Введение в компьютерную графику

Визуализация пространственных данных

Тимофей Самсонов

2020-10-20

## Компьютерная графика

Наука и искусство визуальной коммуникации посредством компьютерного дисплея и инструментом интерактивного взаимодействия с ним.

* Визуальный аспект: компьютер -> человек
* Интерактивный аспект: человек -> компьютер



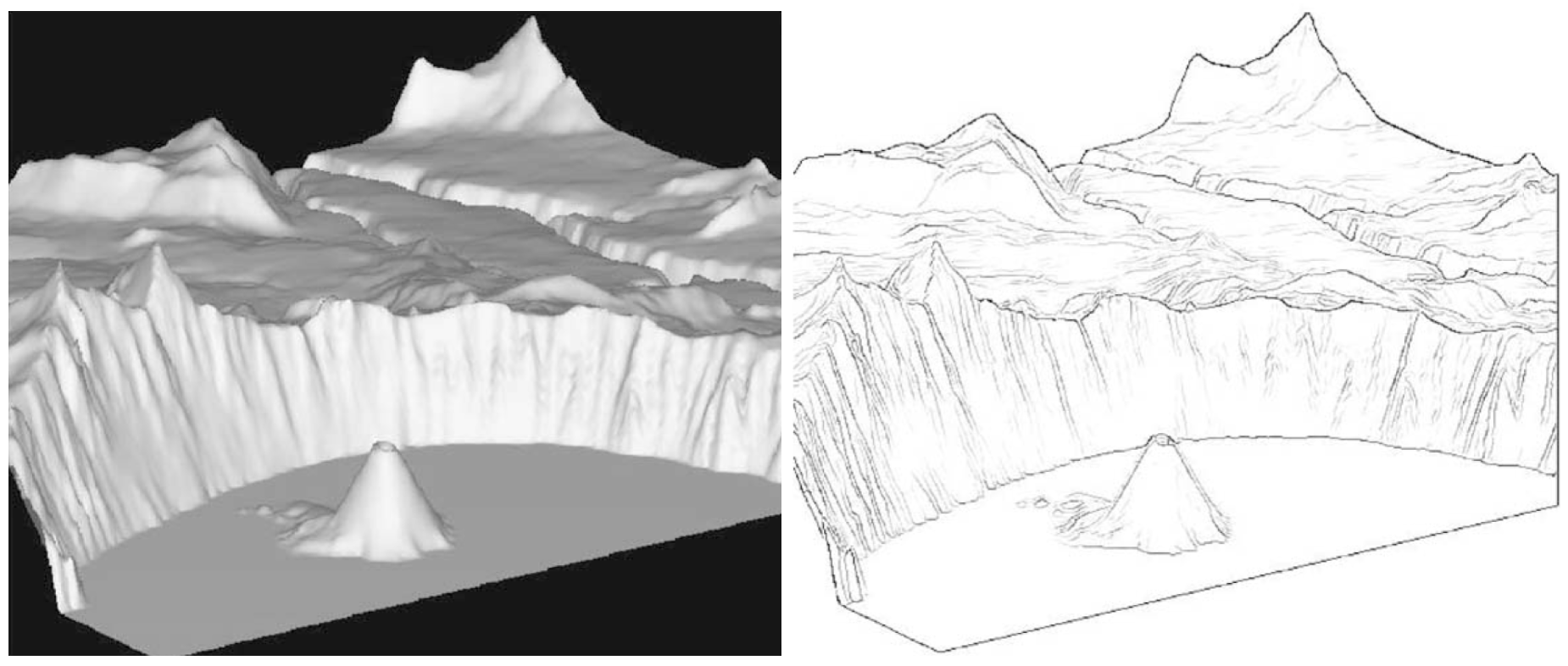
## Мультидисциплинарность

* физика
* математика
* человеческое зрение
* человеко-машинное взимодействие
* программная и аппаратная инженерия
* графический дизайн
* …

## Связь с компьютерным зрением

**Компьютерная графика**: реализация с определенного ракурса внешнего вида сцены, в которую погружены модели объектов под заданными условиями освещения.

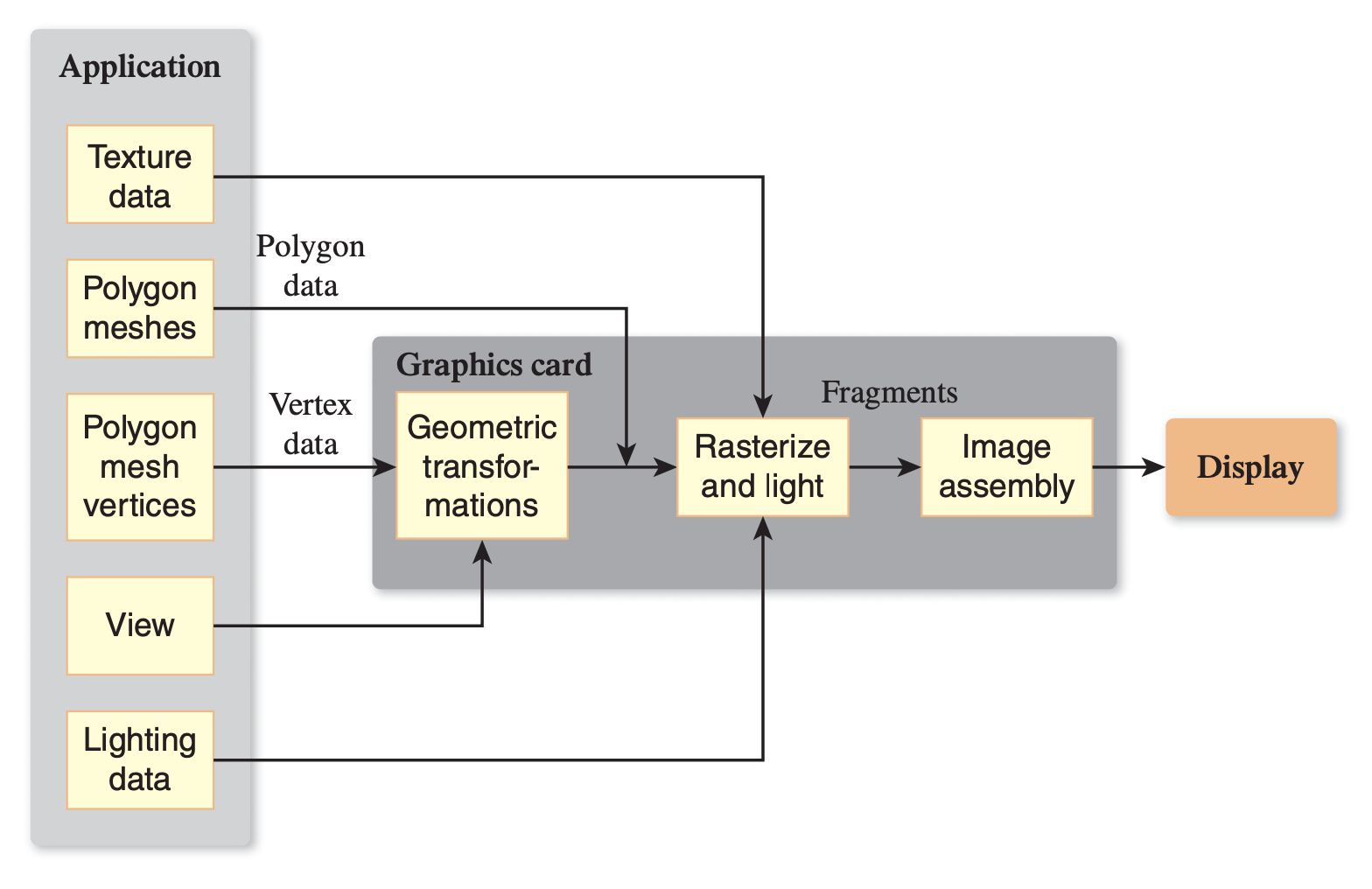
**Компьютерное зрение**: определение параметров сцены и погруженных в нее объектов при заданных допущениях.



:scale 75%

.small[*Lesage, Visvalingam, 2002*]

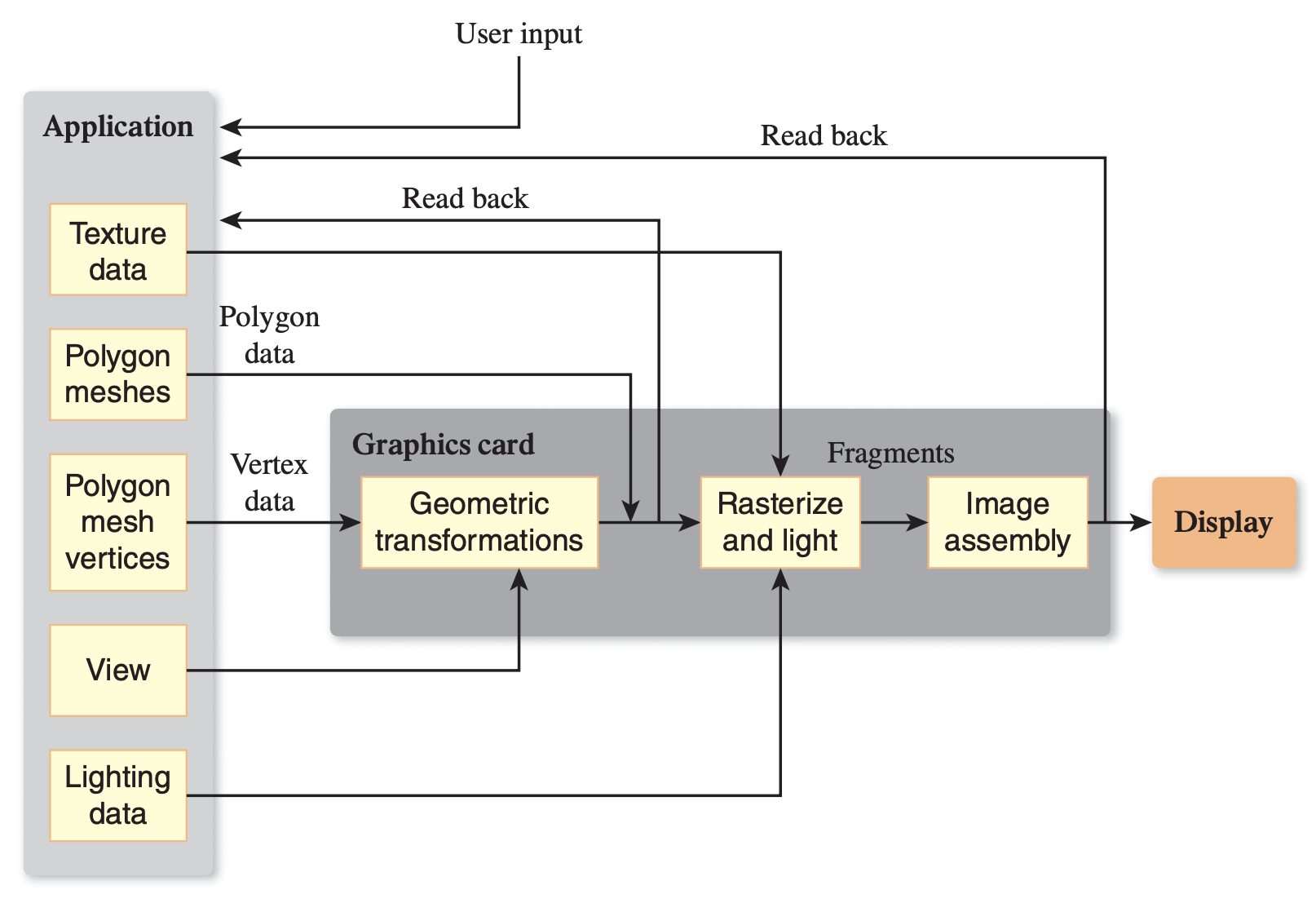
## Графический конвейер



:scale 85%

.small[*Hughes et al, 2013*]

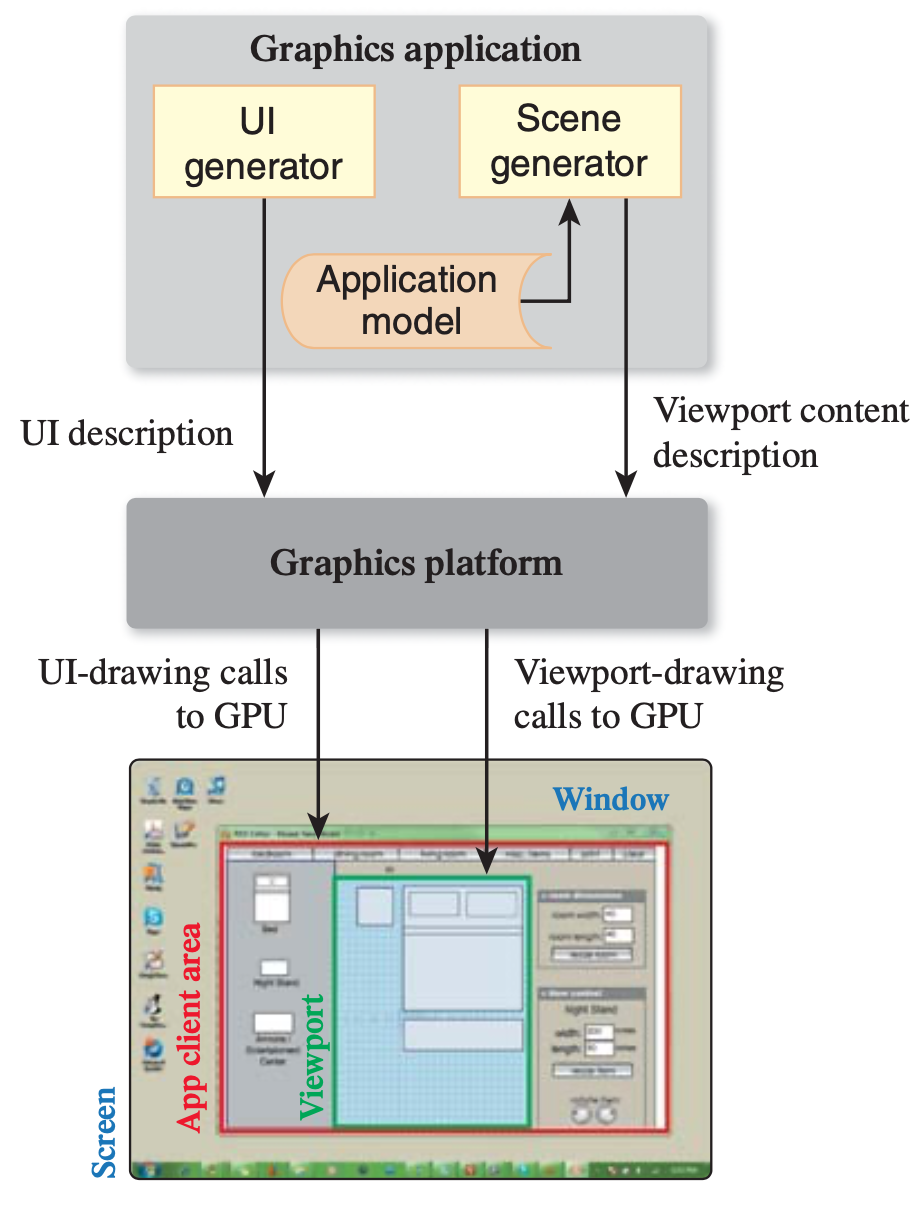
## Графический конвейер



:scale 80%

Взаимодействие с пользователем может оказывать влияние на последующее поведение графического приложения

## Графический конвейер 2D-приложения

.pull-left[ ]

.pull-right[ **Графическая платформа** отвечает за взаимодействие приложения и оборудования

**Модель приложения** (application model) представляет визуализируемые данные

**Клиентская область** (client area) определяет пространство, в котором приложение выполняет отрисовку.

**Порт просмотра** (viewport) определяет часть клиентской области, где генератор сцены выполняет отрисовку модели ]

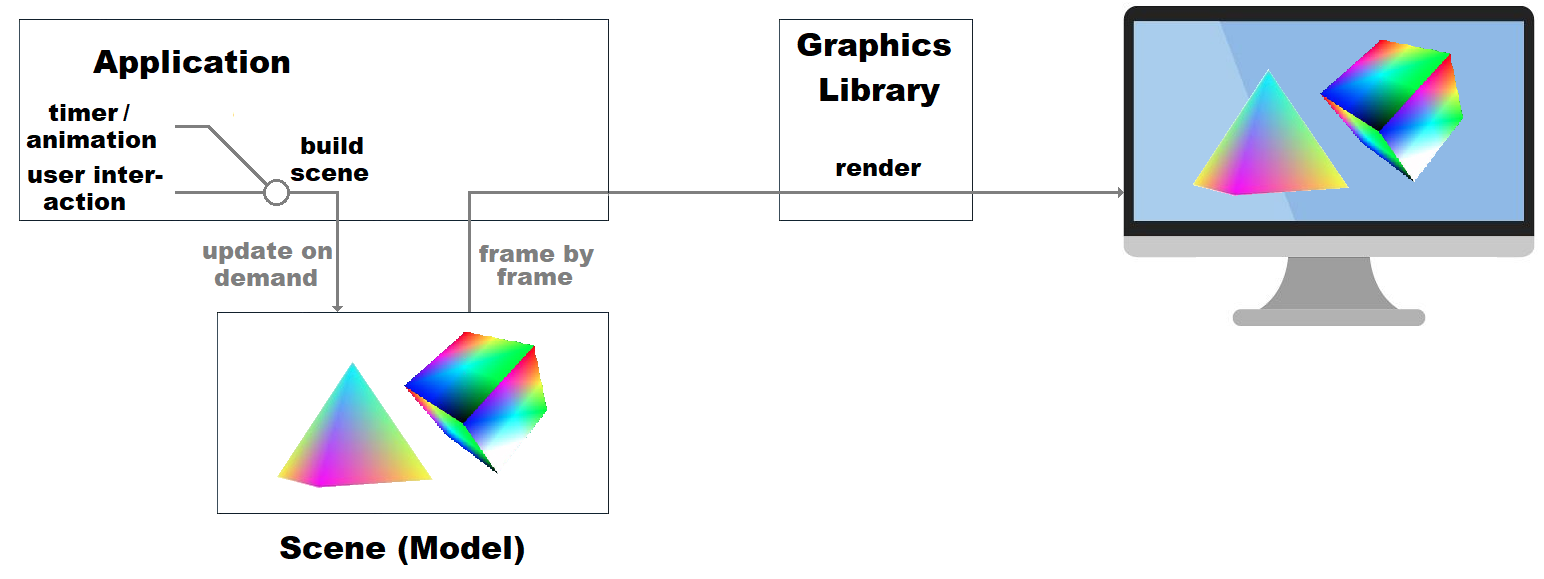
## Эволюция графической платформы

**1980-е — начало 1990-х** *(Apple QuickDraw, Microsoft GDI)*: отрисовка пикселей на прямоугольной канве в целочисленных координатах. Отрисовка .red[*примитивов*]: геометрических форм или битмапов (пиксельных карт). - Точка располагается в левом верхнем углу. - Каждый примитив заполняется путем вызова определенной функции (например, FillRectangle). - Размер объекта зависит от разрешения объекта. Более низкое разрешение (крупные пикселы) — больше размеры объектов.

**Середина 1990-х — н.вр.** Переход к координатам с плавающей точкой позволил решить проблему зависимости размера изображения от системы координат.

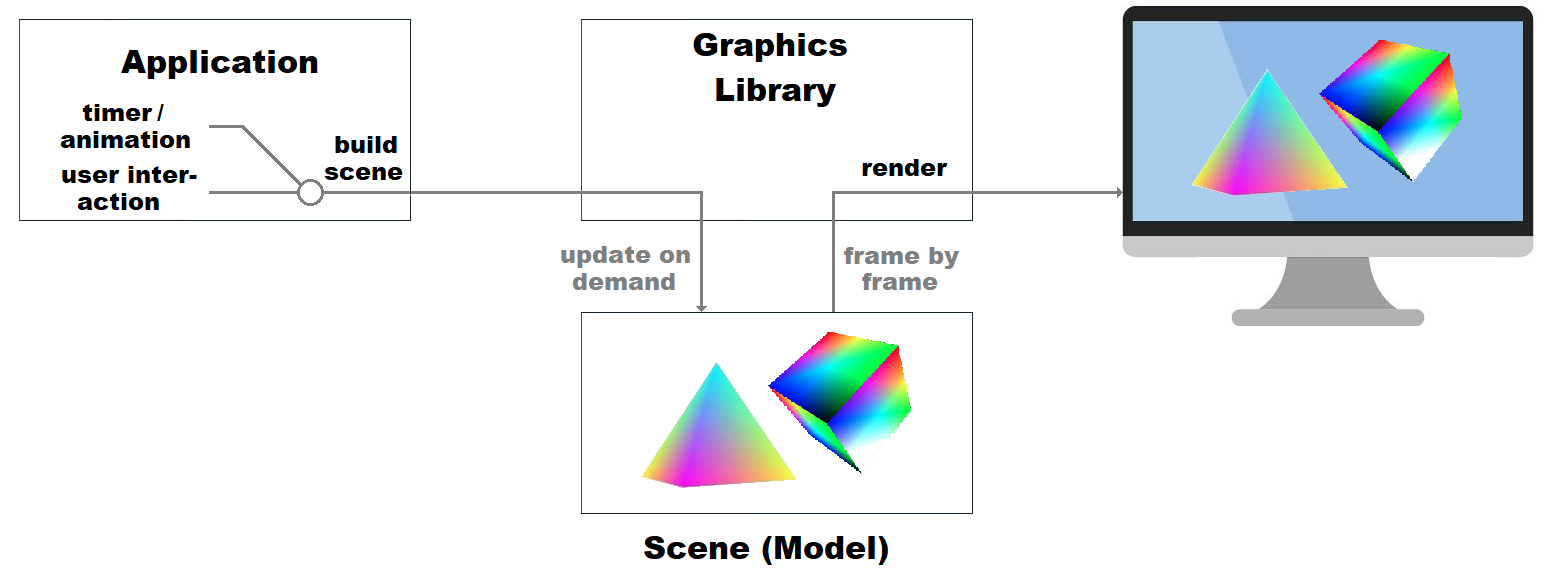
## Непосредственный режим

В .blue[**непосредственном режиме**] *(immediate mode)* клиент через графическую платформу обращается непосредственно к выводящему устройству. - Запись примитивов, переданных приложением, не выполняется. - При смене сцены приложение должно перерисовать ее целиком.



## Сохраненный режим

В .red[**сохраненном режиме**] *(retained mode)* графическая платформа хранит спецификацию сцены в виде специализированной базы данных, которая называется графом сцены *(scene graph)*. - Объекты могут добавляться в граф сцены последовательно. - Каждое изменение графа приводит к перерисовке сцены



## Процедурный и декларативный подходы

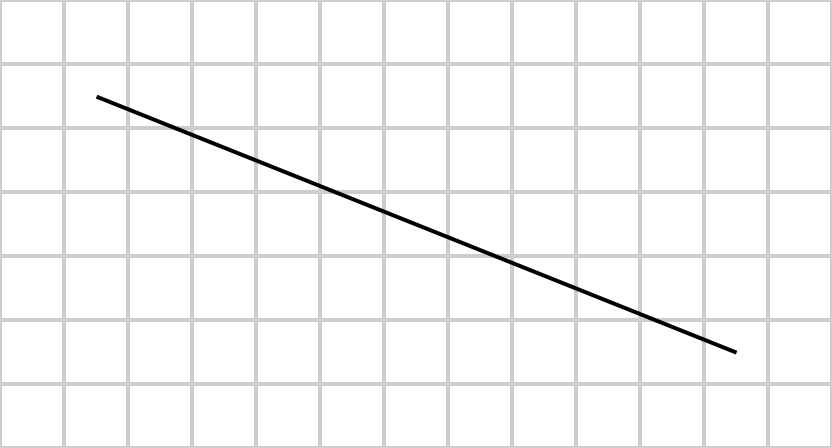
.blue[**Процедурный код**] пишется на императивном языке программирования - Описывает *действия* - Стандарт при взаимодействии с графической платформой - В геоинформационных пакетах применяется на низком уровне, невидимом для пользователя

.red[**Декларативные спецификации**] фиксируются на языке разметки - Описывают *результаты* - Вспомогательное средство в графических платформах - В геоинформационных пакетах применяется на высоком уровне при описании стилей карт (CSS, LYR, QML, SLD, YSLD)

## Процедурный код

Классический низкоуровневый пример — отрисовка линии.

draw\_line(x0, y0, x1, y1)



:scale 70%

.blue[**Алгоритм Брэзенхема**] .small[*Bresenham, J. E.* (1965). “Algorithm for computer control of a digital plotter” (PDF). **IBM Systems Journal**. 4 (1): 25–30. <doi:10.1147/sj.41.0025>.]

## Алгоритм Брезенхэма

Cистема экранных координат , где — столбец (увеличивается слева направо), — строка (увеличивается сверху вниз)

* — начальная точка отрезка в экранных координатах
* — конечная точка отрезка в экранных координатах

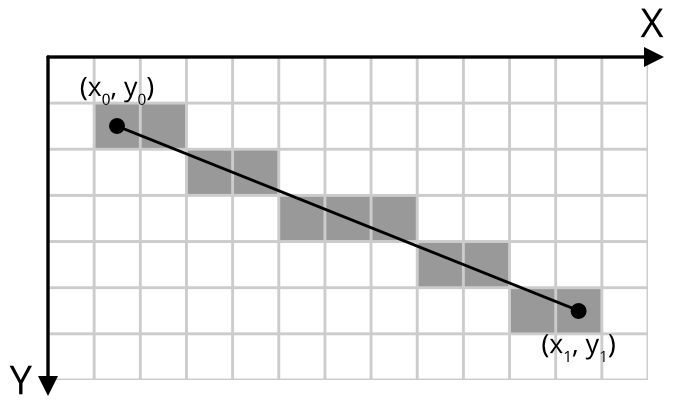
Уравнение отрезка:

Выразив , получаем:

$$y = \color{red}{\underbrace{\frac{y\_1 - y\_0}{x\_1-x\_0}}\_{\textbf{d (уклон)}}} (x - x\_0) + y\_0$$

—

## Алгоритм Брезенхэма

*Стандартно* рассматривается отрисовка линии, которая располагается в секторе В-ЮВ, т.е. идет относительно начальной точки вправо и полого вниз при соблюдении следующих условий: - ; - ; - . 

## Алгоритм Брезенхэма

Для сектора В-ЮВ алгоритм опирается на координаты :

.pull-left[**Вычисляем заранее**: - . - . - $\color{blue}{d = dy / dx}$; - $\color{blue}{y = y\_0}$; - ]

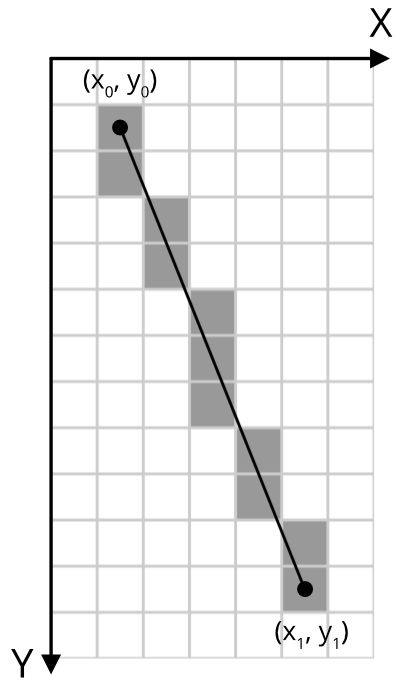
.pull-right[**Для $\color{blue}{x = x\_0, ... x\_1}$, выполняем**: 1. Рисуем пиксель . 2. Обновляем ошибку . 3. Если , то: - $\color{blue}{y = y + \texttt{sign}(dy)}$, - .]

Использование функции позволяет применять алгоритм также и для сектора В-СВ (только будет уменьшаться).

Обозначим этот случай как draw\_line\_byX(x0, y0, x1, y1)

## Алгоритм Брезенхэма

.pull-left[*Альтернативно* рассматривается отрисовка линии, которая располагается в секторе Ю-ЮВ, т.е. идет относительно начальной точки вправо и круто вниз при соблюдении следующих условий: - ; - ; - .]

.pull-right[]

## Алгоритм Брезенхэма

Для сектора Ю-ЮВ алгоритм опирается на координаты :

.pull-left[**Вычисляем заранее**: - . - . - $\color{red}{d = dx / dy}$; - $\color{red}{x = x\_0}$; - ]

.pull-right[**Для $\color{red}{y = y\_0, ... y\_1}$, выполняем**: 1. Рисуем пиксель . 2. Обновляем ошибку . 3. Если , то: - $\color{red}{x = x + \texttt{sign}(dx)}$, - .]

Использование функции позволяет применять алгоритм также и для сектора Ю-ЮЗ (только будет уменьшаться).

Обозначим этот случай как draw\_line\_byY(x0, y0, x1, y1)

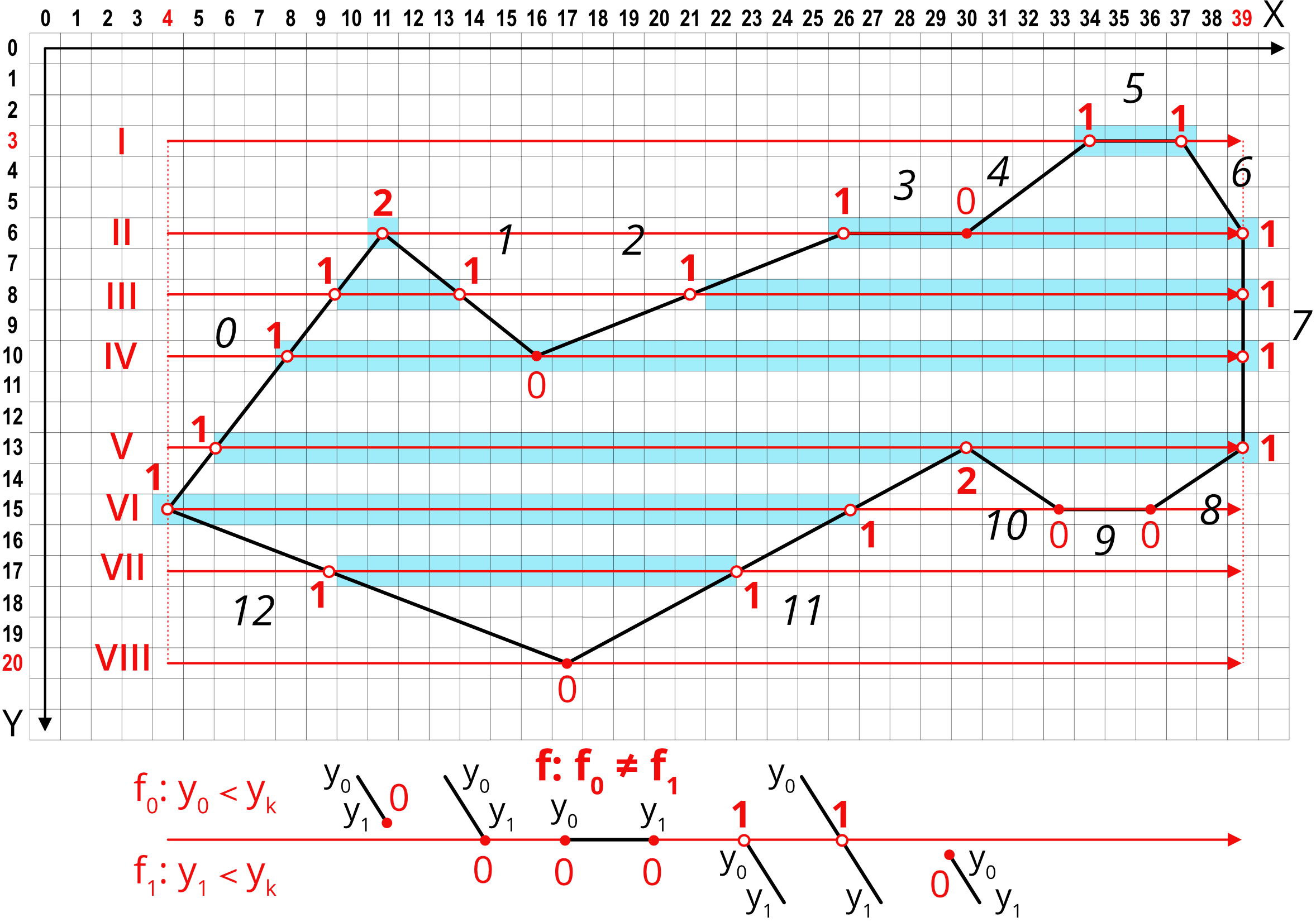
## Алгоритм Брезенхэма

Для оставшихся направлений построение линий реализуется путем перестановки местами начальной и конечной точки. Код результирующей функции на языке **Python**:

def plot\_line(x0, y0, x1, y1):  
 if abs(y1 - y0) < abs(x1 - x0): # пологая линия  
 if x0 > x1:  
 plot\_line\_byX(x1, y1, x0, y0)  
 else:  
 plot\_line\_byX(x0, y0, x1, y1)  
 else:  
 if y0 > y1: # крутая линия  
 plot\_line\_byY(x1, y1, x0, y0)  
 else:  
 plot\_line\_byY(x0, y0, x1, y1)

где abs() — функция вычисления модуля.

## Заливка полигона



:scale 95%

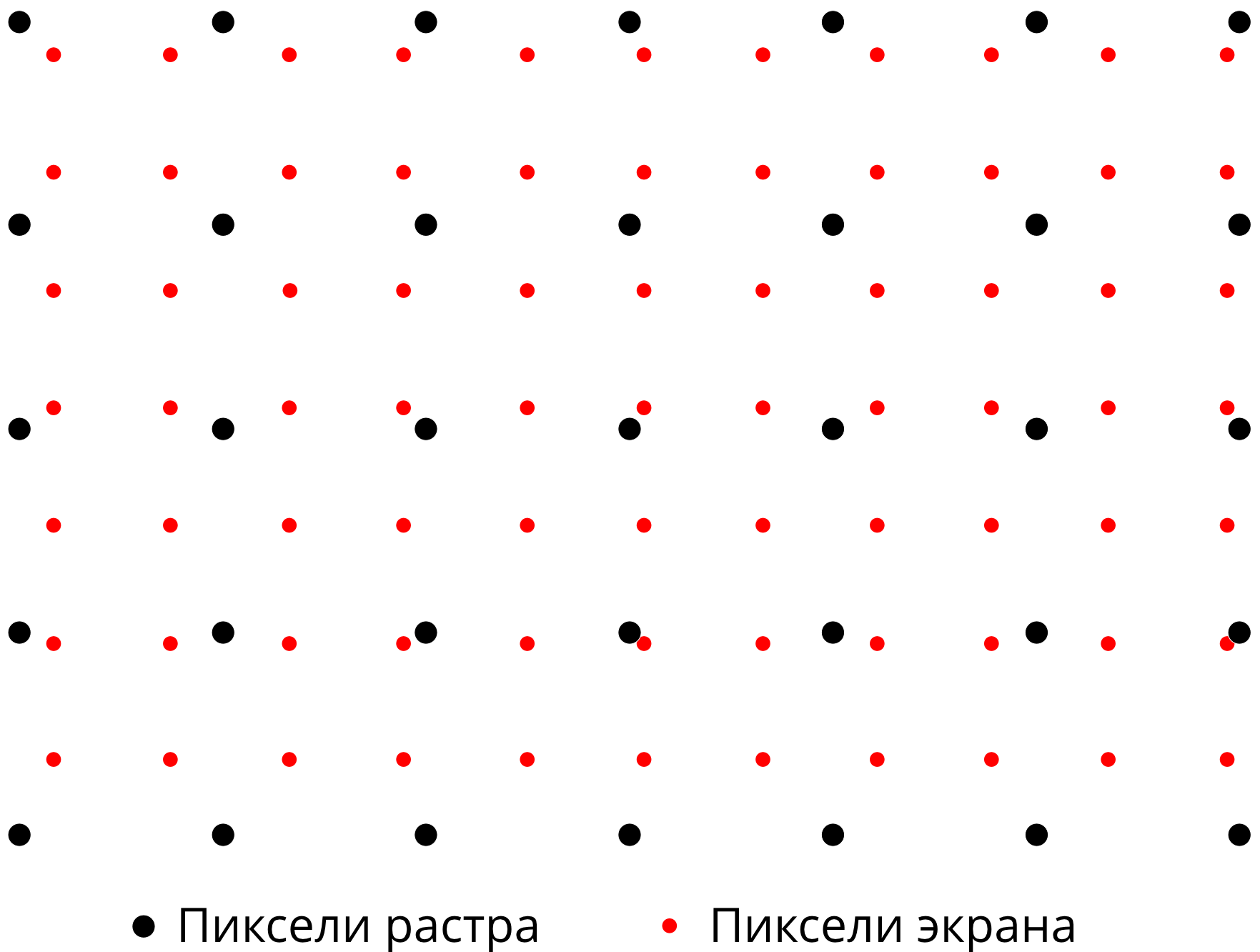
## Заливка полигона

.blue[**Алгоритм сканирующей линии:**]

Для каждого необходимо получить упорядоченное по возрастанию множество абсцисс точек его пересечения с исходными линиями: 1. Для каждой линии : - вычислить флаг пересечения ; - если , то - вычислить ; - добавить в множество . 2. Если не пусто, то упорядочить его по возрастанию и выполнить закрашивание между его элементами в порядке:

## Передискретизация (resampling)

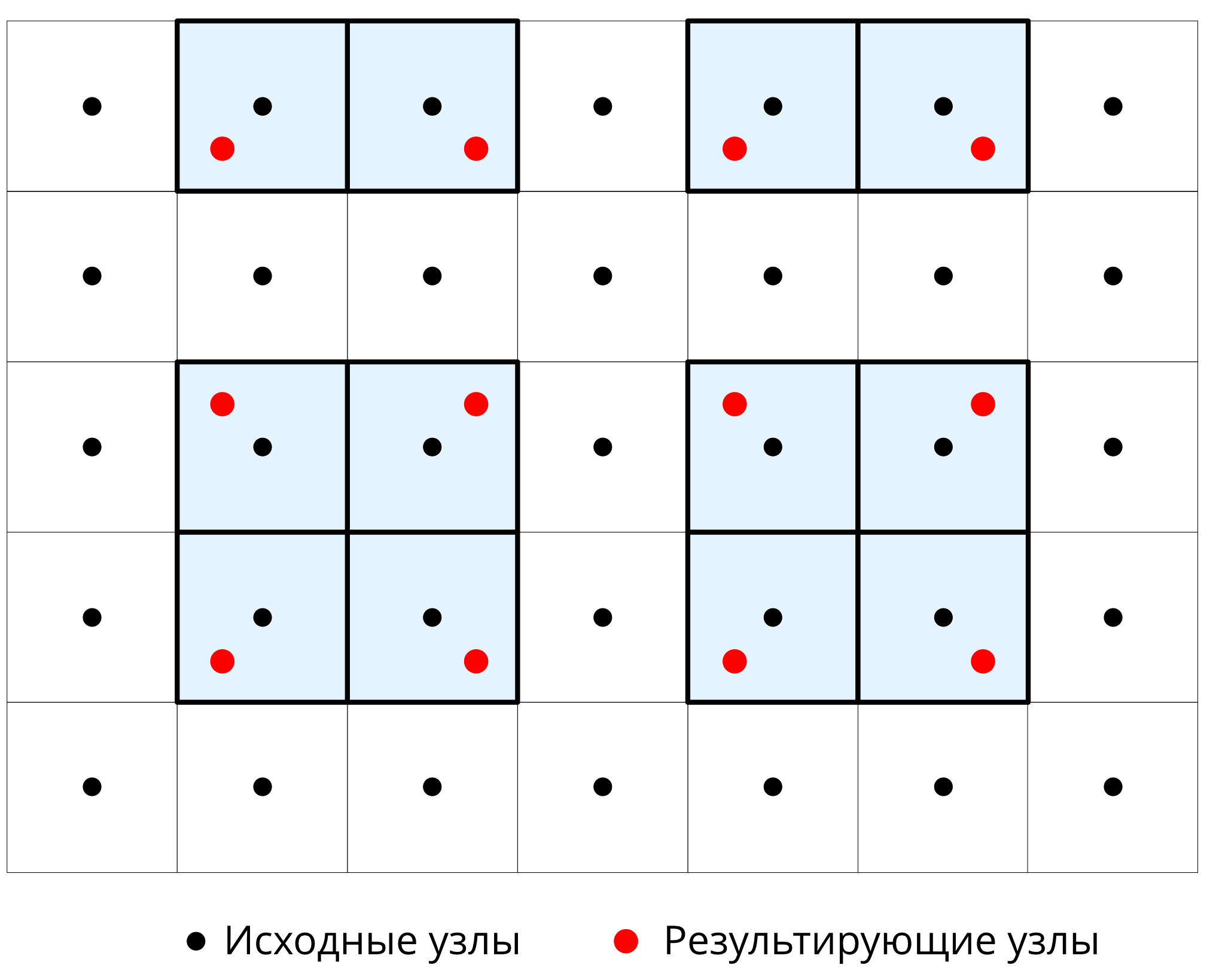
Передискретизация необходима при визуализации растрового изображения на растровом экране. Как правило, разрешение этих растров не совпадает.



:scale 65%

## Метод ближайшего соседа

Используется значение ближайшего пикселя:



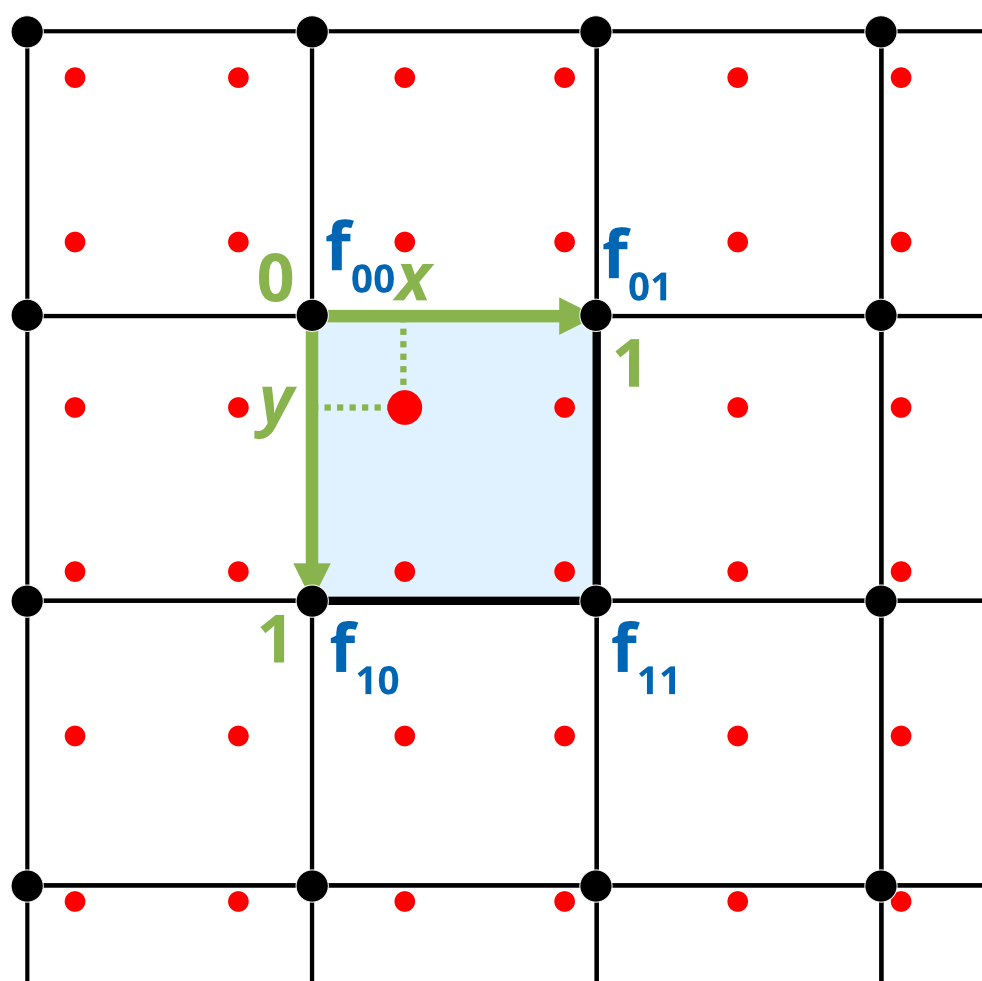
:scale 70%

## Метод билинейной интерполяции

Восстанавливается поверхность в ячейке из 4 узлов:

## :scale 75%

## Метод билинейной интерполяции

.left-40[ ]

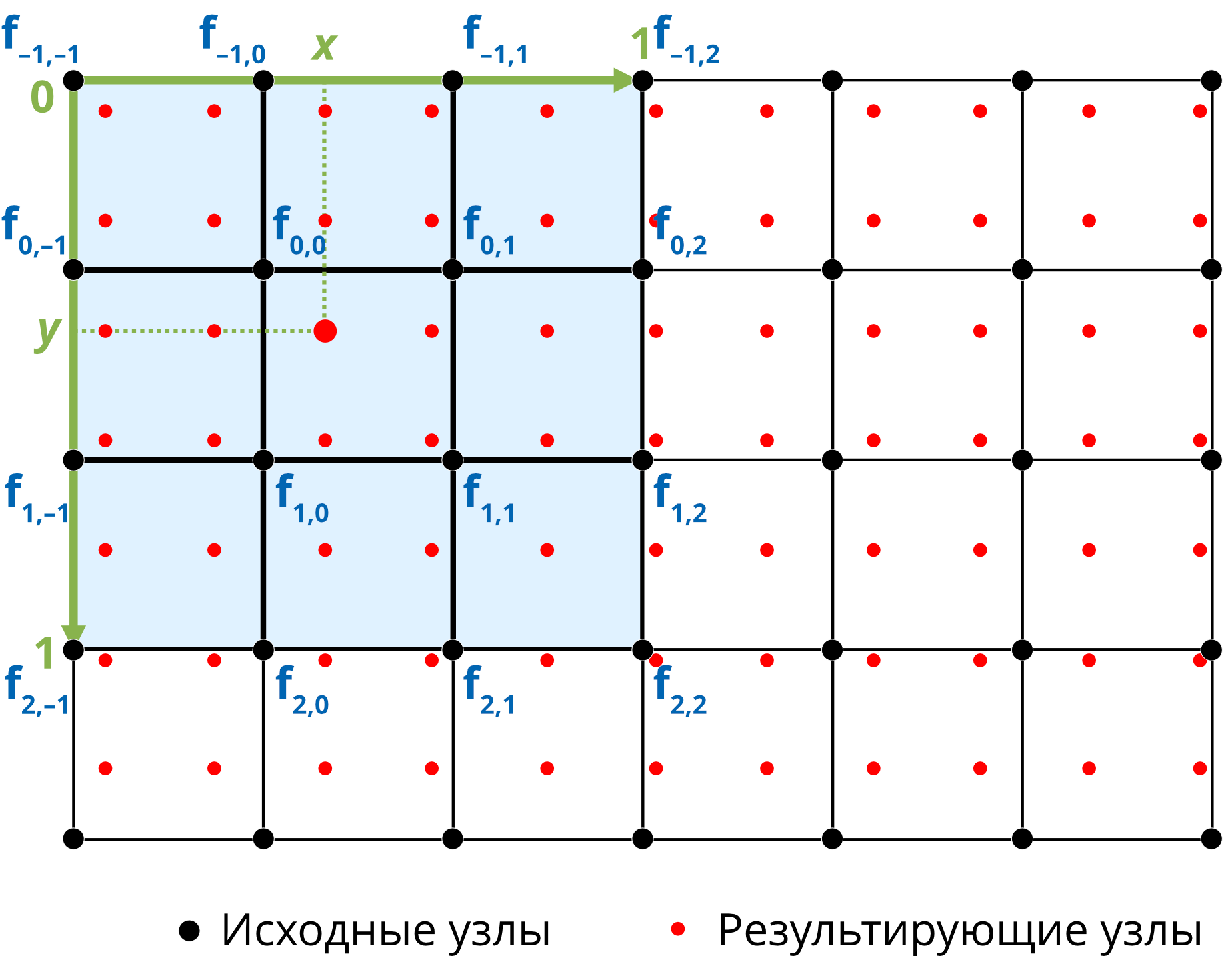
.right-60[ Коэффициенты определяются по 4 точкам:

]

Координаты и меняются в диапазоне от до в пределах ячейки.

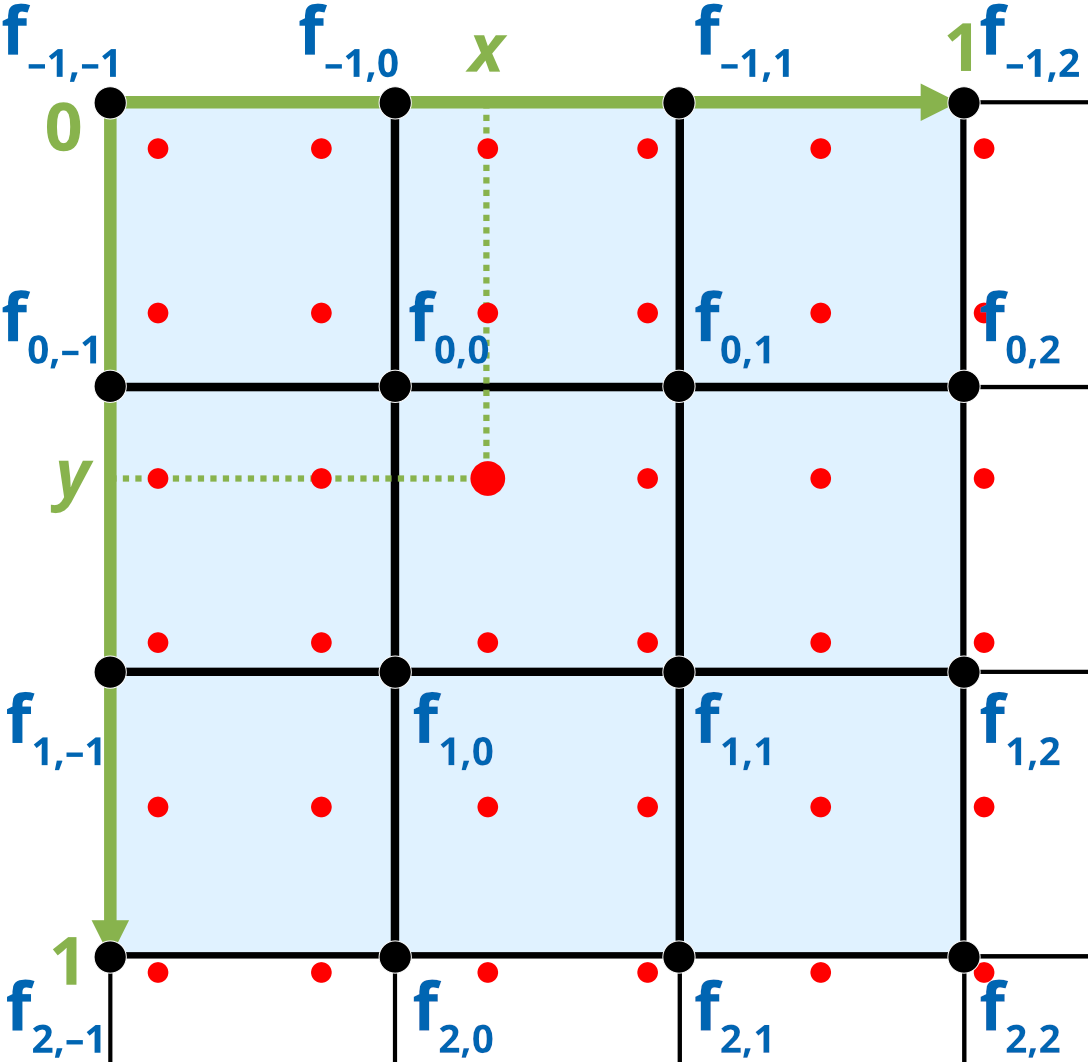
## Метод бикубической интерполяции

Восстанавливается поверхность в 9 ячейках, окружающих узел:



:scale 75%

## Метод бикубической интерполяции

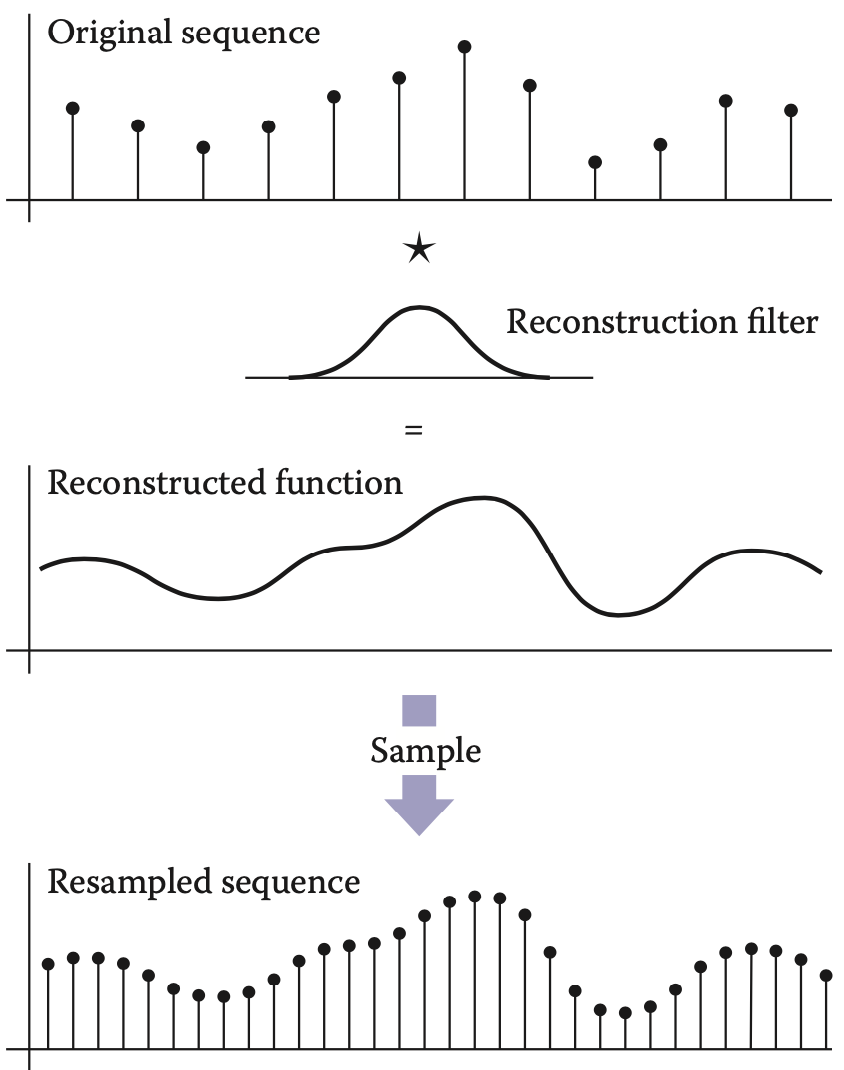
.left-40[ ]

.right-60[ Коэффициенты определяются по 16 точкам:

]

Координаты и меняются в диапазоне от до в пределах окрестности ячеек.

## Передискретизация и фильтрация

.pull-left[  .small[*Marschner et al., 2016*]] .pull-right[ Передискретизацию можно также рассматривать как двухстадийный процесс: - Построение гладкой непрерывной функции путем применения реконструкционного фильтра

* Выборка (семплирование) значений функции в локациях, определенных узлами новой сетки ]

## Передискретизация и фильтрация

Псевдокод одномерной передискретизации:

fun resample(sequence a, float x0, float dx, int n, filter f)  
 b = list(length = n)  
 for i = 0 to n-1 do  
 b[i] = reconstruct(a, f, x0 + i \* dx)  
 return b

Параметр x0 отвечает за расположение первого элемента выборки в системе координат исходной последовательности.

Например, если этот элемент расположен посередине между 0-м и 1-м элементом исходной последовательности, то x0 = 0.5

## Дискретно-непрерывная свёртка

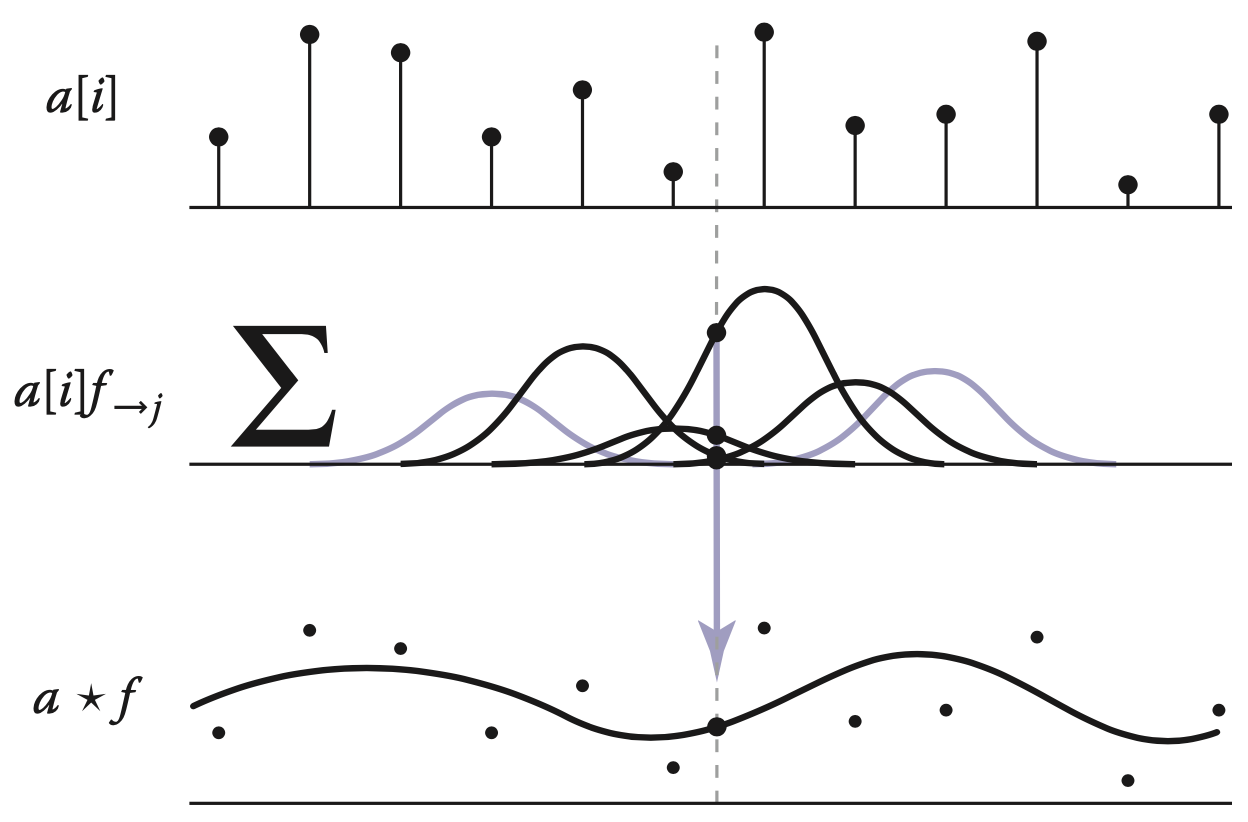
Восстановление непрерывной поверхности происходит путем взвешенного осреднения исходных значений. Веса определяются свёрточным ядром: .left-60[ 

.small[*Marschner et al., 2016*] ] .right-40[ Свертка есть результат произведения функций:

Для ядра ограниченного радиуса :

]

## Дискретно-непрерывная свёртка

.left-60[ 

.small[*Marschner et al., 2016*] ] .right-40[ Восстановленную функцию можно получить путем суммирования произведений фильтра и исходного значения в точках :

]

## Дискретно-непрерывная свёртка

Псевдокод:

fun reconstruct(sequence a, filter f , real x)  
 s=0  
 r = f.radius  
 for i = ⌈x − r⌉ to ⌊x + r⌋ do  
 s = s + a[i]f (x − i)   
 return s

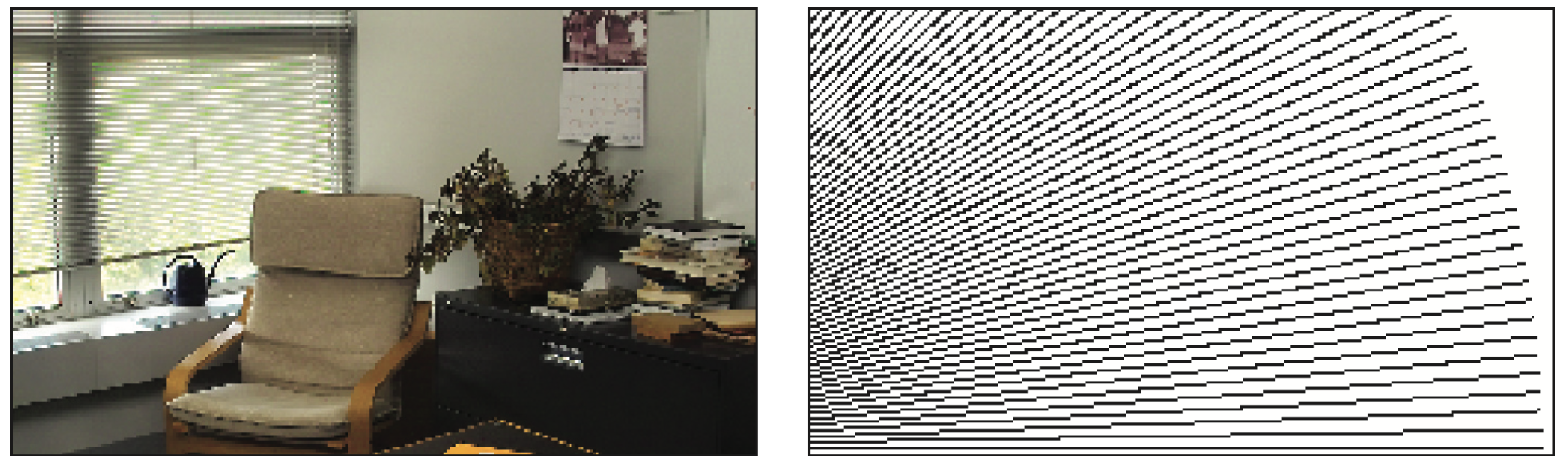
## Популярные фильтры



:scale 70%

## Передискретизация и фильтрация

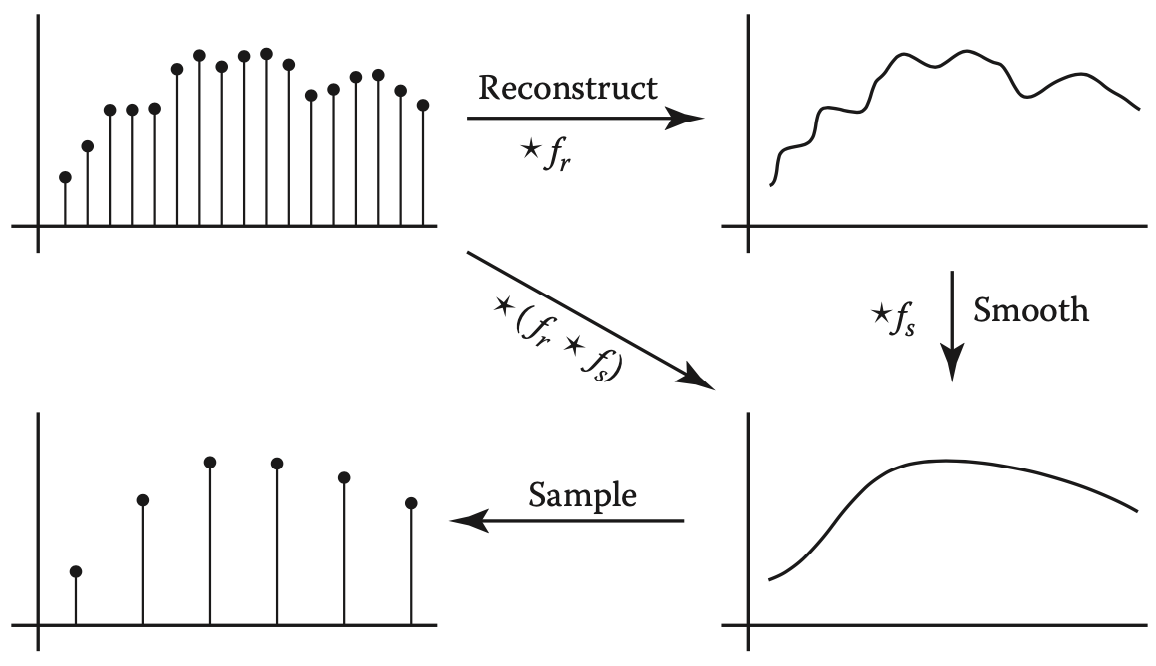
Проблема передискретизации при даунсемплинге (понижении разрешения) заключается в появлении артефактов таких как муар (слева) и зубчатость (справа):

 .small[*Marschner et al., 2016*]

Поэтому при понижении разрешения помимо реконструкционного фильтра необходимо использовать также сглаживающий.

## Передискретизационный фильтр

Поскольку свёрточные фильтры можно перемножать, для комбинирования реконструкции и сглаживания достаточно провести фильтрацию один раз (диагональная стрелка):

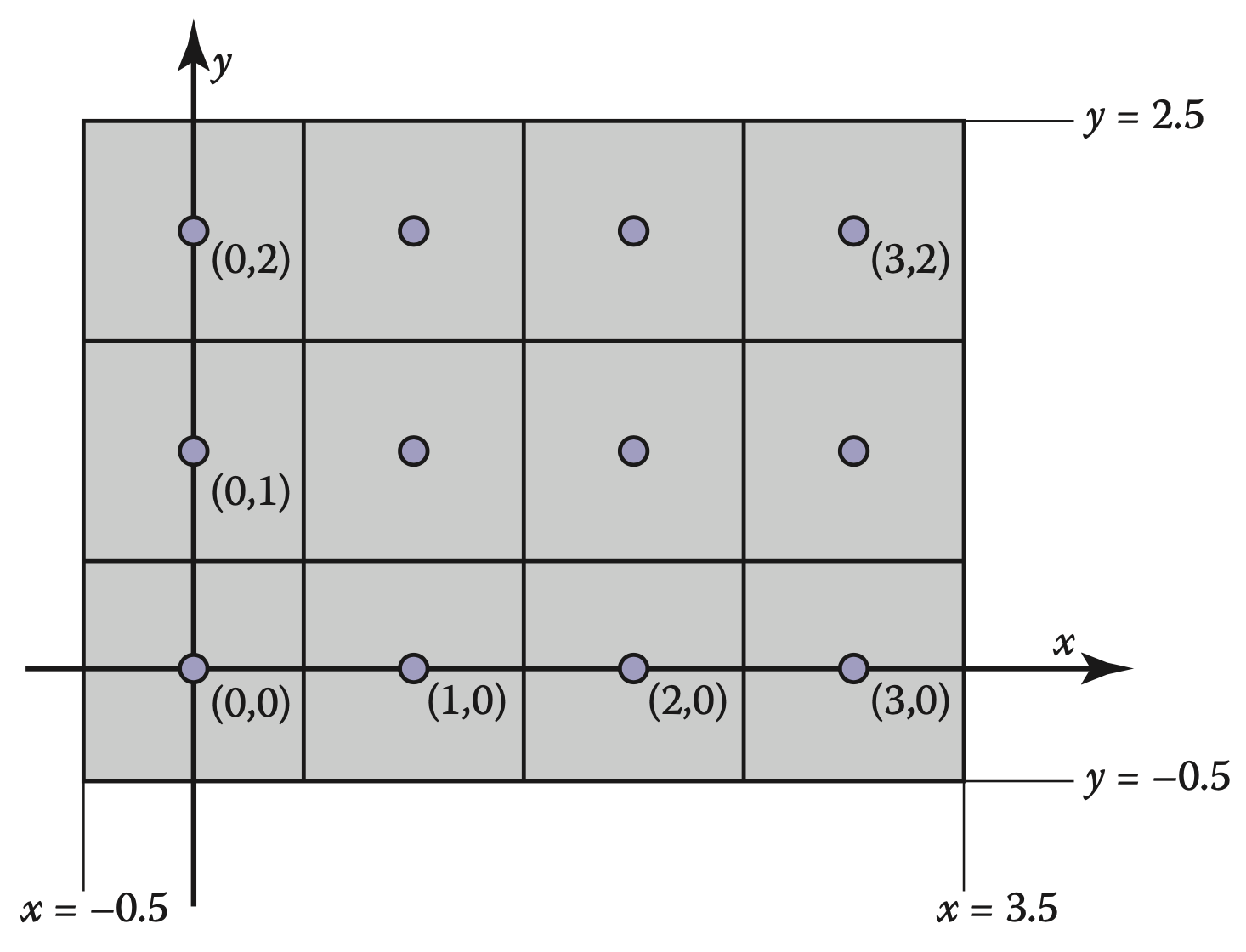


:scale 65%

.small[*Marschner et al., 2016*]

## Передискретизация и фильтрация

Прямоугольный домен изображения размером :



.small[*Marschner et al., 2016*]

## Передискретизация и фильтрация

Псевдокод одномерной сглаживающей передискретизации:

fun resample(sequence a, float xmin, float xmax, int n, filter f)  
 b = list(length = n)  
 r = f.radius  
 dx = (xmax - xmin)/n  
 x0 = xmin + dx/2  
 for i = 0 to n-1 do  
 s = 0  
 x = x0 + i \* dx  
 for j = ceil(x-r) to floor(x+r) do  
 s = s + a[j] \* f(x - j)  
 b[i] = s  
 return b

Параметры xmin и xmax обозначают минимальную и максимальную растровую координату по измерению X. Для полного изображения xmin = -0.5, xmax = nx - 0.5.

## Фильтрация на границе растра

Если фильтр выходит за границу растра, возможно три варианта:

* Не обрабатывать такие пикселы. В этом случае они будут заполнены нулями.
* Обрезать все индексы до минимально и максимально возможной координаты. В этом случае при индексации будет возвращаться , а при индексации будет возвращаться .
* Модифицировать фильтра для граничных ячеек. Например, заменить бикубическую интерполяцию на билинейную. Если используется сверточный фильтр, то уменьшение числа его элементов требует ренормализации значений (разделить на сумму элементов, перекрывающих фильтруемый растр).

## Метод бикубической интерполяции

Keys (1981) показал, что бикубическую реконструкцию можно реализовать путем свёртки с использованием следующего фильтра:

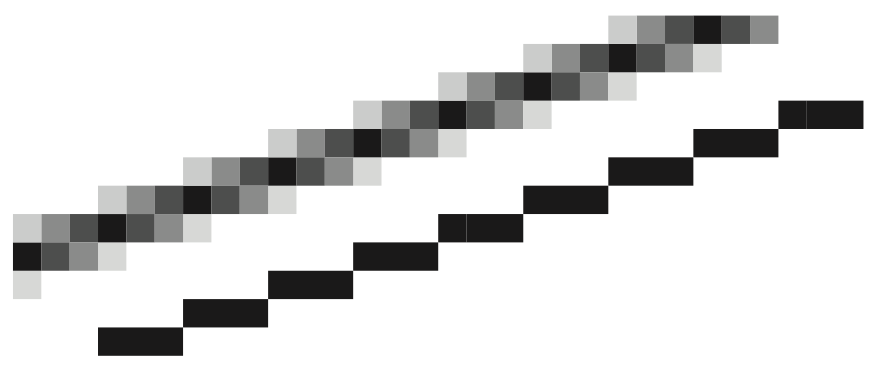
где или .

*R. Keys.* Cubic convolution interpolation for digital image processing // **IEEE Transactions on Acoustics, Speech, and Signal Processing** — 1981. — Vol. 29, no. 6. — P. 1153—1160. — <doi:10.1109/TASSP.1981.1163711>

## Сглаживание (antialiasing)

При крупном размере пиксела выводящего устройства ступенчатость линий и границ полигонов становится заметной.

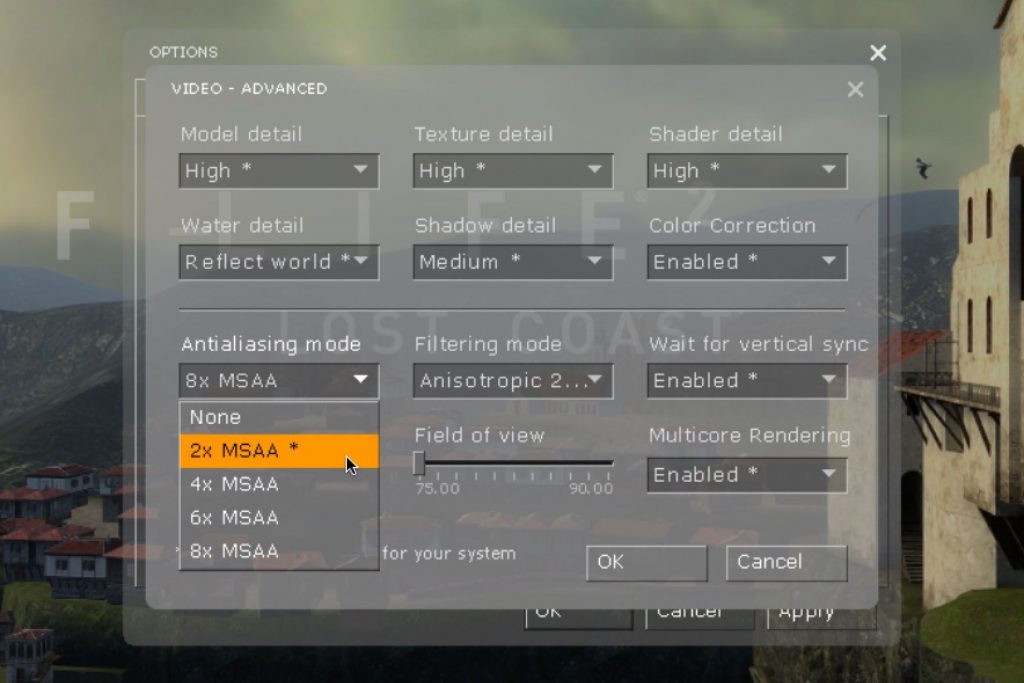
Для устранения эффекта ступенчатости используется **сглаживание** (*antialiasing*).



:scale 75%

Вверху — со сглаживанием, внизу — без (.small[*Marschner et al., 2016*])

## Сглаживание (antialiasing)



:scale 80%

.small[<https://www.neogamr.net/what-is-anti-aliasing/>]

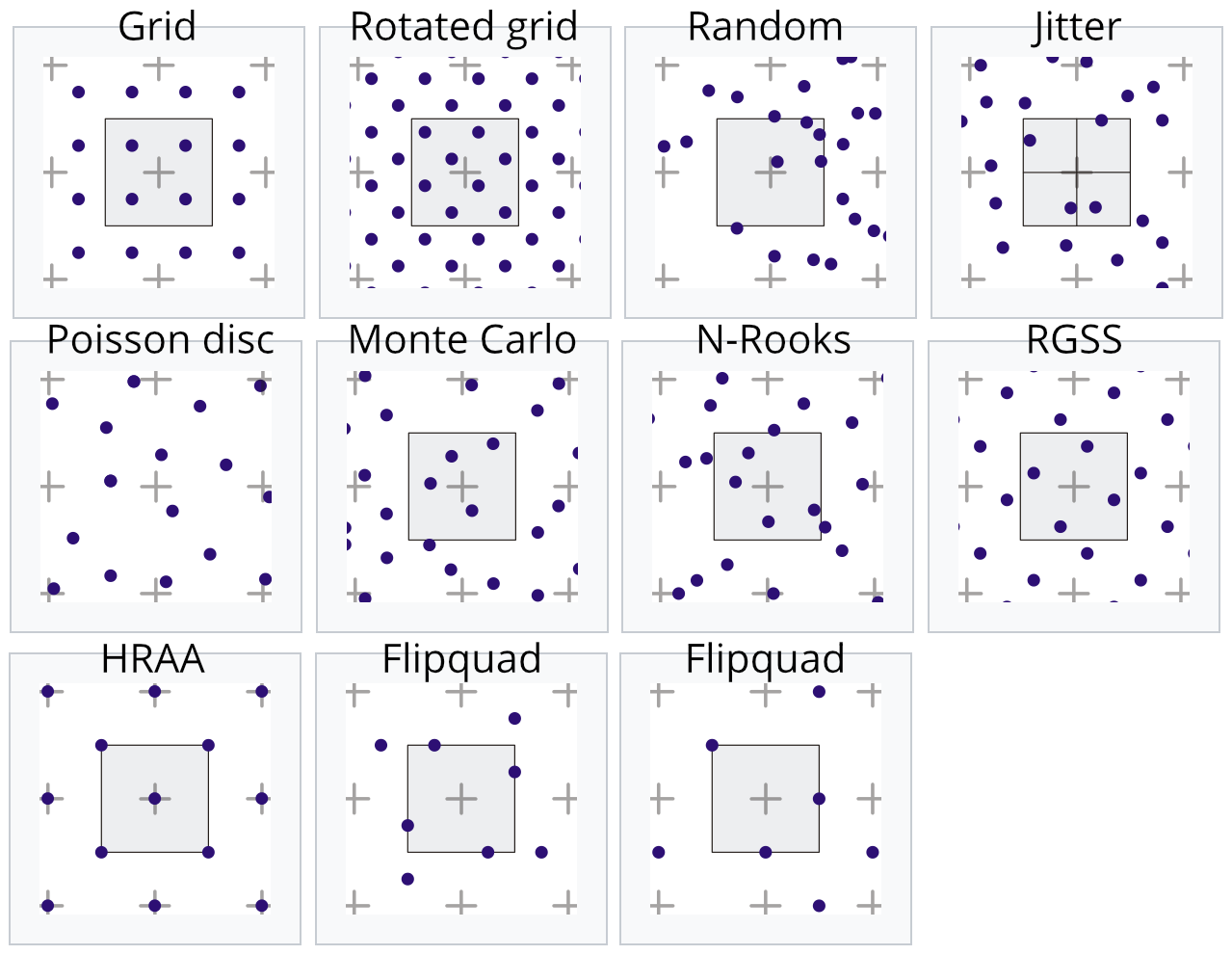
## Суперсемплинг (antialiasing)

Пусть результирующее растровое изображение имеет размеры .

1. Создается буферное изображение размером , где — коэффициент сглаживания (положительное натуральное число).
2. Объекты отрисовываются в .
3. Производится передискретизация с использованием блочной фильтрации (*box filter*). Значение результирующего пиксела равняется среднему арифметическому значений , попадающих в его пределы:

где — выборочное значение (сэмпл), взятое с .

## Стратегии семплирования

 .small[<https://en.wikipedia.org/wiki/Supersampling>]

## Аффинное преобразование

В системах двумерной визуализации данных фундаментальную роль играют *аффинные преобразования*.

**Аффинное преобразование** есть преобразование вида:

где — *обратимая матрица* (квадратная матрица, определитель которой отличен от нуля) и .

С помощью аффинных преобразований осуществляется **навигация** по карте: *масштабирование, перемещение, поворот изображения*.

Частные случаи этих операций могут быть связаны с центрированием относительно объекта, а также “вписыванием” объекта или множества объектов в пределы экрана.