

Генерация интерактивного редактора изображений на базе Canvas API методами промпт-инжиниринга

Пайдершев Алексей Константинович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В статье исследуются возможности больших языковых моделей при проектировании веб-приложений для попиксельной обработки графических данных на примере генерации интерактивного редактора изображений в среде Qwen Studio. Опора на нативный Canvas API и структуру ImageData позволяет реализовать полноценный графический редактор без привлечения сторонних библиотек, сохраняя парадигму Zero-Dependency. Применение детализированного промпт-инжиниринга делегирует нейросети функции системного архитектора, однако эксперимент выявляет итеративный характер генеративной разработки: первоначальный код потребовал многократной доработки для устранения архитектурных недочётов в логике обрезки, системе курсора и обработке событий мыши. Полученные результаты демонстрируют высокую эффективность ИИ в задачах математической обработки пикселей и слоёв, но подтверждают необходимость строгого инженерного контроля со стороны разработчика.

Ключевые слова: Qwen Studio, нейросети, промпт-инжиниринг, вайб-кодинг, Canvas API.

Generating an interactive image editor based on the Canvas API using prompt engineering techniques.

Paydershev Alexey Konstantinovich

Sholom-Aleichem Priamursky State University

Student

Abstract

The article explores the capabilities of large language models in designing web applications for pixel-by-pixel processing of graphical data, using the example of generating an interactive image editor in the Qwen Studio environment. Relying on the native Canvas API and the ImageData structure makes it possible to implement a full-fledged graphic editor without using third-party libraries, while maintaining the Zero-Dependency paradigm. The use of detailed prompt engineering delegates the functions of a system architect to the neural network; however, the experiment reveals the iterative nature of generative development: the initial code required multiple revisions to eliminate architectural flaws in the cropping logic, the cursor system, and mouse event handling. The results obtained demonstrate the high

efficiency of AI in tasks involving the mathematical processing of pixels and layer compositing, but they confirm the need for strict engineering control by the developer.

Keywords: Qwen Studio, neural networks, prompt-engineering, viber coding, Canvas API.

1 Введение

1.1 Актуальность

Современная веб-разработка всё чаще сталкивается с проблемой избыточного усложнения: даже простые интерактивные приложения обвешиваются десятками зависимостей, что приводит к раздуванию размеров бандлов и снижению производительности. На этом фоне парадигма Zero-Dependency, предполагающая использование исключительно нативных браузерных технологий, приобретает особую научную и практическую ценность. Однако реализация сложных графических приложений на чистом Canvas API требует глубокого понимания попиксельной обработки, композитинга слоёв и математических преобразований цветовых пространств – задач, традиционно считающихся прерогативой опытных разработчиков.

Актуальность данного исследования обусловлена необходимостью проверить, способна ли большая языковая модель, выступая в роли Senior-архитектора, самостоятельно спроектировать многокомпонентную систему обработки изображений, где математическая точность формул напрямую влияет на визуальный результат. Редактор изображений выступает идеальным экспериментальным полигоном: в отличие от шахматного движка, где логика дискретна, здесь нейросеть должна реализовать непрерывные вычисления над массивами пикселей, организовать неразрушающее редактирование и обеспечить отзывчивый интерфейс при одновременной работе нескольких инструментов.

1.2 Обзор исследования

Фундаментальные принципы конструирования интерактивных браузерных продуктов на базе HTML5 подробно раскрыты в исследованиях С.В. Рогатина и В. А. Антошкина [1], где авторы обосновывают универсальность нативных технологий для создания мультимедийных приложений. Концепцию вайб-кодинга как педагогического и инженерного инструмента развивает А. Б. Рахмонов, выделяя его функциональные возможности для современного образовательного процесса [2]. Строгий инженерный контроль за результатами генерации предлагает И. В. Елманов, обосновывая необходимость методологии Spec Driven Development для верификации сгенерированного кода [3]. Применимость генеративных моделей в сложных предметных областях подтверждают работы Д. Р. Мухаметова, исследующего влияние ИИ на формирование политических образов [4], а также труд А. Д. Степанова, Л. А. Богданова, Д. К. Шишковой и А. Г. Кутихина, демонстрирующий результативность нейросетевых алгоритмов в трансляционной кардиологии [5].

1.3 Цель исследования

Цель исследования – оценить эффективность и границы применимости больших языковых моделей (на примере Qwen Studio) при автономной генерации веб-приложений, требующих попиксельной математической обработки данных и организации многокомпонентной архитектуры пользовательского взаимодействия в парадигме Zero-Dependency.

2 Материалы и методы

Методология исследования базируется на комбинации детализированного промпт-инжиниринга и строгого архитектурного ограничения: всё приложение конструируется в виде единого HTML-файла без привлечения сторонних библиотек, фреймворков и предзагруженных медиафайлов. В качестве графического движка выбран нативный Canvas API, а обработка изображений реализуется через прямую манипуляцию структурой ImageData – массивом пикселей в цветовом пространстве RGBA.

Особое внимание при составлении запроса к нейросети уделяется математической точности алгоритмов: формулы преобразования яркости, контраста и насыщенности задаются в явном виде, что позволяет объективно оценить способность ИИ генерировать корректный вычислительный код. Архитектура приложения предполагает разделение на три функциональных блока: модуль неразрушающих фильтров, систему рисования с композитингом слоёв и инструмент обрезки с двухкликерной логикой взаимодействия.

Ключевой особенностью методологии является итеративный подход: первоначальная генерация рассматривается как прототип, который последовательно дорабатывается через серию уточняющих промптов. Это позволяет зафиксировать не только сильные стороны ИИ, но и типичные архитектурные просчёты, возникающие при генерации сложных интерактивных систем.

3 Результаты

Первый этап исследования – формирование промпта, описывающего архитектуру редактора изображений.

Промт: «Действуй как Senior Frontend разработчик. Твоя задача – написать полный, готовый к запуску код веб-приложения «Редактор изображений». Приложение должно быть реализовано в одном файле и работать локально в браузере. Никаких сторонних библиотек – только нативные HTML, CSS и JavaScript с Canvas API.

Архитектурные требования:

- Используй Canvas API для всех манипуляций с изображением.
- Храни оригинал изображения отдельно, чтобы фильтры можно было применять с нуля.
- Реализуй систему истории отмены/повтора.

Требования к интерфейсу:

- Современный, минималистичный дизайн. Тёмная тема с мягкими тенями и скруглёнными углами.

- Расположение: левая боковая панель с инструментами, основная область с canvas, верхняя панель с действиями над файлом.

- Кнопки верхней панели: «Загрузить изображение», «Экспорт PNG», «Сбросить», «Отменить», «Повторить».

Секции боковой панели:

- «Фильтры» – слайдеры для: Яркость, Контраст, Насыщенность. Плюс кнопки-переключатели: «Сепия», «Инверсия», «Ч/Б».

- «Рисование» – переключатель режима рисования, выбор цвета, слайдер размера кисти, переключатель инструментов: «Кисть» / «Ластик».

- «Обрезка» – кнопка входа в режим обрезки.

- Если изображение не загружено, иконку с текстом «Загрузите изображение для начала работы».

- Все фильтры должны применяться одновременно и неразрушающе.

Выведи полный код в одном блоке, чтобы я мог сохранить его как index.html и сразу запустить».

Результат генерации: https://disk.yandex.ru/d/4D_eB7pd1DU17A

В результате первичной генерации получен полностью функциональный прототип редактора изображений (рис. 1). Нейросеть строго соблюла архитектурные ограничения, реализовав приложение в едином файле объёмом около 1200 строк кода. Интерфейс успешно построен средствами нативного HTML и CSS с использованием CSS-переменных.

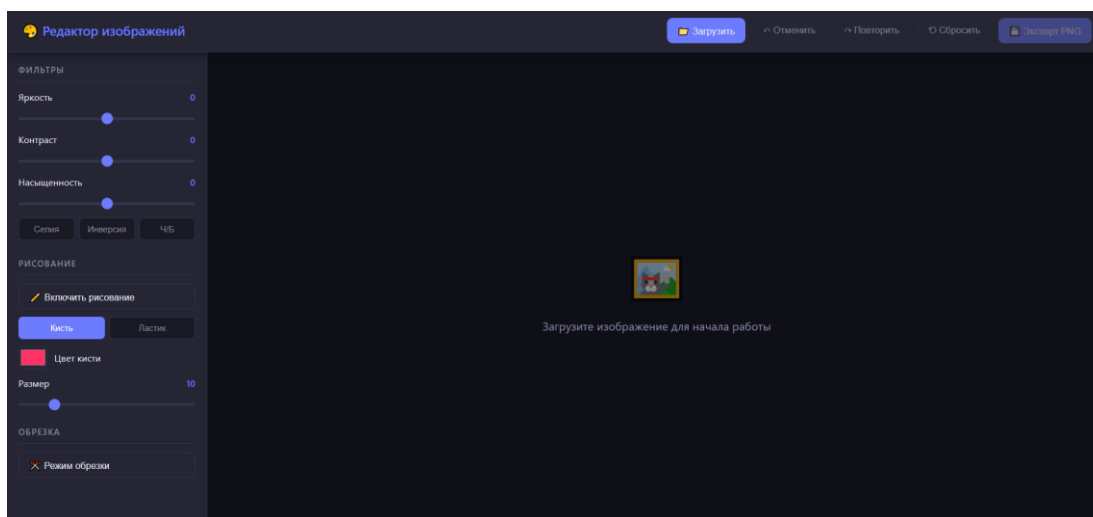


Рисунок 1. Интерфейс приложения

Ключевым достижением первичной генерации стала корректная реализация модуля попиксельной обработки. Нейросеть самостоятельно вывела математические формулы преобразования яркости, контраста и насыщенности, а также применила классическую матрицу сепии. Все фильтры применяются неразрушающе: оригинал изображения хранится в отдельной переменной `originalImageData`, и при каждом изменении слайдера вычисления

производятся заново от исходных пикселей, что исключает накопление ошибок квантования (рис. 2).

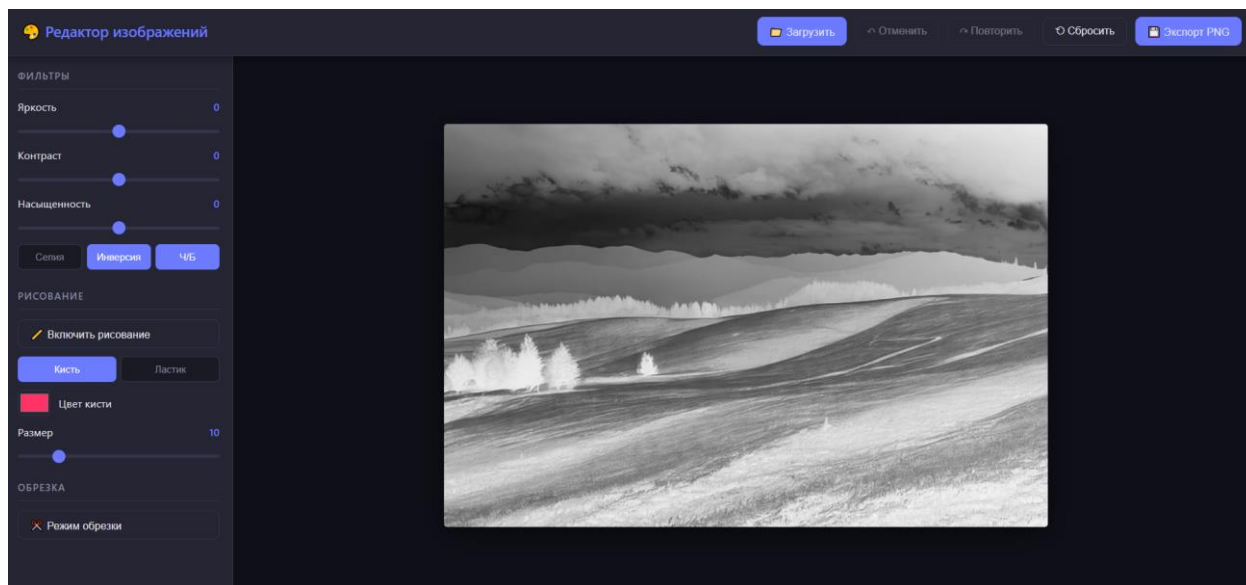


Рисунок 2. Применение фильтров к изображению

Также важно отметить, что все фильтры работают независимо и могут накладываться друг на друга.

Однако на этом этапе выявился ряд проблем: огика обрезки работала неверно. При попытке обрезать изображение, окно обрезки не фиксировалось и нажатие кнопки «Обрезать» не представлялось возможным.

агрузка изображения не работала посредством перетягивания изображения в окно редактора.

тмена действия не работала посредством нажатия горячих клавиш «Ctrl+Z».

Для решения данных проблем использован протм: «Исправь и добавь следующее:

брезка работает некорректно. Алгоритм обрезки должен быть такой:

- клик по изображению, появляется растягивающееся окно обрезки
- второй клик по изображению, окно обрезки фиксируется
- нажатие на кнопку "обрезать" обрезает изображение

обавь отмену действия нажатием сочетания клавиш «Ctrl+Z»

обавь загрузку изображения посредством перетягивания изображения в окно редактора»

Результат генерации: <https://disk.yandex.ru/d/3oD7CJ1hjlUYwQ>

Во второй версии изображения добавлена возможность загрузки изображения путём перетаскивания (рис. 3), отмена действия нажатием сочетания клавиш «Ctrl+Z» и исправлен режим обрезки (рис. 4-5).



Рисунок 3. Окно загрузки изображения

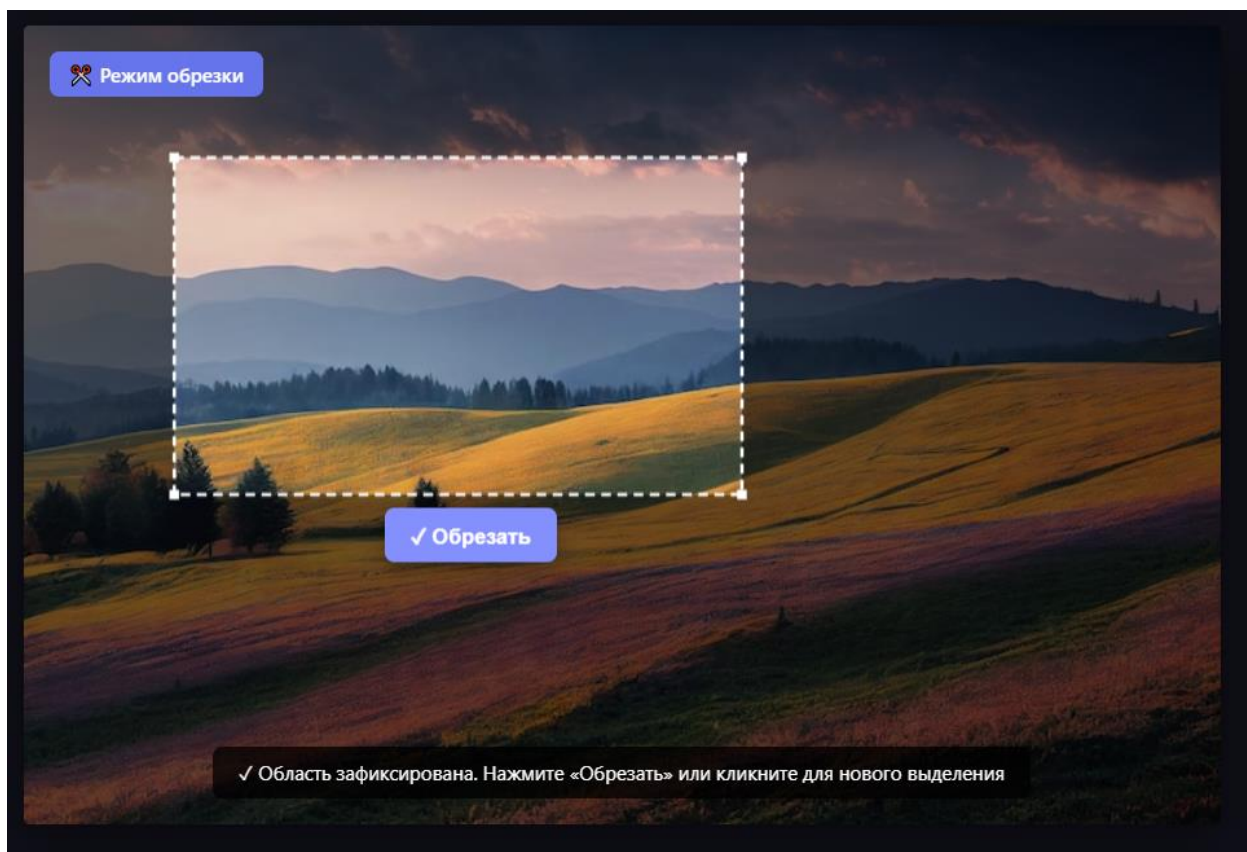


Рисунок 4. Окно обрезки

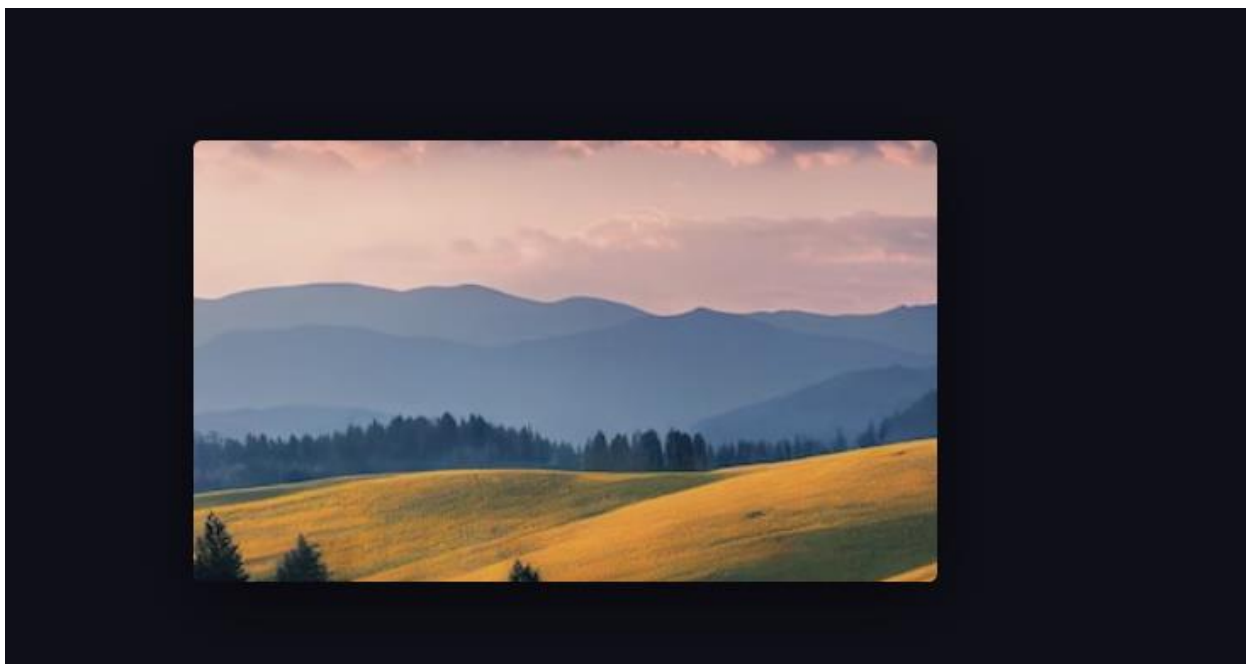


Рисунок 5. Обрезка изображения

Следующая проблема проявилась в режиме рисования: при наведении курсора на изображение стандартный курсор браузера исчезал, но замена ему не предоставлялась, что дезориентировало пользователя. Кроме того, при выходе мыши за поля рисование прерывалось, и возврат курсора не возобновлял процесс, даже если левая кнопка мыши оставалась зажатой.

Промт для исправления: «В режиме рисования при наведении курсора на изображение, он пропадает. Добавь курсор, который будет соответствовать размеру и цвету кисти.

Также в режиме рисования при выходе за поля изображения рисование прерывается. Сделай следующий алгоритм рисования:

- Нажатие и удержание ЛКМ - кисть рисует на изображении
- Выход за поля рисунка - рисование прекращается
- Возврат курсора на изображение при зажатой ЛКМ - возобновление рисования»

Результат генерации: <https://disk.yandex.ru/d/oZP-QSg-Okf1lw>

Нейросеть написала новый код, в котором исправлены ошибки рисования, о которых говорилось ранее. Режим рисования не прерывается, а курсор стал соответствовать цвету и размеру кисти (рис. 6).

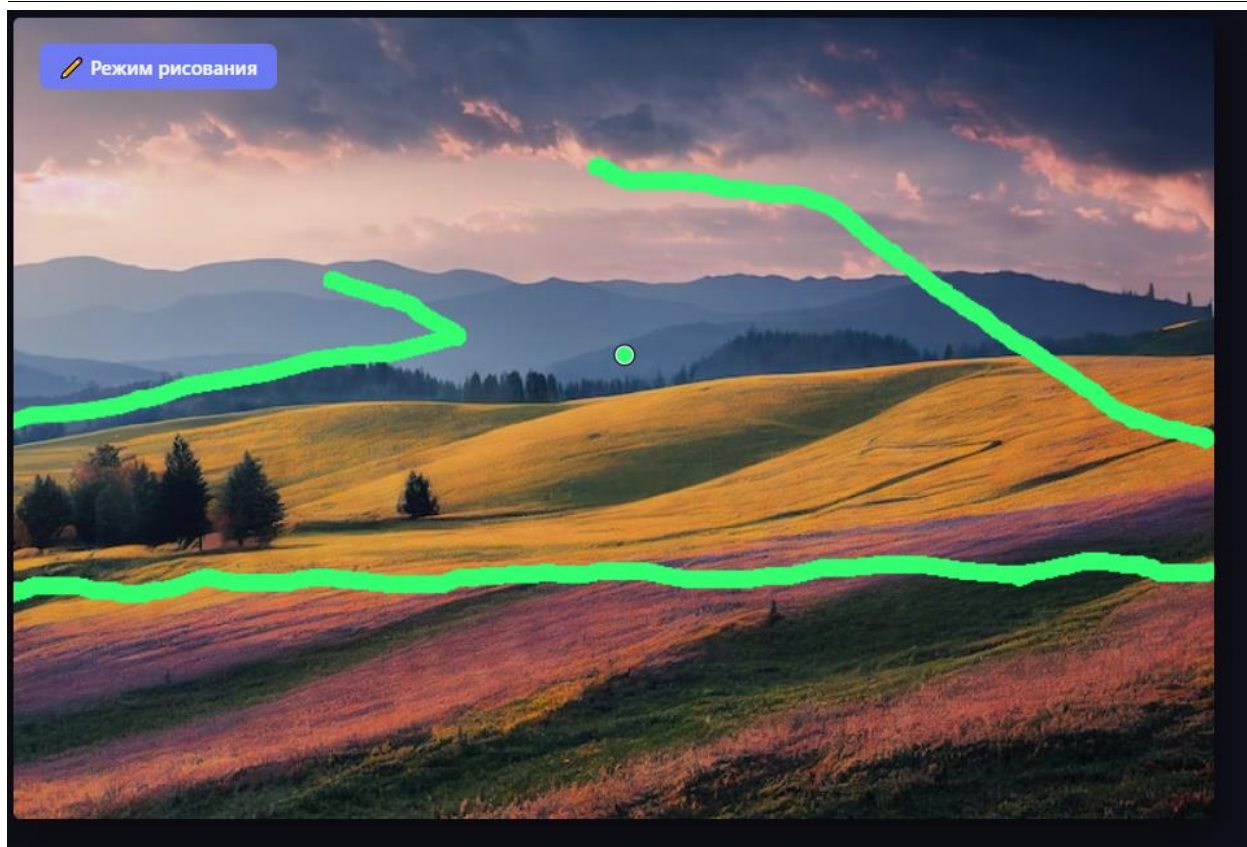


Рисунок 6. Режим рисования

В третьей версии приложения возникла проблема, которая уже решалась ранее. Перетаскивание изображения для загрузки стало невозможным. Искусственный интеллект объяснил это так: «Проблема была в конфликте обработчиков событий. Обработчики на document перехватывали события dragover и drop раньше, чем они доходили до canvasArea».

Для решения этой проблемы написан промт: «Перестало работать перетаскивание для загрузки изображения. Исправь».

Результат генерации: <https://disk.yandex.ru/d/do4dJKyJXESSvA>

Нейросеть переписала код и ошибка пропала.

Суммируя опыт доработок, можно выделить типичные архитектурные просчёты, допущенные нейросетью при первичной генерации:

- Упрощённая логика взаимодействия - инструмент обрезки реализован по принципу mousedown-mousemove-mouseup, что удобно для разработчика, но неудобно для пользователя. Потребовалось введение конечного автомата с явным управлением состояниями.

- Игнорирование UX-деталей - отсутствие визуальной обратной связи для курсора в режиме рисования показывает, что ИИ фокусируется на функциональности, но упускает эргономику.

- Конфликты на разных уровнях DOM - обработчики событий на document и на дочерних элементах конкурируют за приоритет, что требует понимания механизма всплытия событий и фаз перехвата.

Каждая доработка требовала от разработчика чёткой формулировки проблемы и желаемого поведения, что подтверждает тезис о переходе роли человека от кодера к верификатору архитектуры и постановщику задач.

4 Вывод

Проведённое исследование подтверждает высокую эффективность больших языковых моделей при генерации веб-приложений, требующих попиксельной математической обработки данных. Эксперимент по созданию интерактивного редактора изображений на базе Canvas API в парадигме Zero-Dependency продемонстрировал способность нейросети самостоятельно выводить формулы преобразования цветовых пространств, организовывать неразрушающее редактирование и реализовывать композитинг слоёв средствами нативного JavaScript.

Однако ключевым открытием исследования стал итеративный характер генеративной разработки. Первичная генерация, хотя и создала работоспособный прототип, содержала ряд архитектурных недочётов: упрощённую логику взаимодействия, игнорирование UX-деталей, конфликты обработчиков событий и неоптимальное разделение состояний. Каждая из этих проблем потребовала отдельной итерации доработки с уточняющим промптом, что подтверждает необходимость строгого инженерного контроля.

Полученные результаты позволяют сформулировать важное правило: генеративный подход в парадигме Zero-Dependency идеален для задач математической обработки структурированных данных (попиксельные фильтры, композитинг, вычислительная геометрия), но требует многократной верификации и доработки в задачах проектирования сложных пользовательских сценариев. Разработчик в этом тандеме выступает не как кодер, а как Senior-архитектор и UX-дизайнер, диктующий логические и эргономические законы системы.

В перспективе описанная методология может быть применена для оперативного прототипирования образовательных инструментов, где попиксельная обработка изображений используется для визуализации алгоритмов, анализа данных или интерактивного обучения компьютерному зрению.

Библиографический список

1. Рогатин С. В., Антошкин В. А. Разработка компьютерных игр по технологии html5 //Информатика и прикладная математика. 2015. №. 21. С. 12-16. URL: <https://elibrary.ru/item.asp?id=25030492>
2. Рахмонов А. Б. Вайб-кодинг как инструмент цифровой педагогики: функциональные возможности и механизмы реализации //The Lingua Spectrum. 2025. Т. 12. №. 2. С. 552-556. URL: <https://lingvospektr.uz/index.php/lngsp/article/view/1329>
3. Елманов И. В. Инженерный подход к ии: использование spec driven development для валидации vibe coding //Вестник науки. 2025. Т. 4. №. 12

- (93). С. 716-723. URL: <https://cyberleninka.ru/article/n/inzhenernyy-podhod-k-ii-ispolzovanie-spec-driven-development-dlya-validatsii-vibe-coding>
4. Мухаметов Д. Р. Политический вайб-маркетинг: возможности искусственного интеллекта для продвижения образа будущего //Известия Тульского государственного университета. Гуманитарные науки. 2025. №. 3. С. 27-33. URL: <https://cyberleninka.ru/article/n/politicheskiy-vayb-marketing-vozmozhnosti-iskusstvennogo-intellekta-dlya-prodvizheniya-obraza-buduschego>
5. Степанов А. Д., Богданов Л. А., Шишкова Д. К., Кутихин А. Г. Применение инструментов на основе искусственного интеллекта в экспериментальной и трансляционной кардиологии: обзор //Российский кардиологический журнал. 2026. Т. 31. №. S2. С. 114-122. URL: <https://cyberleninka.ru/article/n/primenenie-instrumentov-na-osnove-iskusstvennogo-intellekta-v-eksperimentalnoy-i-translyatsionnoy-kardiologii-obzor>