Лабораторная работа 4  
Документация

## Цели работы:

1. Освоение алгоритмов растеризации: Изучить и реализовать различные алгоритмы рисования линий и окружностей, такие как пошаговый алгоритм, алгоритм ЦДА (DDA) и алгоритм Брезенхема.
2. Создание динамической сетки: Разработать систему координат с возможностью масштабирования и панорамирования, обеспечивающую корректное отображение сетки и подписей.
3. Интерактивный интерфейс: Реализовать пользовательский интерфейс с возможностью взаимодействия через мышь (панорамирование и зумирование) и элементы управления (кнопки и ползунки).
4. Оптимизация визуализации: Обеспечить точное совпадение рисуемых объектов с линиями сетки, а также плавное изменение масштаба без потери читаемости подписей, вычислять время реализации.

## Инструменты разработки:

HTML — для создания структуры интерфейса.

CSS — для стилизации элементов.

JavaScript — для обработки пользовательского ввода, реализации логики алгоритмов растеризации и управления графическими элементами на Canvas

GitHub Pages — для размещения веб-приложения в общем доступе.

FileReader API — для чтения содержимого файлов в браузере.

PyCharm — среда разработки.

## Использованные библиотеки и технологии:

1. HTML — разметка веб-страницы для создания элементов интерфейса:

<canvas> — для отрисовки графических элементов и сетки координат.

<input> — для ввода координат начальной и конечной точек линии, а также радиуса окружности.

<button> — для выбора алгоритмов рисования и генерации случайных координат.

<div> — для организации структуры страницы и размещения различных секций интерфейса.

<label> — для обозначения полей ввода и других элементов управления.

<span> — для отображения текущего масштаба.

2. CSS — для стилизации интерфейса:

-Использование внешнего CSS-файла для оформления элементов интерфейса, таких как кнопки, канвас и панели управления.

-Создание адаптивных стилей для обеспечения корректного отображения на различных устройствах и экранах.

-Применение эффектов наведения и активного состояния для улучшения пользовательского опыта.

3. JavaScript — логика обработки взаимодействия и реализации алгоритмов:

-Canvas API — для отрисовки графических элементов, сетки координат, линий и окружностей.

-Event Listeners — для обработки событий пользователя, таких как клики, перетаскивания мыши и прокрутка колесика.

-Math Functions — для выполнения математических вычислений, необходимых при реализации алгоритмов рисования.

-Performance API — для измерения времени выполнения алгоритмов и отображения результатов пользователю.

4. GitHub Pages — для размещения приложения:

- Публикация HTML, CSS и JS файлов, что позволяет сделать приложение доступным для пользователей.

## Функционал приложения:

1. Панорамирование:

Позволяет пользователю перемещать видимую область графика путем нажатия и перетаскивания мышью.

2. Масштабирование:

Реализовано с помощью колесика мыши и ползунка, обеспечивая плавное увеличение и уменьшение масштаба графика.

3. Рисование линий и окружностей:

Пользователь может выбирать различные алгоритмы для рисования линий и окружностей, такие как пошаговый алгоритм, ЦДА и Брезенхем.

4. Случайная генерация координат:

Кнопка для генерации случайных координат начальной и конечной точек линии, а также радиуса окружности.

5. Отображение времени выполнения:

Показывает время, затраченное на выполнение выбранного алгоритма рисования.

6. Динамическая сетка с подписями:

Сетка координат отображается на канвасе с динамическими подписями, адаптирующимися под текущий масштаб для обеспечения читаемости.

## Теоретические аспекты:

1. Пошаговый алгоритм (function stepByStep)

Пошаговый алгоритм рисования линии представляет собой самый простой метод построения прямой линии на пиксельной сетке. Он заключается в поочередном увеличении координат от начальной точки до конечной, основываясь на шагах по одной или обеим осям.

Принцип Работы:

-Начинаем с начальной точки (x1, y1).

-На каждом шаге увеличиваем координаты x и/или y в направлении конечной точки (x2, y2).

-Продолжаем шаги до достижения конечной точки.

2. Алгоритм ЦДА (function DDA)

Алгоритм ЦДА является улучшением пошагового метода, который использует дробные шаги для более точного и равномерного распределения пикселей по линии. Он основывается на вычислении приращений по осям x и y для достижения гладкости линии.

Принцип Работы:

-Вычисляем разницу по осям dx = x2 - x1 и dy = y2 - y1.

-Определяем количество шагов steps как максимальное значение из |dx| и |dy|.

-Вычисляем приращения xIncrement = dx / steps и yIncrement = dy / steps.

-Начинаем с начальной точки (x1, y1) и на каждом шаге добавляем приращения к координатам.

-Рисуем пиксель в округленных координатах (Math.round(x), Math.round(y)).

3. Алгоритм Брезенхема (function bresenham)

Алгоритм Брезенхема является одним из наиболее эффективных и популярных методов рисования линий на пиксельной сетке. Он использует только целочисленные операции, что делает его быстрым и подходящим для аппаратного ускорения.

Принцип Работы:

-Вычисляем разницы dx = |x2 - x1| и dy = |y2 - y1|.

-Определяем направления движения по осям sx и sy в зависимости от положения конечной точки.

-Инициализируем ошибку err = dx - dy.

-На каждом шаге:

-Рисуем пиксель в текущей позиции (x, y).

-Вычисляем удвоенную ошибку e2 = 2 \* err.

-Если e2 > -dy, корректируем ошибку и перемещаемся по x.

-Если e2 < dx, корректируем ошибку и перемещаемся по y.

Повторяем шаги до достижения конечной точки.

4. Алгоритм Брезенхема для рисования окружностей (function bresenhamCircle)

1. Алгоритм Брезенхема для Окружностей

Алгоритм Брезенхема для окружностей — это эффективный метод построения окружности на пиксельной сетке, использующий симметрию окружности и целочисленные операции для минимизации вычислительных затрат.

Принцип Работы:

-Инициализируем координаты (x, y) с начального положения (0, r), где r — радиус окружности.

-Устанавливаем начальную ошибку d = 3 - 2 \* r.

-На каждом шаге:

-Рисуем пиксели в восьми симметричных точках относительно центра (xc, yc).

-Если d < 0, корректируем ошибку d += 4 \* x + 6.

-Иначе, корректируем ошибку d += 4 \* (x - y) + 10 и уменьшаем y на 1.

-Увеличиваем x на 1.

Повторяем шаги до тех пор, пока x не станет больше y.

## Ход работы:

1. Создание интерфейса:

-HTML и CSS: Разработан базовый макет страницы с канвасом для рисования, панелью управления кнопками и ползунком для масштабирования, а также полями ввода для координат.

-Canvas: Установлены размеры канваса и добавлены стили для улучшения визуального восприятия (границы, фон, курсор при взаимодействии).

2. Реализация панорамирования и зумирования:

-Панорамирование: Добавлены обработчики событий mousedown, mousemove, mouseup и mouseleave для отслеживания и реализации перемещения канваса.

-Зумирование: Реализовано через событие wheel мыши и изменение значения ползунка. Введены ограничения на минимальный и максимальный масштаб для предотвращения чрезмерного уменьшения или увеличения.

3. Разработка сетки координат:

-Сетка: Реализована функция drawGrid, которая рисует вертикальные и горизонтальные линии сетки, определяя шаг сетки (gridStep) на основе текущего масштаба.

-Подписи: Добавлены подписи на осях X и Y, которые адаптируются под масштаб, отображая только определенные координаты для сохранения читаемости.

4. Реализация алгоритмов рисования:

-Функции plotPixel и resetPlotPixel: Созданы для рисования отдельных пикселей и сброса предыдущих координат для корректного соединения линий.

-Алгоритмы: Реализованы функции stepByStep, DDA, bresenham для линий и bresenhamCircle для окружностей, каждая из которых возвращает время выполнения для отображения пользователю.

-Отображение результатов: После выполнения алгоритма вызывается функция redraw, которая перерисовывает сетку и заново рисует последние линии или окружности.

5. Добавление описания алгоритмов на сайт:

Описание алгоритмов: Добавлен раздел, отображающий подробное описание выбранного алгоритма, а также иллюстрация его работы с сопроводительными вычислениями.

6. Оптимизация и исправление ошибок:

-Совпадение сетки и рисования: Первоначально возникла проблема с несовпадением сетки и рисуемых объектов, решенная путем корректного определения gridStep и использования одинакового масштаба при рисовании.

-Минимальный масштаб: Установлен минимальный масштаб 5 для предотвращения чрезмерного уменьшения, которое приводило к плотной сетке и затрудняло чтение подписей.

-Интенсивность зума: Уменьшена интенсивность зума (zoomIntensity) для более плавного изменения масштаба, что улучшило пользовательский опыт.

-Размеры элементов: Настроены размеры точек и толщины линий в зависимости от масштаба для обеспечения пропорционального отображения при разных уровнях зума.

7. Тестирование и отладка:

-Проверка соответствия: Проведено тестирование для подтверждения того, что рисуемые линии и окружности точно совпадают с линиями сетки.

-Пользовательский интерфейс: Проверена функциональность панорамирования и зумирования, убедившись в плавности и корректности отображения.

-Случайные координаты: Проверено корректное отображение случайно сгенерированных координат и окружностей.

-Отображение описаний: Проверено корректное отображение описания алгоритма.

## Заключение:

В ходе выполнения лабораторной работы успешно разработано интерактивное веб-приложение для визуализации алгоритмов растеризации на основе HTML, CSS и JavaScript с использованием Canvas. Приложение позволяет пользователю панорамировать и масштабировать график, а также выбирать различные алгоритмы для рисования линий и окружностей. Были решены ключевые проблемы с совпадением сетки и рисуемых объектов, а также обеспечена плавность и удобство взаимодействия. Дополнительно, внедрена функциональность отображения подробного описания выбранных алгоритмов с пошаговыми примерами выполнения, что значительно повышает образовательную ценность приложения. Полученные результаты демонстрируют эффективность применяемых методов и алгоритмов для создания точной и гибкой системы визуализации.