Удаление невидимых граней

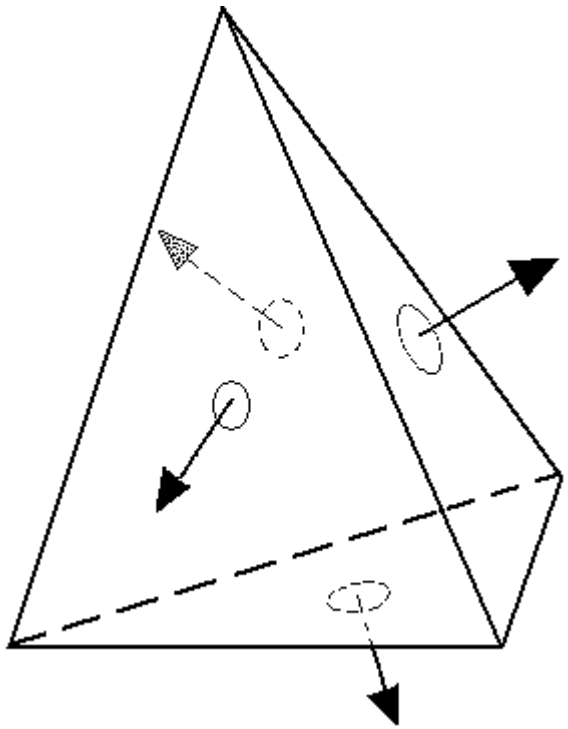
Компьютерная графика

Каркасные модели

- В 1960 году впервые были использованы формулы для преобразования 3d координат в 2d координаты в качестве основы для визуализации.
- Использовал их Уильям Феттер (William Fetter), который заодно придумал еще и термин "компьютерная графика". С помощью его программы можно было рисовать 3d объекты из линий (проволочные).

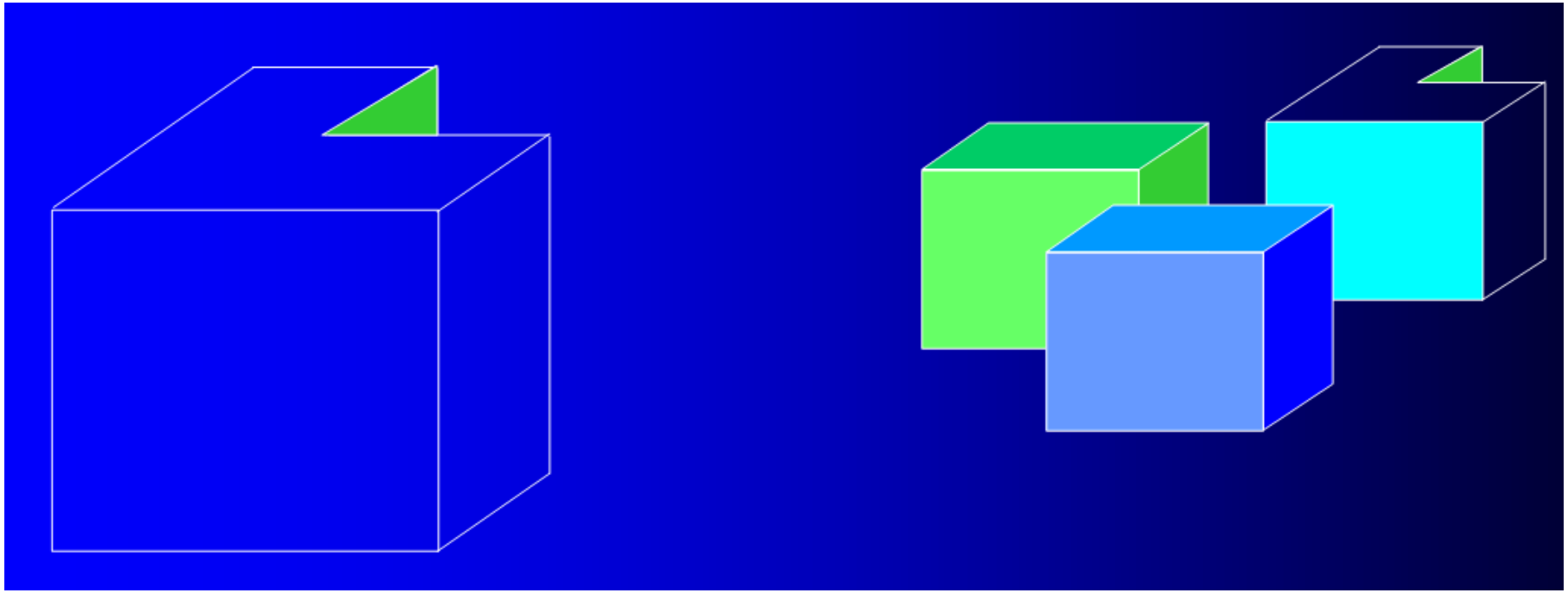
Отсечение нелицевых граней

front-faced



back-faced

Лицевые, но невидимые грани



Классификация алгоритмов

Основные идеи

Алгоритмы 3D

- Оперируем геом. примитивами, проверяем пересечение объектов друг с другом, в результате – список видимых объектов и их частей.
- Объекты из списка могут отображаться с любой точностью, но вычисления видимых частей – с высокой.
- Сложность алгоритмов зависит от количества объектов: в среднем для n объектов $\rightarrow O(n^2)$

Алгоритмы 2D

- Находим ближ. точки сцены к наблюдателю и для каждого пикселя изображения отображаем только такие точки.
- Точность на уровне разрешения устройства. Не требуется высокой точности вычислений.
- Зависимость сложности алгоритмов от числа пикселей и количества объектов: для с пикселей и n объектов $\rightarrow O(c\ n)$.

Особенности алгоритмов: вопрос

?D

- Объект анализируется 1 раз.
- Пиксель сцены может перерисовываться несколько раз.

?D

- Пиксель сцены рисуется 1 раз.
- Отношения между объектами для данного пикселя анализируются несколько раз.

Особенности алгоритмов: ответ

2D

- Объект анализируется 1 раз.
- Пиксель сцены может перерисовываться несколько раз.

3D

- Пиксель сцены рисуется 1 раз.
- Отношения между объектами для данного пикселя анализируются несколько раз.

Классификация алгоритмов

Алгоритмы 3D

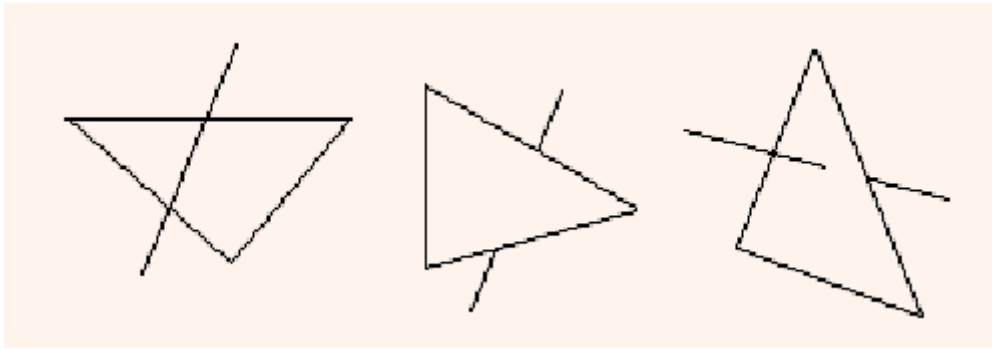
- Робертса (1963)
- Алгоритм Аппеля (1967)
- Вейлера-Айзертона (1977)
- BSP-деревьев (1969-91)

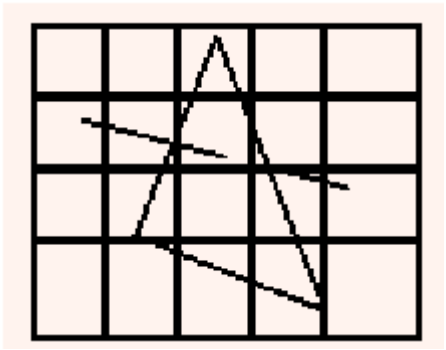
Алгоритмы 2D

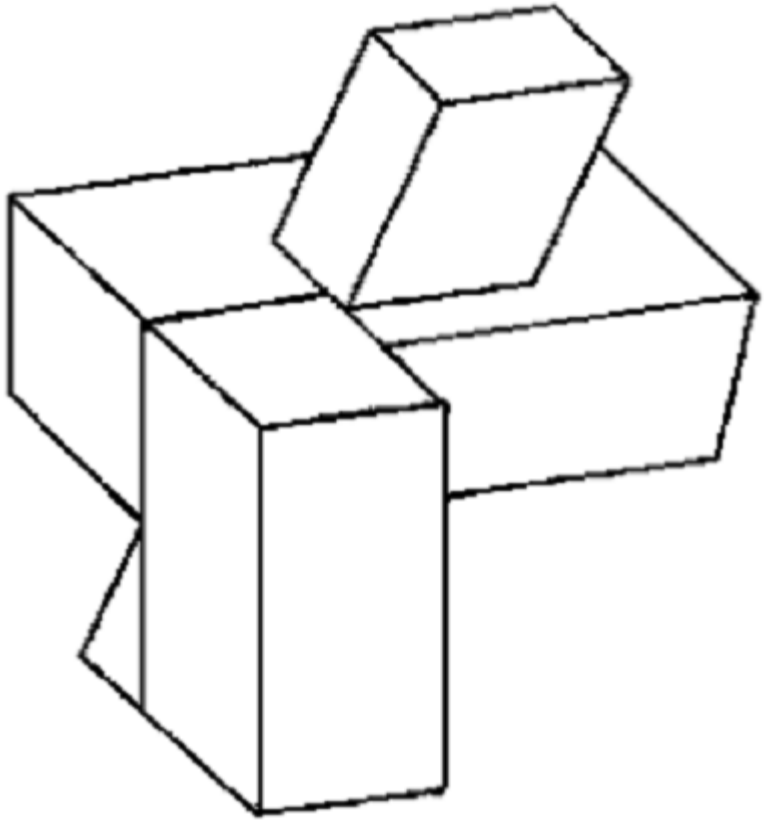
- Построчного сканирования (1967)
- Варнока (1969)
- Трассировки лучей (1968)
- Плавающего горизонта (1972)
- «Художника» + Ньюэла(1972)
- Z-буфера (1974)
- A-буфера (1984)

Алгоритм Робертса

- Лоуренс Робертс (Lawrence Roberts)
- 1963
- $0(n^2)$







Требования к сцене

- В алгоритме Робертса требуется, чтобы все изображаемые тела или объекты были выпуклыми.
- Невыпуклые тела должны быть разбиты на выпуклые части.
- В алгоритме выпуклое многогранное тело с плоскими гранями должно представляться набором пересекающихся плоскостей.

Классификация алгоритмов

Алгоритмы 3D

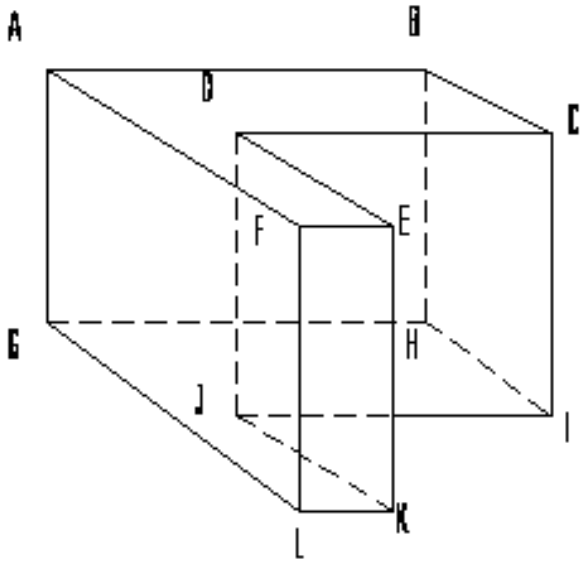
- Робертса (1963)
- Алгоритм Аппеля (1967)
- Вейлера-Айзертона (1977)
- BSP-деревьев (1969-91)

Алгоритмы 2D

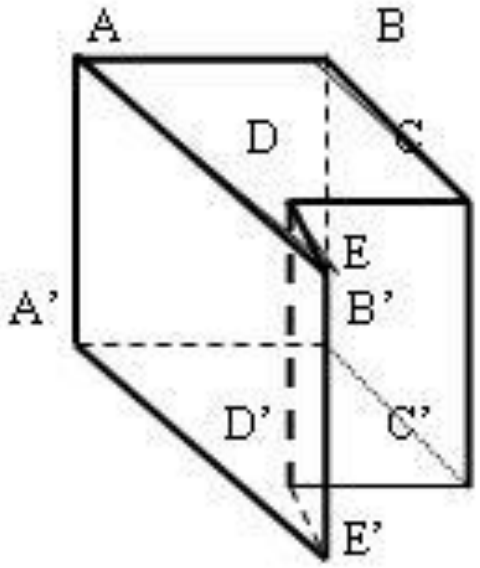
- Построчного сканирования (1967)
- Варнока (1969)
- Трассировки лучей (1968)
- Плавающего горизонта (1972)
- «Художника» + Ньюэла(1972)
- Z-буфера (1974)
- A-буфера (1984)

Алгоритм Аппеля

- Артур Аппель (Arthur Appel) 1967 (в то время работал в IBM), создал алгоритм удаления невидимых ребер (в том числе и частично скрытых).



Алгоритм Аппеля



Каждая точка видима только тогда, когда её **количественная невидимость** =0.

Контурная линия многогранника - есть ломаная из рёбер, для которых одна грань - лицевая, а другая - нелицевая. Например: ABCC'D'DEE'A'A

Количественная невидимость точек ребра изменяется, на 1 при прохождении ребра позади контурной линии.

Классификация алгоритмов

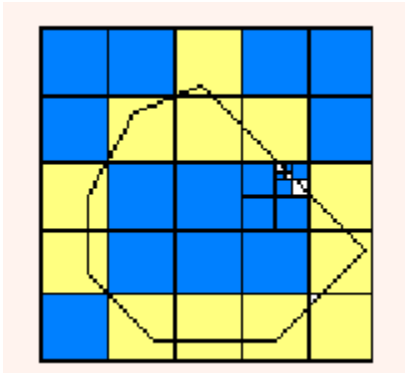
Алгоритмы 3D

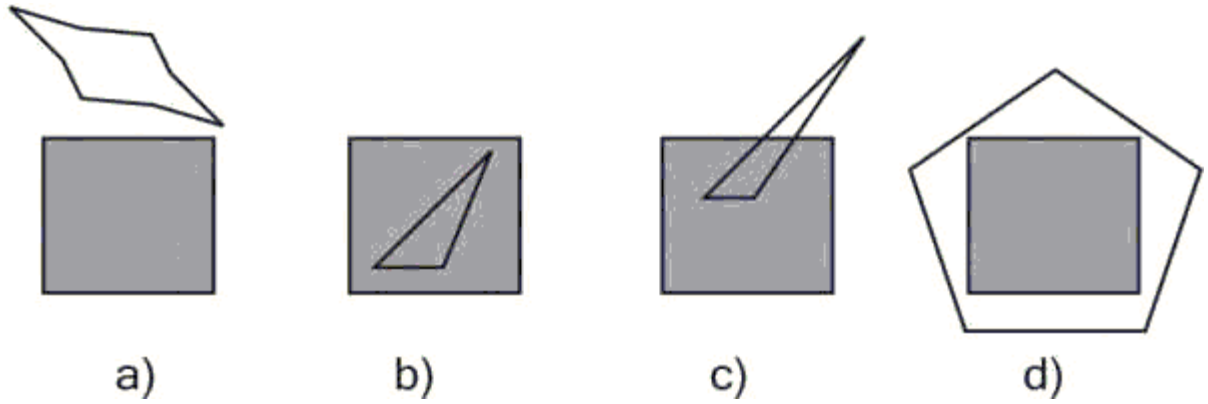
- Робертса (1963)
- Алгоритм Аппеля (1967)
- Вейлера-Айзертона (1977)
- BSP-деревьев (1969-91)

Алгоритмы 2D

- Построчного сканирования (1967)
- Варнока (1969)
- Трассировки лучей (1968)
- Плавающего горизонта (1972)
- «Художника» + Ньюэла(1972)
- Z-буфера (1974)
- A-буфера (1984)

Алгоритм Варнока (1969)





Классификация алгоритмов

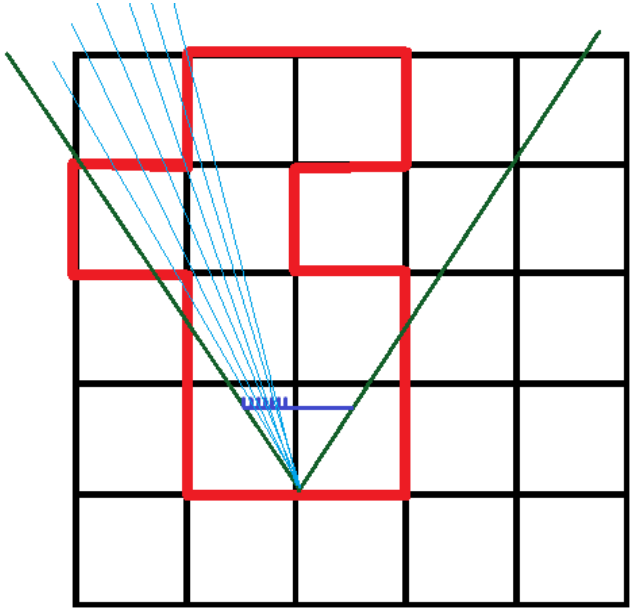
Алгоритмы 3D

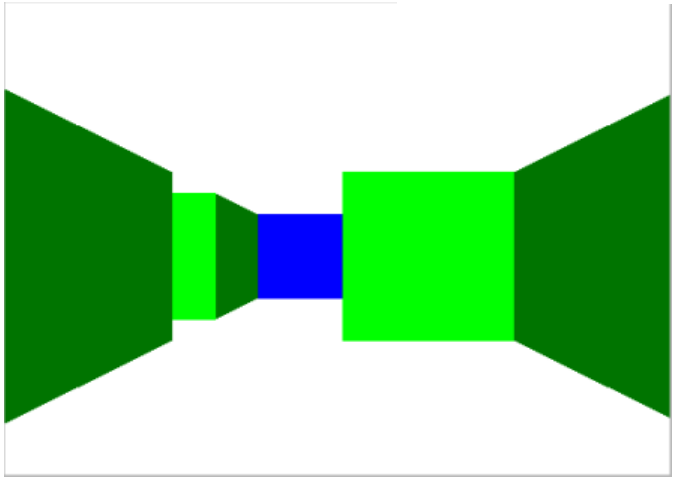
- Робертса (1963)
- Алгоритм Аппеля (1967)
- Вейлера-Айзертона (1977)
- BSP-деревьев (1969-91)

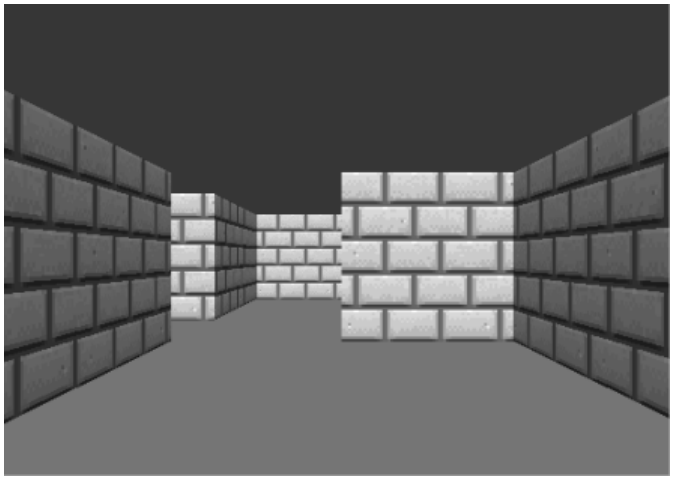
Алгоритмы 2D

- Варнока (1969)
- Построчного сканирования (1967)
- Трассировки лучей (1968)
- Плавающего горизонта (1972)
- «Художника» + Ньюэла(1972)
- Z-буфера (1974)
- A-буфера (1984)

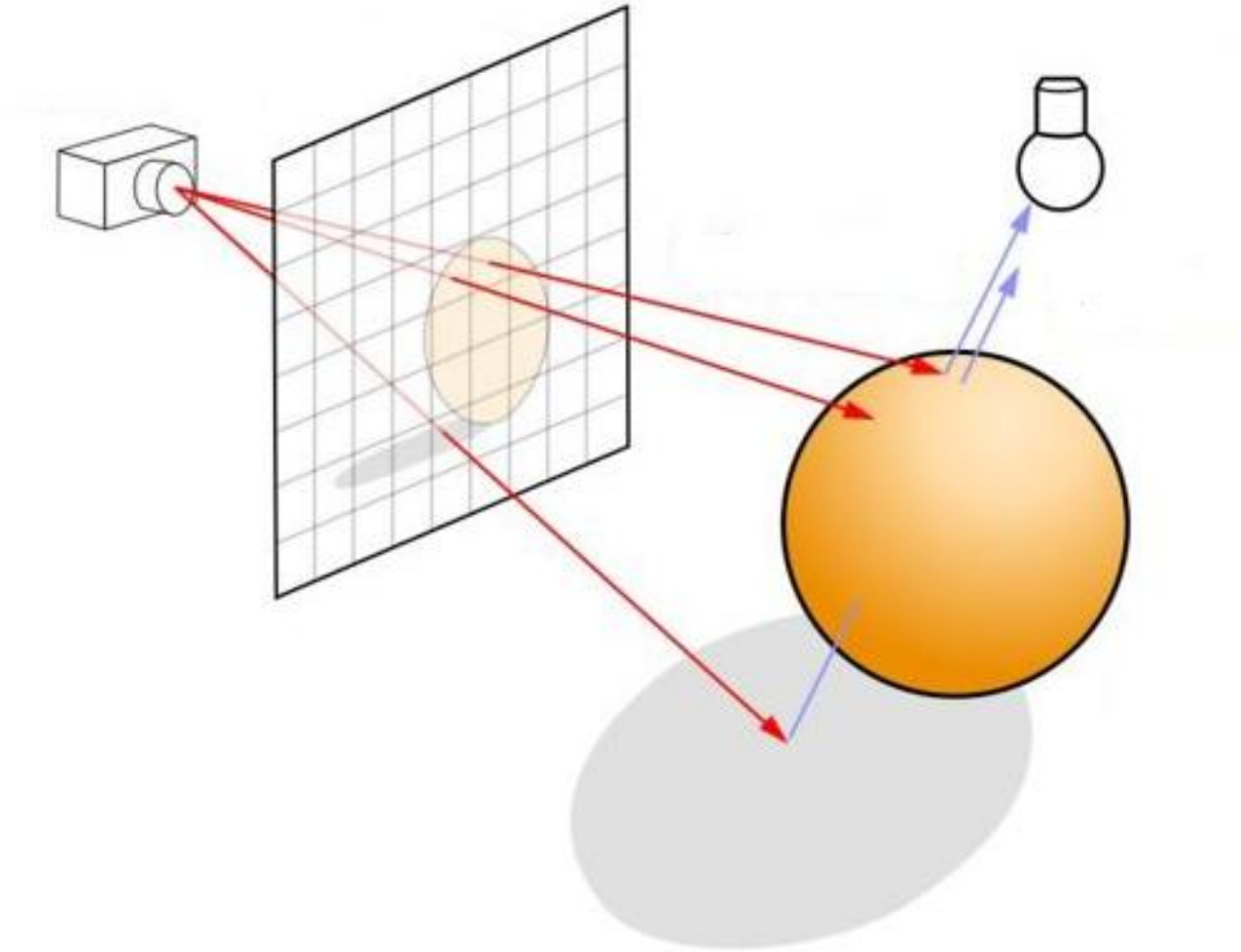
Трассировка лучей. Ray casting



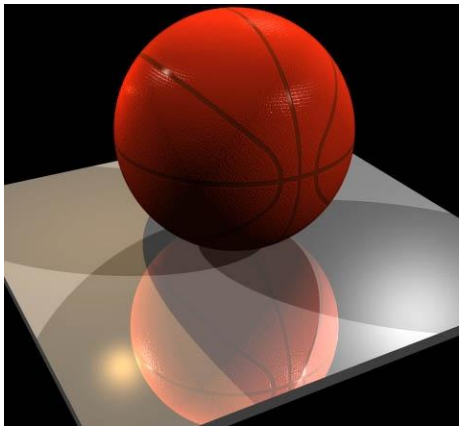
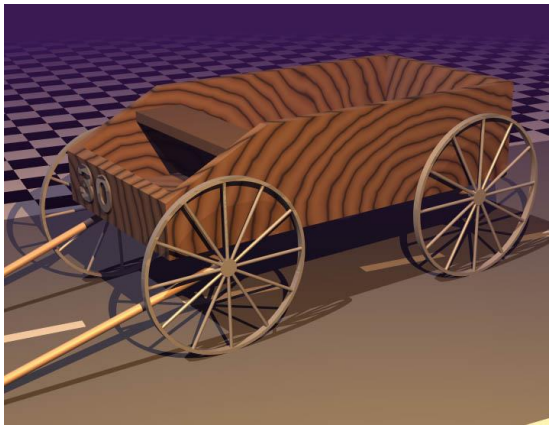
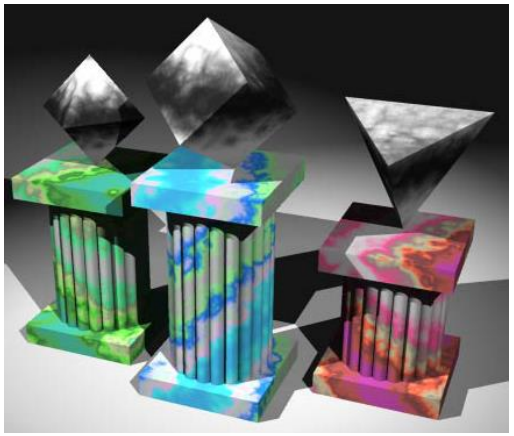
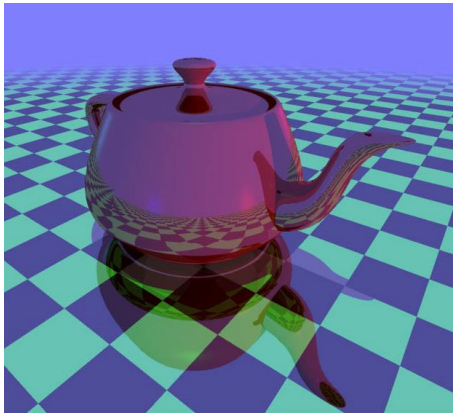
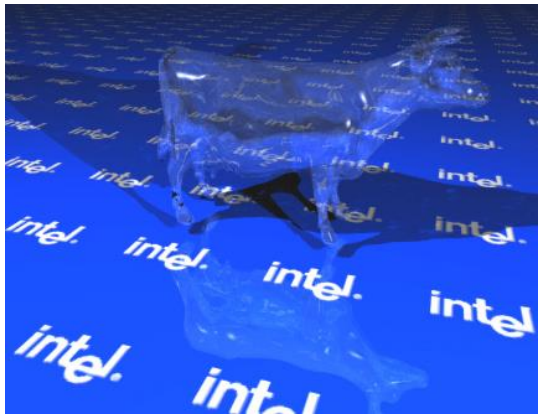
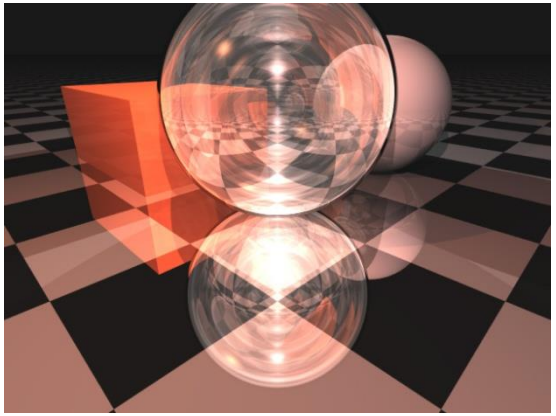
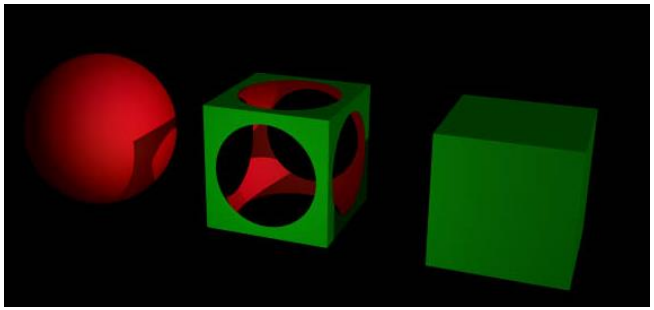
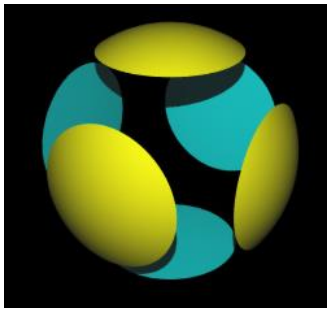




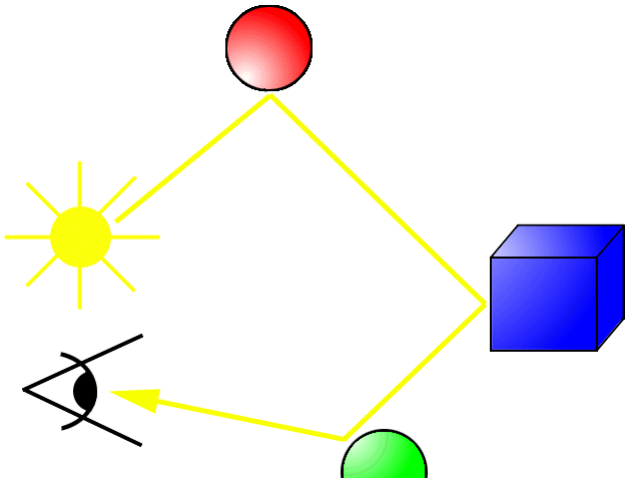
Трассировка лучей. Ray tracing

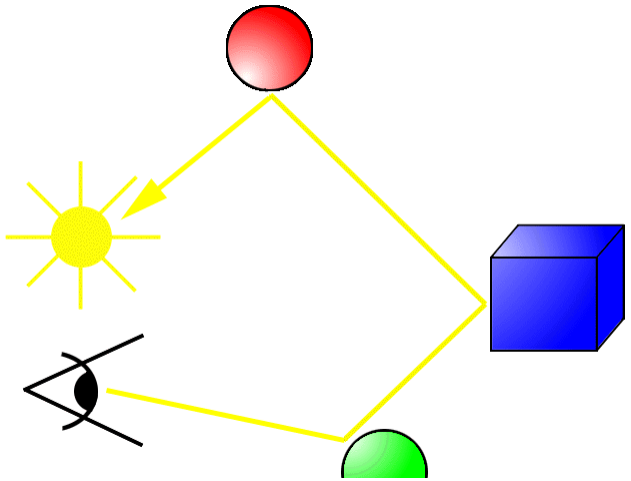


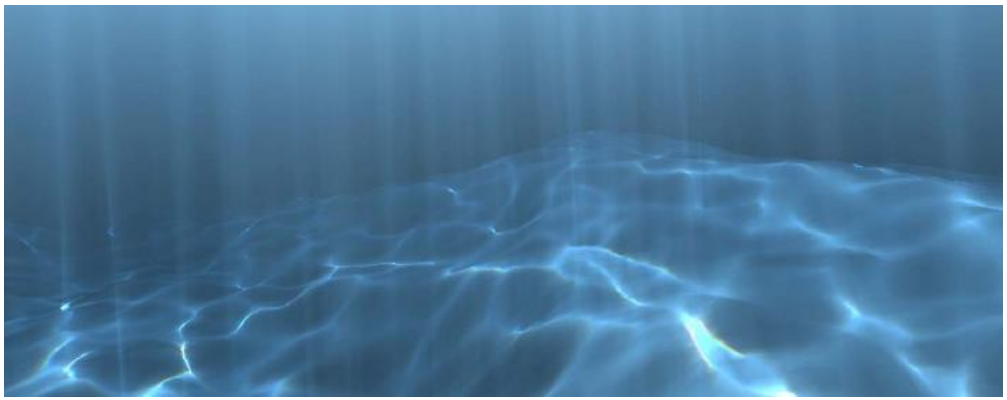
Ray tracing

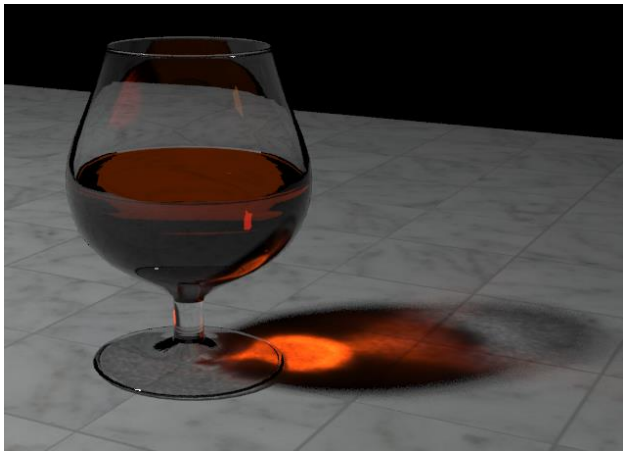









Forward vs Backward









Классификация алгоритмов

Алгоритмы 3D

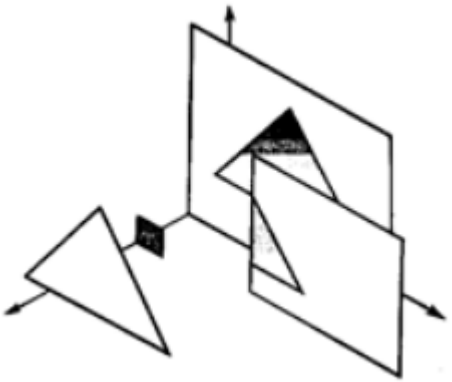
- Робертса (1963)
- Алгоритм Аппеля (1967)
- Вейлера-Айзертона (1977)
- BSP-деревьев (1969-91)

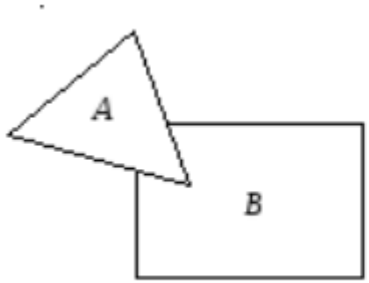
Алгоритмы 2D

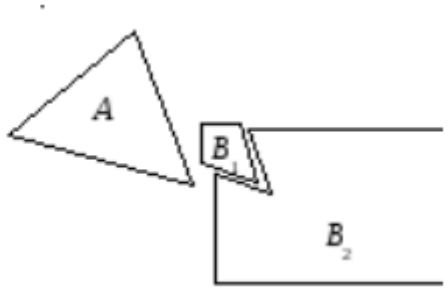
- Построчного сканирования (1967)
- Варнока (1969)
- Трассировки лучей (1968)
- Плавающего горизонта (1972)
- «Художника» + Ньюэла(1972)
- Z-буфера (1974)
- A-буфера (1984)

Алгоритм Вейлера-Азертона

- Предварительная сортировка по глубине.
- Отсечение по границе ближайшего к точке наблюдения многоугольника.
- Удаление многоугольников, экранируемых более близкими к точке наблюдения многоугольниками.
- Если требуется, то рекурсивное разбиение и новая сортировка.







Классификация алгоритмов

Алгоритмы 3D

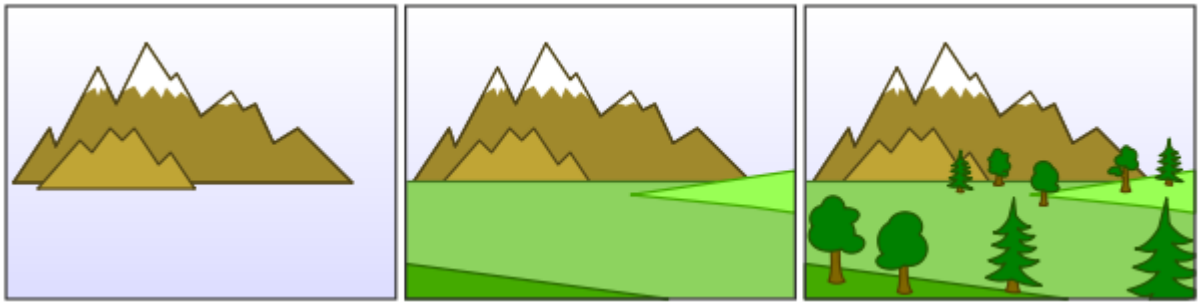
- Робертса (1963)
- Алгоритм Аппеля (1967)
- Вейлера-Айзертона (1977)
- BSP-деревьев (1969-91)

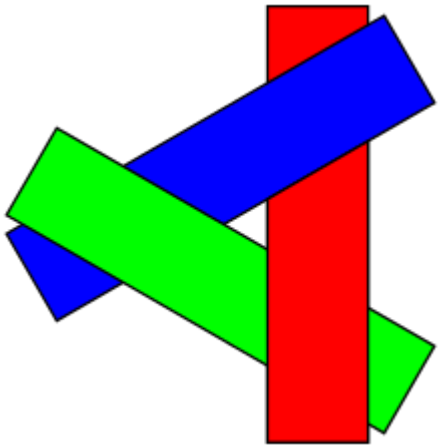
Алгоритмы 2D

- Построчного сканирования (1967)
- Варнока (1969)
- Трассировки лучей (1968)
- Плавающего горизонта (1972)
- «Художника» + Ньюэла(1972)
- Z-буфера (1974)
- A-буфера (1984)

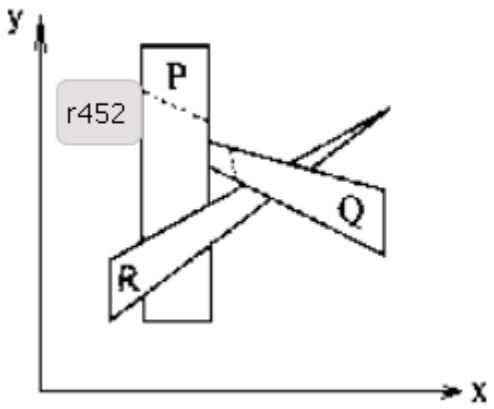
Алгоритм художника + Ньюэла (1972)

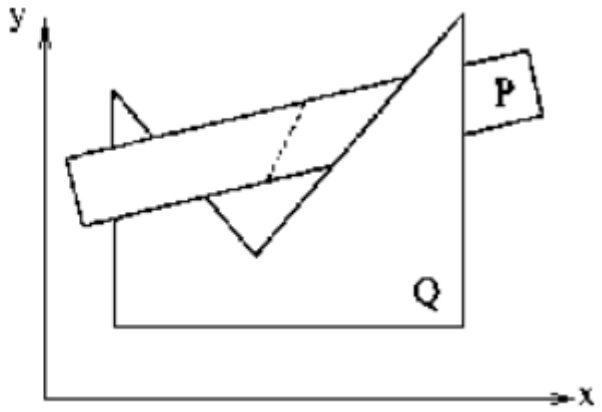
Алгоритм Ньюэла-Нюэла-Санча





Алгоритм Ньюэла-Нюэла-Санча

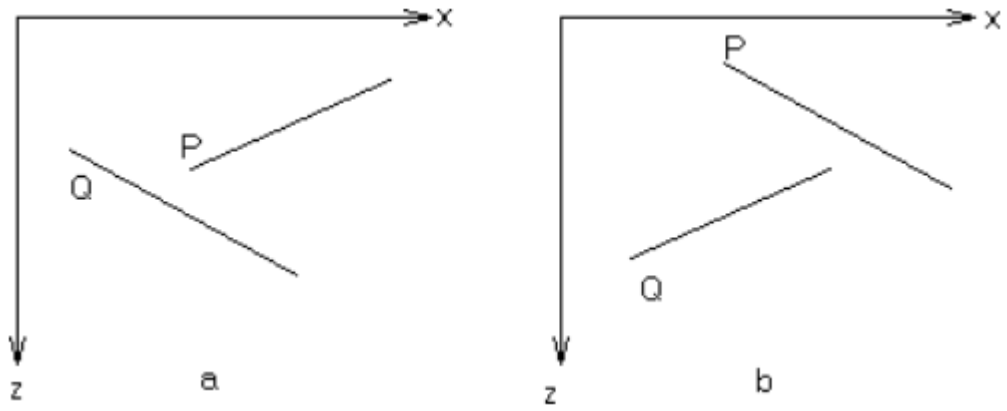




Тесты для перекрывающихся многоугольников

- Верно ли, что прямоугольные объемлющие оболочки P и Q не перекрываются по x ?
- Верно ли, что прямоугольные оболочки P и Q не перекрываются по y ?
- Верно ли, что P целиком лежит по ту сторону плоскости, несущей Q, которая расположена дальше от точки наблюдения?
- Верно ли, что Q целиком лежит по ту сторону плоскости, несущей P, которая ближе к точке наблюдения?
- Верно ли, что проекции P и Q не перекрываются?

Каждый из этих тестов применяется к каждому элементу {Q}. Если ни один из них не дает положительного ответа, то P может закрывать Q.



Классификация алгоритмов

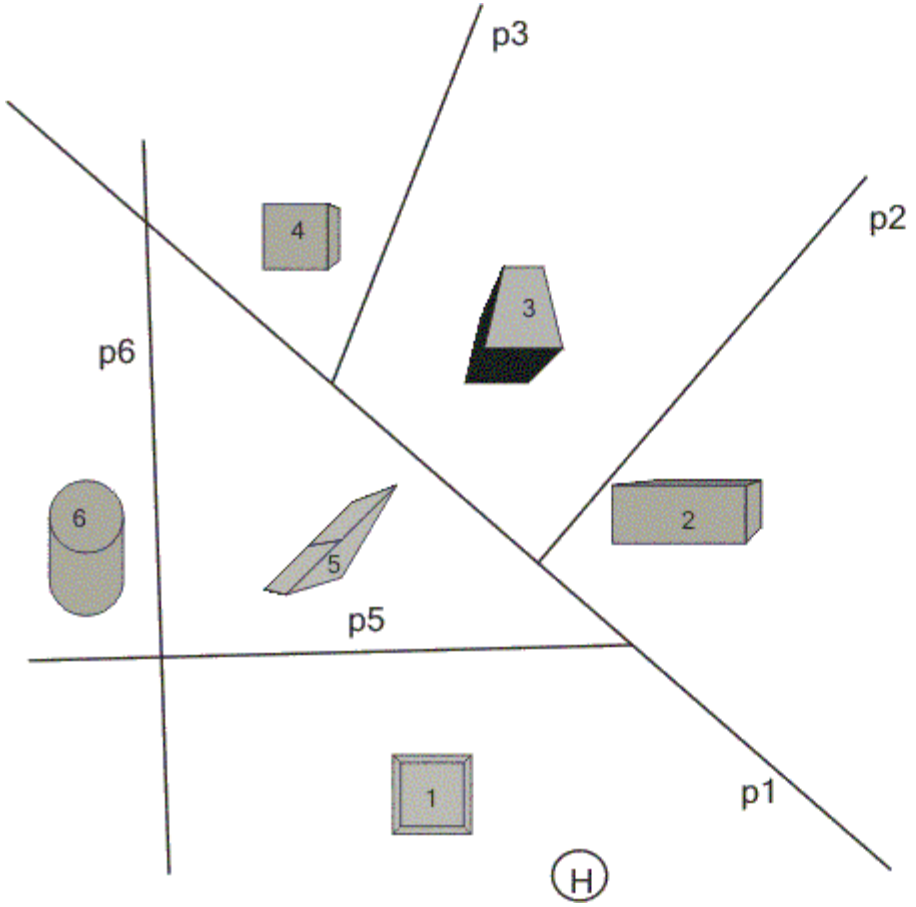
Алгоритмы 3D

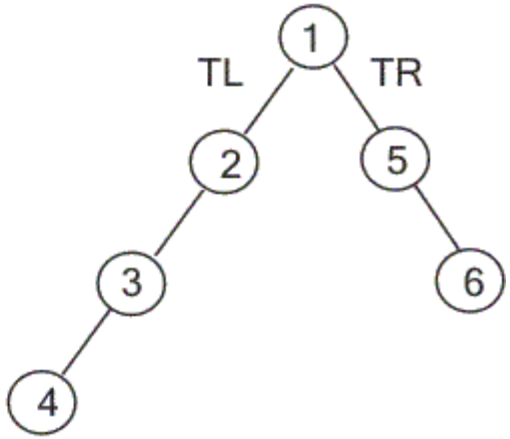
- Робертса (1963)
- Алгоритм Аппеля (1967)
- Вейлера-Айзертона (1977)
- BSP-деревьев (1969-91)

Алгоритмы 2D

- Построчного сканирования (1967)
- Варнока (1969)
- Трассировки лучей (1968)
- Плавающего горизонта (1972)
- «Художника» + Ньюэла(1972)
- Z-буфера (1974)
- A-буфера (1984)

Метод двоичного разбиения пространства





Метод двоичного разбиения пространства

Рисуем дерево (T) :

Если наблюдатель находится в положительной полуплоскости, то:

- Если правое поддерево TR не пусто, рисуем дерево (TR).
- Рисуем корневую грань.
- Если левое поддерево TL не пусто, рисуем дерево (TL).

Иначе

- Если левое поддерево TL не пусто, рисуем дерево (TL).
- Рисуем корневую грань.
- Если правое поддерево TR не пусто, рисуем дерево (TR).

Классификация алгоритмов

Алгоритмы 3D

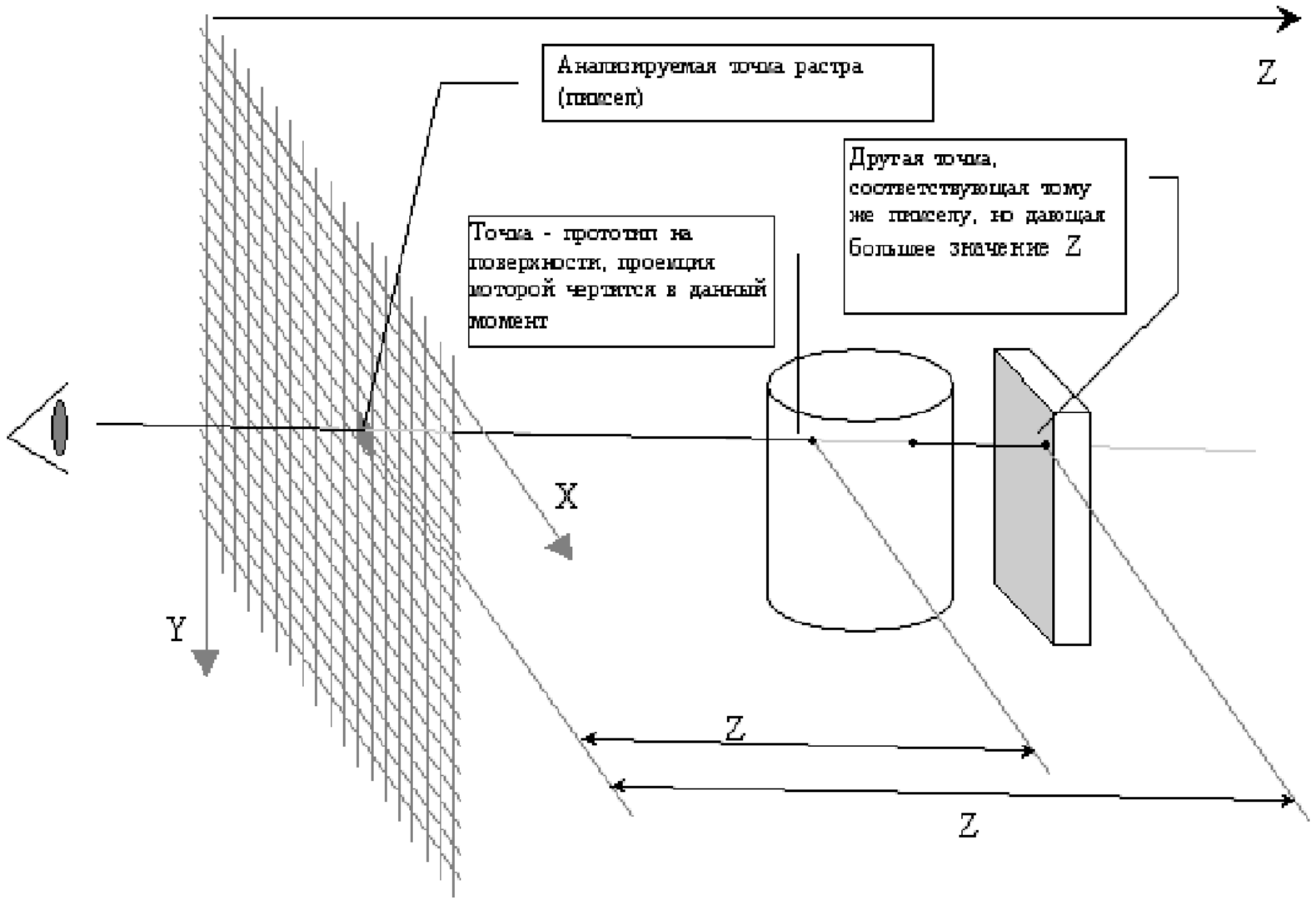
- Робертса (1963)
- Алгоритм Аппеля (1967)
- Вейлера-Айзертона (1977)
- BSP-деревьев (1969-91)

Алгоритмы 2D

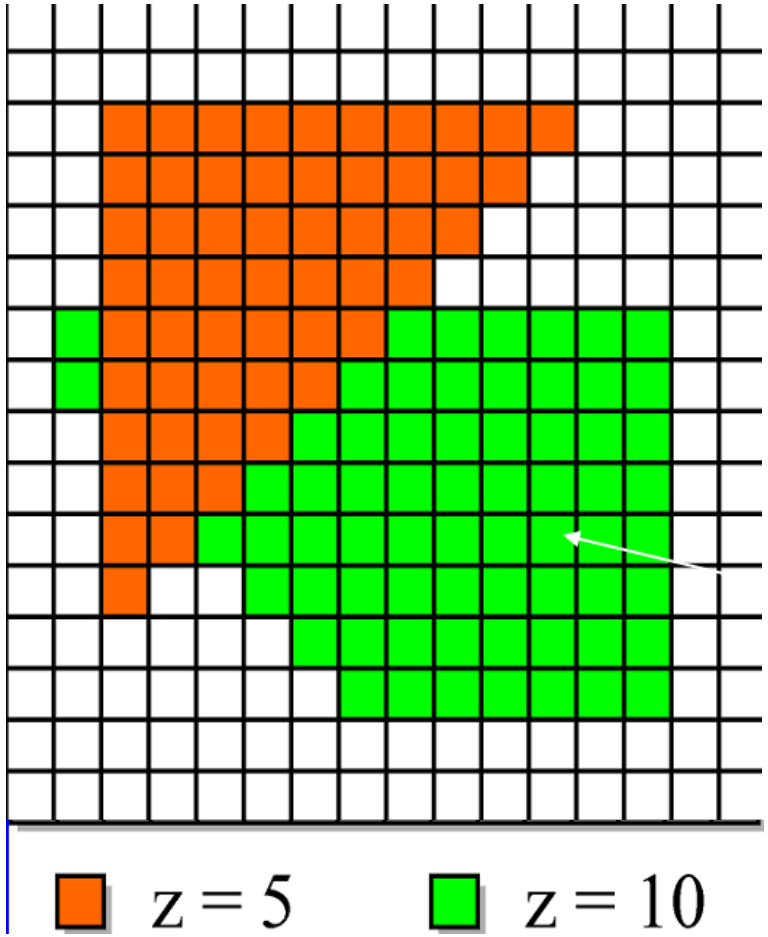
- Построчного сканирования (1967)
- Варнока (1969)
- Трассировки лучей (1968)
- Плавающего горизонта (1972)
- «Художника» + Ньюэла(1972)
- Z-буфера (1974)
- A-буфера (1984)

Метод z-буфера

- Эд Кэтмул 1974



Метод z-буфера

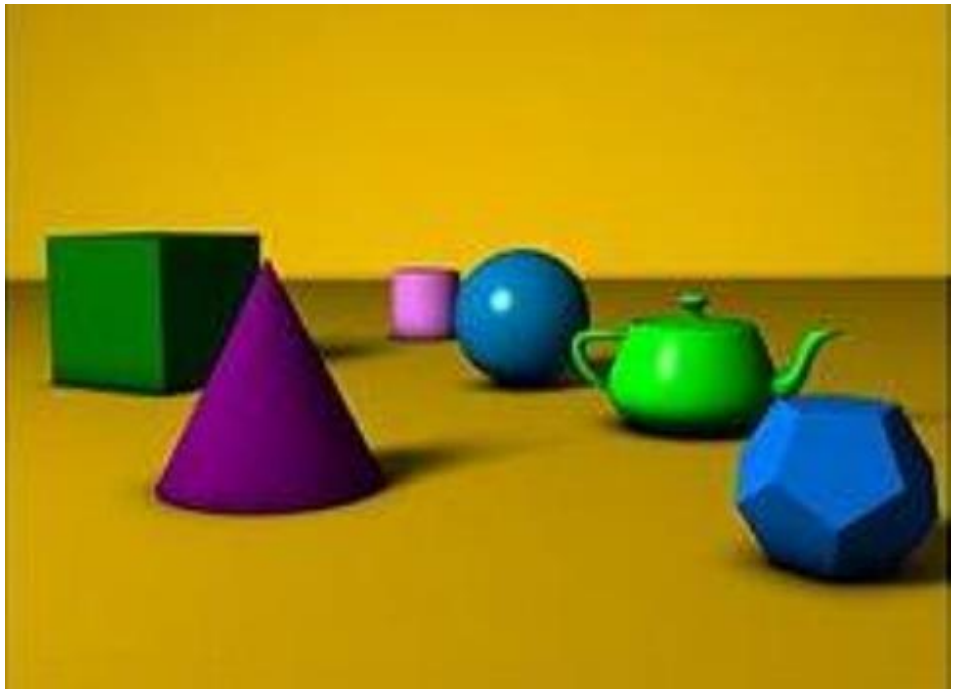


Изначально, буфер инициализируется значением $z = z_{max}$

```
for each pixel in polygon:
    if (pixel z < buffer z) then
        buffer z = pixel z
        fill pixel in raster
```

Метод z-буфера

- Буфер кадра размером 512x512x24 бит в комбинации с z-буфером размером 512x512x20 бит требует почти 1.5 мегабайт памяти.



Формальное описание алгоритма z-буфера

- заполнить буфер кадра фоновым значением интенсивности или цвета
- заполнить z-буфер минимальным значением z
- преобразовать каждый многоугольник в растровую форму в произвольном порядке;
- для каждого пиксела(x, y) в многоугольнике вычислить его глубину z(x, y)
- сравнить глубину z(x, y) со значением Z буфер(x, y), хранящимся в z -буфере в этой же позиции
- если z(x, y) > Z буфер (x, y), то записать атрибут этого многоугольника (интенсивность, цвет и т. п.) в буфер кадра и заменить Z буфер(x, y) на z (x, y)
- в противном случае никаких действий не производить
- в качестве предварительного шага там, где это целесообразно, применяется удаление нелицевых граней

Почему z-буфер так популярен?

- Прост в реализации на «железе»
- Память для z-буфера уже не дорогая
- Разнородность использующихся примитивов – не ограничиваемся только полигонами
- Нелимитированная возможная сложность сцены
- Отсутствие необходимости вычисления пересечений объектов
- сцены друг с другом

Недостатки z-буфера

- Дополнительная память и дополнительные требования к каналу передачи данных (полосе пропускания)
- Напрасная трата времени на отрисовку невидимых объектов
- Недостаток точности Z- координат

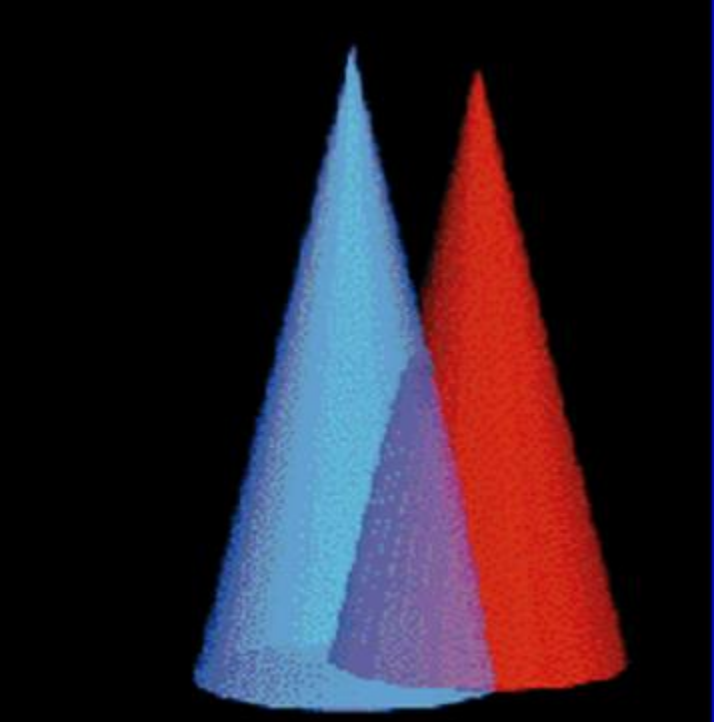
Классификация алгоритмов

Алгоритмы 3D

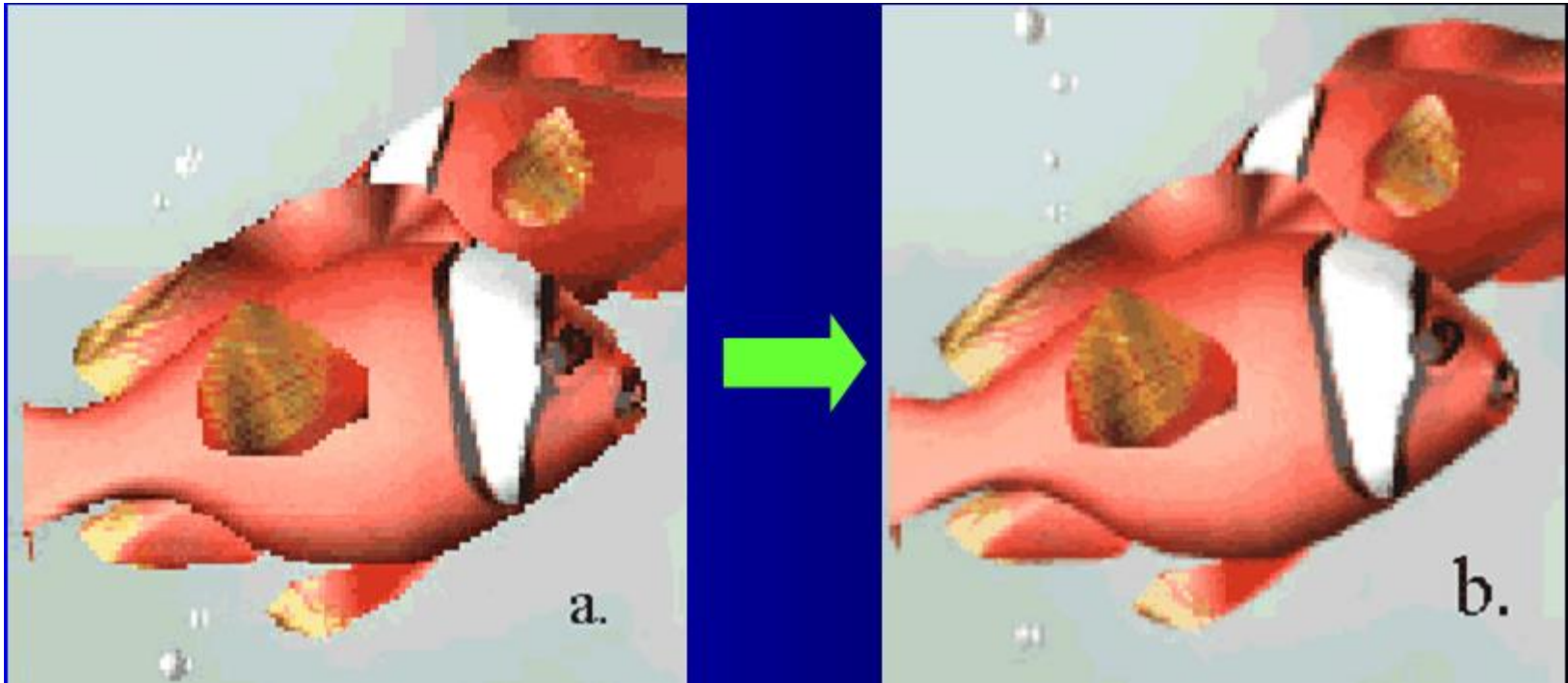
- Робертса (1963)
- Алгоритм Аппеля (1967)
- Вейлера-Айзертона (1977)
- BSP-деревьев (1969-91)

Алгоритмы 2D

- Построчного сканирования (1967)
- Варнока (1969)
- Трассировки лучей (1968)
- Плавающего горизонта (1972)
- «Художника» + Ньюэла(1972)
- Z-буфера (1974)
- A-буфера (1984)



Алгоритм A-буфера



Алгоритм A-буфера

- Область буфера в алгоритме называется буфером накопления, так как в ней в дополнение к значениям глубин хранятся различные данные о поверхности

Алгоритм A-буфера

Данные по поверхности включают следующие поля:

- значения интенсивностей RGB-компонентов
- параметр непрозрачности (процент прозрачности)
- глубина
- процент охвата площади
- идентификатор поверхности
- другие параметры, требуемые для визуализации поверхности.

Классификация алгоритмов

Алгоритмы 3D

- Робертса (1963)
- Алгоритм Аппеля (1967)
- Вейлера-Айзертона (1977)
- BSP-деревьев (1969-91)

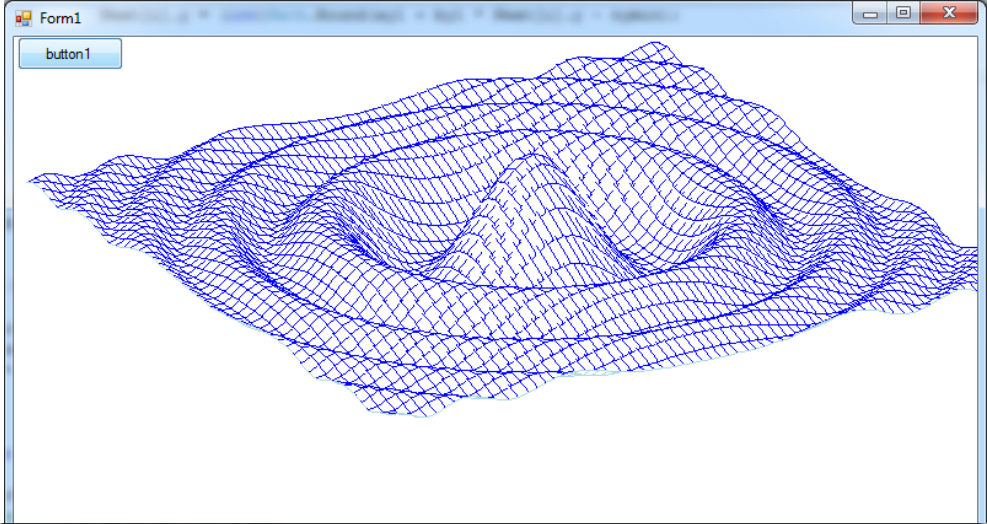
Алгоритмы 2D

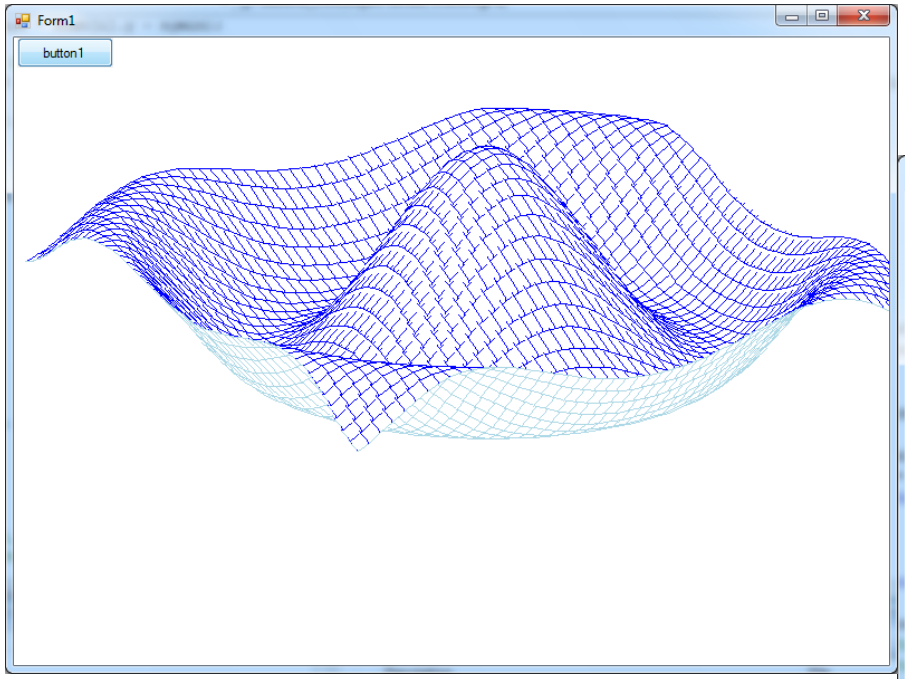
- Построчного сканирования (1967)
- Варнока (1969)
- Трассировки лучей (1968)
- Плавающего горизонта (1972)
- «Художника» + Ньюэла(1972)
- Z-буфера (1974)
- A-буфера (1984)

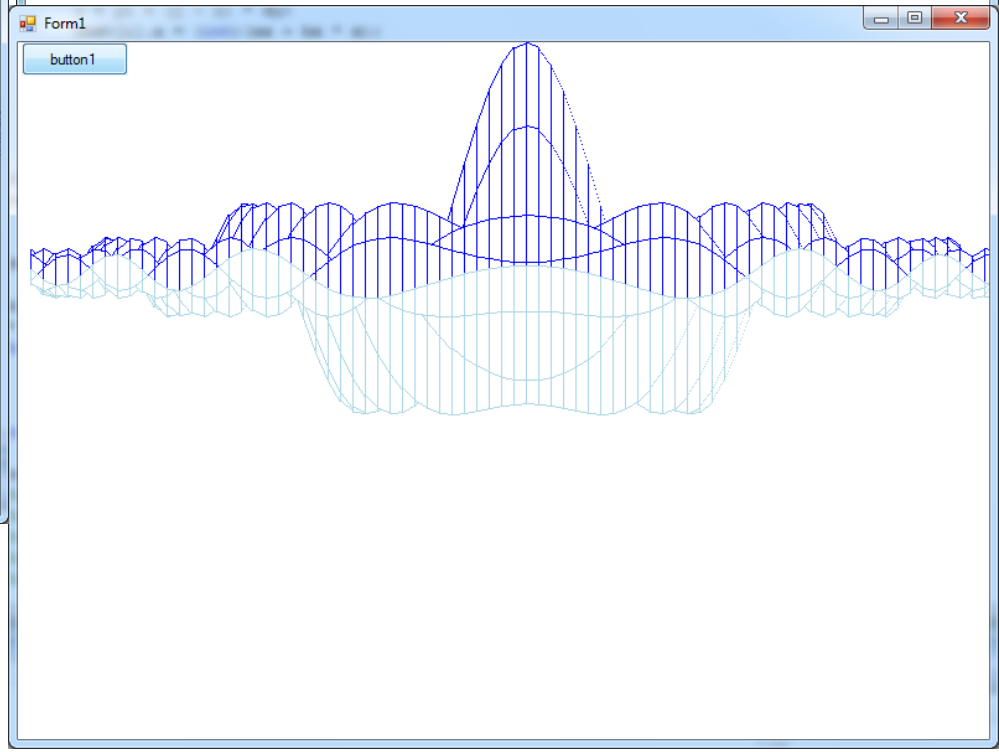
Графики функций f(x,y)

Алгоритм плавающего горизонта

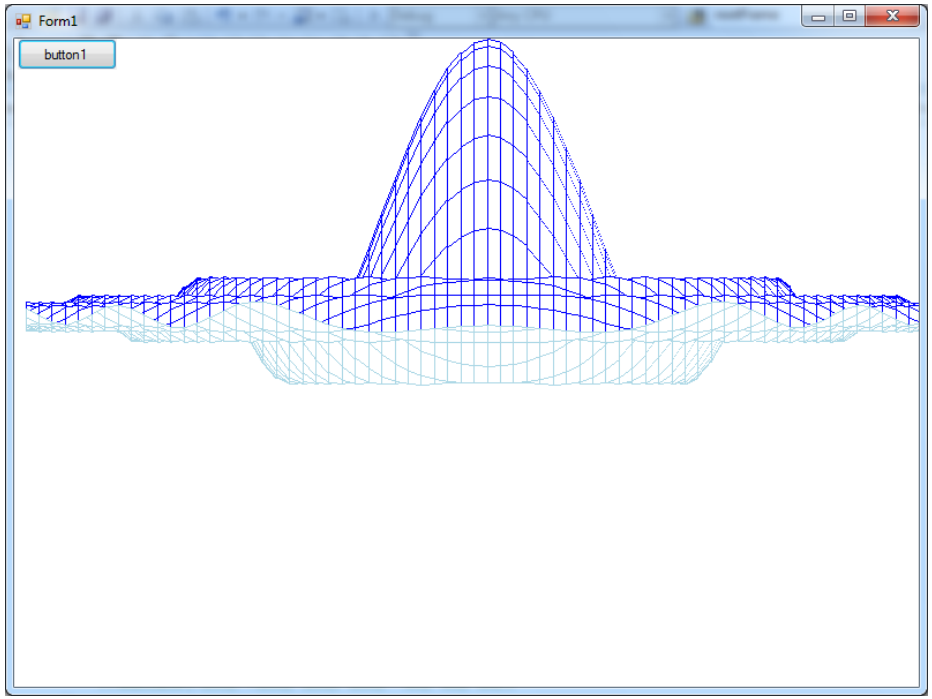
```
double r = x * x + y * y + 1;
return 5 * (Math.Cos(r) / r + 0.1);
```

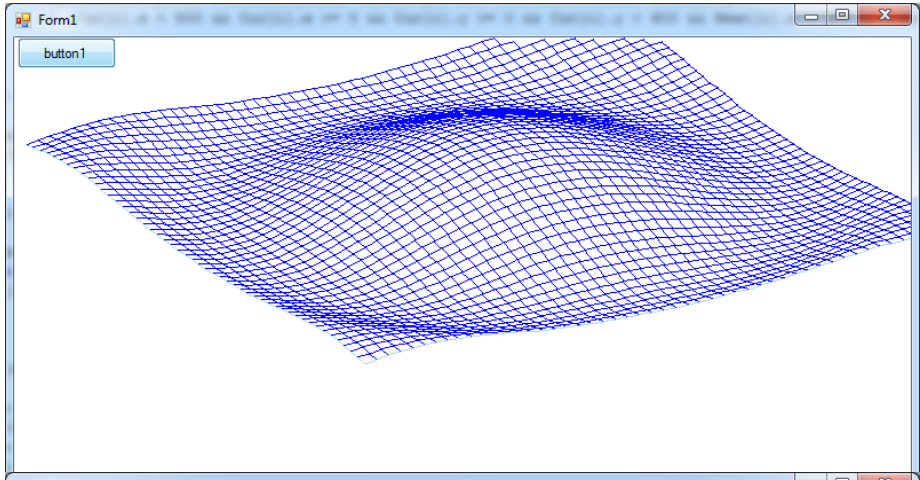


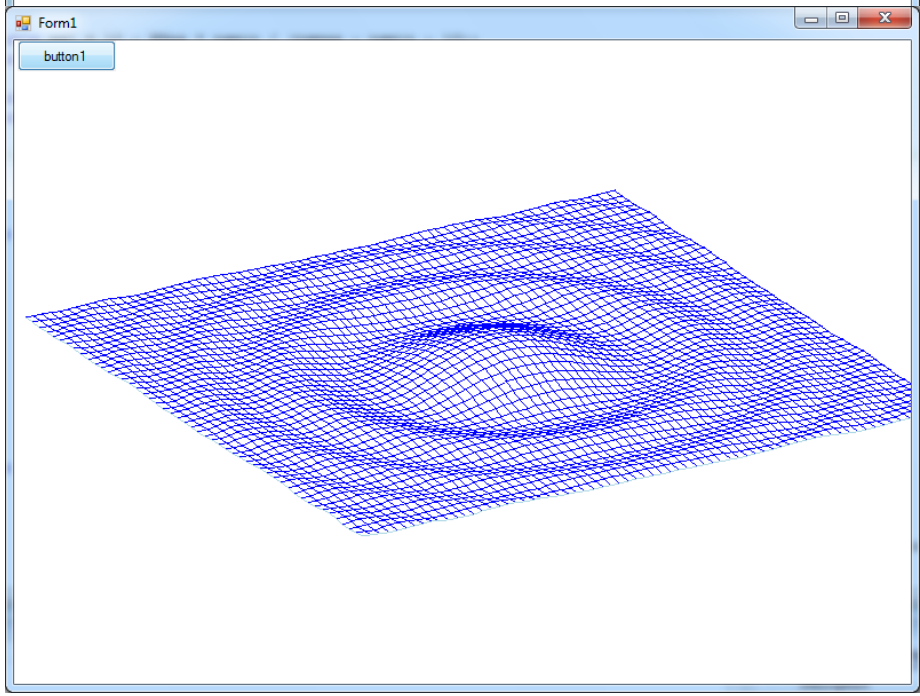




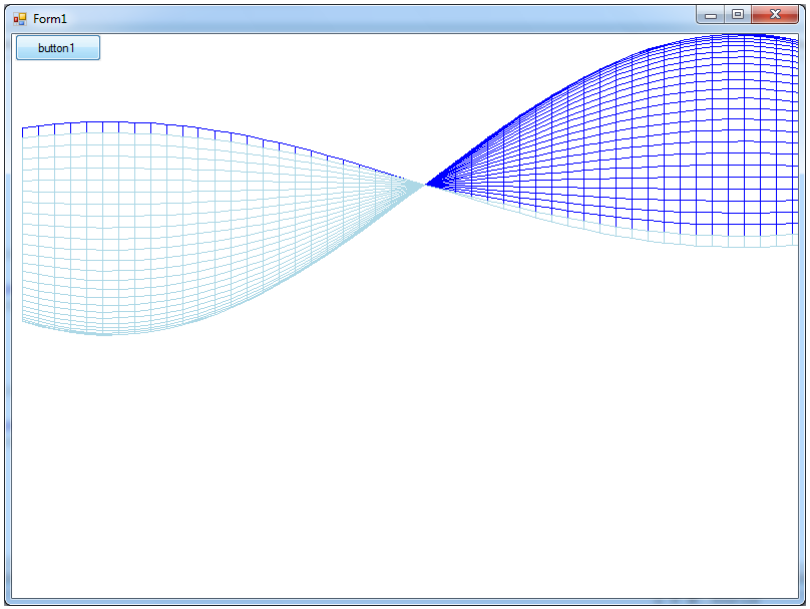
```
double r = x * x + y * y;
return Math.Cos(r) /(r+1);
```

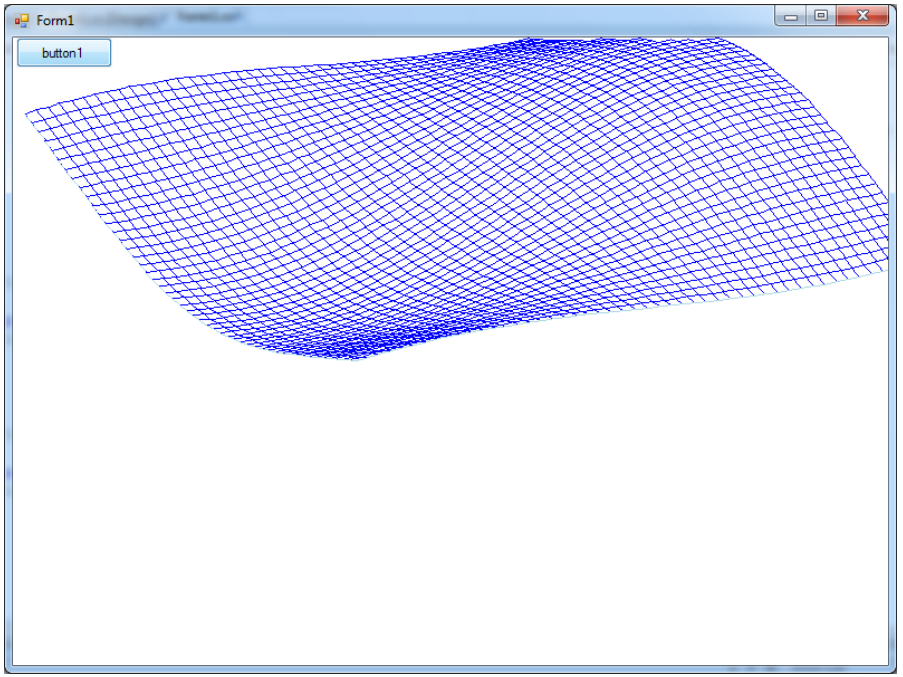




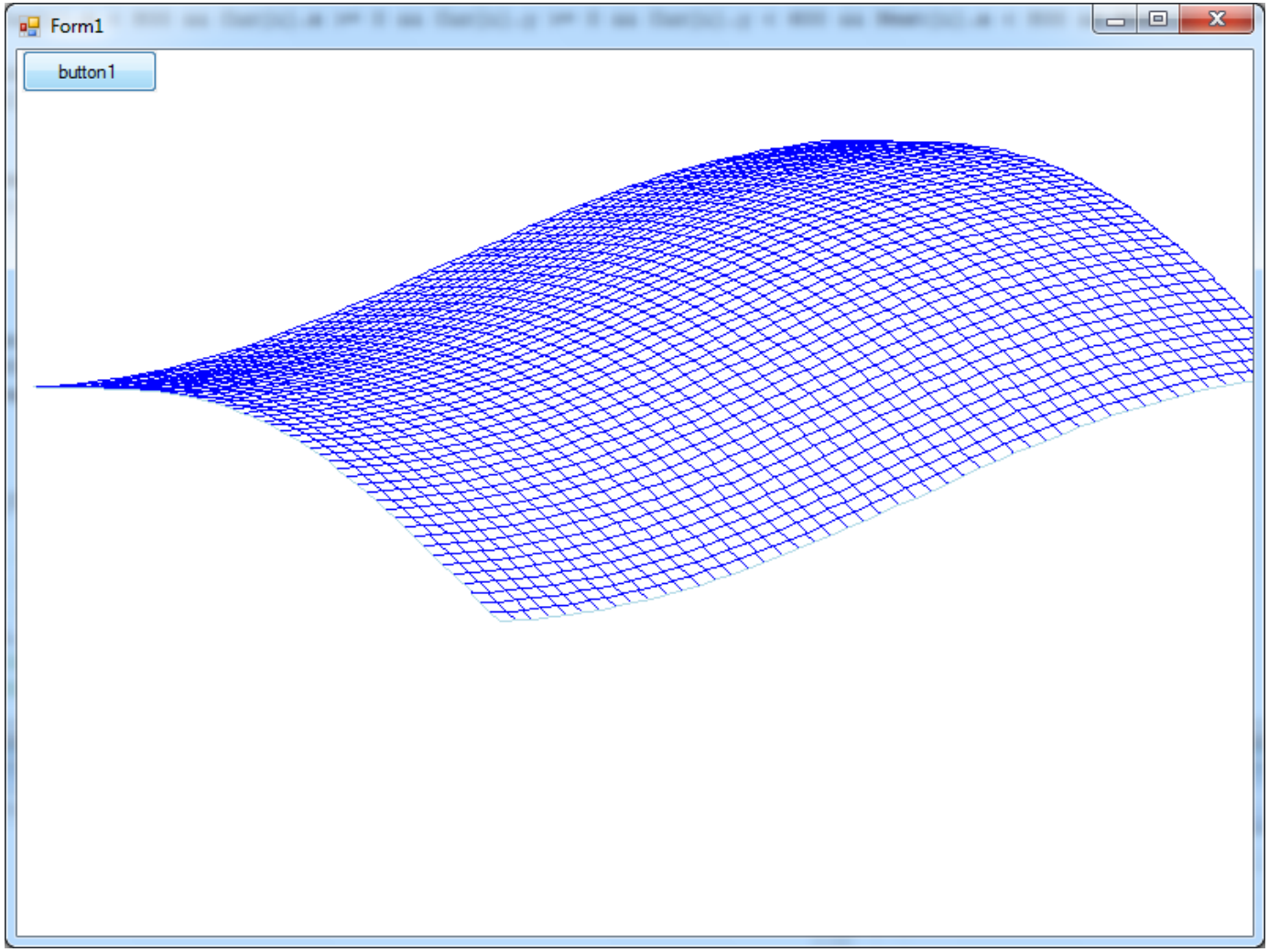


return Math.Sin(x) * Math.Cos(y);

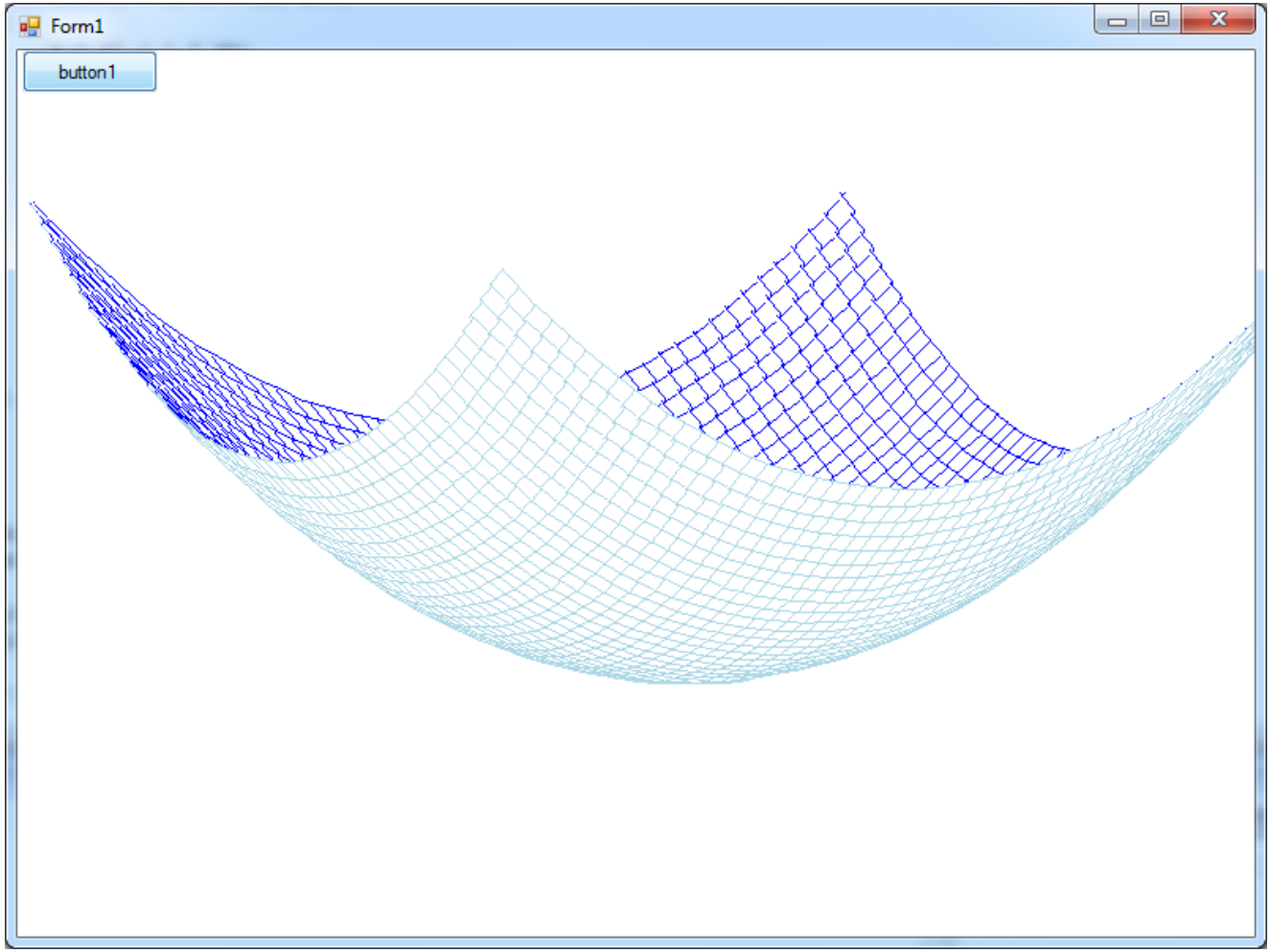




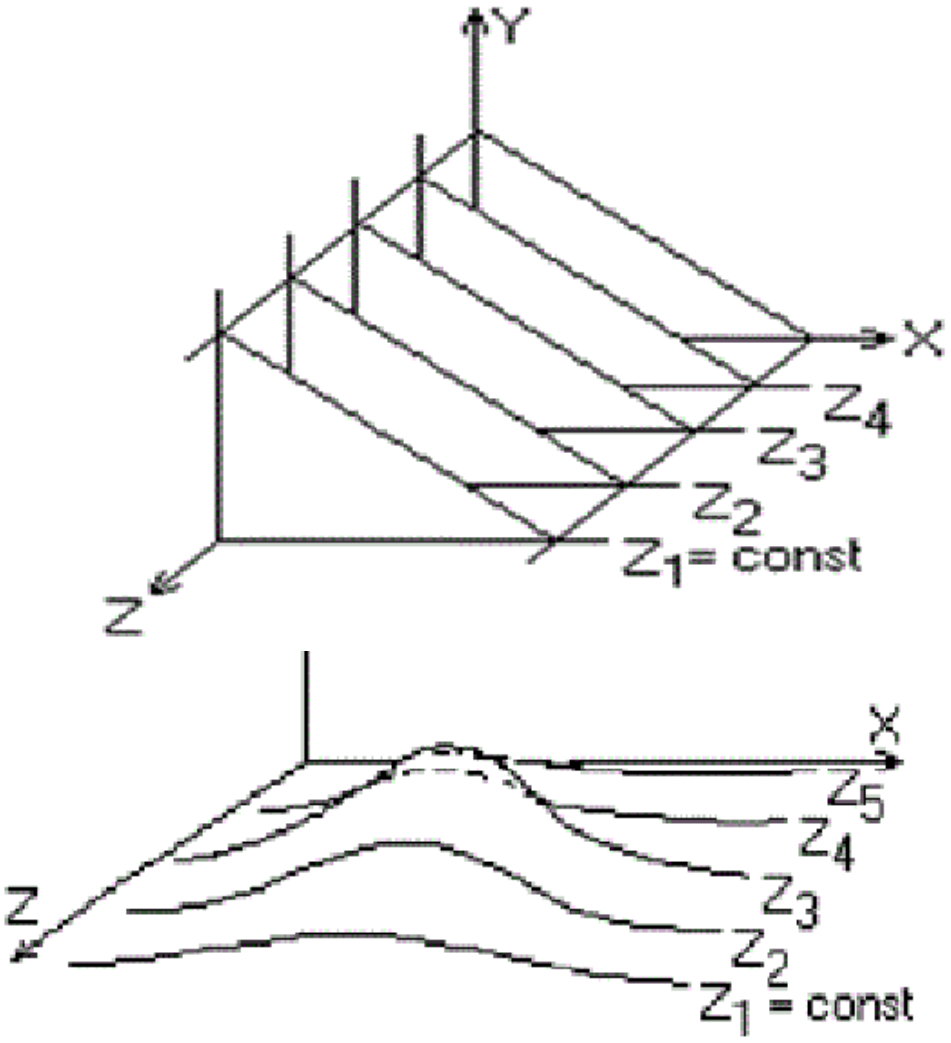
return Math.Sin(x)+Math.Cos(y);



return (x * x + y * y);



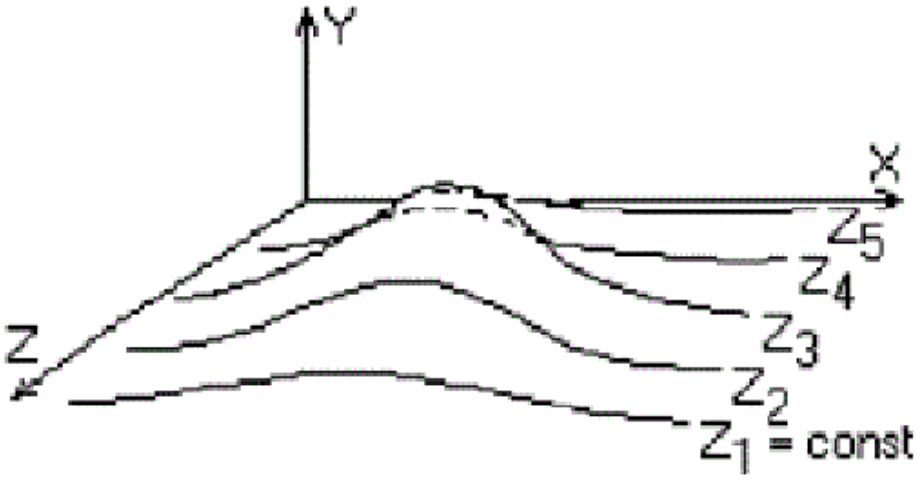
Алгоритм плавающего горизонта



Главная идея данного алгоритма заключается в сведении трёхмерной задачи к двумерной путем пересечения исходной поверхности последовательностью параллельных секущих плоскостей, имеющих постоянные значения координат x, y или z.

Алгоритм плавающего горизонта

Функция F (x, y, z) = 0 сводится к последовательности кривых, лежащих в каждой из этих параллельных плоскостей, например, последовательности y = f (x, z) или x = g(y, z), где z постоянна на каждой из заданных параллельных плоскостей.



Для хранения максимальных значений y для каждого значения x используется массив, длина которого равна числу различимых точек (пикселей) по оси x.

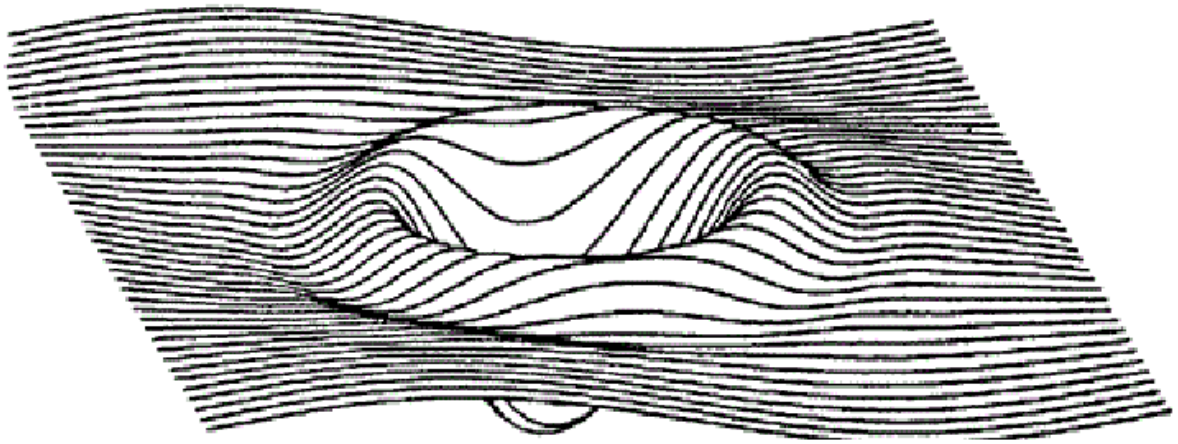
Линии горизонта или контурные линии

$Y = f_k(X)$ – проекция $y = f(x,z_k)$ на картинную плоскость,
где (X,Y) – координаты на картинной плоскости.

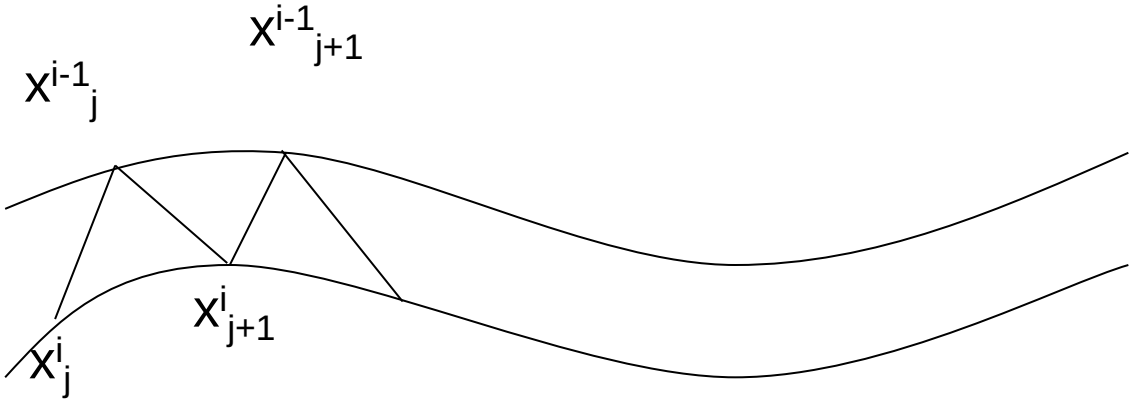
Контурные линии $Y^k_{max}(X)$ и $Y^k_{min}(X)$ определяются следующими соотношениями:

$Y^k_{max}(X) = \max Y_i(X)$

$Y^k_{min}(X) = \min Y_i(X)$



Вывод сеткой



Растровая реализация

Однако если для каждого значения x нельзя указать (вычислить) соответствующее ему значение y, то в таком случае используется линейная интерполяция значений y между известными значениями

Формальная запись алгоритма

- Если на текущей плоскости при некотором заданном значении x соответствующее значение y на кривой больше максимума или меньше минимума по y для всех предыдущих кривых при этом x, то текущая кривая видима. В противном случае она невидима.

- Если на участке от предыдущего x_n до текущего x_{n+k} значения x видимость кривой изменяется, то вычисляется точка пересечения x_i. Или можно использовать алгоритм Брезенхема

- Если на участке от x_n до x_{n+k} сегмент кривой полностью видим, то он изображается целиком; если он стал невидимым, то изображается фрагмент от x_n до x_i; если же он стал видимым, то изображается фрагмент от x_i до x_{n+k}.

- Заполнить массивы верхнего и нижнего плавающих горизонтов.