Материал и методы. В формате DICOM представлены результаты МРТ-сканирования головного мозга с контрастным усилением. Данные организованы в виде последовательности срезов, где каждый содержит: двумерное изображение исследуемой области, набор метаданных с информацией о размерах пикселей, толщине среза, пространственной ориентации. В основу разработки легли модели CNN и U-Net сегментации МРТ-снимков. Работа над проектом велась на языке Python с применением библиотек NumPy, Pillow, OpenCV, TensorFlow, PyTorch, PyQt и Pydicom. Официальная документация и сайты разработчиков использовались как основа для корректной реализации и настройки необходимых модулей.

Результаты и их обсуждение. Для взаимодействия с пользователем разработано оптимизированное приложение. При открытии конкретного снимка врач сразу видит исходное изображение, предсказанную маску и их полупрозрачное наложение. Благодаря проработанной обработке ошибок и сохранению состояния приложения все этапы взаимодействия оказываются надёжными и устойчивыми к непредвиденным сбоям.

Также был проведён анализ выбранной архитектуры нейронной сети, в ходе которого её характеристики были сопоставлены с ближайшим конкурентом – SegNet. На основе проведённых экспериментов было доказано, что архитектура U-Net обеспечивает более высокую точность сегментации, несмотря на несколько большее время обучения. Учитывая важность точности при работе с медицинскими изображениями, особенно в задачах локализации и сегментации опухолей, правильность выбора U-Net подтверждена как с практической, так и с теоретической точек зрения.

Заключение. Приложение объединяет все этапы работы с МРТ-снимками: от выбора папки с DICOM-изображениями и автоматической подготовки срезов до прогнозирования масок и наглядного отображения результатов. Понятный и простой интерфейс позволяет врачу сосредоточиться на анализе данных, не беспокоясь о технических деталях.

1. Использование сверточных нейронных сетей для решения задач классификации в неконтролируемых условиях / Н.Д. Никонов, Т.В. Никонова, О.Е. Рубаник, Е.А. Корчевская // Веснік Віцебскага дзяржаўнага ўніверсітэта. – 2023. – № 2. – С. 5–11. URL: https://rep.vsu.by/handle/123456789/39417 (дата обращения 10.03.2025).

РЕАЛИЗАЦИЯ МЕХАНИЗМА ОТМЕНЫ И ПОВТОРА ДЕЙСТВИЙ НА ОСНОВЕ СТЕКА В ВЕКТОРНОМ РЕДАКТОРЕ

Радевич И.Н.,
учащийся 3 курса Оршанского колледжа ВГУ имени П.М. Машерова,
г. Орша, Республика Беларусь
Научный руководитель – Романцов Д.Ю., магистр техн. наук, преподаватель

Ключевые слова. Отменить/повторить, стек, полный снимок, паттерн Хранитель, графический редактор.

Keywords. Undo/redo, stack, full snapshot, graphic editor, Memento.

Механизм отмены и повтора действий является неотъемлемой частью современных графических редакторов, таких как Adobe Illustrator, CorelDRAW или Inkscape. Он позволяет пользователям безопасно экспериментировать с дизайном, исправлять ошибки и возвращаться к предыдущим состояниям проекта.

Материалы и методы. Для реализации работы механизма используется среда разработки Visual Studio 2022 Community Edition и язык программирования C#. В рамках разработки редактора векторной графики функция отмены и повтора действий реализована на основе двух стеков, undoStack – стек [1] для хранения предыдущих состояний холста, используемый для отмены действий и redoStack – стек для хранения состояний, которые были отменены, чтобы их можно было вернуть. Каждый элемент стека – это список фигур, который хранит текущее состояние холста (все фигуры на нём). Такой подход, известен как «полный снимок» (full snapshot), он является одной из распространённых стра-

тегий в объектно-ориентированном программировании и часто ассоциируется с паттерном Memento (Хранитель) [2]. Он прост в реализации, но может быть ресурсоёмким при больших объёмах данных, что делает актуальными альтернативные подходы.

Цель данной работы – разработать и реализовать алгоритм управления историей изменений в простом редакторе векторной графики, использующий паттерн Хранитель и двухстековую модель с применением полного снимка, для обеспечения гибкости, безопасности экспериментирования и эффективного восстановления предыдущих состояний холста.

Результаты и их обсуждение.

1. Сохранение состояния перед изменением.

Когда пользователь выполняет действие, которое меняет холст (например, рисует линию), приложение делает следующее.

a. Программа создает копию текущего списка всех фигур. Каждая фигура со всеми своими свойствами (координаты, цвет, обводка) дублируется.

b. Текущая копия помещается на вершину стека undoStack.

c. Чтобы предотвратить чрезмерное использование памяти, размер стека ограничивается (например, 50 последних действий). Когда лимит превышается, самая старая копия удаляется.

d. Стек redoStack полностью очищается. Это происходит потому, что любое новое действие делает неактуальной всю цепочку отмененных действий, которая хранилась для возможного возврата.

Пример. Если пользователь рисует линию, перед добавлением её на холст приложение сохраняет копию текущего состояния (все фигуры до появления новой фигуры). Эта копия готова для отмены, если пользователь решит убрать линию (рисунок 1).

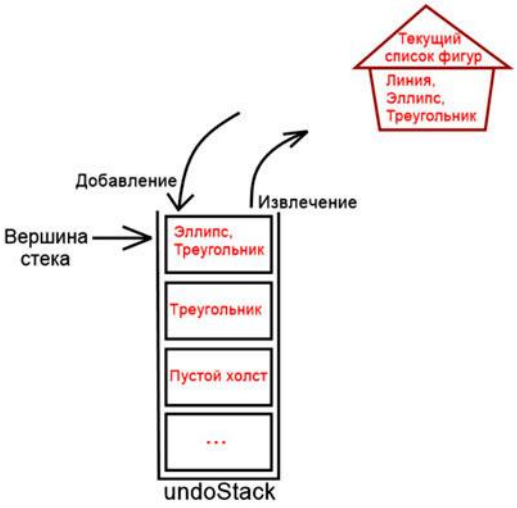

Рисунок 1 – Сохранение состояния перед изменением холста

2. Отмена действия

Когда пользователь нажимает Cntl+Z, чтобы отменить последнее действие, происходит следующее.

a. Проверяется, не пуст ли стек undoStack. Если стек окажется пустым, это означает, что нечего отменять.

b. Если стек оказался не пустым. Текущее состояние холста (которое мы хотим отменить) сохраняется в стек redoStack. Это позволяет в будущем вернуть только что отмененное действие.

c. Затем берётся верхняя копия списка фигур из стека для отмены и холст полностью заменяется его содержимым. Все фигуры, их позиции, цвета и другие свойства возвращаются к тому состоянию, которое было до последнего действия.

d. Холст перерисовывается, чтобы показать восстановленное состояние.

Пример. Если пользователь добавил линию, а затем нажал Ctrl+Z, приложение вернёт холст к состоянию без этой линии, используя сохранённый ранее список фигур. Текущий холст (с линией) сохраняется в стопке для возврата, чтобы пользователь мог вернуть его обратно (рисунок 2).

3. Возврат действия

Когда пользователь нажимает Ctrl+Y, чтобы вернуть отменённое действие, происходит следующее.

a. Проверяется, есть ли в стеке для возврата сохранённый список фигур.

b. Если есть, текущий список фигур сохраняется в стек для отмены, а последнее состояние из стека возврата становится текущим

c. Холст перерисовывается, показывая восстановленное состояние.

Пример. Если пользователь отменил добавление линии (Ctrl+Z), а затем нажал Ctrl+Y, приложение восстановит холст с этой линией, используя список фигур из стека для возврата (рисунок 3).

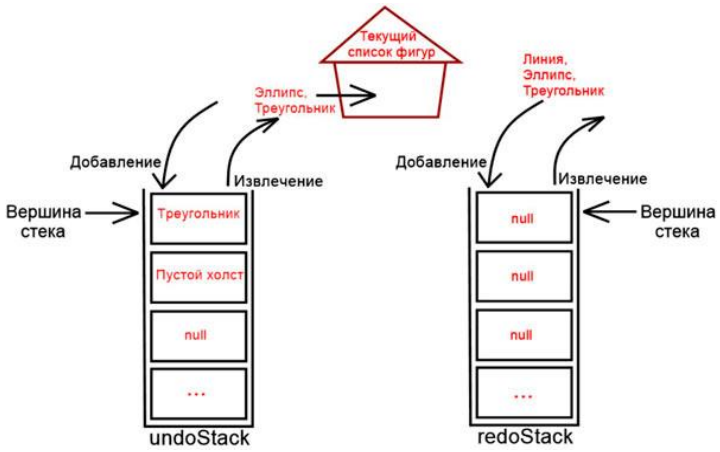

Рисунок 2 – Отмена действия

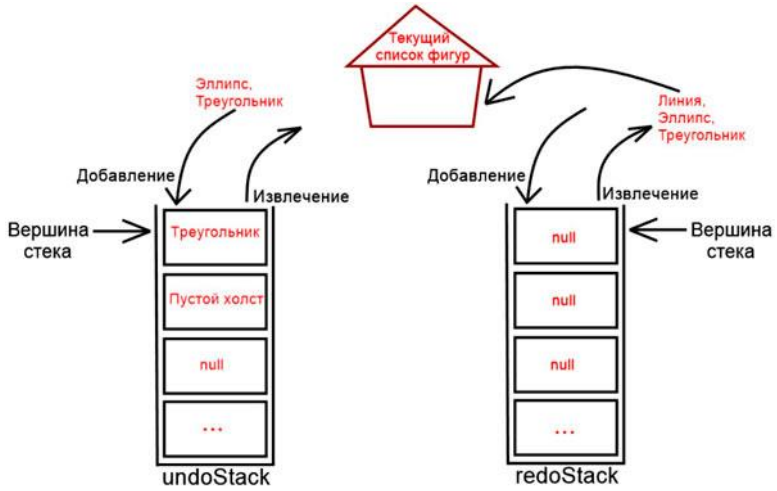

Рисунок 3 – Повтор действия

Заключение. Механизм undo/redo в редакторе использует два стека, что просто, надёжно и удобно для отмены и возврата действий. Он легко реализуется, но потребляет много памяти из-за копирования всех фигур при каждом изменении, а ограничение в 50 состояний может не хватить для больших проектов. Для улучшения возможно: 1) использовать паттерн Command для хранения только операций и их параметров; 2) сжимать данные в стеках; 3) сохранять состояния на диск.

1. Кормен, Т. Алгоритмы: построение и анализ, 2-е издание. Пер. с англ. / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн. – М.: Вильямс, 2005. – 1296 с., ил.

2. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – СПб.: Питер, 2015. – 368 с.

ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ СИМУЛЯТОРА НА ОСНОВЕ РЕТРОУСТРОЙСТВА «EMU CITIZEN ED-3000RX»

Силевцов Д.М.,
учащийся 4 курса Оршанского колледжа ВГУ имени П.М. Машерова,
г. Орша, Республика Беларусь
Научный руководитель – Юржиц С.Л., магистр, преподаватель

Ключевые слова. Симуляция, ретроустройства, электронные записные книжки, CITIZEN ED-3000RX, мобильные устройства, цифровое наследие.

Keywords. Simulation, retro devices, electronic notebooks, CITIZEN ED-3000RX, mobile devices, digital heritage.

Ретро-устройства (электронные записные книжки, персональные компьютеры, игровые консоли и др.) имеют историческую и техническую ценность. Современные технологии позволяют создавать их программные симуляторы, сохраняя функциональность устаревших систем [1].

Одним из примеров ретро-устройств стали электронные записные книжки, которые являлись частью повседневной жизни и профессиональной деятельности во второй половине XX века. Эти компактные устройства обеспечивали удобное хранение и управление информацией, позволяя легко получать доступ к контактам, заметкам, расписанию и другим данным. В конце 1980-х и начале 1990-х годов эти устройства получили широкое распространение, заложив основы для будущих мобильных устройств. Одним из примеров таких устройств является модель «CITIZEN ED-3000RX», выпущенная в 1990 году, которая продемонстрировала интеграцию цифровых технологий в повседневную жизнь, упрощая хранение личных и деловых данных. Эти устройства способствовали повышению эффективности работы и планирования, предлагая удобные способы ведения записей и организации информации. Исследование таких устройств, позволяет оценить уровень развития электроники в определенный исторический период, а также выявить принципы, которые легли в основу современных мобильных решений, что позволит воссоздавать функциональность на современных вычислительных платформах, обеспечивая доступ к оригинальным возможностям.

Целью данного исследования является разработка симулятора электронной записной книжки «CITIZEN ED-3000RX», способного воспроизводить функциональность на современных компьютерах.

Основные задачи исследования включают:

– анализ аппаратной и программной архитектуры устройства;

– изучение принципов работы и архитектуры электронных записных книжек на примере «CITIZEN ED-3000RX»;

– оценку их роли в развитии мобильных устройств и цифровых технологий;

– исследование возможностей воссоздания функционала ретро-устройства с применением современных технологий.