow{L}$$
$$\overrightarrow{U}^* = R^* \cdot \overrightarrow{U}$$
$$\overrightarrow{R}^* = \overrightarrow{U}^* \times \overrightarrow{L}^*$$

4 Оптимізація

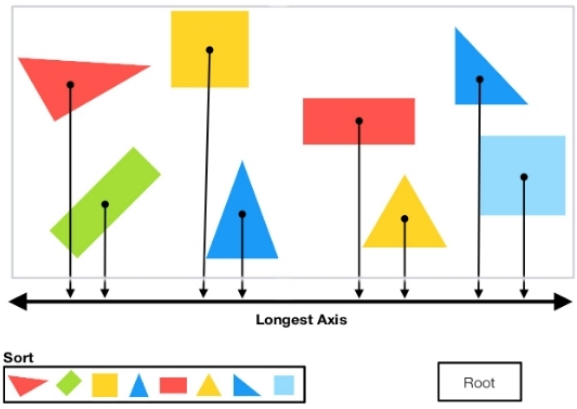
4.1 Ієрархія обмежуючих об'ємів

Ієрархія обмежуючих об'ємів (Bounding Volume Hierarchy - BVH) - це древовидна структура множини геометричних об'єктів. Усі геометричні об'єкти загорнуті в обмежувальні об'єми, що утворюють листові вузли дерева. Ці вузли потім групуються у вигляді невеликих наборів та укладаються в більші обмежувальні об'єми. Вони, у свою чергу, також групуються та укладаються в інші великі обмежувальні об'єми рекурсивно, з часом утворюючи деревову структуру з єдиним обмежуючим об'ємом у верхній частині дерева.

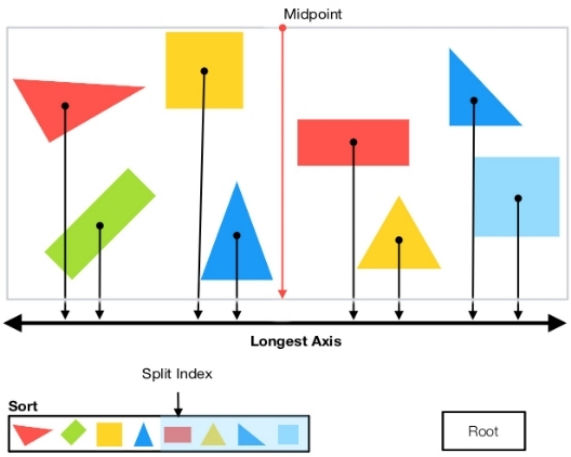
4.1.1 BVH алгоритм



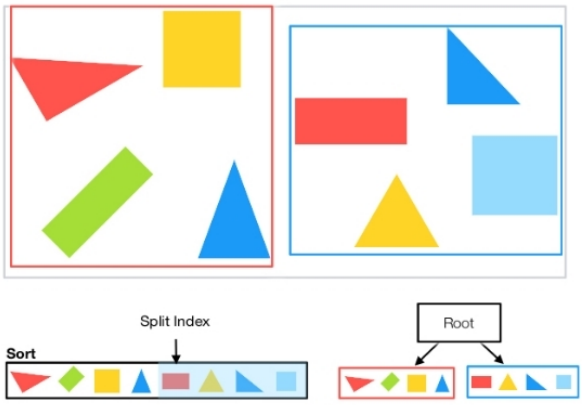
(а)



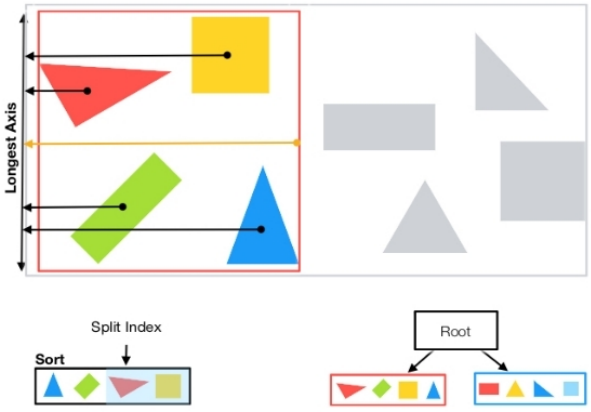
(б)



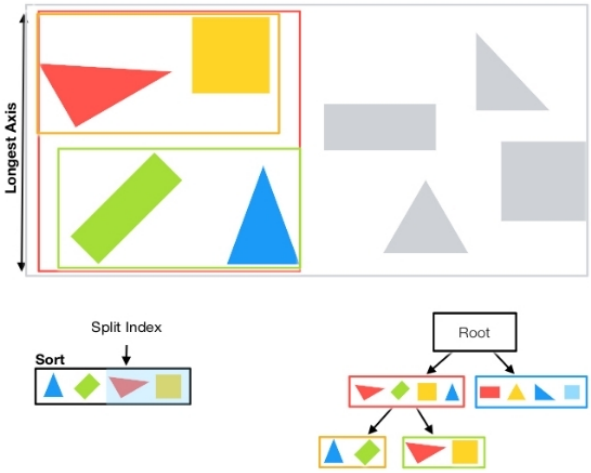
(в)



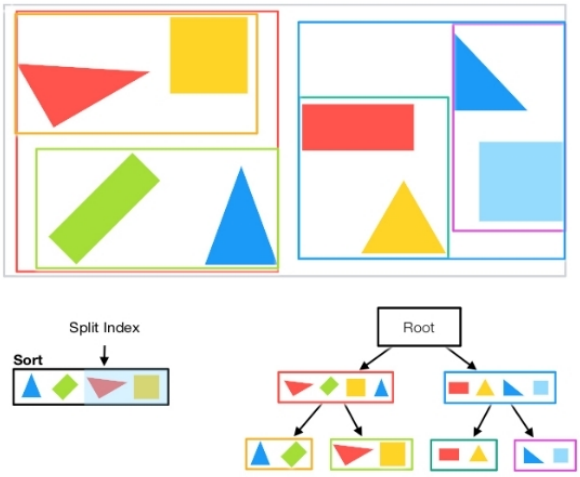
(г)



(д)



(е)



(ж)

(AABB - axis-aligned bounding box)

1. Створити кореневий вузол.
2. Створити AABB, що обмежує кожен об'єкт на сцені.
3. Знайти AABB кореневого вузла (Рис.(а)).
4. Знайти найдовшу вісь AABB і відсортувати усі об'єкти за цим напрямком (Рис.(б)).
5. Знайти середню точку (split index), яка ділить кореневий AABB (Рис.(в)).
6. За допомогою середньої точки ділимо сцену на ліву та праву частини.

7. Для кожної частини створити AABB, що міститьвідповідні об'єкти.

8. Створити лівий і правий вузол у бінарному дереві та знайти їх відповідні AABB (Рис.(г)).

9. Повторити кроки $4-8$ для кожного вузла, поки кожен вузол не містить максимум двох об'єктів.

Оскільки BVH алгоритм має деревовидну структуру, то не доведеться шукати перетин променя з кожним об'єктом. Складність алгоритму перетину променя з об'єктом буде $O(\log N)$, що має велику перевагу порівняно з $O(N)$.

4.2 Обчислення на графічному процесорі

Оскільки трасуванні усіх променей є незалежним процесом, то це можно оптимізувати. Найбільш очевидний спосіб прискорення роботи трасувальника променів - трасувати кілька променів одночасно. Оскільки кожен промінь, що виходить з камери, незалежний від усіх інших, а більшість структур даних призначені тільки для читання, ми можемо трасувати по одному променю на кожне ядро графічного процесора без особливих труднощів і складнощів з синхронізацією. Це можно зробити за допомогою OpenCL. OpenCL (від англ. Open Computing Language) — фреймворк для створення комп'ютерних програм, пов'язаних з паралельними обчисленнями на різних графічних (англ. GPU) і центральних процесорах (англ. CPU).

Висновки

В ході дипломної роботи був розглянуті алгоритми та розроблена програма, яка моделює трасування променів. За допомогою променів можна змоделювати реалістичне відображення. Були розглянуті способи задання об'єктів на сцені (аналітично та неаналітично), модель освітлення об'єктів та способи оптимізації. Подальше вдосконалення алгоритму для підвищення його ефективності є відкритим питанням. Існують ще алгорими для оптимізації та прискорення розрахунків, які не були розглянуті. Такими прикладами є оптимізація тіней, субдискретизація тощо.

Список використаних джерел

1. Kevin Suffern. Ray Tracing from the Ground Up. – A. K. Peters, Ltd. : Los Altos, California

2. Eric Lengyel. Mathematics for 3D Game Programming and Computer Graphics, Third Edition

3. Eric Haines. Pat Hanrahan, Robert L. Cook, James Arvo. An Introduction to Ray Tracing

4. Jamis Buck. The Ray Tracer Challenge: A Test-Driven Guide to Your First 3D Renderer

5. Ryoji Tsuchiyama. The OpenCL Programming Book

6. https://www.realtimerendering.com/raytracing/Ray%20Tracing%20in%20a%20Weekend.pdf

7. http://www.pbr-book.org/3ed-2018/Primitives_and_Intersection_Acceleration/Bounding_Volume_Hierarchies.html

8. http://hugi.scene.org/online/hugi24/coding%20graphics%20chris%20dragan%20raytracing%20shapes.htm

Додаток А. Демонстрація роботи програми