



# *Компютърна Графика и Презентации*

Алгоритми за Визуализация

*доц. д-р Александър Пенев*

# *Визуализация*

Построяване на изображение съответстващо на модел.

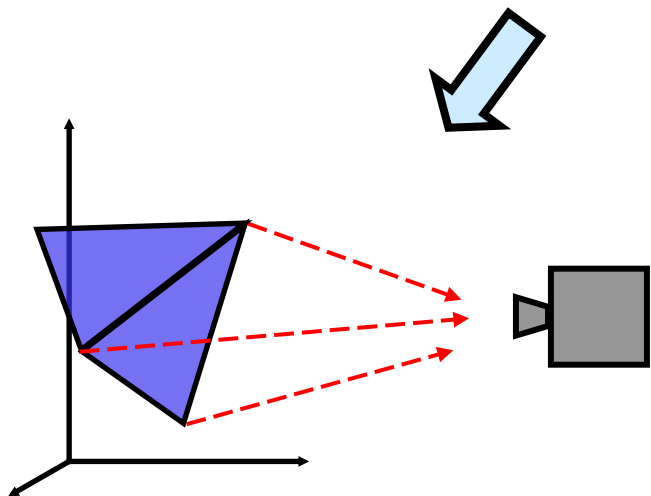
Операция по преобразуване на представяне на двумерни/тримерни обекти в графично изображение.

# Класификация

- ❖ Алгоритми работещи в обектното пространство;
  - ❖ Алгоритми работещи в екранното пространство;
  - ❖ Алгоритми работещи в екранното и в обектното пространство.
- 
- ❖ Отстраняване на невидимите линии и повърхности;
  - ❖ Построяване на реалистични изображения.

# Алгоритми за визуализация

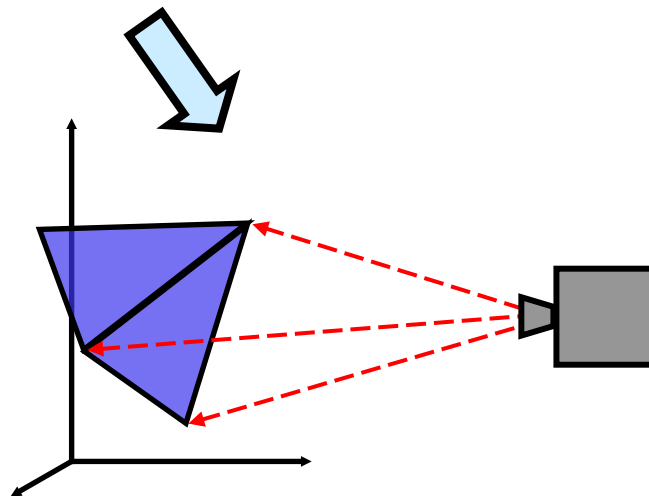
## ВИЗУАЛИЗАЦИЯ



### В обектното п-во:

От сцената напред към  
камерата/наблюдателя.

*(Пример: Z-буфер)*

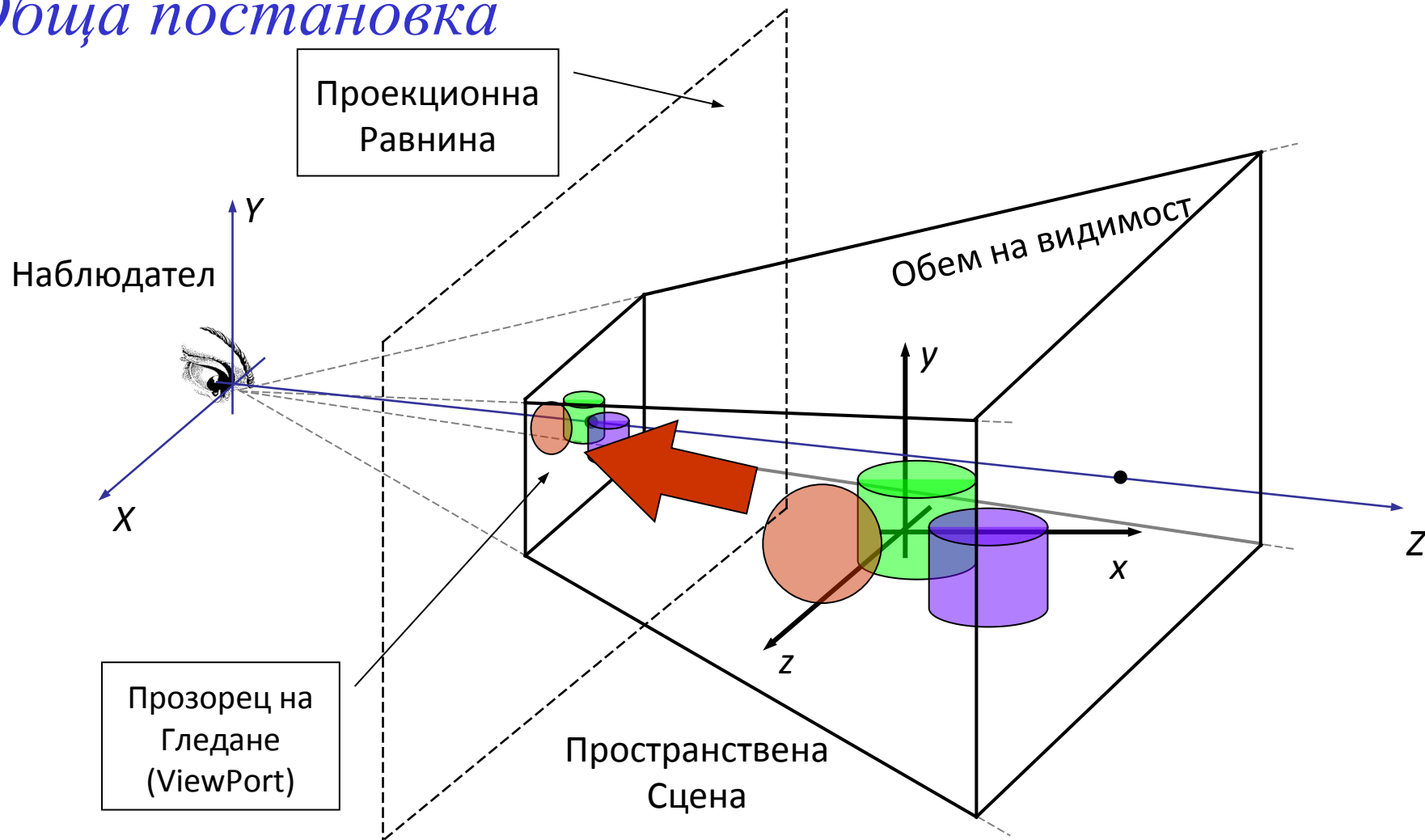


### В екранното п-во:

От камерата назад към  
сцената/геометрията.

*(Пример: Ray tracing)*

# Обща постановка



# Алгоритми за Визуализация

- ❖ Алгоритъм на плаващият хоризонт;
- ❖ Алгоритъм на Робъртс;
- ❖ Алгоритъм на Варнок;
- ❖ Разбиване на криволинейни повърхности;
- ❖ Алгоритъм, използващ Z-буфер;
- ❖ Алгоритъм, използващ списък на приоритетите;
- ❖ Алгоритми за поредово сканиране;
- ❖ Интервален алгоритми за поредово сканиране;
- ❖ Radiosity;
- ❖ Трасиране на лъчи (Ray Tracing);



# *Алгоритъм на плаващия хоризонт*

# *Алгоритъм на плаващия хоризонт*

Алгоритъмът на плаващият хоризонт се използва най-често за отстраняването на невидимите линии при тримерно представяне на функции, описващи повърхности във вида  $F(x,y,z)=0$

# *Алгоритъм на плаващия хоризонт*

Основната идея е да се визуализират последователно сечения на функцията с различни равнини (например  $z=\text{const}$ ), започвайки от най-близката до наблюдателя.

# *Алгоритъм на плаващия хоризонт*

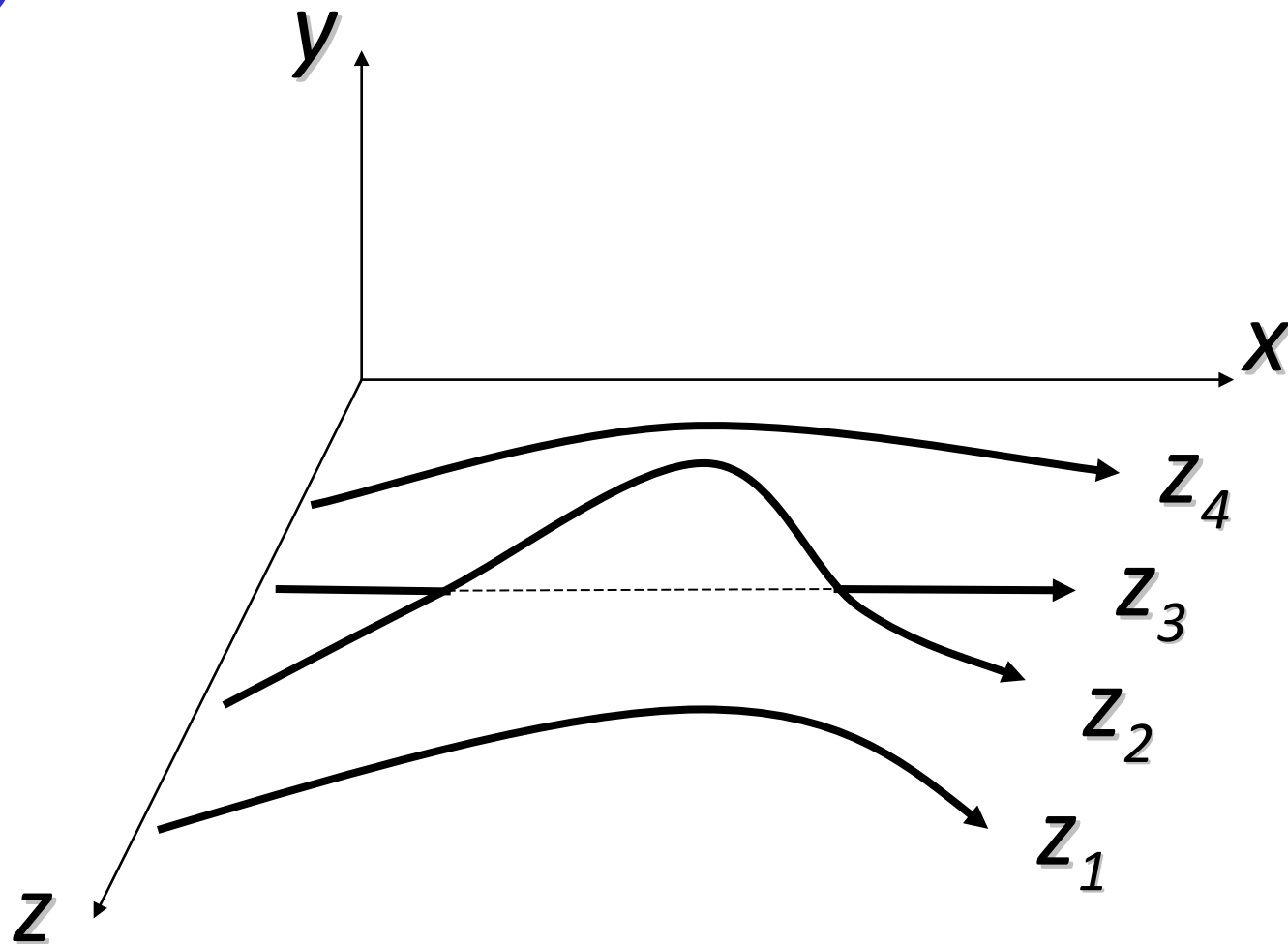
Всяко сечение е крива, която се визуализира по  $x$ , като се показват само тези точки, за които  $y$  не е по-малко от това на предишните сечения.

## *Алгоритъм на плаващия хоризонт*

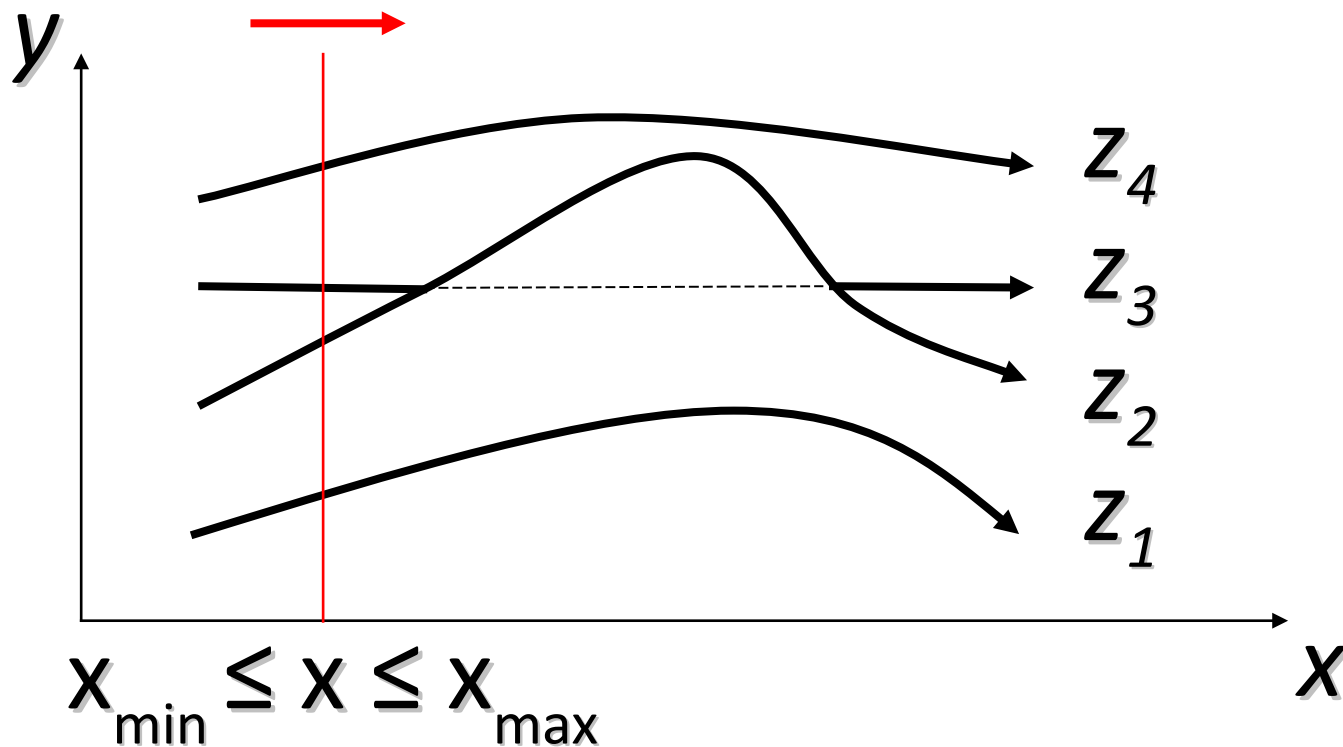
За целта се използва масив съхраняващ  
максималните стойности на  $y$  за всяко  $x$  до  
момента.

Това е така нареченият **хоризонт**.

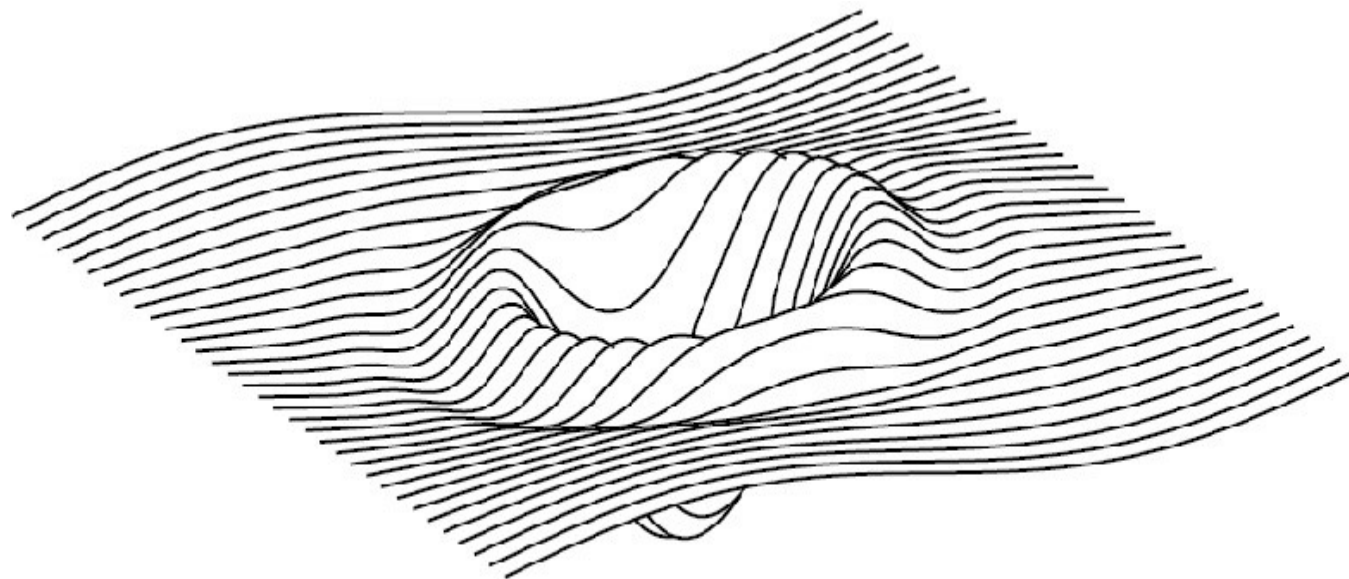
# Пример



# Пример



## Пример 2



# *Алгоритъм на Робъртс*

# Алгоритъм на Робъртс

Основната идея е за тримерните тела (изпъкнали многостени, описани чрез стените си) да се пресметнат ъглите между посоката на гледане на наблюдателя и нормалните вектори на всички стени.

Това става чрез **скалярно** произведение на вектора на посоката на гледане и всеки от нормалните вектори на стените.



L. Roberts

# Алгоритъм на Робъртс

Телата трябва да са изпъкнали. Нормалните вектори да сочат в посока “навън” от тялото.

Определят се кои стени не са видими.



## Алгоритъм на Робъртс

Определят се кои стени не са видими в зависимост от ъглите. В зависимост от видимостта на стените ребрата може да се класифицират като:

- ❖ Видими;
- ❖ Контурни;
- ❖ Невидими.

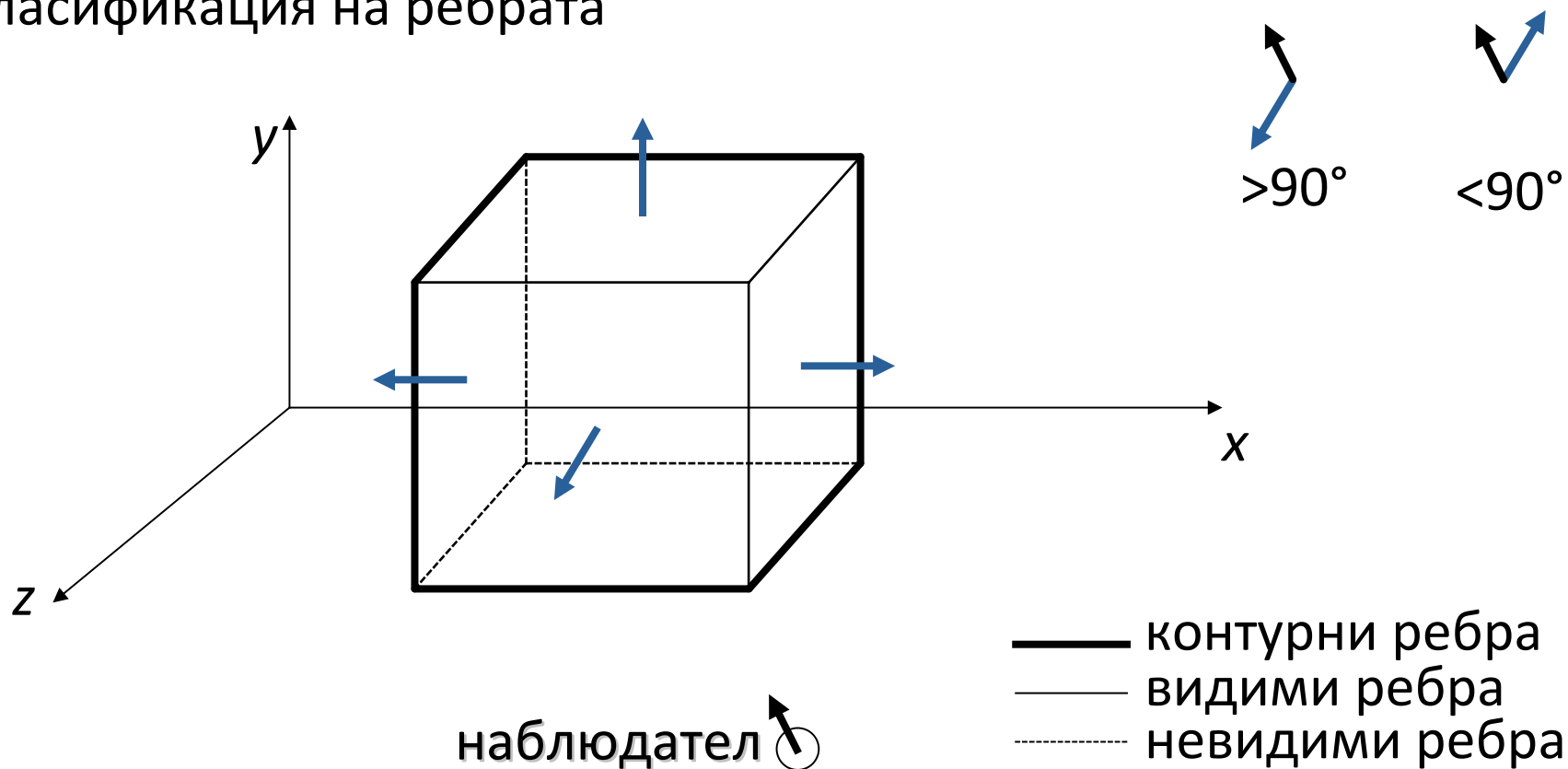
## *Алгоритъм на Робъртс*

За всички видими се прави проверка за закриване от всички други тела и се определят видимите части.

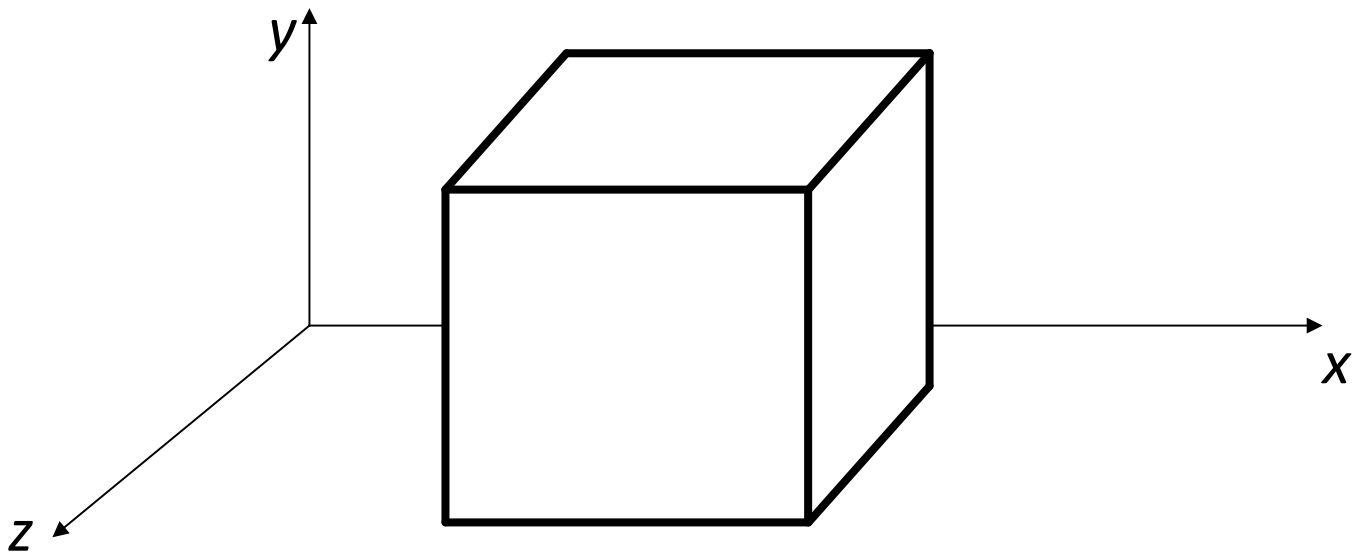
Визуализират се всички без невидимите.

# Алгоритъм на Робъртс

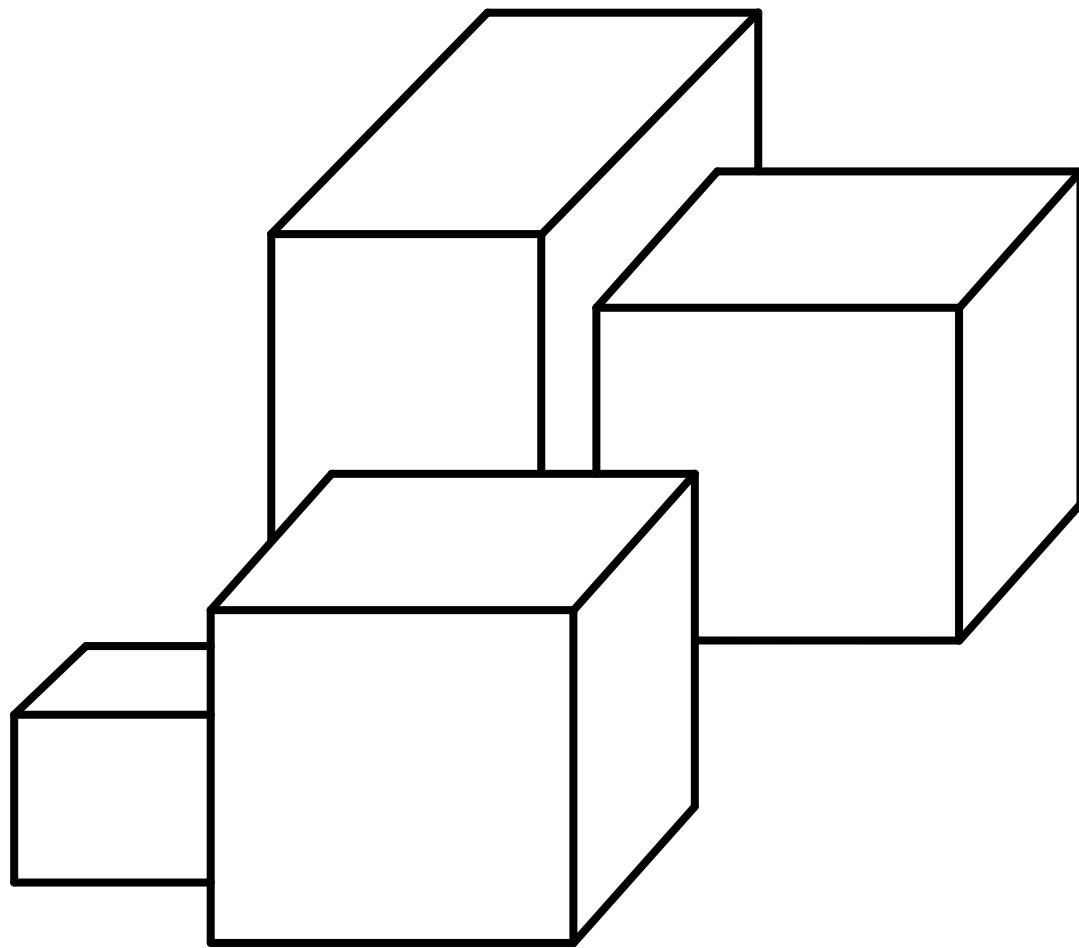
## Класификация на ребрата



# Алгоритъм на Робъртс



# Алгоритъм на Робъртс

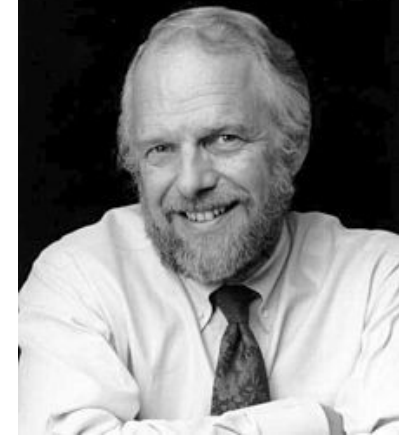


# *Алгоритъм на Варнок*

# Алгоритъм на Варнок

Идеята е че за обработката на области, съдържащи малко информация, се изразходват малко време и усилие.

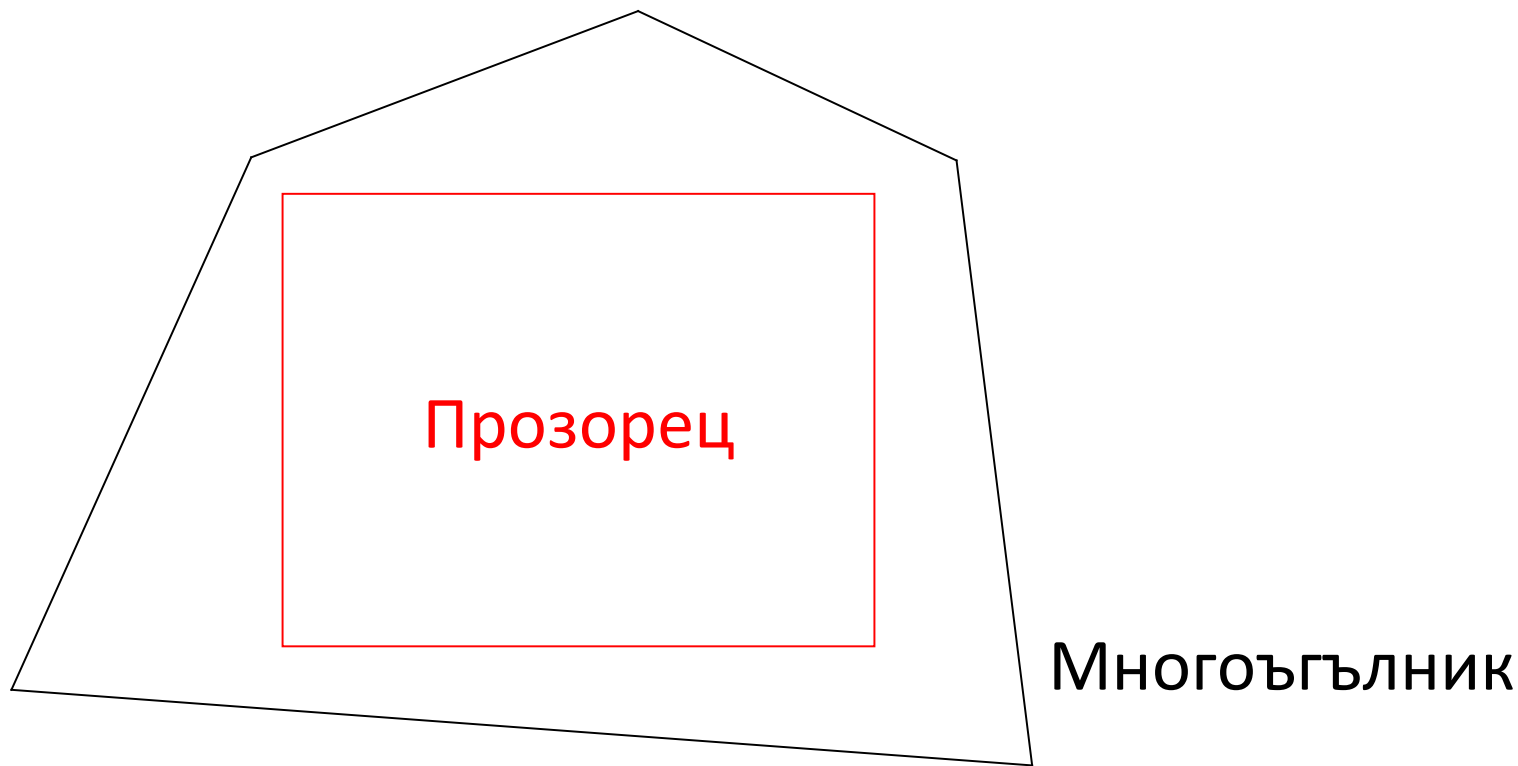
Използва се свойството на изображението **кохерентност**.



J. Warnock

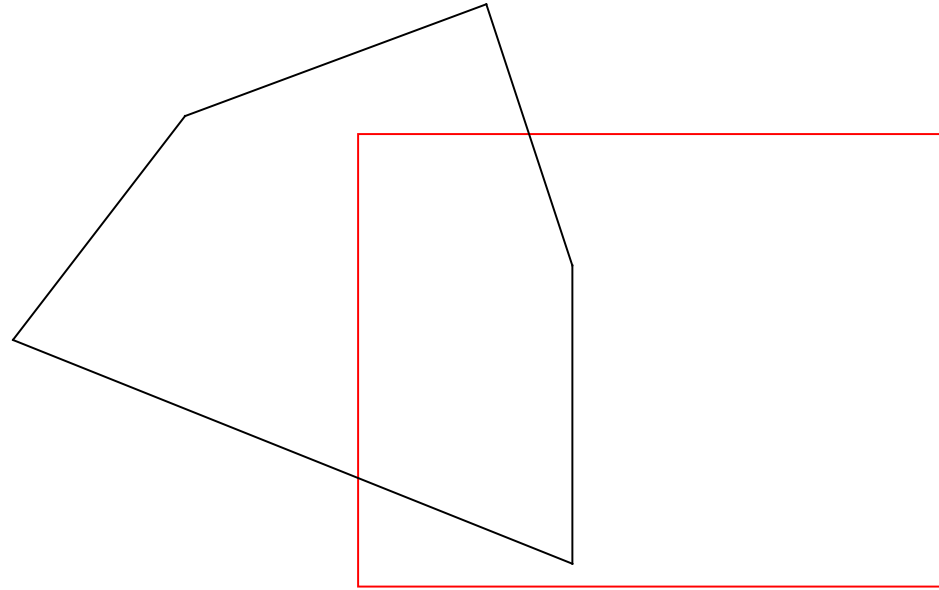
По-голямата част от времето и труда се ангажират от области с високо информационно съдържание.

## Типизация на многоъгълник относно прозорец



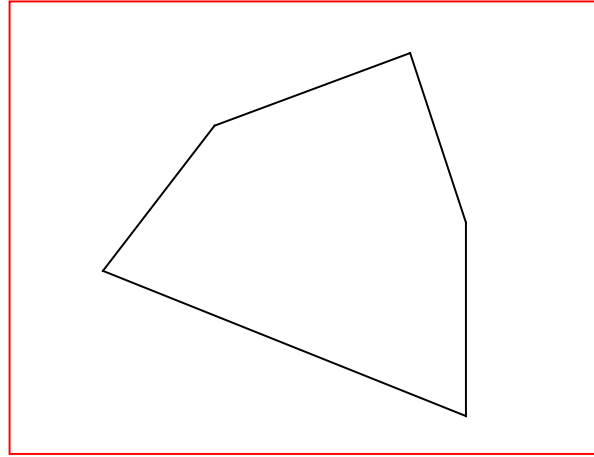
**Обхващащ**, ако прозореца се намира изцяло вътре в многоъгълника.

## Типизация на многоъгълник относно прозорец



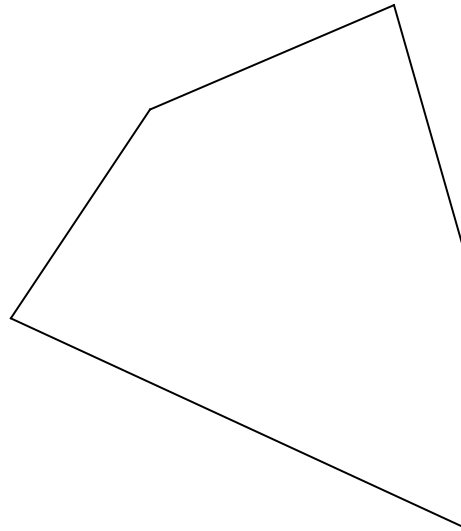
**Пресичащ**, ако вътрешността и границата на многоъгълника имат общи точки с вътрешността и границата на прозореца.

## Типизация на многоъгълник относно прозорец



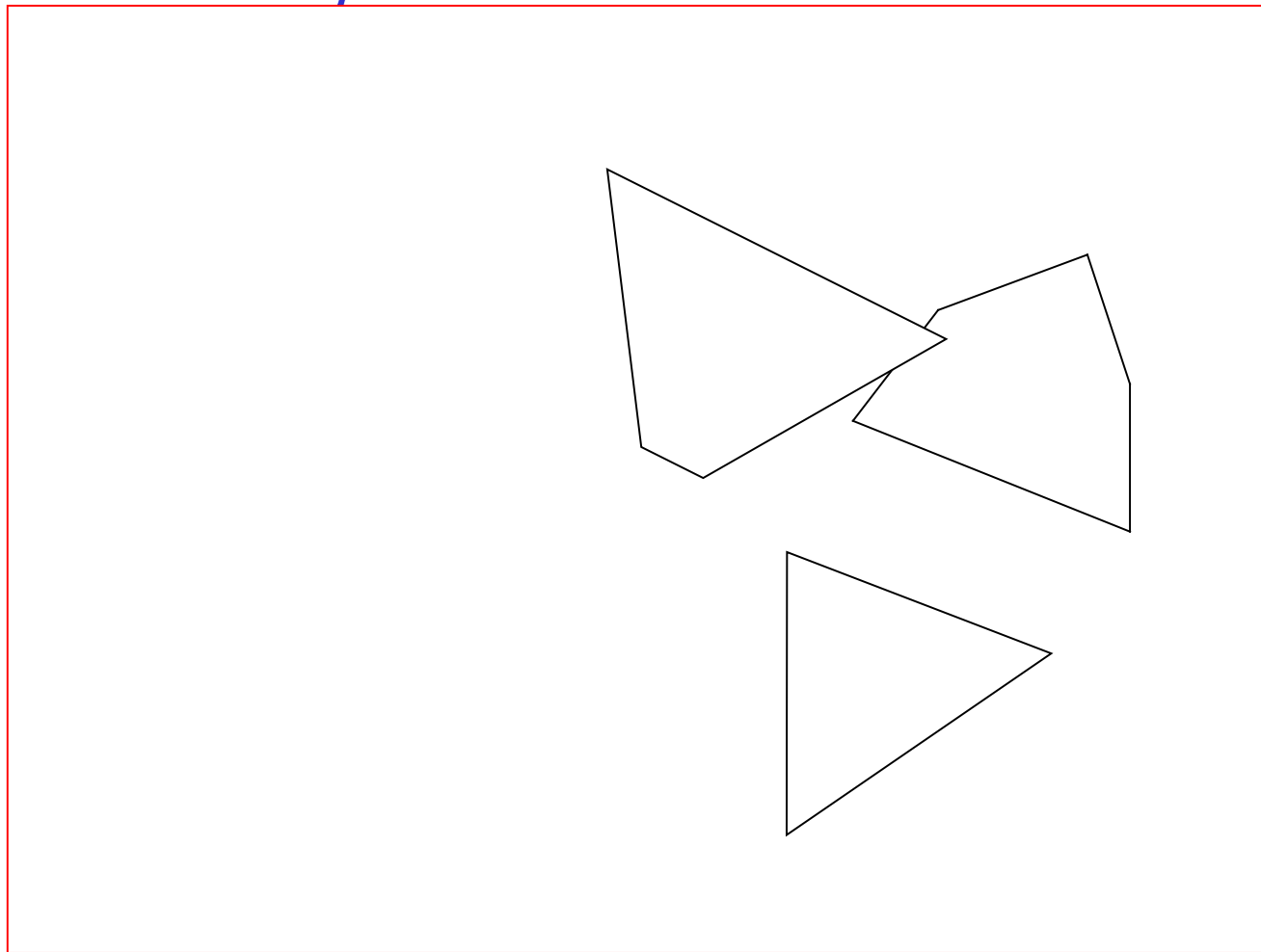
**Вътрешен**, ако той се намира изцяло вътре в прозореца.

## Типизация на многоъгълник относно прозорец

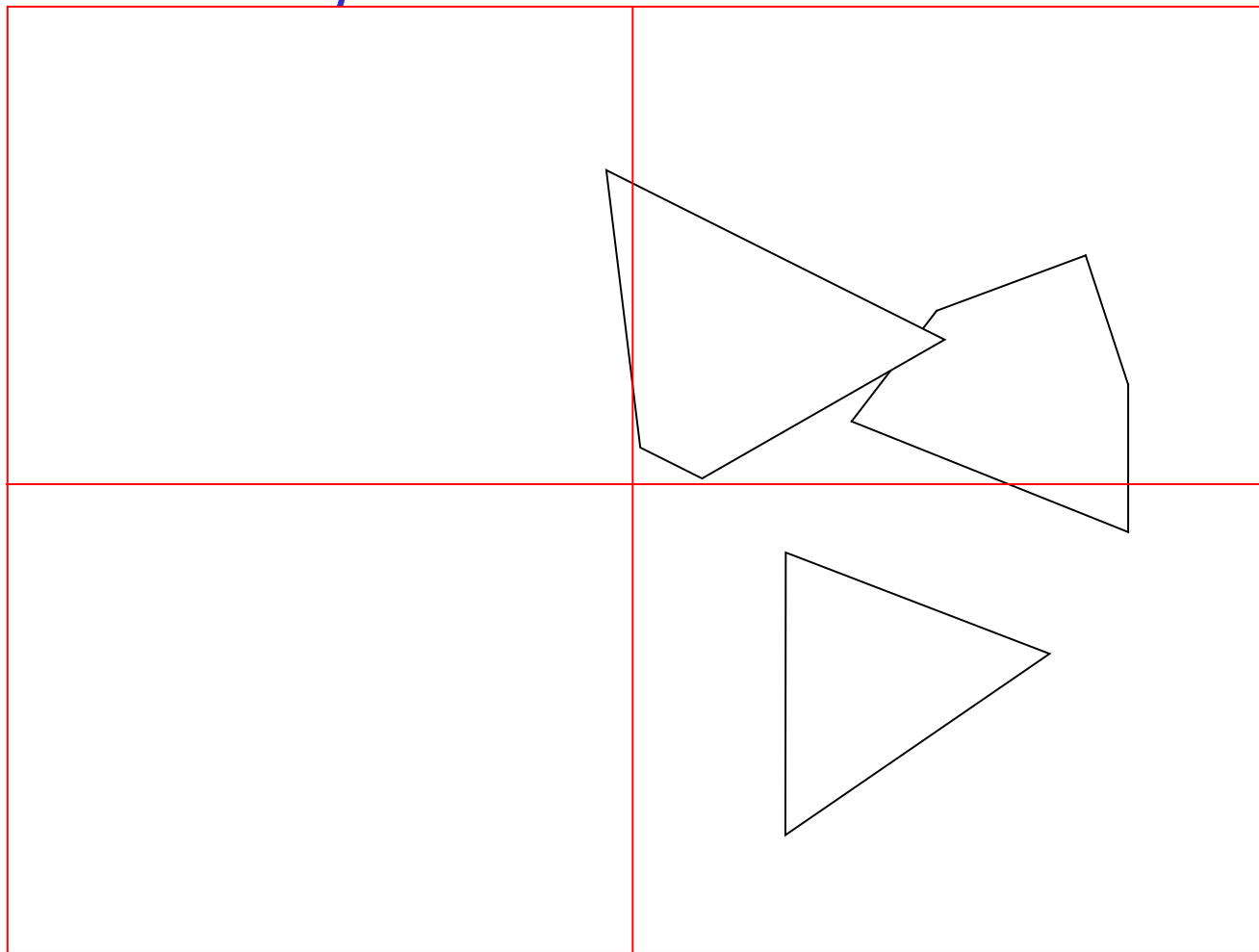


**Външен**, ако той се намира изцяло извън прозореца.

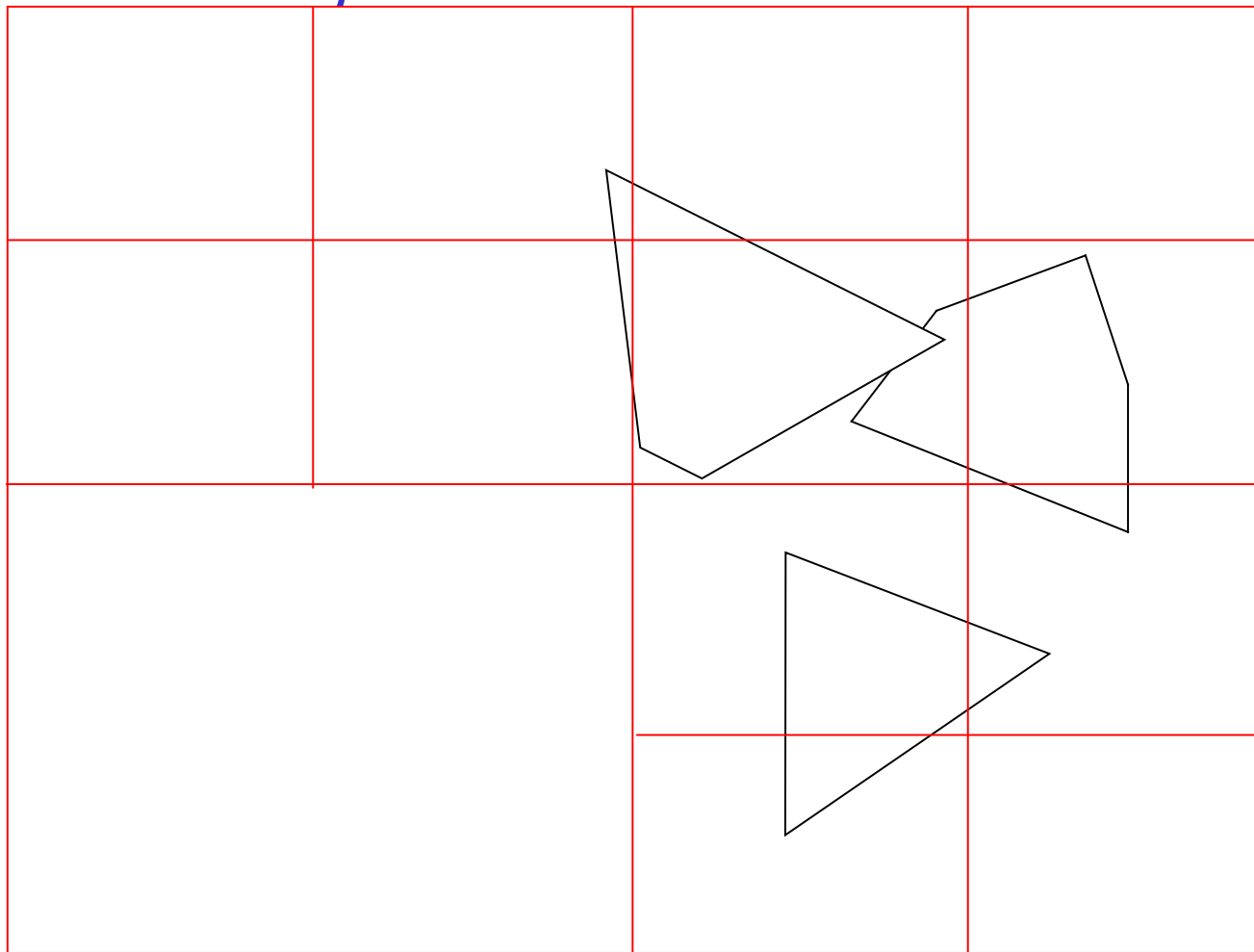
# Алгоритъм на Варнок



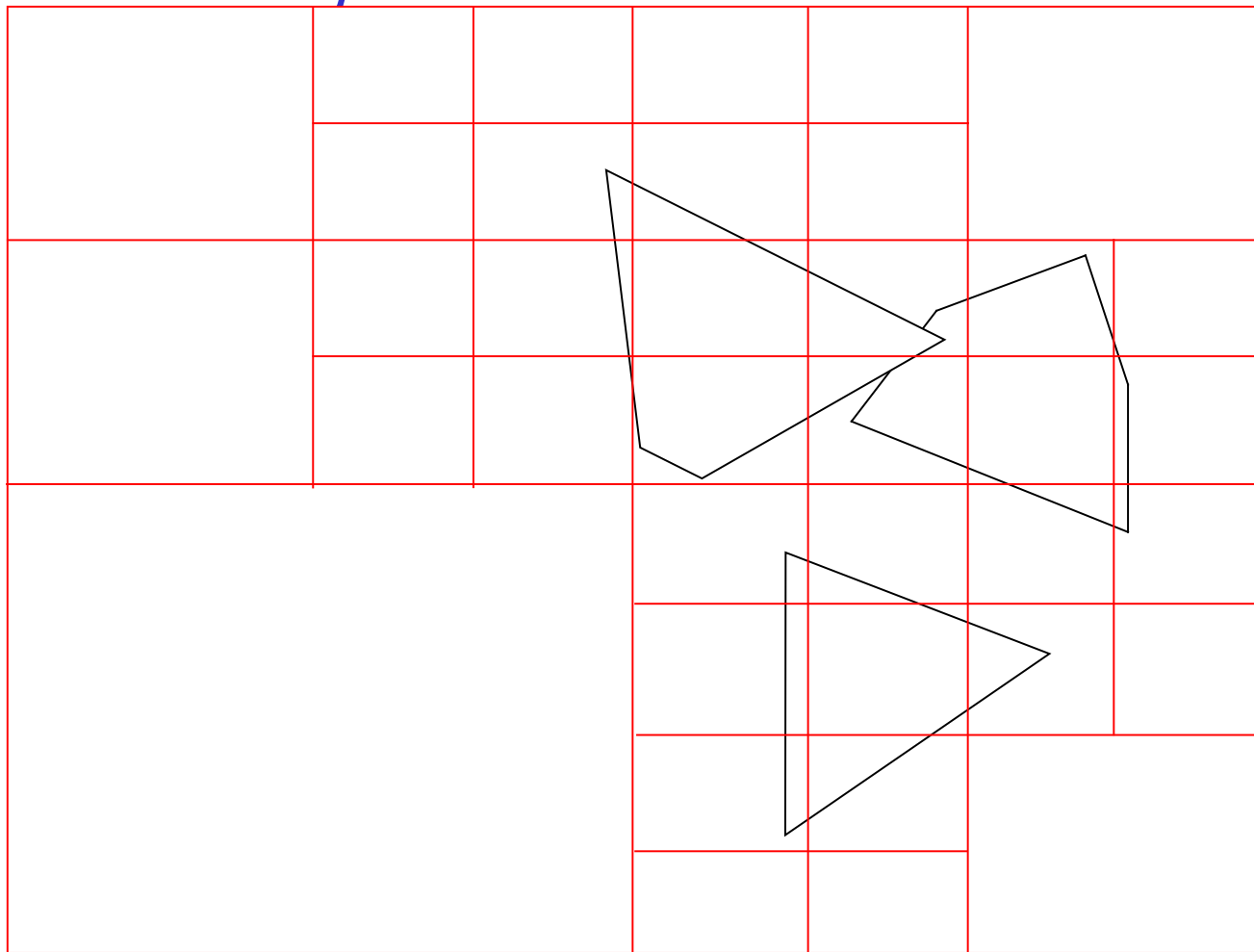
# Алгоритъм на Варнок



# Алгоритъм на Варнок



# Алгоритъм на Варнок



# *Алгоритъм, използващ Z-буфер*

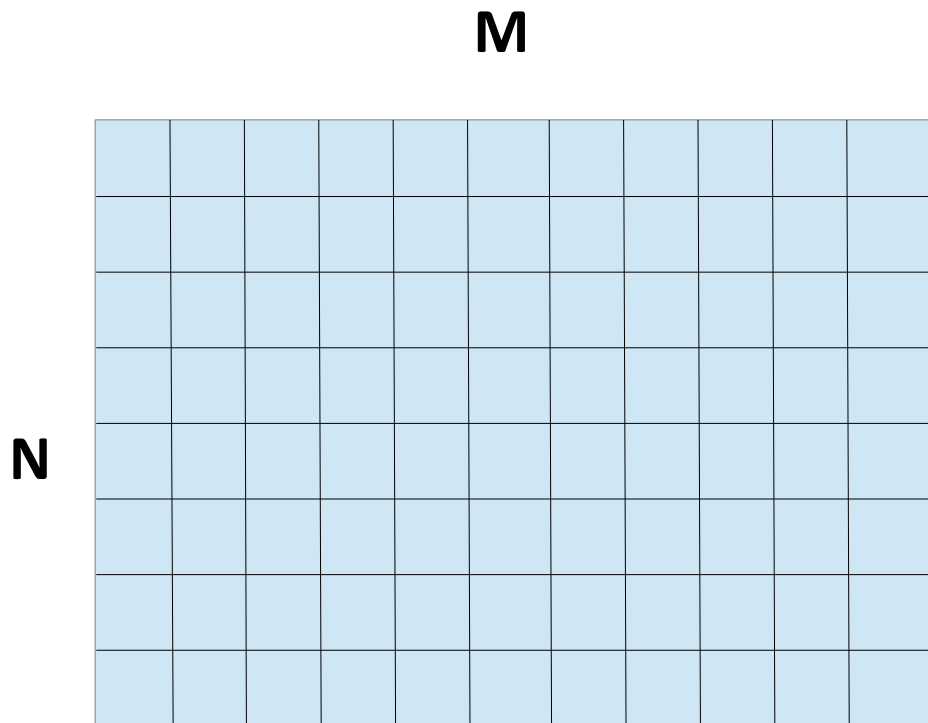
# Алгоритъм, използващ Z-буфер



Ed. Catmull

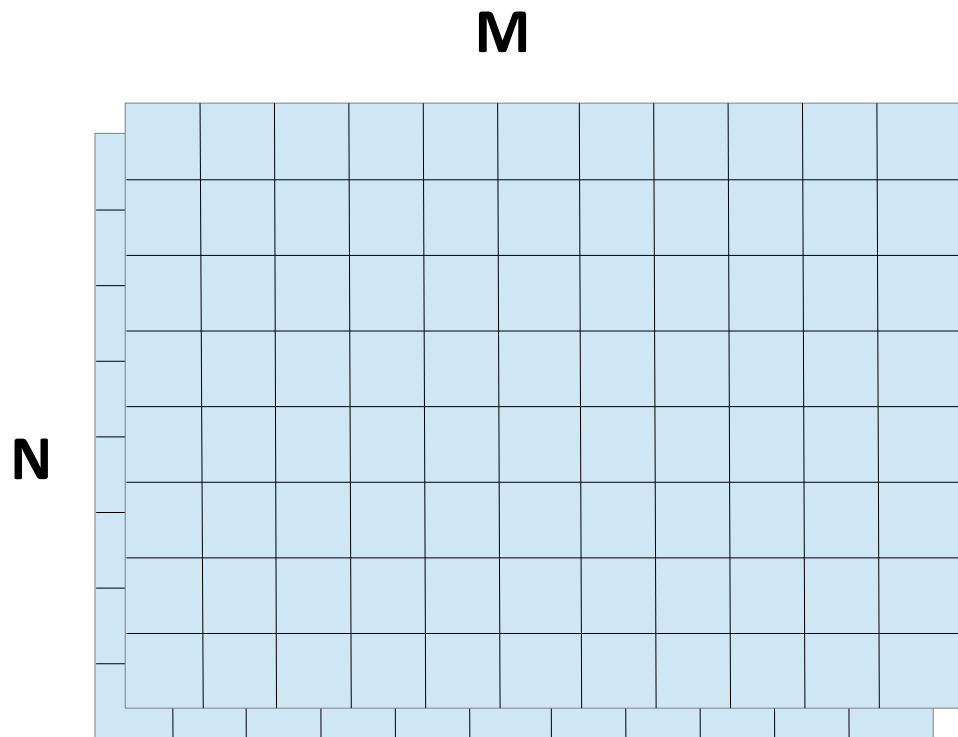
- ❖ Това е един от най-простите алгоритми за отстраняване на невидими повърхности;
- ❖ За пръв път той е предложен от Кетмул;
- ❖ Алгоритъмът работи в пространството на изображенията. Идеята за Z-буфер е просто обобщение на идеята за буфер на кадъра.

# Алгоритъм, използващ Z-буфер



Цветовете, които съответстват на всеки пиксел се записват в матрица  $M \times N$ , която наричаме буфер на кадъра (Frame buffer или Color buffer).

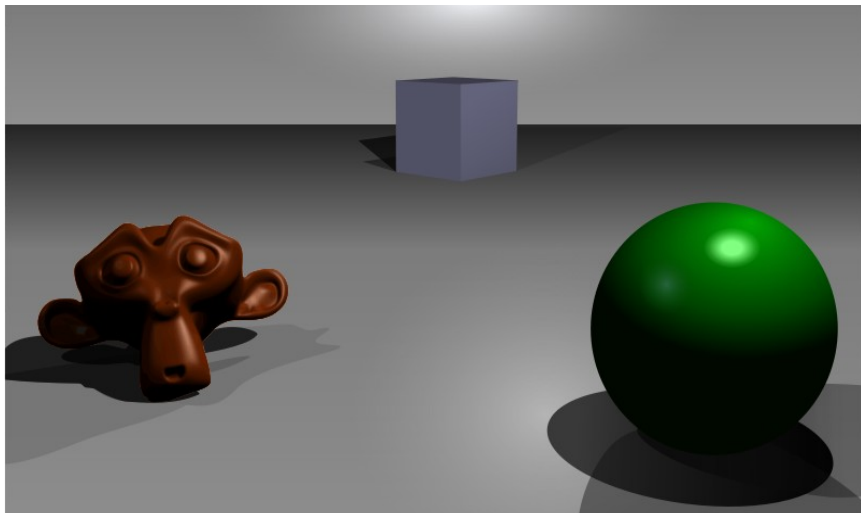
# Алгоритъм, използващ Z-буфер



Цветовете, които съответстват на всеки пиксел се записват в матрица  $M \times N$ , която наричаме буфер на кадъра (Color Buffer).

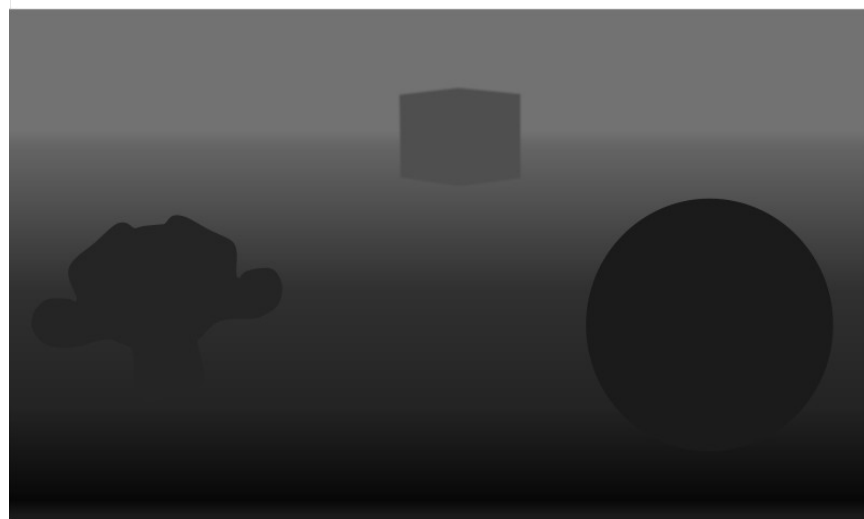
Използва се втори буфер – на дълбочината (Depth Buffer), в който се записва най-близкото до наблюдателя разстояние.

## Пример



Буфер на цвета  
(Color Buffer)

Буфер на дълбочината  
(Depth Buffer)



# *Алгоритъм, използващ списък на приоритетите*

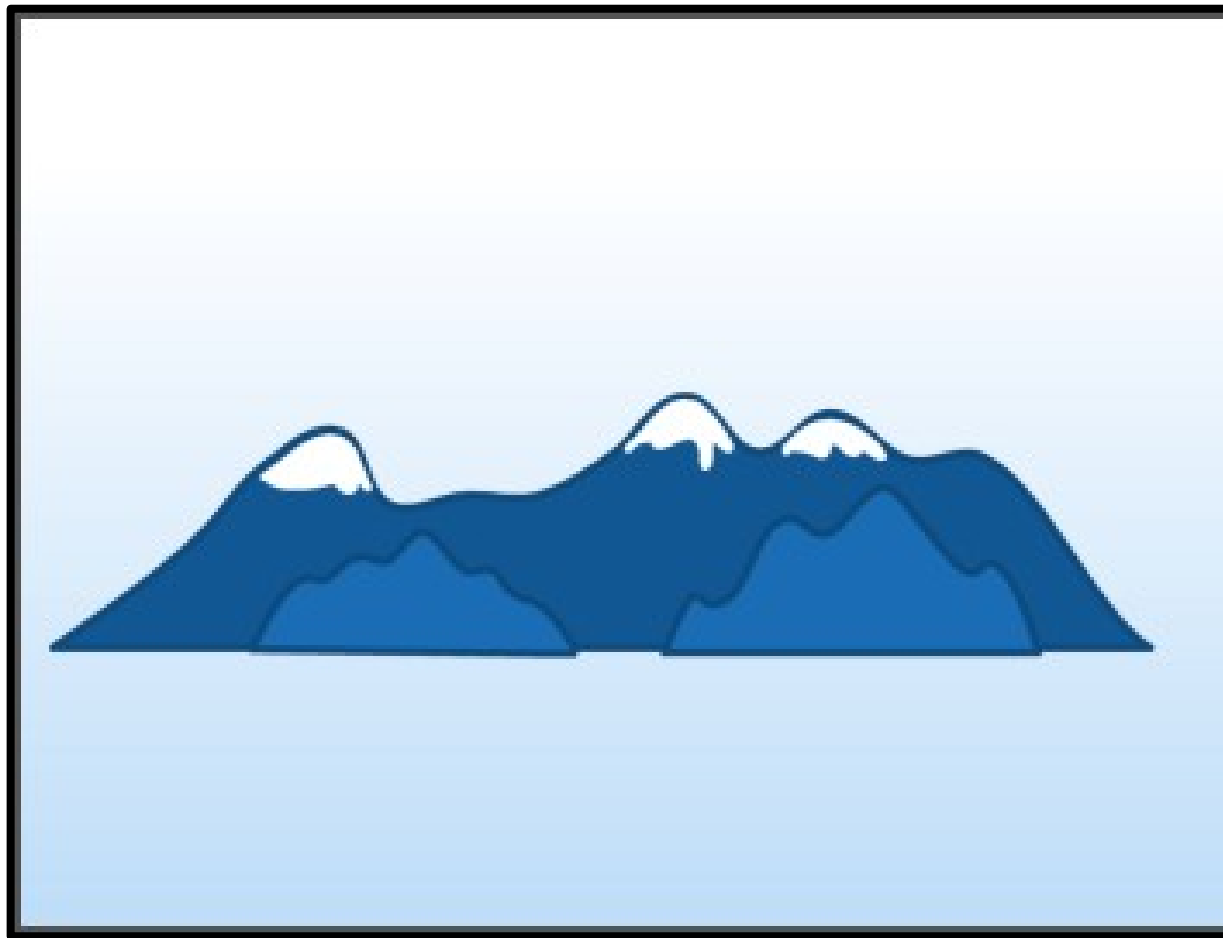
*(Алгоритъм на художника)*

## *Алгоритъм, използващ списък на приоритетите*

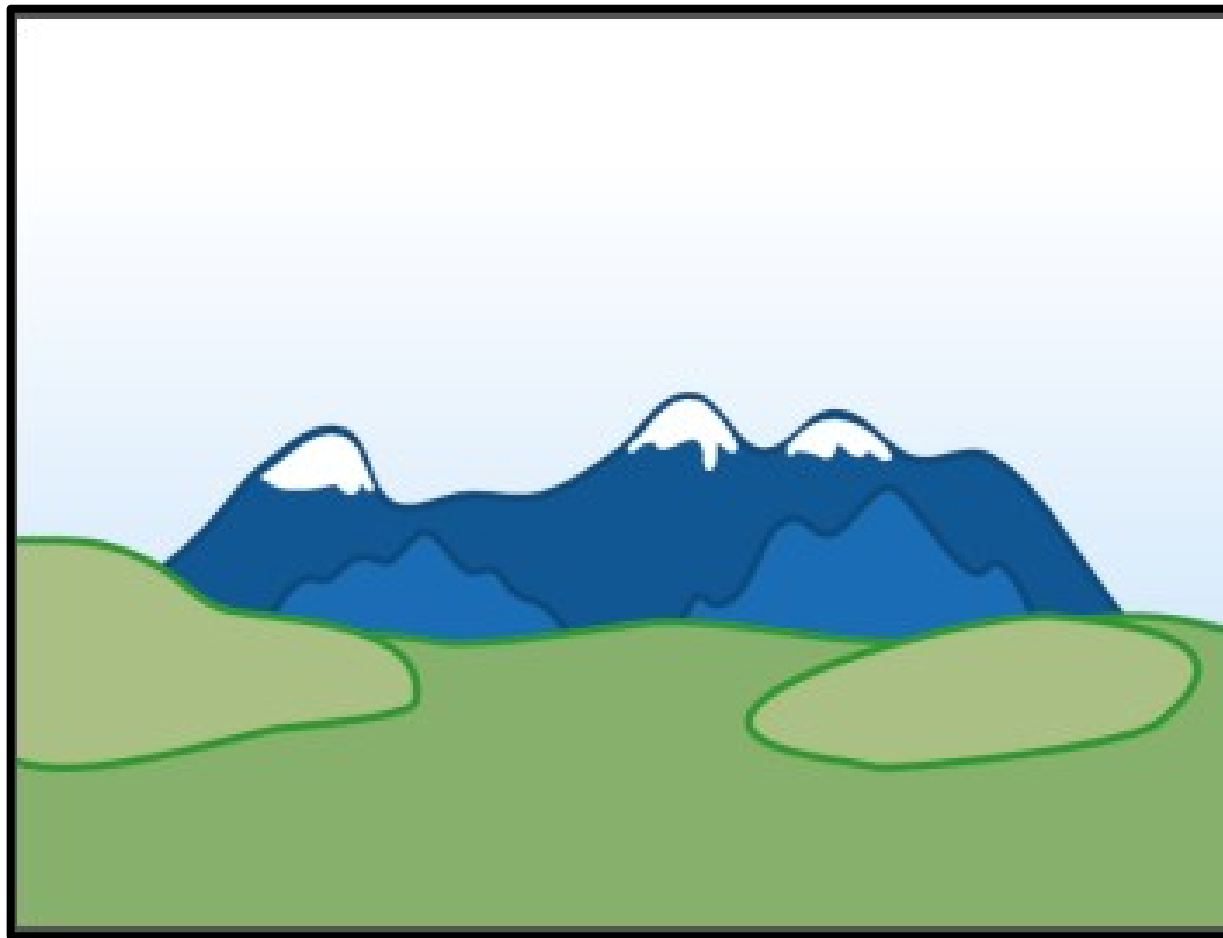
Основната идея е многоъгълниците да се наредят по някакъв критерии (например по отдалеченост от наблюдателят) и да се изрисуват в обратна посока.

Този алгоритъм още се нарича  
Алгоритъм на художника.

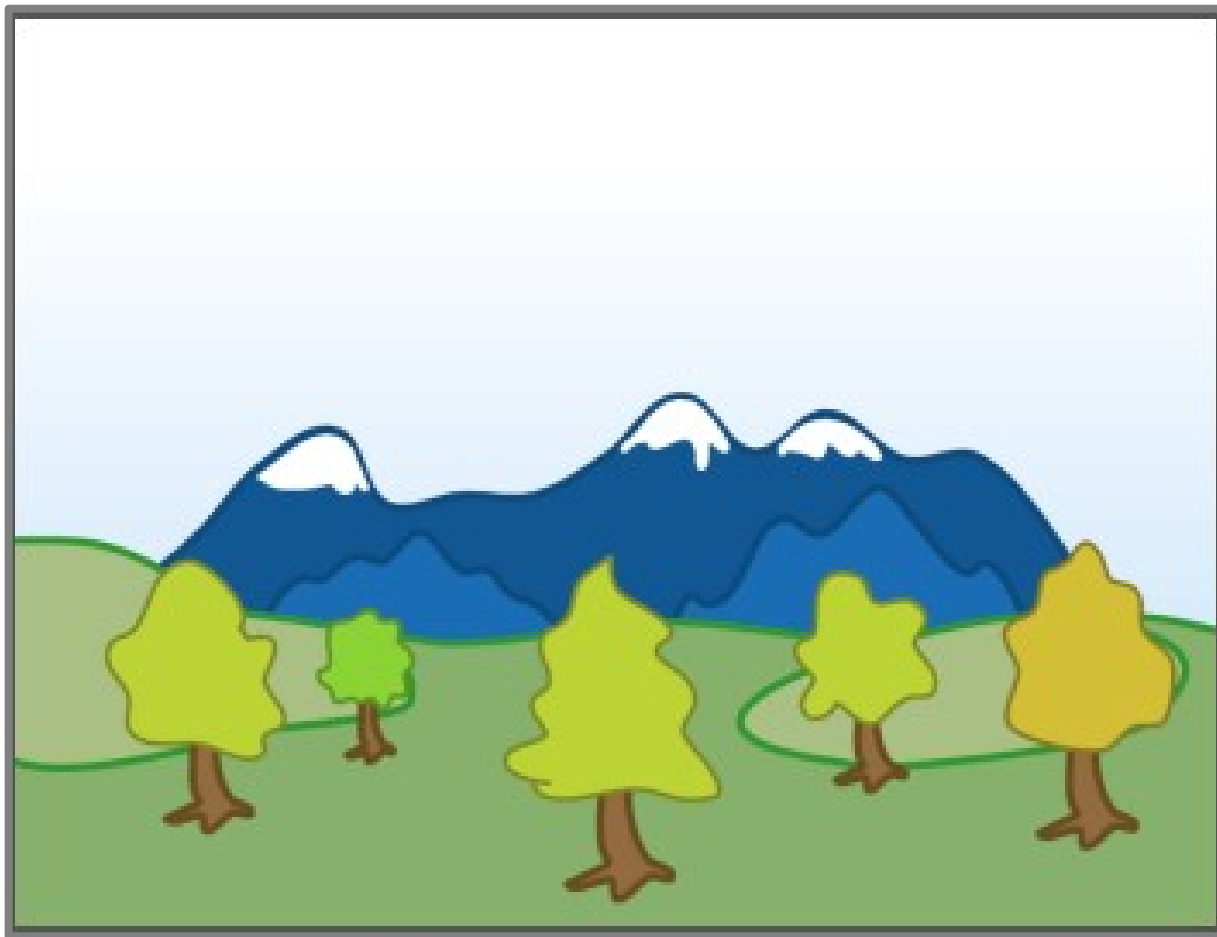
# Пример



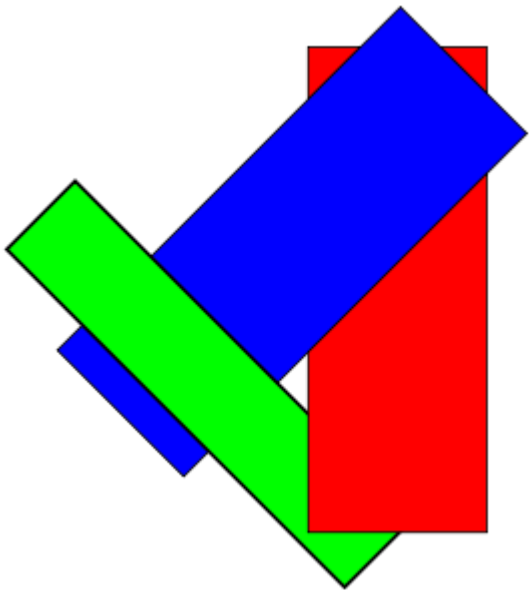
# Пример



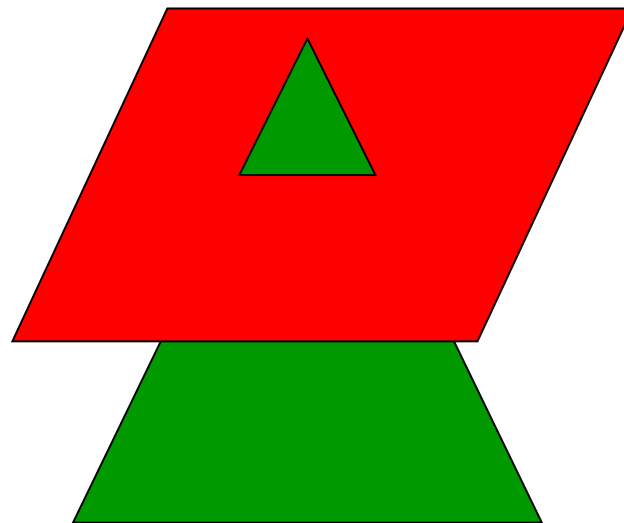
# Пример



# Проблеми



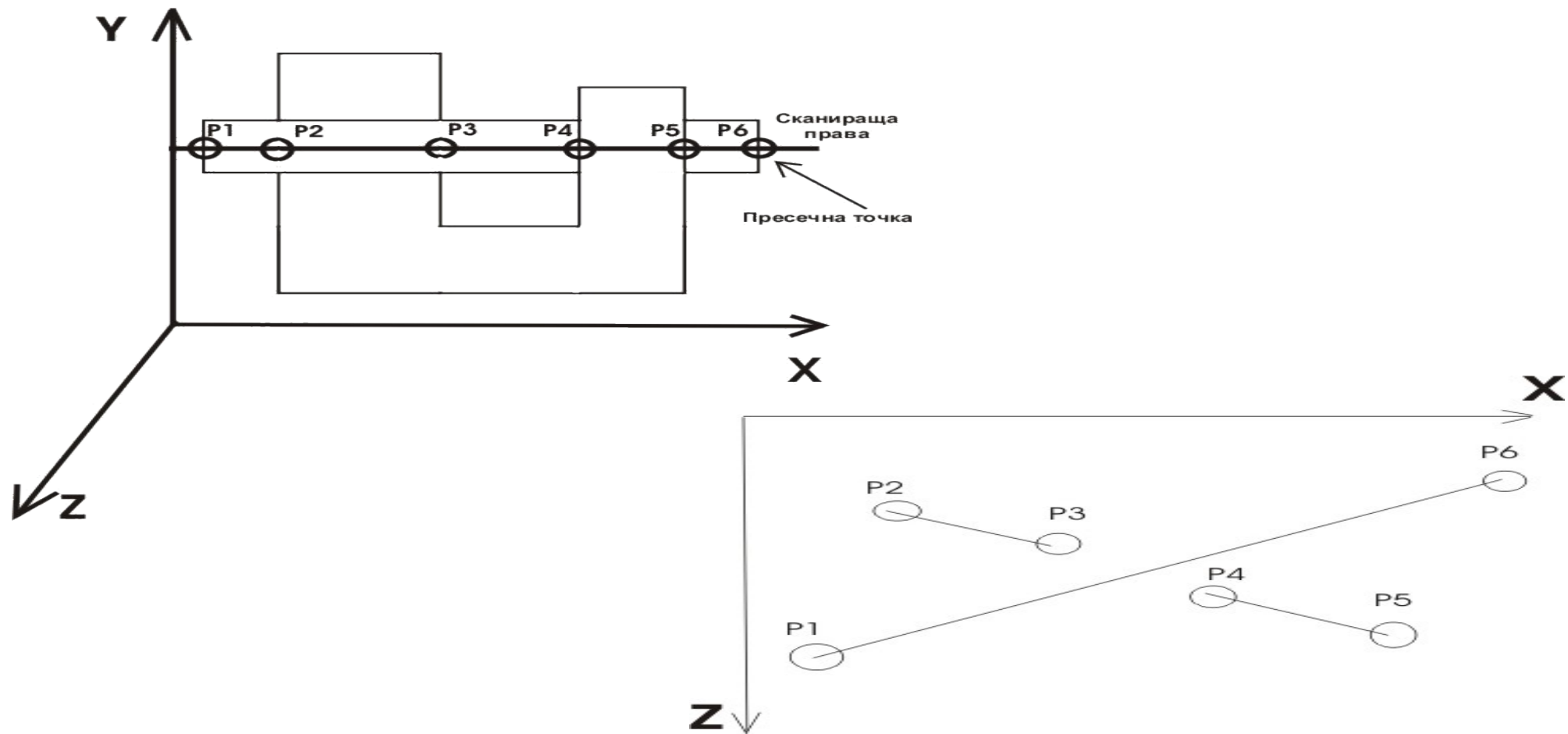
Циклично  
припокриване



Проникване

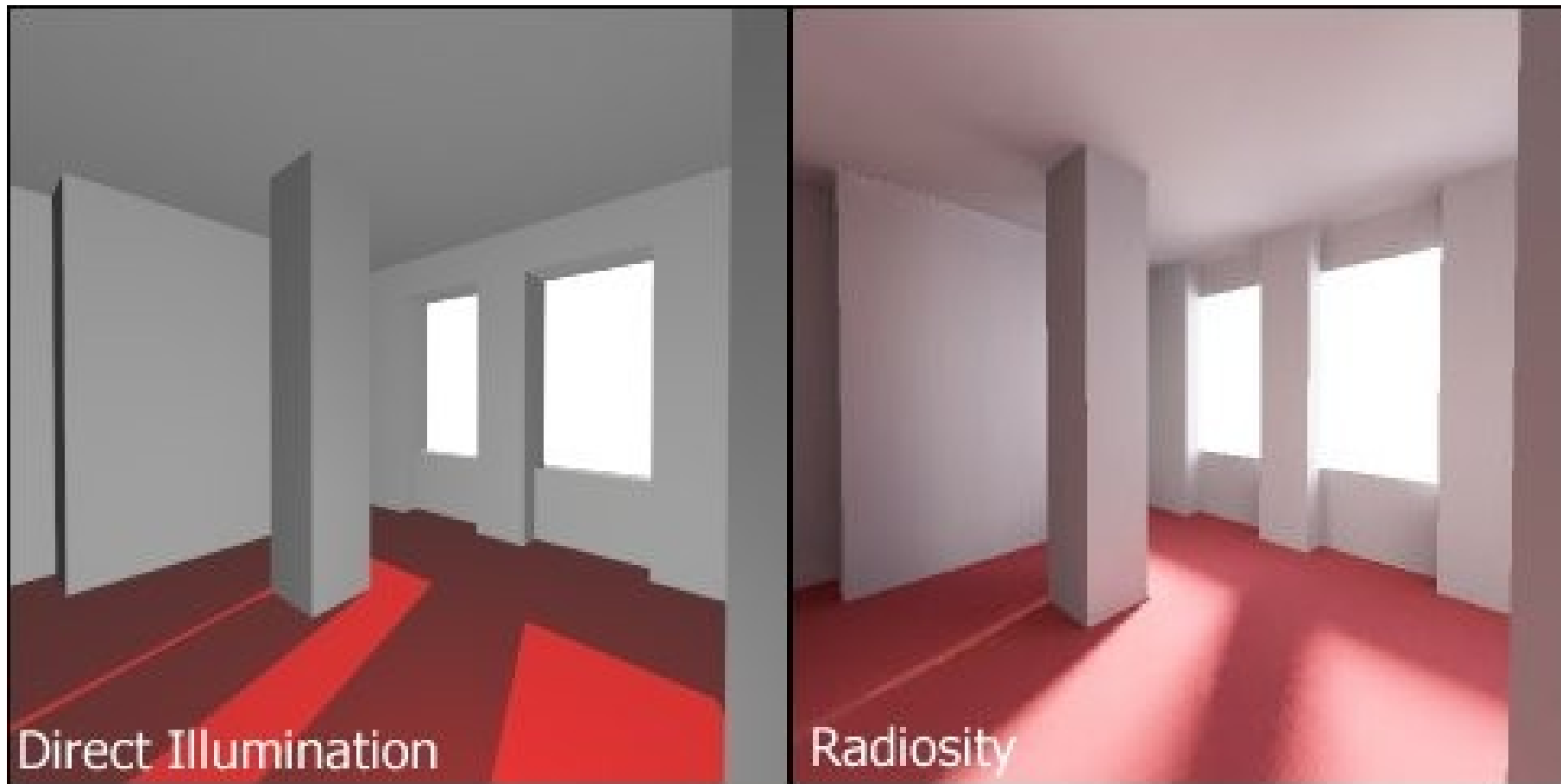
# *Интервален алгоритъм за Поредово сканиране*

# Поредово сканиране (интервален алгоритъм)

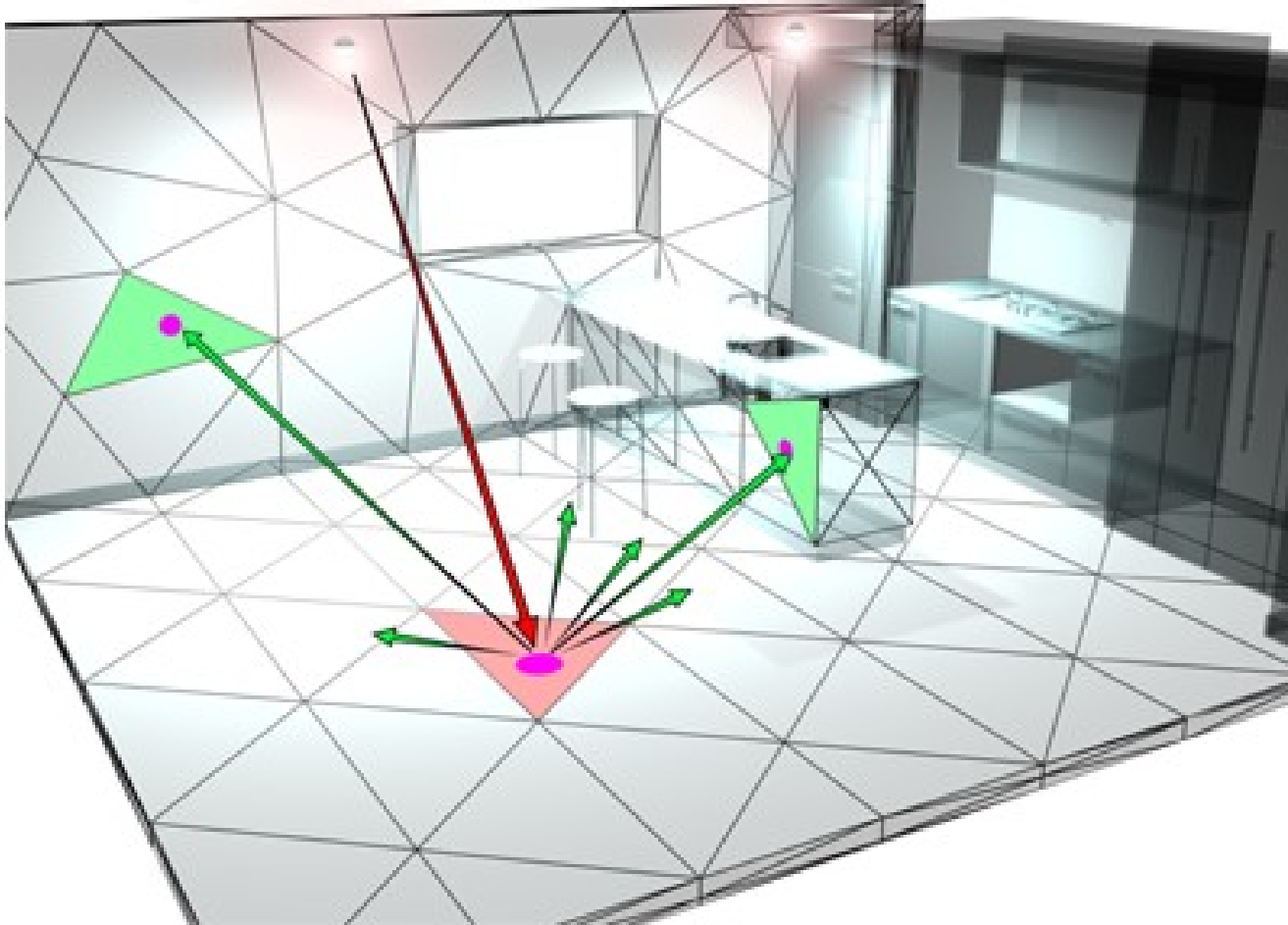


# *Radiosity*

# Локално срещу Глобално осветяване



# Radiosity



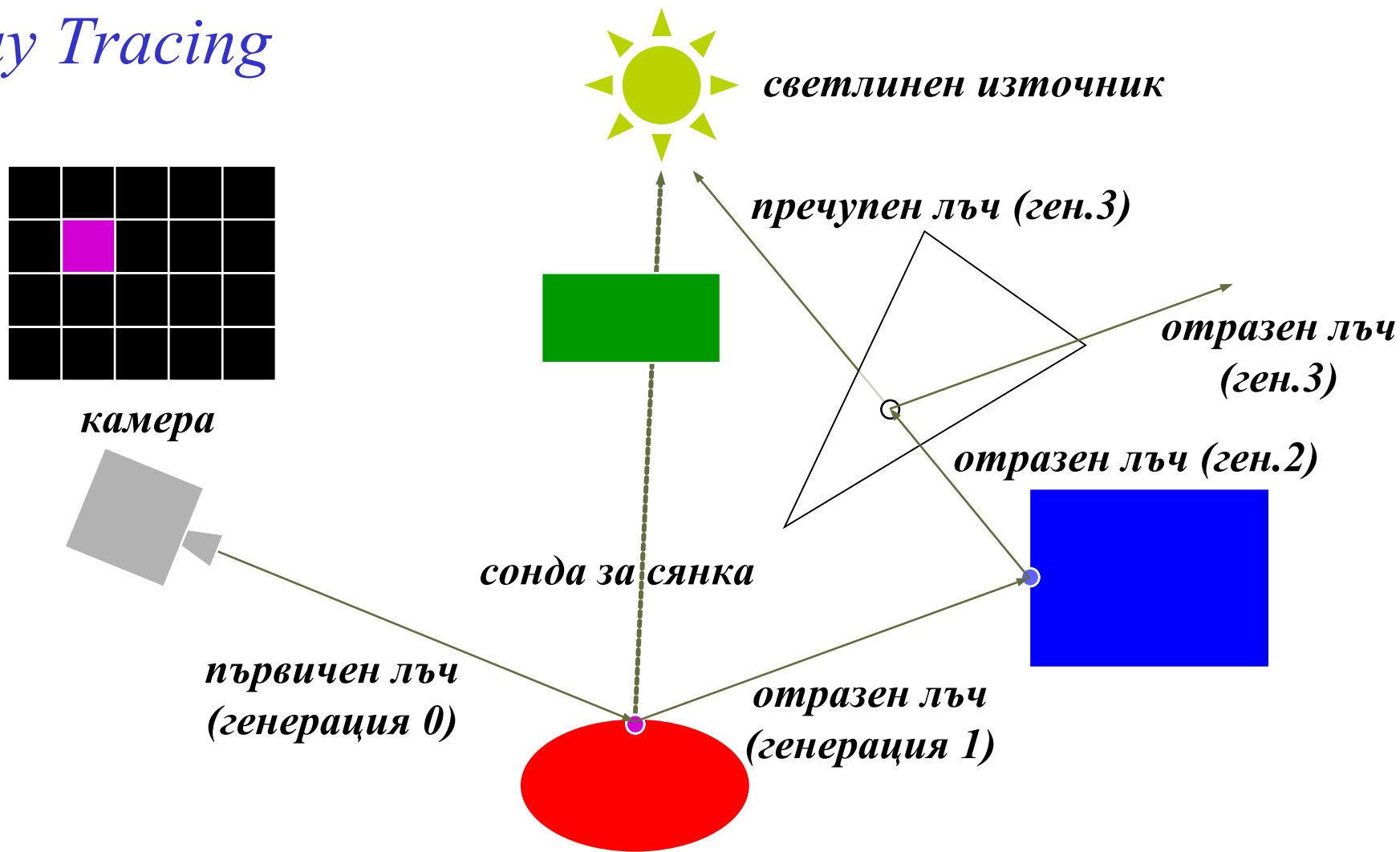
Геометрията на сцената се разбива на много малки повърхности, след което се пресмята пренасянето на светлината между всеки две от тях.

# Пример



# *Ray Tracing*

# Ray Tracing



## Пример 1



## Пример 2



# Въпроси?

*apenev@uni-plovdiv.bg*

