Разработка графических приложений

БЕЛЯЕВСКИЙ КИРИЛЛ ОЛЕГОВИЧ, АСП.

KIRILL.BELIAEVSKII@SPBPU.COM

Программа курса

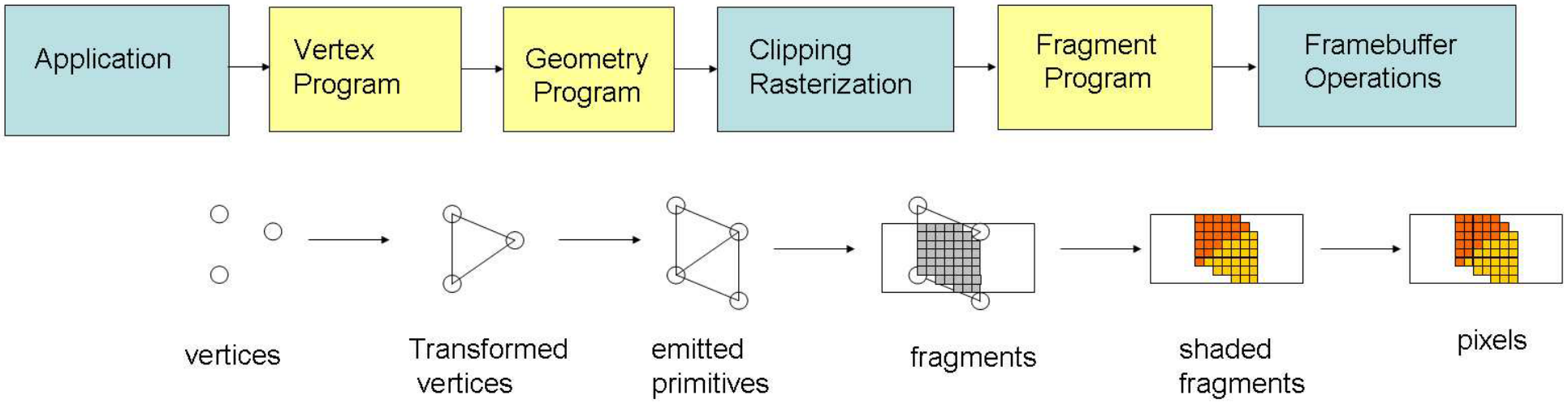
Три лекции:

- Трехмерные модели
- Пространственные преобразования
- Растеризация
- Раскраска и текстурирование
- Z-буферизация

Одна лабораторная:

- Растеризация трехмерных моделей на CPU

Конвейер визуализации

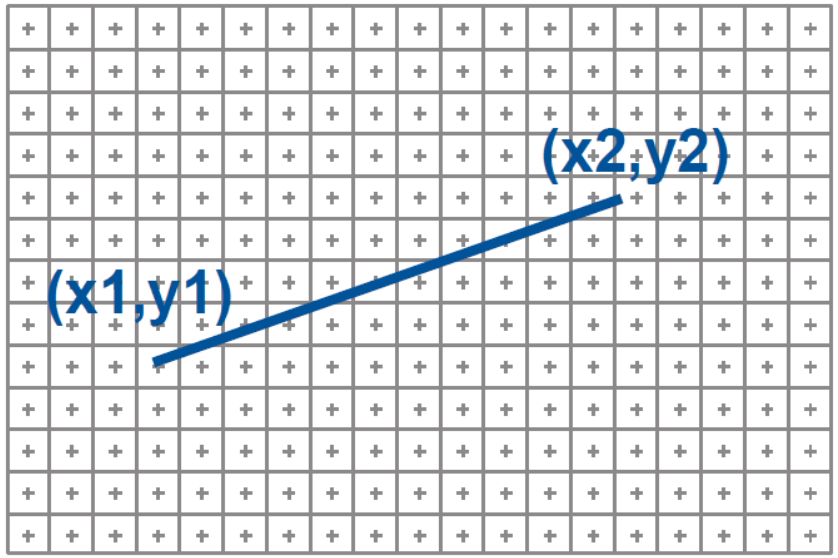


Буфер кадра

Двумерный массив пикселей

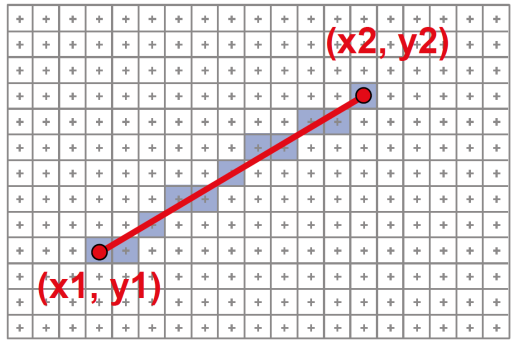
Каждый пиксель может иметь свой цвет

Оконные координаты: центр пикселя как целое значение

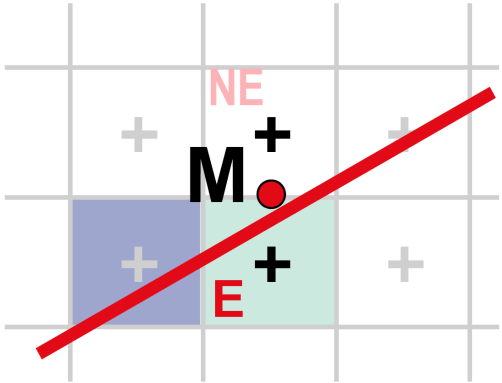


Алгоритмы растеризации линий

Наивный алгоритм растеризации



Bresenham DDA



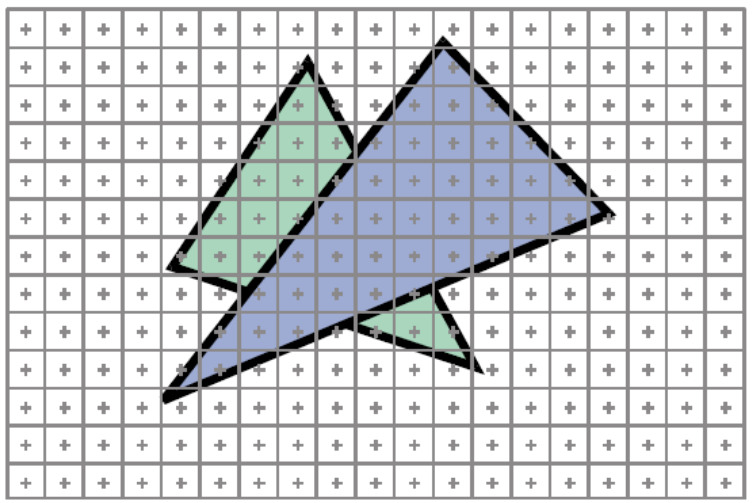
Конвертация в 2Д

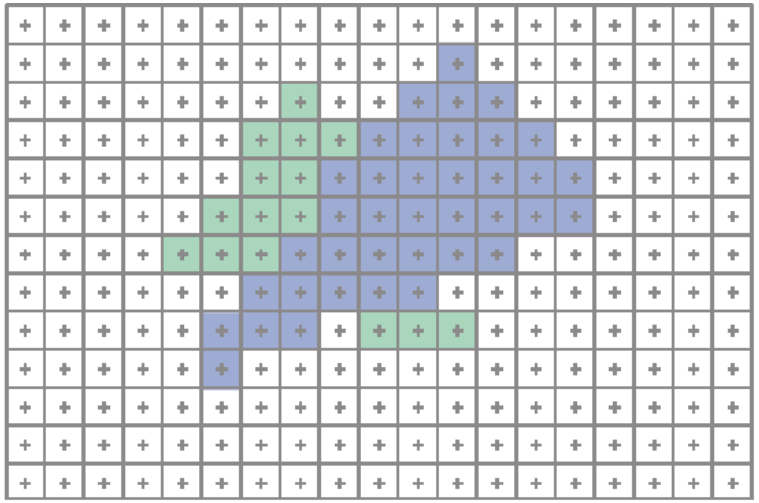
Геометрические примитивы

- 2D: point, line, polygon, circle...
- 3D: point, line, polyhedron, sphere...

Примитивы непрерывны, экран дискретен

Растеризация вычисляет дискретное приближение в терминах пикселей





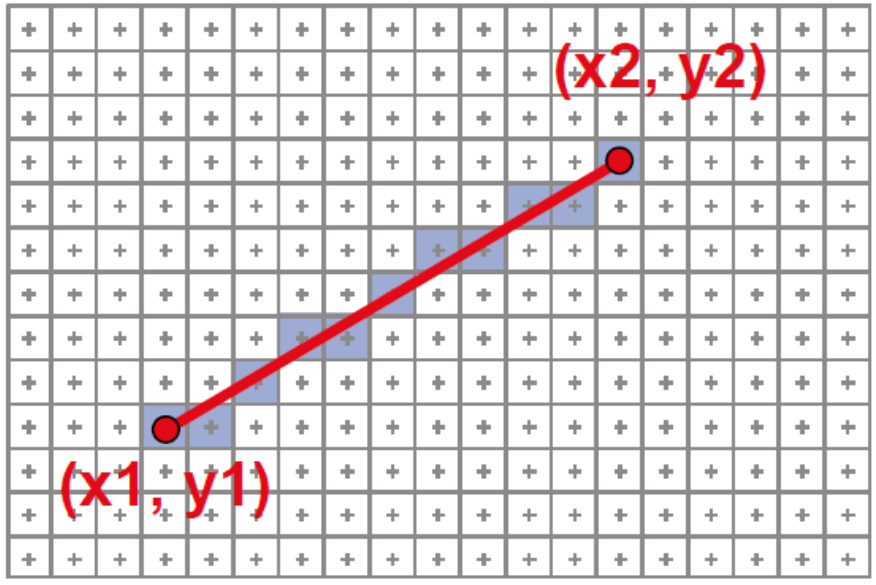
Наивная растеризация линий

Дано:

- Координаты сегмента линии (integers x1, y1; x2, y2)
- x2 > x1, y2 > y1

Вычислить:

- Набор пикселей x;y, принадлежащих сегменту линии

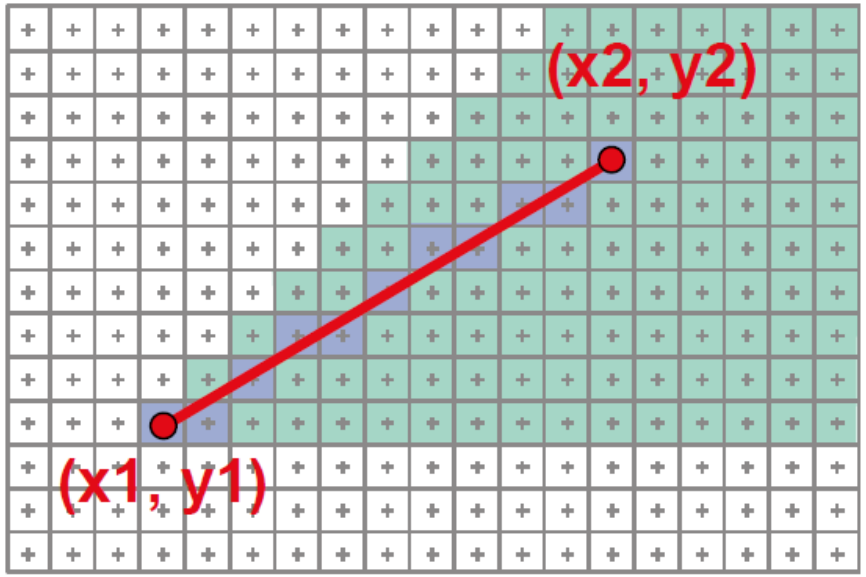


Дизайн алгоритма

Пусть m – крутизна линии, m = dy / dx

Алгоритм корректен при 0 < m < 1

Почему?

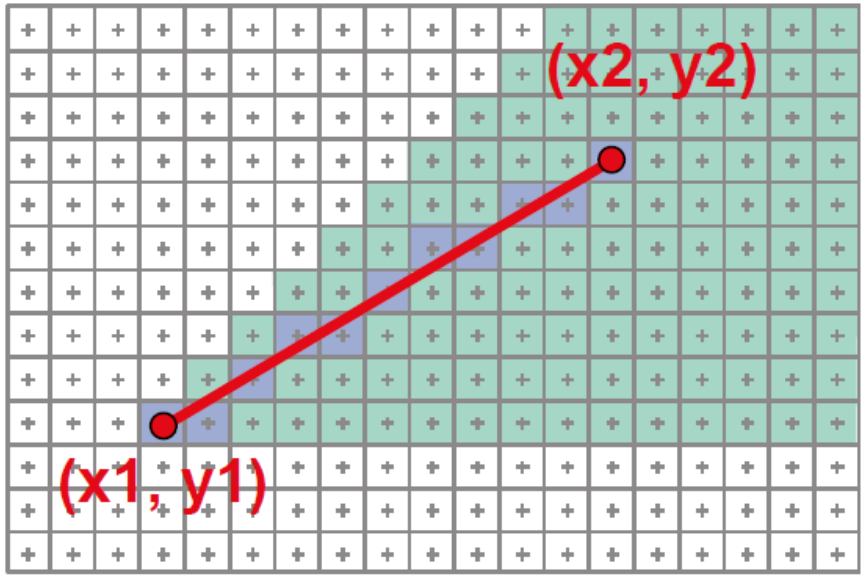


Дизайн алгоритма

Ровно один пиксель на колонку

- Меньше -> отсутствие соединений

- Больше -> слишком толстая линия

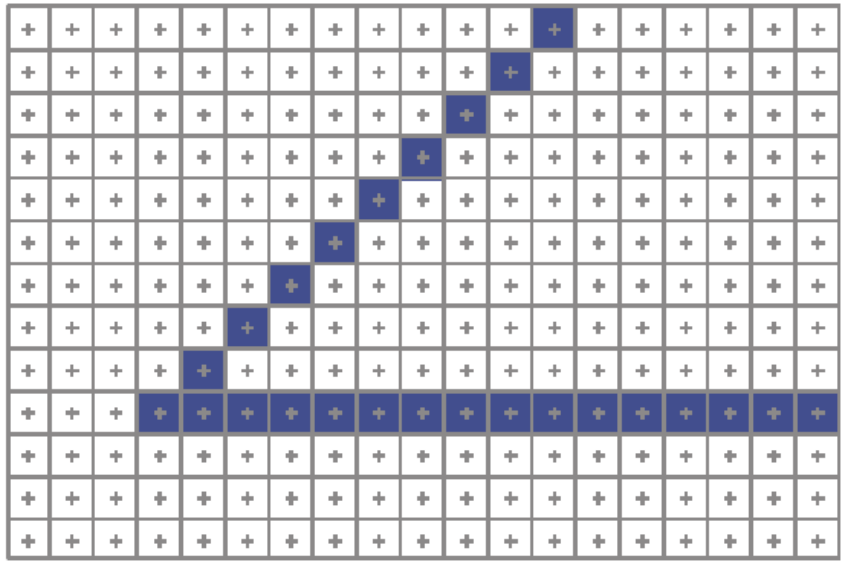


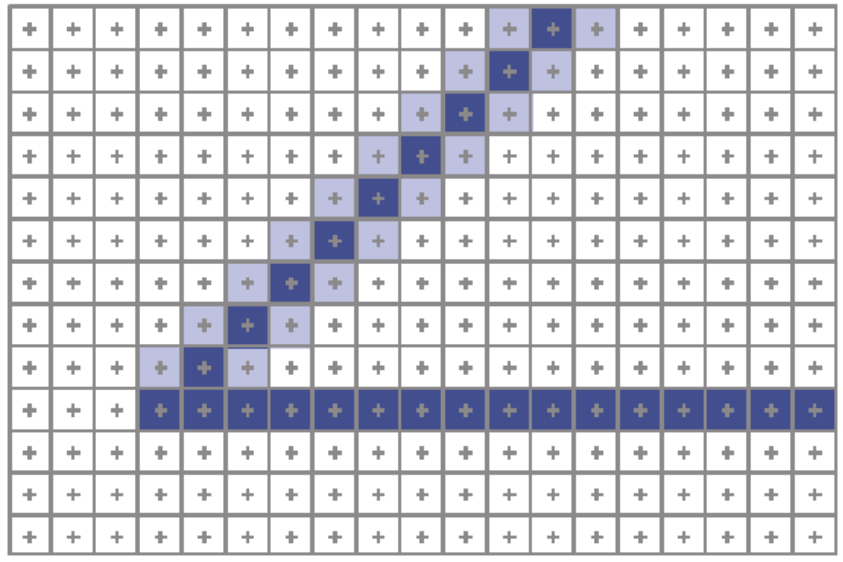
Дизайн алгоритма

Яркость линии может варьироваться в зависимости от уклона

Как это компенсировать?

- Антиалиазинг





Наивный алгоритм, первый вариант

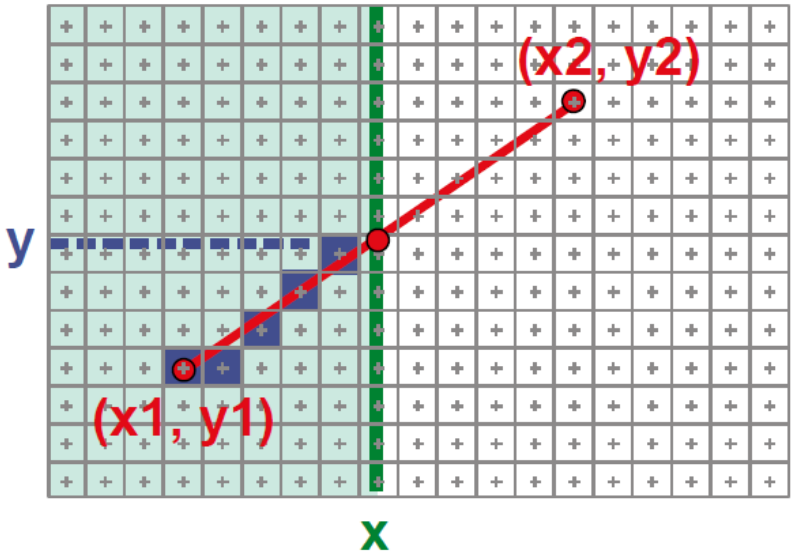
Рассчитать y как функцию от x

- Двигая вертикальное окно сканирования от x1 до x2

Таким образом:

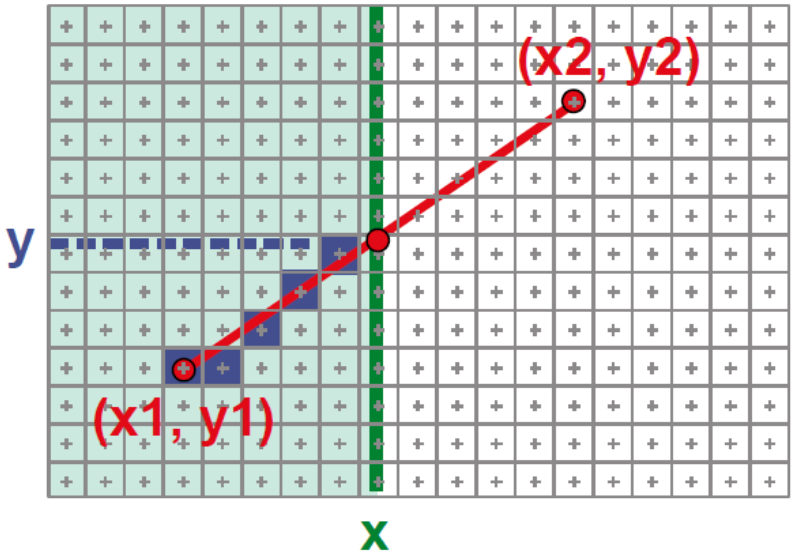
$$y = y_1 + \frac{x - x_1}{x_2 - x_1}(y_2 - y_1) = y_1 + m(x - x_1)$$

$$m = \frac{dy}{dx}$$



Наивный алгоритм, первый вариант

1. Рассчитать y как функцию от x
2. Округлить y
3. Установить значение пикселя (x, floor(y(x)))



Инкрементальный алгоритм

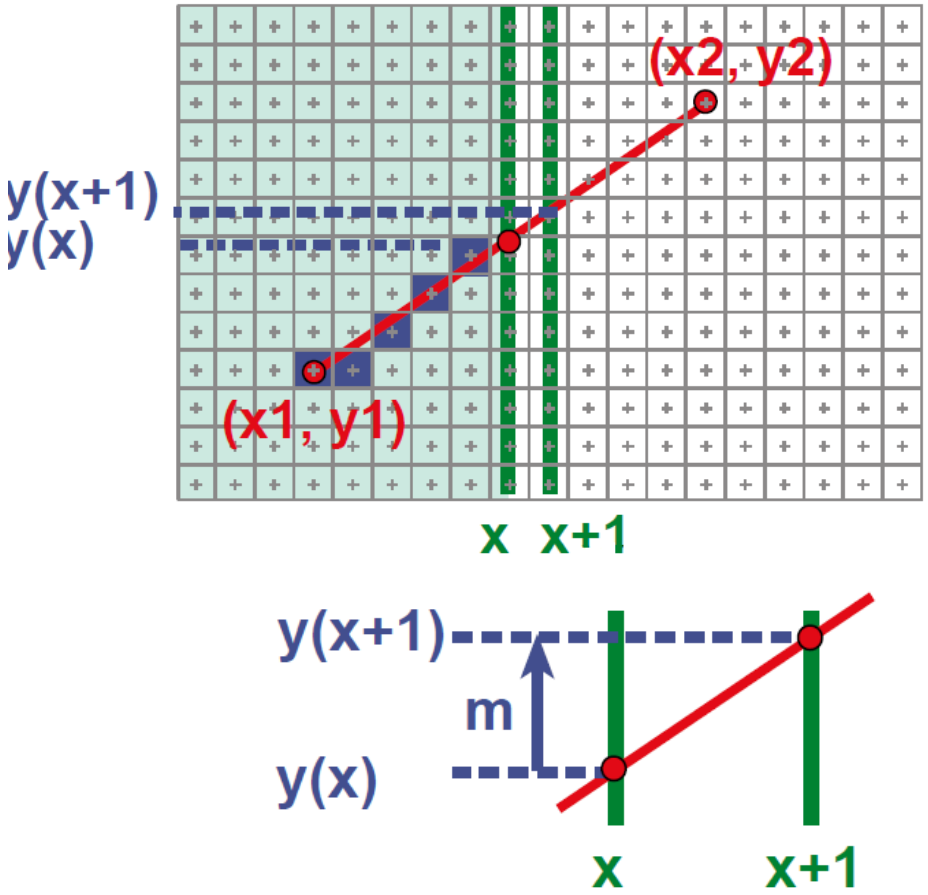
Расчет значения y можно упростить:

Инкрементный подход:

- y += m на каждом шаге алгоритма вместо вычисления полного значения

Таким образом:

- Начать в точке (x1; y1)
- Инкрементировать y на значение уклона m на каждом шаге алгоритма
- Обратите внимание: x – integer, но y - float



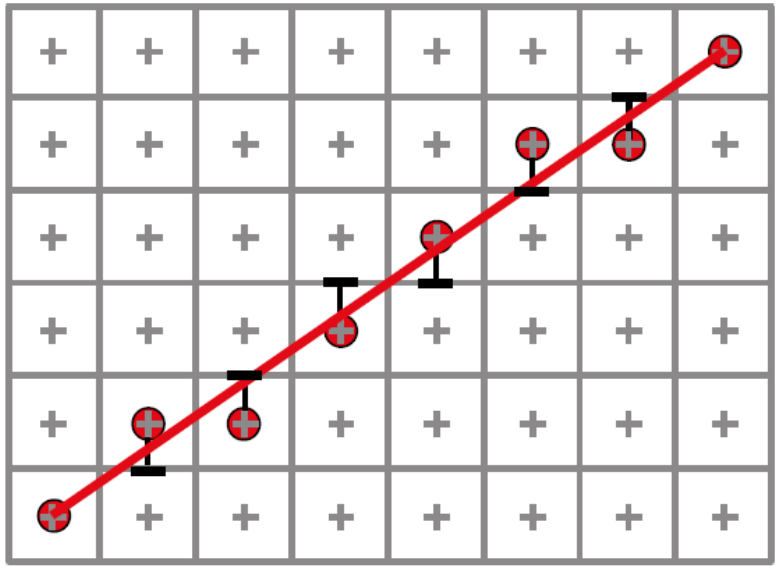
Вопросы?

Алгоритм Брезенхэма

Выделяет пиксель, ближайший к сегменту по вертикали

Результат такой же, как и у наивногоалгортма

Более эффективен, позволяет использовать только целочисленные вычисления



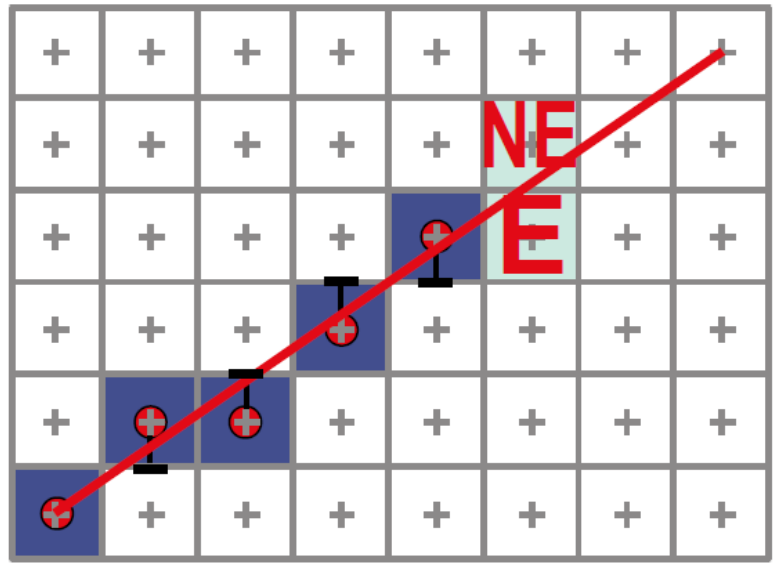
Алгоритм Брезенхэма

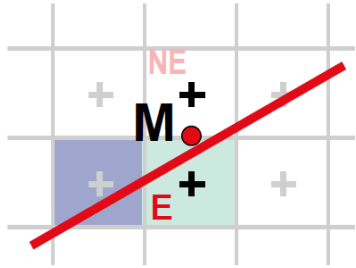
Наблюдение:

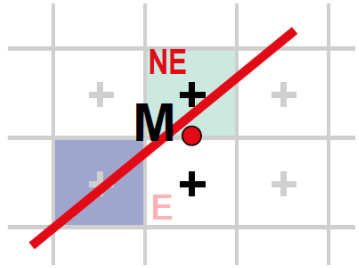
- Следующий после пикселя **P** (xp, yp) будет либо **E** либо **NE**

Какой пиксель выбрать?

- Выберите E, если сегмент проходит ниже или через среднюю точку M
- Выберите NE, если сегмент проходит над M







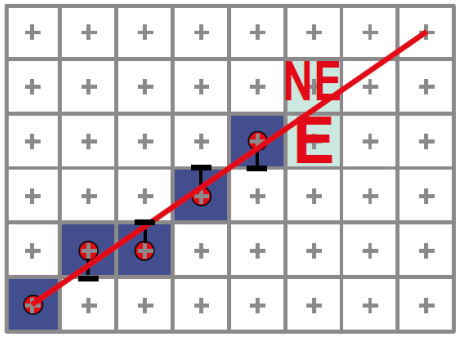
Реализация

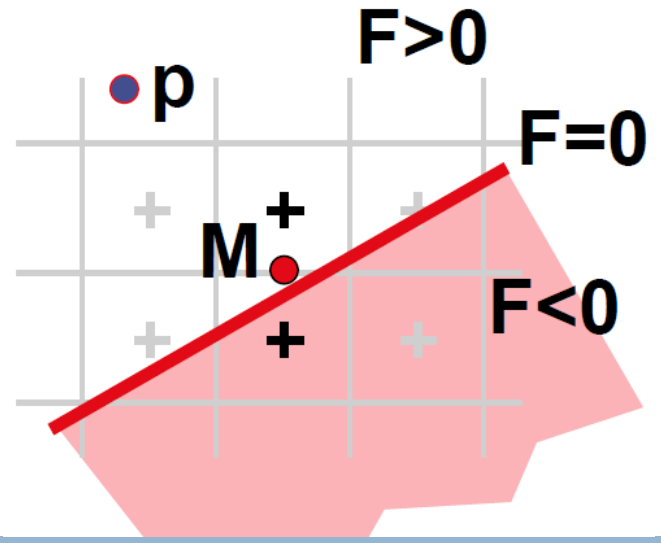
Используем неявное уравнение F для базовой линии L

- F(x;y) = 0, где F(x;y) = y – mx – b
- F положительна над L, равна нулю на L и отрицательна под L

Введем метрику ошибки *e* = -F(x,y)

- Выберем NE если e > 0.5
- Выберем E если e < 0.5





Использование метрики ошибки

Вычислим *e’* при приращении координаты x

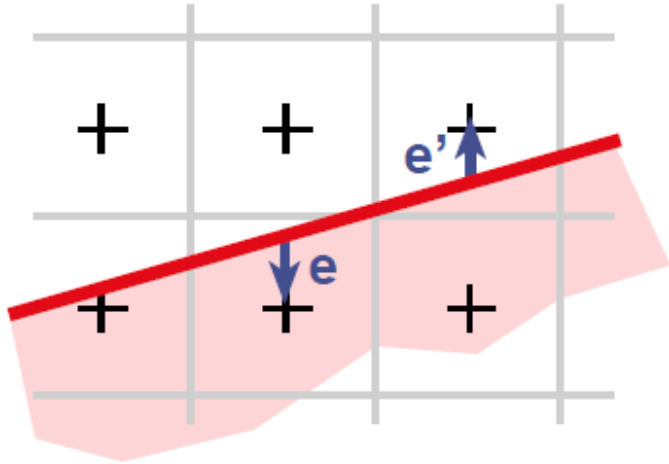
- e’ = e + m

Если e’ < 0.5:

- Выбрать E

Иначе:

- Выбрать NE
- Уменьшить e на единицу

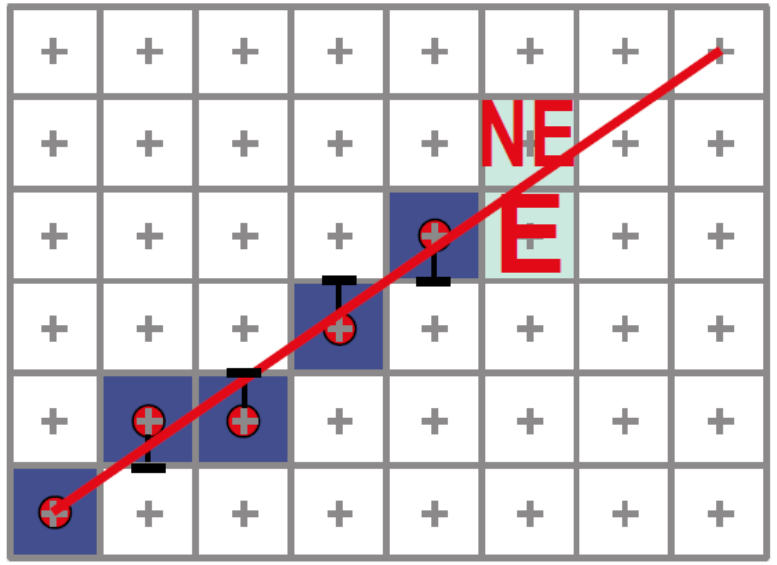


Итоговый алгоритм

Инициализировать x,y,e

Цикл x1::x2

- Закрашивание
- e = e + m
- Если e > 0.5
  - Увеличить y
  - Уменьшить e на 1



Вопросы?

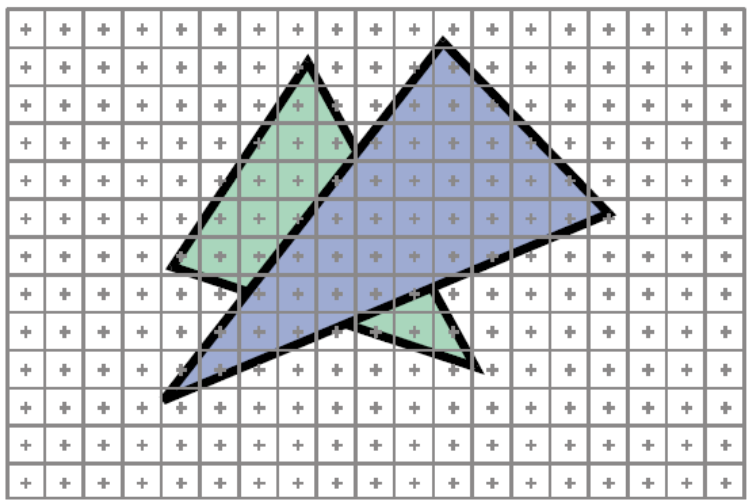
Конвертация в 2Д

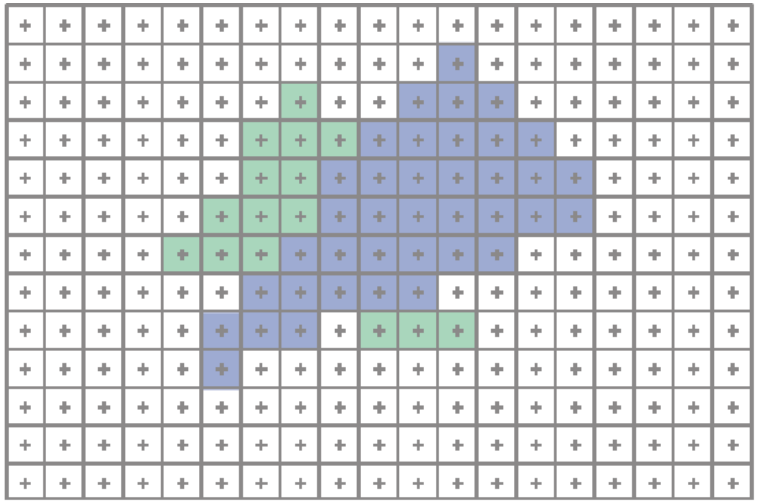
Геометрические примитивы

- 2D: point, line, polygon, circle...
- 3D: point, line, polyhedron, sphere...

Примитивы непрерывны, экран дискретен

Растеризация вычисляет дискретное приближение в терминах пикселей





Алгоритмы растеризации треугольников

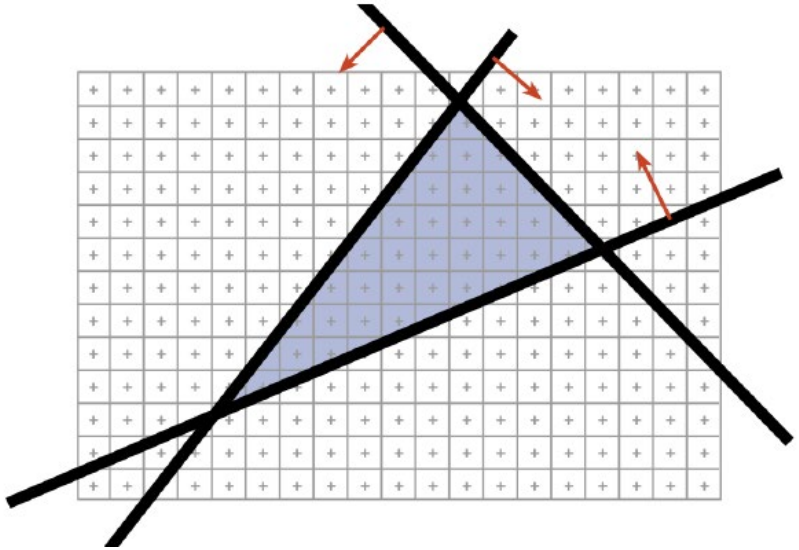
Грани трехмерного треугольника проецируются на экран как сегменты линий

Внутренняя часть треугольника представляет собой множество точек, находящихся во всех трех полупространствах, определенных этими линиями

$$E_i(x, y) = a_i x + b_i y + c_i$$

(x, y) внутри треугольника, если:

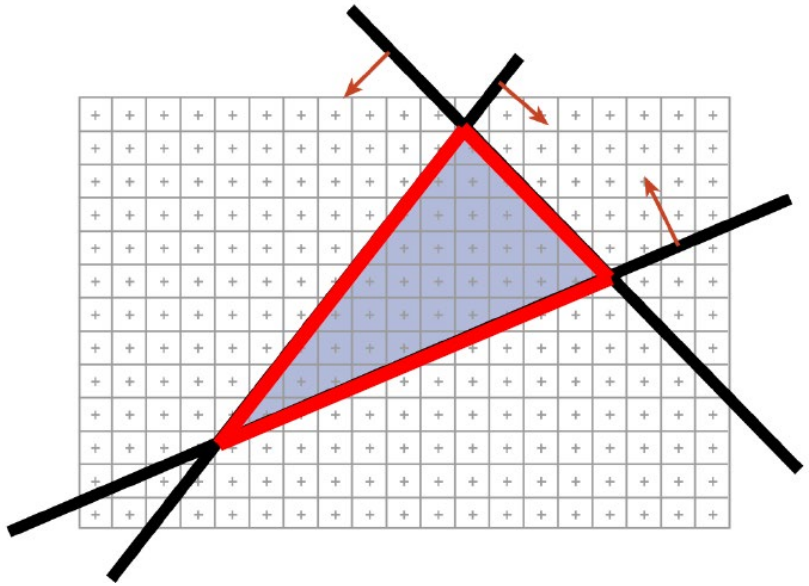
$$E_i(x, y) \geq 0, \forall i = 1,2,3$$



Brute Force Rasterizer

1. Рассчитать коэффициенты E_1, E_2 и E_3 из спроецированных на экран вершин
   - т.н. «Настройка треугольника», вычисляет a_i, b_i, c_i для $i = 1,2,3$

2. Для каждого пикселя (x, y)
   - Вычислить принадлежность треугольнику для центра пикселя
   - Если значение положительно – пиксель внутри



Проблемы?

Если треугольник мал, то требуется много ненужных вычислений для тестирования каждого пикселя

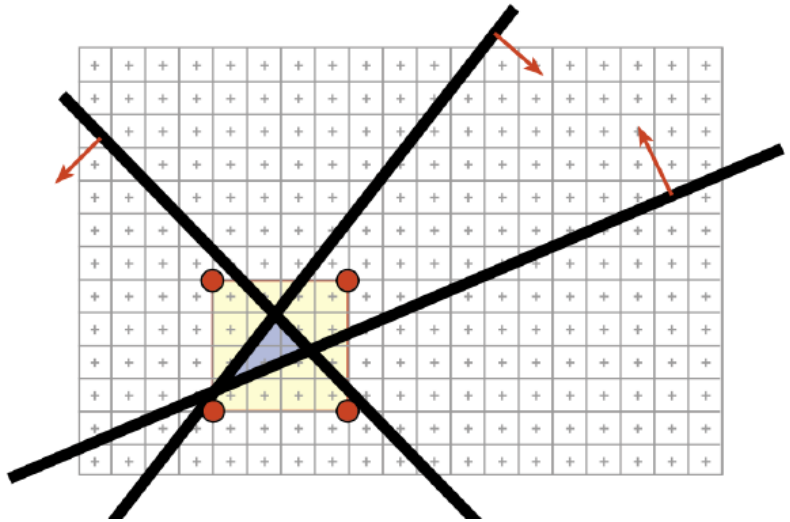
Простая оптимизация

Оптимизация:

- Производить сканирование только тех пикселей, которые находятся внутри *ограничивающего прямоугольника*

Как получить такой прямоугольник?

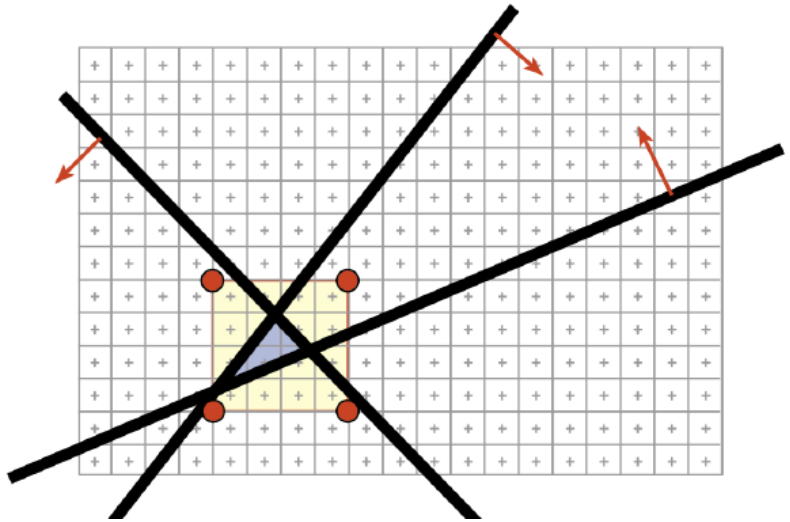
- $X_{min}, X_{max}, Y_{min}, Y_{max}$ от спроецированных на экран вершин треугольника



Алгоритм

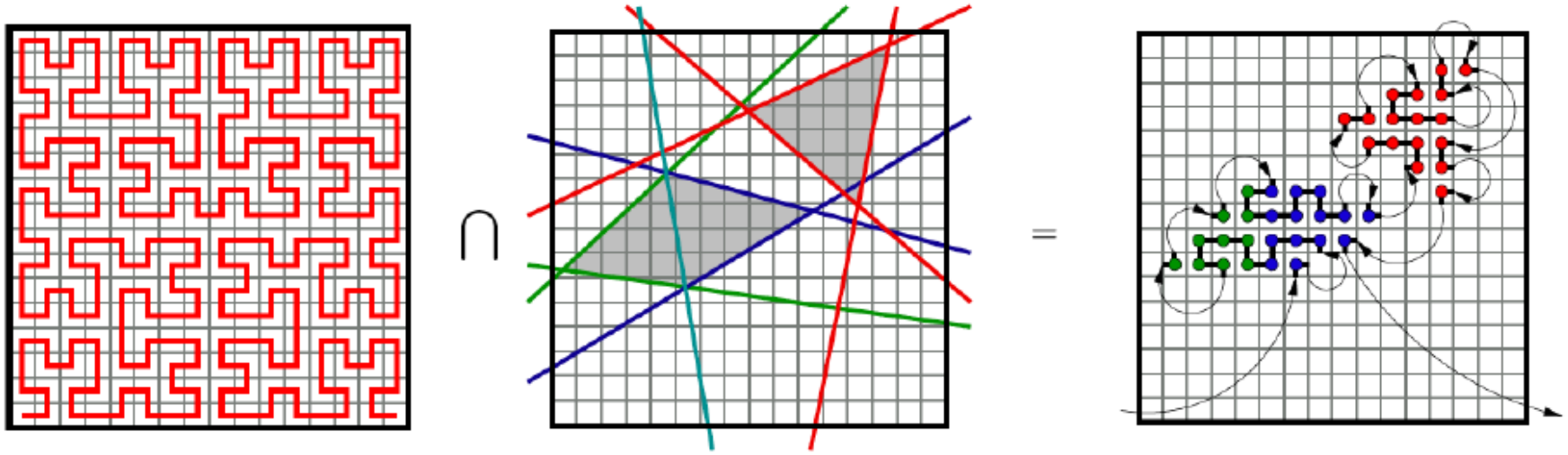
For every triangle

- Compute projection for vertices, compute the Ei
- Compute bbox, clip bbox to screen limits
- For all pixels in bbox
  - Evaluate edge functions $a_i x + b_i y + c_i$
    - If all > 0
      - Framebuffer[x,y] = triangleColor



Вопросы?

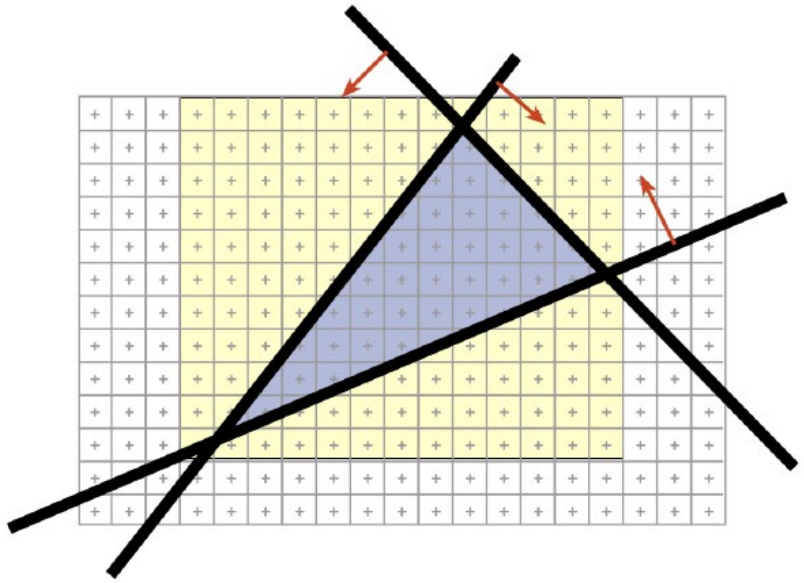
Более продвинутые алгоритмы растеризации:Hilbert curve rasterizer by McCool, Wales and Moule.



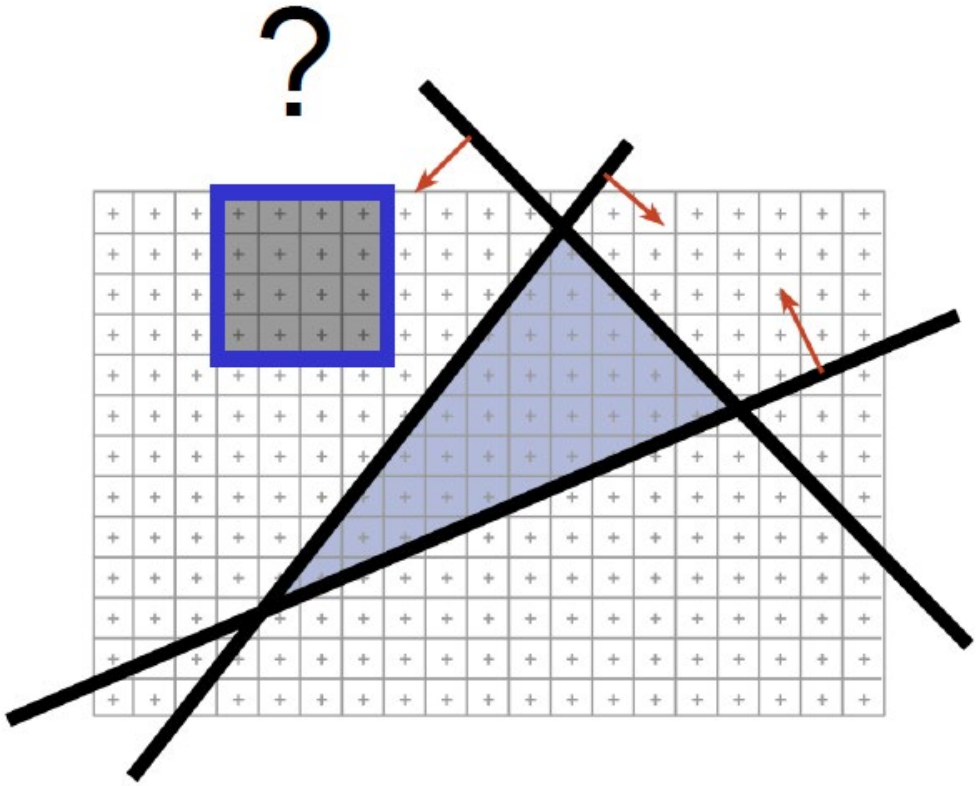
Больше оптимизации

Мы выполняем вычисления для большого количества ненужных пикселей

Как можно это исправить?



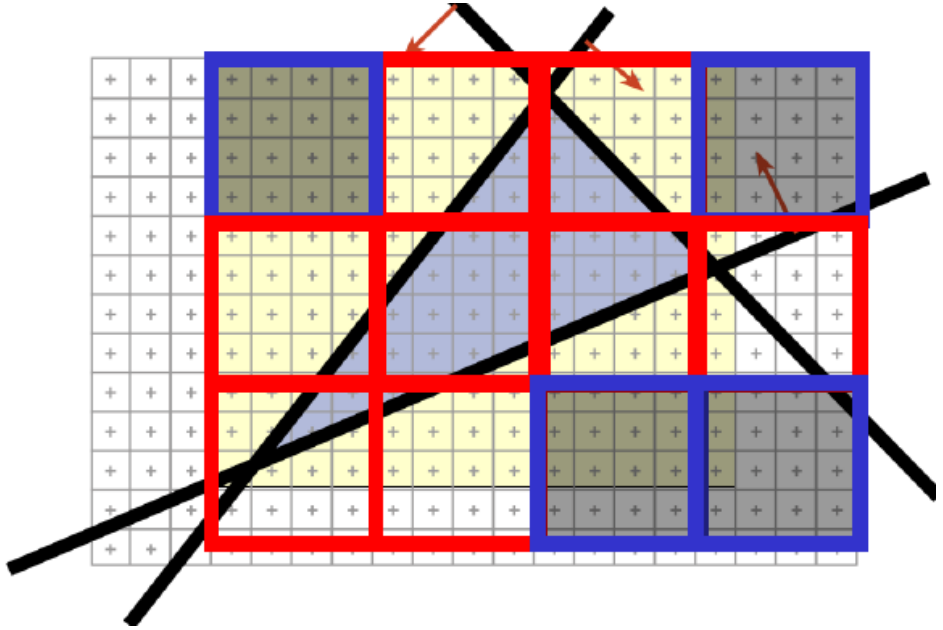
Больше оптимизации



Больше оптимизации

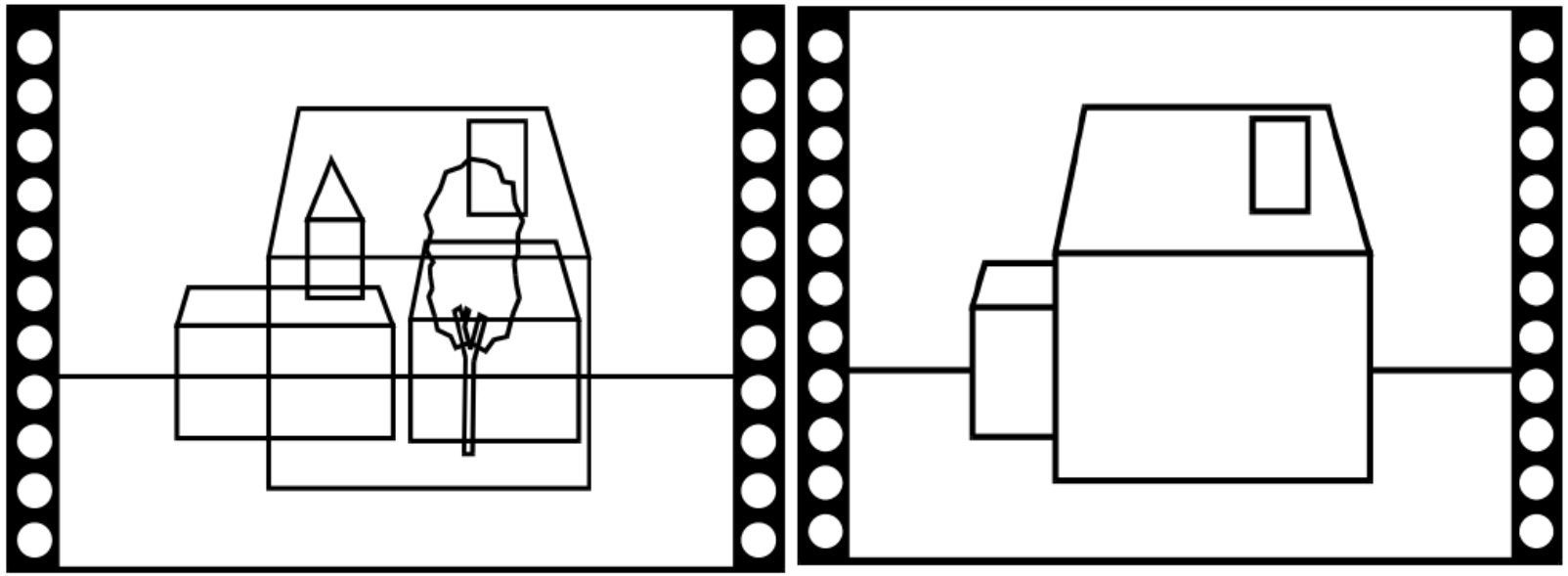
Иерархическая растеризация!

- Тестирование блоков пикселей перед переходом на попиксельный уровень (большие блоки можно пропускать сразу)
- Обычно используется два уровня
- Можно также проверить, находится ли весь блок внутри треугольника. Тогда можно пропустить тесты всех пикселей для еще большего ускорения.



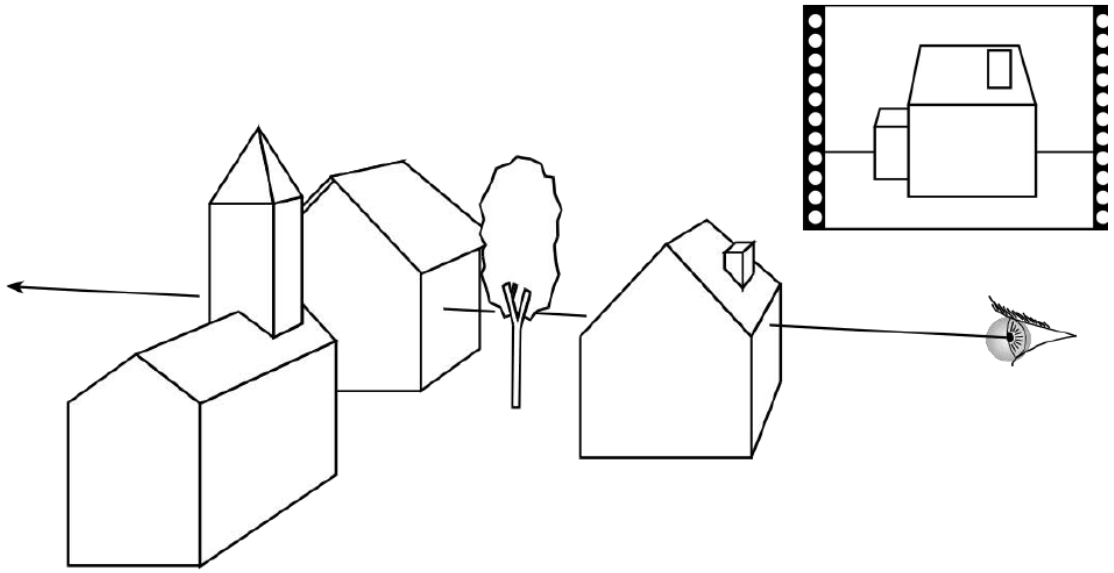
Видимость

Как мы узнаем, какие части видимы / впереди?



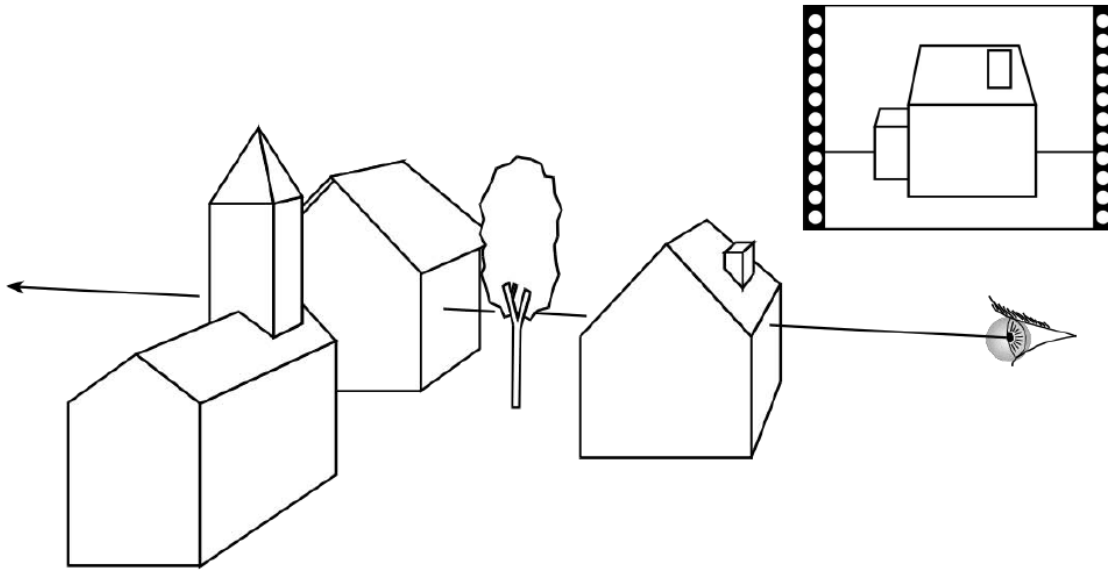
Видимость

Определяем пересечение с ближайшим объектом



Видимость

Как определить ближайший ли пиксель при растеризации?

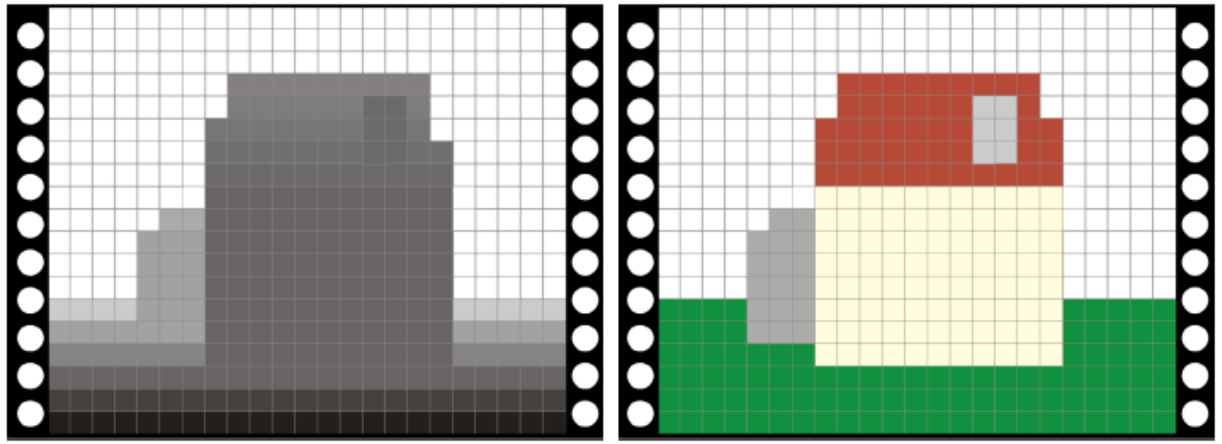


Буфер глубины (z-буффер)

Используется в дополнение к буферу кадра

Хранит дистанцию до камеры для каждого пикселя

Цвет (и значение глубины) пикселя обновляется только если *newz* ближе, чем значение в z-буффере



Псевдокод алгоритма

For every triangle

Compute Projection, color at vertices

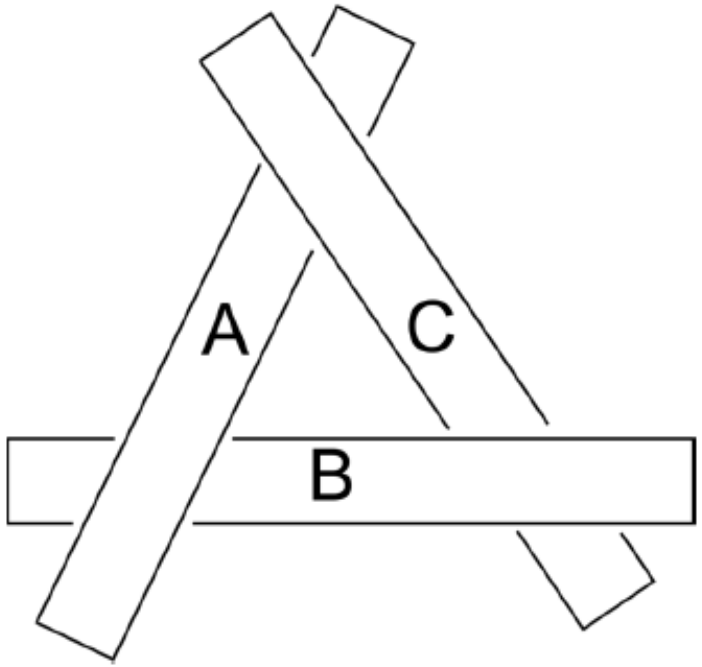
Setup line equations

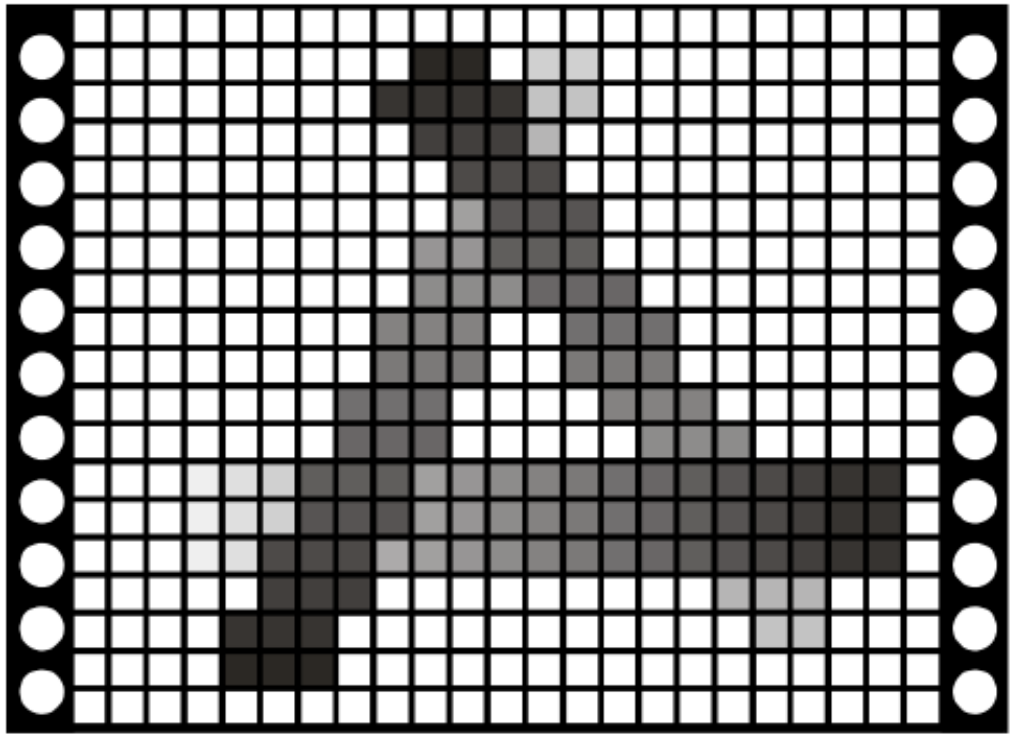
Compute bbox, clip bbox to screen limits

For all pixels in bbox

- Increment line equations
- **Compute curentZ**
- Compute currentColor
- If all line equations>0 *//pixel [x,y] in triangle*
  - **If currentZ<zBuffer[x,y] *//pixel is visible***
    - Framebuffer[x,y]=currentColor
    - **zBuffer[x,y]=currentZ**

Работает для сложных случаев!





Интерполяция в пространстве экрана

Как получить значение глубины для каждого пикселя?

- Мы знаем только значение глубины для вершин
- Нужно интерполировать значения внутри треугольника из вершин

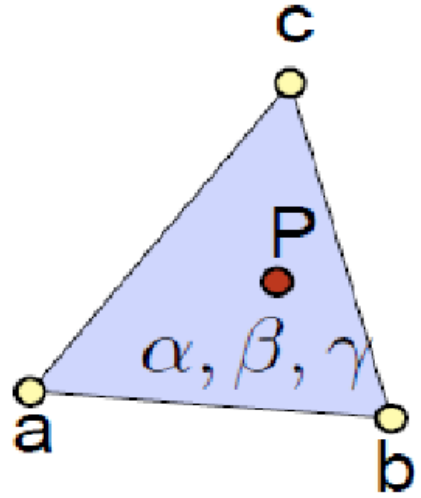
Барицентрические координаты

Барицентрические координаты для треугольника (a, b, c)

- $P(\alpha, \beta, \gamma) = \alpha\boldsymbol{a} + \beta\mathbf{b} + \gamma\boldsymbol{c}$
- $\alpha + \beta + \gamma = 1, \alpha, \beta, \gamma \geq 0$

Барицентрические координаты могут использоваться для интерполяции любых атрибутов вершины (координаты, текстурные координаты, цвет, …)

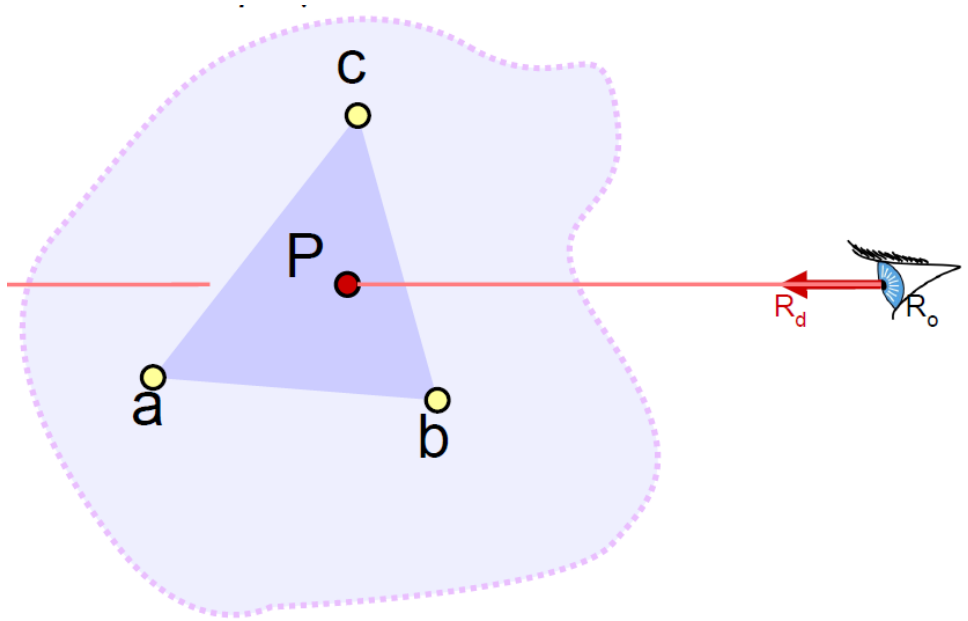
- Все, что линейно изменяется в пространстве объектов
- Включая Z



Барицентрические координаты

Барицентрическое определение плоскости

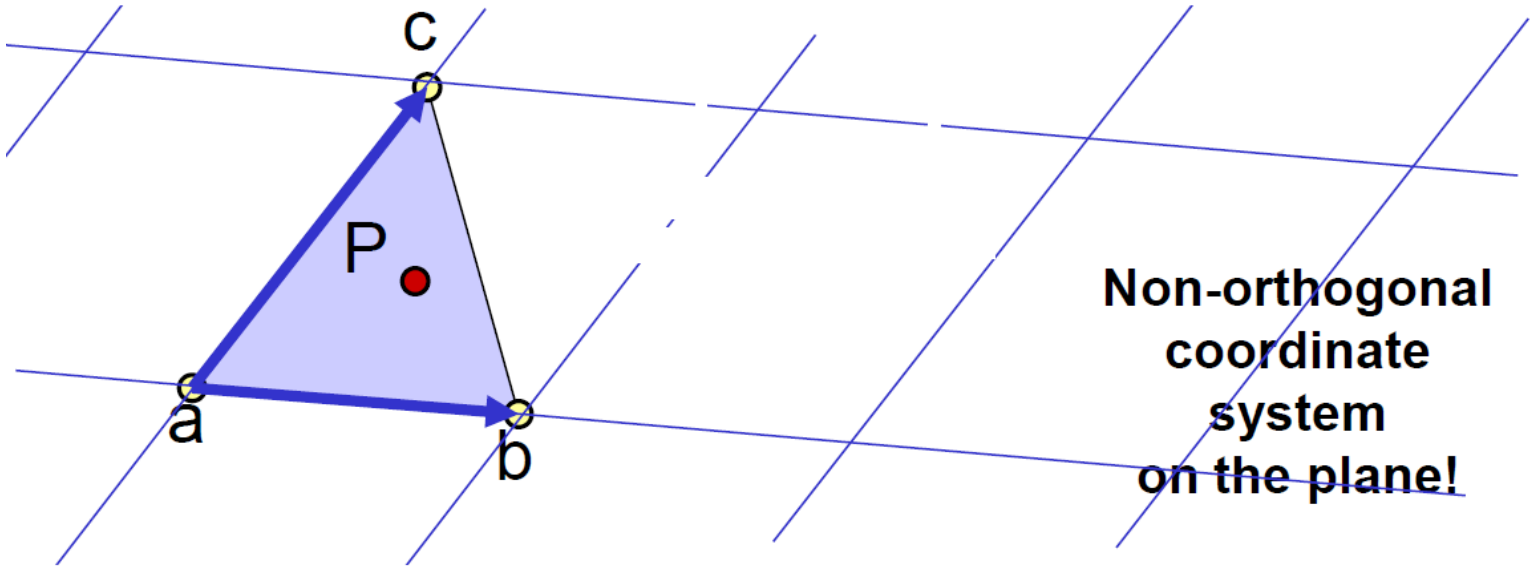
- (невырожденный) треугольник (a, b, c) определяет плоскость
- Любая точка P на этой плоскости может быть записана как:
  - $P(\alpha, \beta, \gamma) = \alpha\boldsymbol{a} + \beta\boldsymbol{b} + \gamma\boldsymbol{c}$, где $\alpha + \beta + \gamma = 1$



Барицентрические координаты

Так как $\alpha + \beta + \gamma = 1$, можно считать $\alpha = 1 - \beta - \gamma$

- $P(\alpha, \beta, \gamma) = \alpha\boldsymbol{a} + \beta\boldsymbol{b} + \gamma\boldsymbol{c}$
- $P(\beta, \gamma) = (1 - \beta - \gamma)\boldsymbol{a} + \beta\boldsymbol{b} + \gamma\boldsymbol{c} = \boldsymbol{a} + \beta(\boldsymbol{b} - \boldsymbol{a}) + \gamma(\boldsymbol{c} - \boldsymbol{a})$

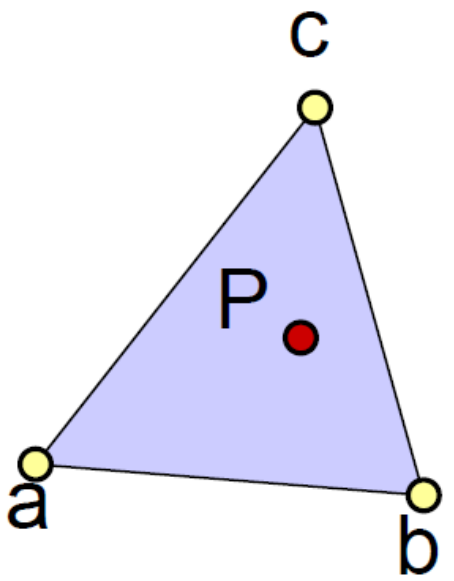


Барицентрическое определение треугольника

Уравнение $P(\alpha,\beta,\gamma)=\alpha\boldsymbol{a}+\beta\boldsymbol{b}+\gamma\boldsymbol{c}$, где $\alpha+\beta+\gamma=1$ параметризует всю плоскость

Если мы примем в дополнение, что $\alpha, \beta, \gamma \geq 0$, то получим треугольник!

- Вместе с $\alpha+\beta+\gamma=1$ это означает, что:
  - $0 \leq \alpha \leq 1 \;\; \& \;\; 0 \leq \beta \leq 1 \;\; \& \;\; 0 \leq \gamma \leq 1$
- Проверка:
  - α = 0, значит P лежит на линии b-c
  - α, β = 0, значит P = c
  - И т.д.



Барицентрическое определение треугольника

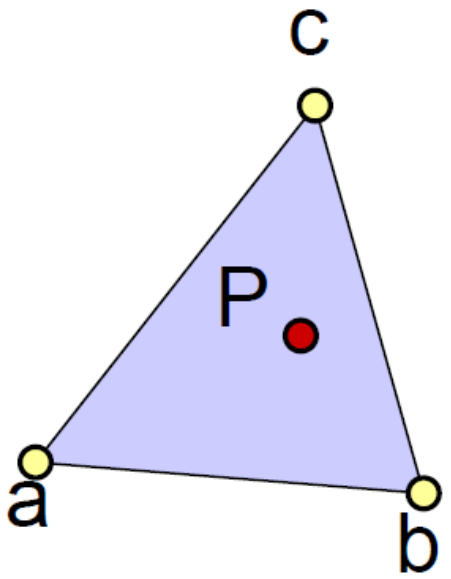
$P(\alpha,\beta,\gamma)=\alpha\boldsymbol{a}+\beta\boldsymbol{b}+\gamma\boldsymbol{c}$

Координаты являются барицентрическими, если:

- $\alpha+\beta+\gamma=1$

Точка находится внутри треугольника, если:

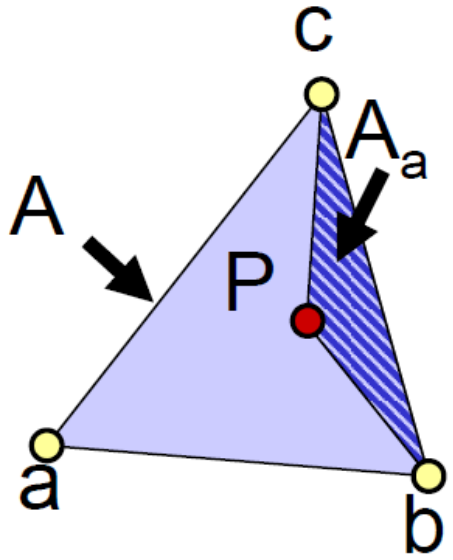
- $\alpha,\beta,\gamma \geq 0$



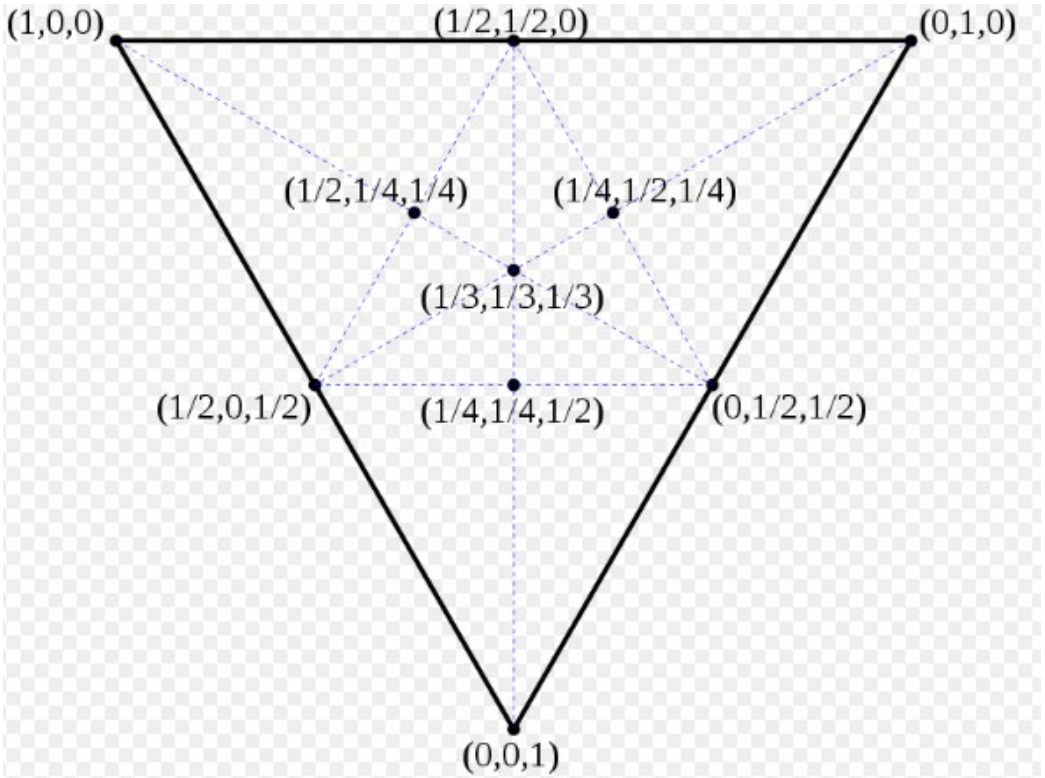
Как рассчитать α, β, γ?

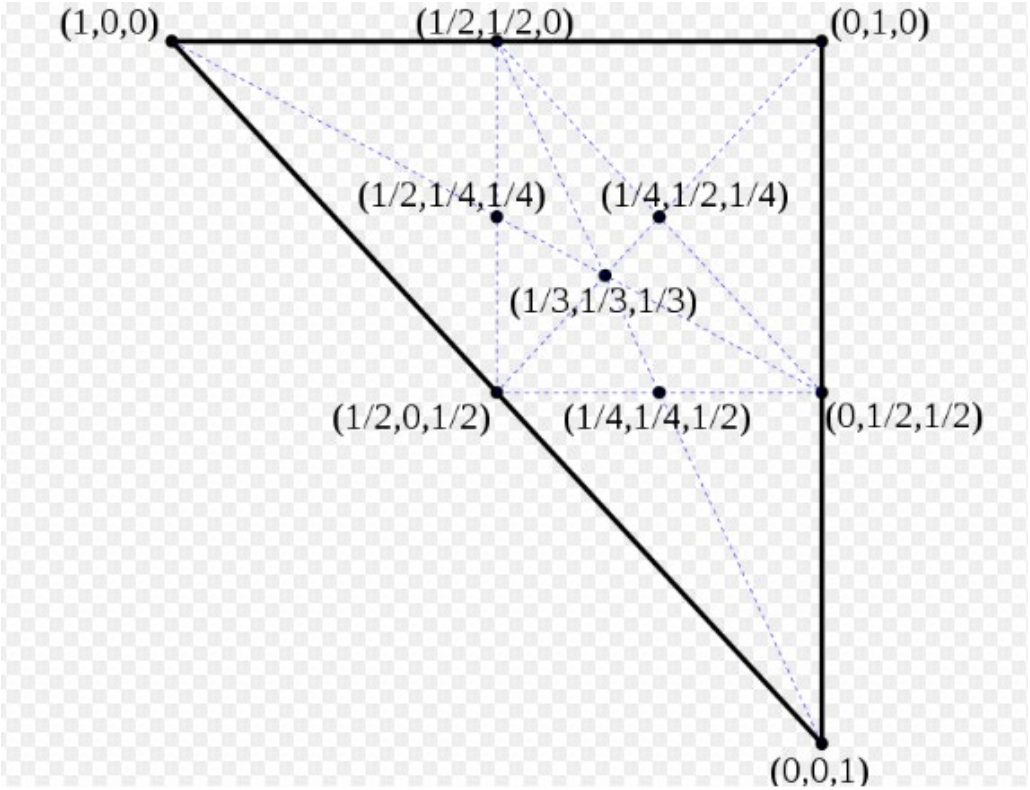
Отношение площади противоположного треугольника к общей площади

- $\alpha = \frac{A_a}{A}, \ \beta = \frac{A_b}{A}, \quad \gamma = \frac{A_c}{A}$



Барицентрические координаты треугольника

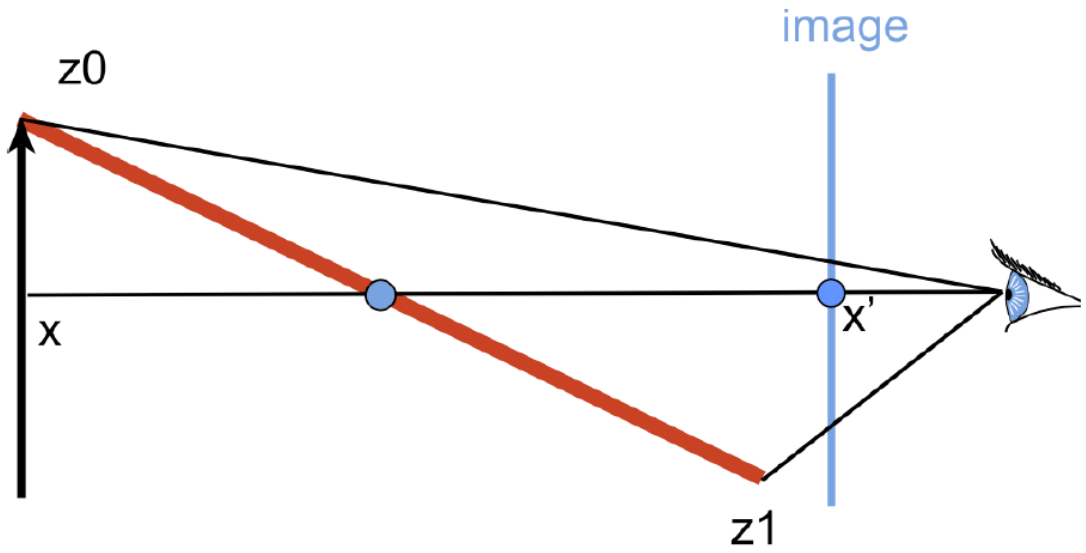




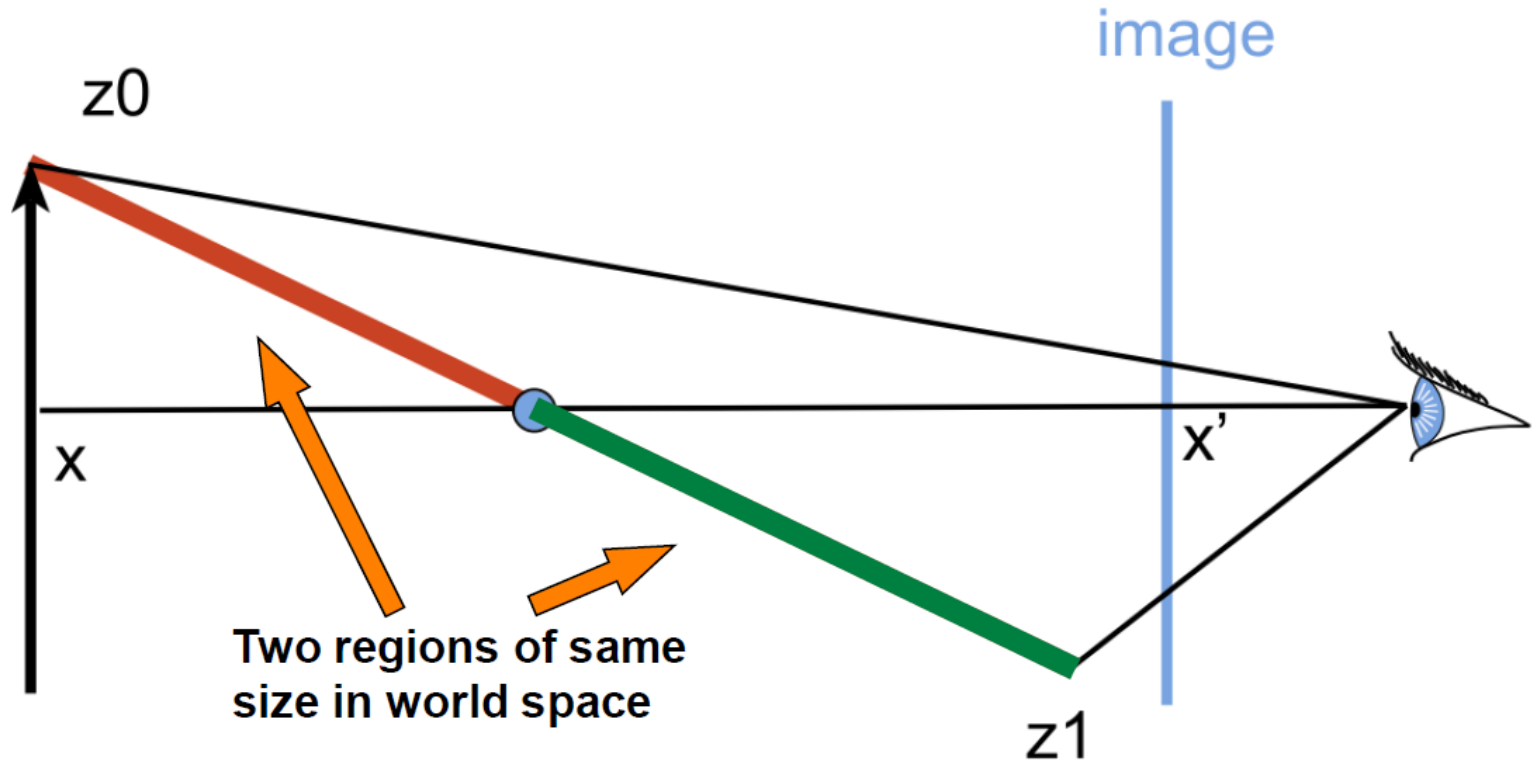
Интерполяция в пространстве экрана

Также необходимо интерполировать цвет, нормали, координаты текстуры и т. д. между вершинами

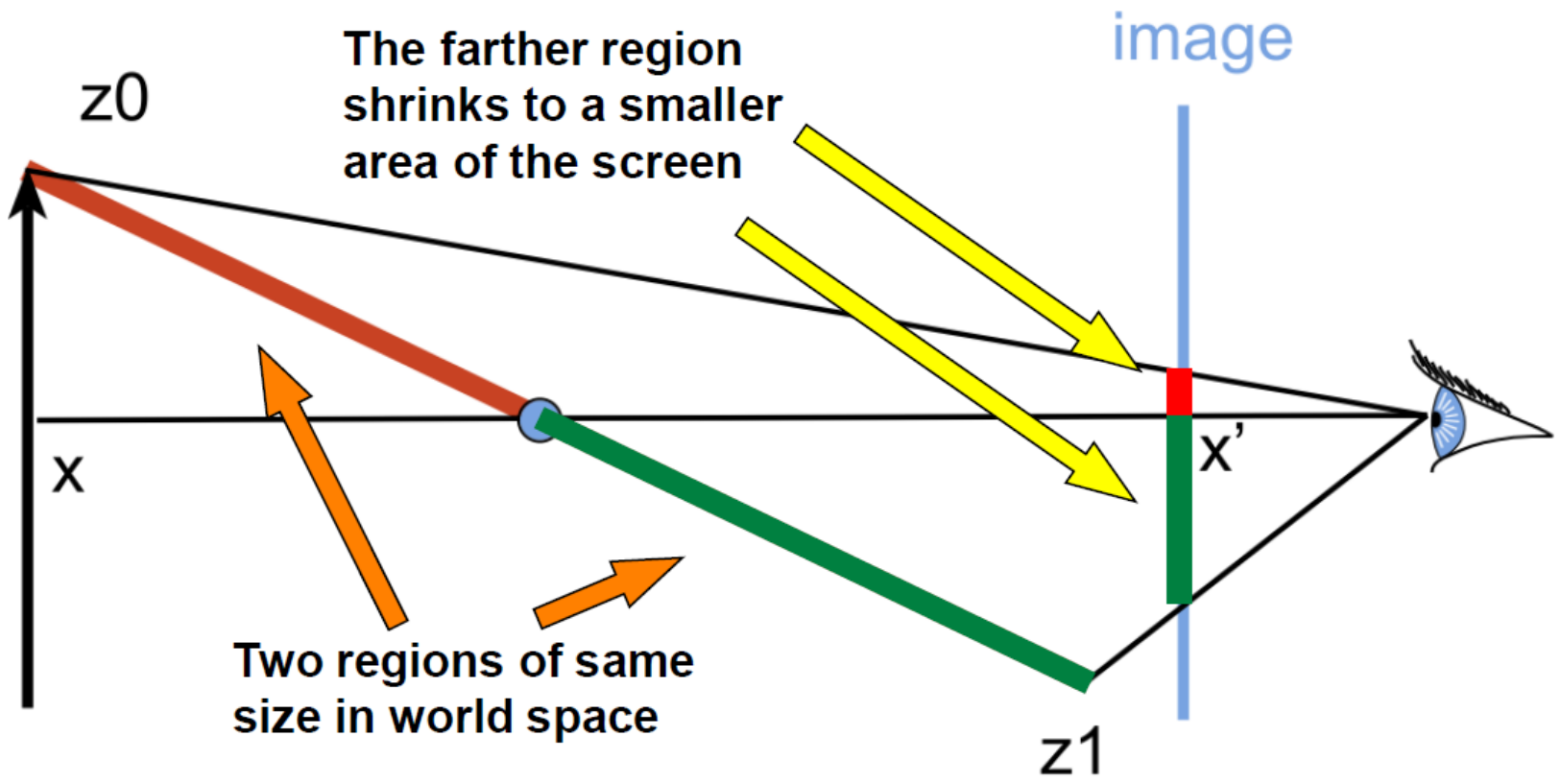
- Можно сделать используя барицентрические координаты
- Есть ли разница между интерполяцией в координатах экрана и интерполяцией в координатах мира?



Интерполяция в пространстве экрана



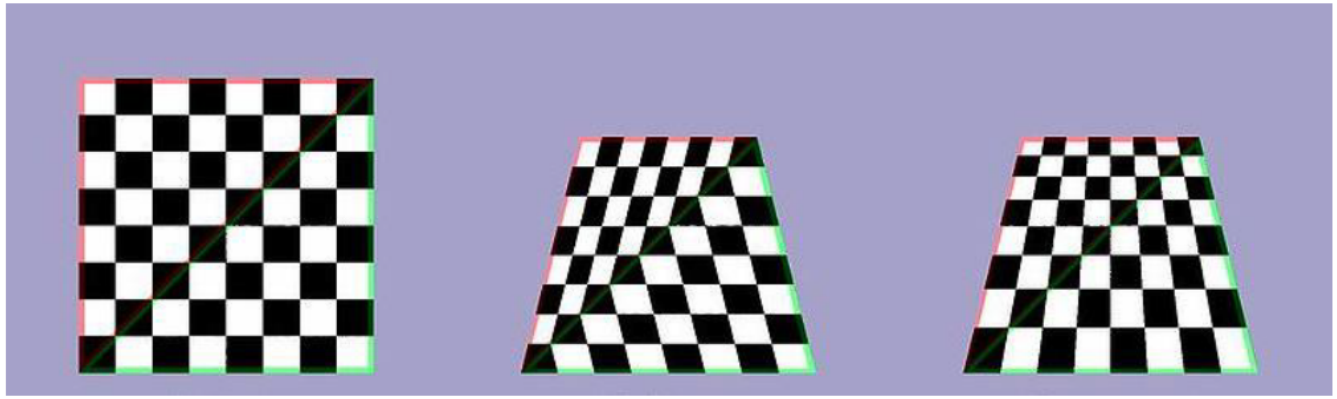
Интерполяция в пространстве экрана



Да, разница есть

Линейное изменение в мировом пространстве не приводит к линейному изменению экранного пространства из-за проекции

- Глядя на шахматную доску под крутым углом: все квадраты имеют одинаковый размер на плоскости, но не на экране



Спасибо за внимание!

Источники:

Ray Casting II

Line Rasterization

Graphics Pipeline & Rasterization

Graphics Pipeline & Rasterization II