

Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Пензенский государственный университет»
Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
по курсу «Программирование»
на тему «Разработка программы сложной структуры
методом нисходящего программирования.
Игра в формате визуальной новеллы»

22.05.2026
взято
А. Генералова А.А.

Выполнил:
студент группы 25BBB1
Разживалов Н.А.

Принял:
к.т.н., доцент Генералова А.А.
к.т.н., доцент Токарев А.Н.

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники
Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ

М.А. Митрохин

« 18 » *февраль* 2016 г

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту *Рартишаву Никите Алексеевичу* Группа *25 ВВВ1*

Тема проекта: *Разработка программы сложной структуры методом нисходящего программирования.*

Исходные данные (технические требования) на проектирование

Игра в формате визуальная новелла. Компьютер выводит на экран изображение с персонажем, игрок проматывает изображение и на его основе делает сюжетные выборы. В драмле сохранения персонажа и выбора.

Обязательные требования к программе:

1. Многомодульность.
2. Использование сложных типов данных – структур, списков, файлов.
3. Режим работы видеосистемы – текстовый / графический.
4. Устройство ввода информации – клавиатура, мышь.
5. Пользовательский интерфейс должен быть построен на основе меню и панели инструментов.
6. Наличие заставки.
7. Операционная система MS Windows.
8. Язык программирования – Си и Ассемблер.
9. Среда разработки ПО – Microsoft Visual Studio.

Объем работы по курсу

1. Расчетная часть

Разработка программы.

2. Графическая часть

Схема данных, схема ресурсов системы, схема работы системы, иерархическая структура программы, схема взаимодействия программы.

3. Экспериментальная часть

Отладка программы.

Срок выполнения проекта по разделам

В соответствии с графиком.

Дата выдачи задания " 11 " февраля 20

Дата защиты проекта " 2 " июня 20

Руководитель ✓ Гусев А. Н.

Задание получил " 11 " февраля 2026 г.

Студент Рам Рамматов Н. А.

Содержание

Введение.....	3
Постановка задачи.....	6
Выбор решения.....	8
Описание разработки программы.....	10
Отладка и тестирование.....	10
Описание программы.....	10
Руководство пользователя.....	16
Приложение А Листинги программы.....	20
Приложение В Результаты работы программы.....	28

Введение

На современном этапе развития информационных технологий наблюдается устойчивый рост интереса к интерактивным программным системам, которые находят применение в самых различных сферах деятельности человека — от профессиональной и образовательной до развлекательной. Данный рост обусловлен стремлением пользователей к более активному участию в процессе взаимодействия с цифровым контентом, а также необходимостью повышения эффективности восприятия информации за счёт интерактивных механизмов. В этих условиях особое значение приобретают программные продукты, способные не только предоставлять информацию, но и адаптироваться под действия пользователя, формируя тем самым индивидуальный пользовательский опыт.

Особое место среди подобных систем занимают компьютерные игры, представляющие собой сложные программные продукты, сочетающие в себе элементы визуализации, логики и пользовательского взаимодействия. Современные игры включают в себя множество компонентов, таких как графические движки, системы обработки ввода, сценарные механизмы и средства управления ресурсами. В рамках данной области одним из наиболее специфичных и в то же время востребованных направлений является жанр визуальных новелл, который отличается акцентом не на динамическом геймплее, а на сюжетной составляющей и взаимодействии пользователя с историей.

Визуальная новелла представляет собой разновидность интерактивного программного продукта, сочетающего в себе элементы художественного текста, графического сопровождения и пользовательского управления. Основу таких приложений составляет сценарий, разбитый на отдельные сцены или кадры, которые последовательно отображаются пользователю. Важнейшей особенностью визуальных новелл является их нелинейная структура. В отличие от линейных сценариев, где последовательность событий строго фиксирована и не зависит от действий пользователя, визуальная новелла

реализует разветвлённую систему повествования. Такая система может быть представлена в виде дерева, где каждая вершина соответствует определённому состоянию сюжета, а переходы между ними определяются действиями пользователя. При этом каждая ветка может содержать собственную последовательность событий, что существенно увеличивает общее количество возможных вариантов прохождения. В результате один и тот же программный продукт может предоставлять множество различных путей развития сюжета и несколько альтернативных концовок. Данная особенность делает визуальные новеллы особенно привлекательными для пользователей, поскольку они обеспечивают высокий уровень вовлечённости и позволяют исследовать альтернативные сценарии развития событий. Пользователь получает возможность не только наблюдать за развитием истории, но и активно участвовать в её формировании, что значительно повышает интерес к процессу взаимодействия с программой. Это отличает визуальные новеллы от многих других типов программного обеспечения, где взаимодействие ограничивается однократным прохождением.

Разработка подобного программного обеспечения требует от программиста глубокого понимания ряда ключевых аспектов. В первую очередь, это архитектура операционной системы, в рамках которой осуществляется выполнение программы. Необходимо учитывать особенности работы с окнами, обработкой сообщений и управлением ресурсами. Также важным является понимание принципов событийно-ориентированного программирования, при котором логика работы приложения определяется не заранее заданной последовательностью команд, а реакцией на действия пользователя.

В рамках данной курсовой работы была поставлена задача разработать упрощённый, но функционально законченный движок визуальной новеллы. Программа реализована на языках C/C++ [1] с использованием нативного Windows API (WinAPI) и библиотеки GDI+ для рендеринга двухмерной графики. Отличительной особенностью реализации является демонстрация

низкоуровневых операций с данными: обработку счетчика кадров планируется реализовать на языке Ассемблера (вставки __asm), а сохранение прогресса осуществлять путем прямой записи бинарных дампов структур данных в файл. Программа должна позволять пользователю перемещаться по сценарию, состоящему из нескольких уровней вложенности, делать выбор между ветками сюжета, перематывать историю назад и сохранять свой прогресс в файл.

Постановка задачи

Целью данной курсовой работы является разработка программного обеспечения «Визуальная новелла» (Visual Novel Engine) — интерактивного приложения для воспроизведения нелинейных мультимедийных сценариев. Программа должна обеспечивать последовательный показ кадров истории, предоставлять пользователю возможность влиять на развитие сюжета путем выбора вариантов действий, а также сохранять текущий прогресс для последующего продолжения.

В первую очередь требуется реализовать механизм последовательного вывода графических кадров, которые составляют основу визуального повествования. Каждый кадр представляет собой отдельную сцену или часть сюжета. Особое внимание следует уделить организации управления просмотром. Пользователь должен иметь возможность переходить к следующему кадру для продолжения развития сюжета, а также возвращаться к предыдущим кадрам. Такая функциональность позволяет более гибко взаимодействовать с программой, даёт возможность повторного просмотра отдельных сцен и обеспечивает дополнительный уровень контроля над процессом прохождения.

Поскольку визуальная новелла предполагает наличие нескольких вариантов развития событий, одной из ключевых задач является разработка способа представления сценария в виде разветвлённой структуры. Такая структура должна отражать возможные пути развития сюжета и обеспечивать корректные переходы между ними. При выборе пользователем определённого варианта программа должна не только определить соответствующую ветку, но и корректно инициализировать её начальное состояние. Это включает установку текущего кадра на начальный, а также обновление информации о текущем уровне вложенности сценария. Важно, чтобы переход происходил строго в пределах допустимых значений, исключая возможность обращения к несуществующим элементам структуры сценария.

Дополнительно важно учитывать, что структура ветвления может быть многоуровневой, то есть каждая ветка может содержать собственные подветки. Это требует реализации механизма управления глубиной вложенности и корректного обновления данных при переходах между уровнями. Программа должна отслеживать текущее положение пользователя в этой иерархии и обеспечивать согласованное изменение состояния при каждом выборе. При реализации данного механизма необходимо обеспечить согласованность всех элементов структуры, чтобы пользователь не мог попасть в некорректное или несуществующее состояние. В определённые моменты времени приложение должно предоставлять пользователю возможность выбрать один из вариантов дальнейшего развития событий.

Важной составляющей является реализация механизма сохранения состояния программы. В рамках данной задачи необходимо обеспечить запись информации о текущем положении пользователя в сценарии. К таким данным относятся номер текущего кадра, выбранные ветки и уровень вложенности. Сохранение этих данных позволяет при повторном запуске программы восстановить состояние и продолжить прохождение с того же момента. Это особенно важно для приложений с нелинейным сценарием, где прохождение может занимать продолжительное время.

Исходя из этого, программа должна быть построена с использованием многомодульного подхода, включать в себя сложные типы данных – массивы, структуры и работу с файлами [2]. Режим работы видеосистемы – графический. Устройство ввода-вывода – компьютерная мышь. Программа должна включать заставку и логически понятный интерфейс.

Выбор решения

Исходя из поставленных задач, для реализации программного продукта были выбраны языки программирования **C/C++ и Assembler**, а в качестве среды разработки — Microsoft Visual Studio 2022.

Выбор языков C и C++ обусловлен рядом их существенных преимуществ: язык C обеспечивает эффективную работу с памятью, простоту и предсказуемость выполнения программ [3], тогда как C++ расширяет его возможности за счёт поддержки объектно-ориентированного программирования; их совместное использование позволяет объединить высокую производительность и контроль над ресурсами, характерные для языка C [4], с гибкостью и удобством разработки, обеспечиваемыми средствами C++, что делает данный набор технологий универсальным инструментом для создания системного и прикладного программного обеспечения, включая приложения, работающие с графикой и в реальном времени.

Использование вставок на языке ассемблера было обусловлено необходимостью продемонстрировать работу с низкоуровневыми операциями [5]. К преимуществам использования ассемблера относятся возможность полного контроля над регистровой моделью процессора, минимальные накладные расходы при выполнении операций, а также высокая скорость обработки простых вычислений [6]. Использование ассемблера способствует более глубокому пониманию внутренних механизмов работы компьютера, включая принципы выполнения инструкций, работу с регистрами, стеком и памятью [7]. Это имеет важное образовательное значение, так как позволяет сформировать у разработчика целостное представление о том, как высокоуровневые конструкции преобразуются в машинный код [8]. Несмотря на то, что в современных условиях ассемблер используется достаточно редко, его применение в рамках данной работы носит учебный характер и позволяет закрепить знания о принципах работы процессора.

В качестве среды разработки была выбрана Microsoft Visual Studio 2022, которая является одной из наиболее мощных и функциональных IDE для разработки приложений под Windows. Среда поддерживает работу с проектами различной сложности, обеспечивая удобную навигацию по исходному коду, управление файлами и настройку параметров сборки. Особое значение имеют встроенные средства отладки, которые позволяют детально анализировать процесс выполнения программы. Возможность пошагового выполнения кода, установки точек останова, просмотра значений переменных и анализа стека вызовов значительно упрощает процесс выявления и устранения ошибок. Это особенно важно при разработке приложений с большим количеством взаимосвязанных компонентов. Дополнительно Visual Studio обеспечивает полную поддержку работы с WinAPI и стандартными библиотеками Windows, что является необходимым условием для разработки графических приложений. Среда также предоставляет удобные инструменты для компиляции и сборки проекта, автоматизируя процесс создания исполняемого файла.

Выбранная совокупность языков программирования и среды разработки полностью соответствует поставленным задачам, обеспечивает высокую производительность приложения, позволяет продемонстрировать как базовые, так и углублённые навыки программирования, а также позволяет продемонстрировать широкий спектр навыков программирования, включая работу с системными ресурсами, графикой, структурами данных и низкоуровневыми механизмами выполнения программ.

Описание разработки программы

Разработанная программа состоит из нескольких модулей, которые реализовывались в следующем порядке:

- 1) ConsoleApplication2.cpp – основной файл программы
- 2) MediaFallback.h – реализация отображения изображений и видео
- 3) Choices.h – файл, хранящий все сюжетные ветки, все массивы с количеством максимальных кадров в каждой из них, номер текущего кадра

По порядку реализации модулей видно, что использовался нисходящий подход проектирования. Сначала была создана оболочка окна и цикл обработки сообщений, затем подключалась графика, и в последнюю очередь добавлялась логика ветвлений и сохранений.

На ассемблере реализован переход к следующему или предыдущему кадру за счёт команд инкремента (INK) и декремента (DEC), которые эквивалентны командам операнд++ и операнд-- языка Си, выполняющие увеличение и вычитание единицы из регистра. Для того, чтобы поместить и вынести число в регистр используется операция MOV.

Работа с файлами в коде используется для сохранения и загрузки текущих ветки и кадра в файл save.txt. В него заносятся элементы массивов, отвечающие за номер текущего кадра и путь к нынешнему адресу этого кадра.

Отладка и тестирование

В качестве среды разработки была выбрана программа Microsoft Visual Studio 2022. Программа обладает всеми средствами необходимыми при разработке и отладке программы. Тестирование проводилось во время разработки и также после завершения разработки.

Описание программы

ConsoleApplication2.exe является основным модулем программы. При запуске программы производится проверка наличия файла save.txt, в котором храниться текущее сохранение с указанием выбранной ветки и кадра в ней, в случае если этот файл ещё не создан, пользователю выводится заставка, объяснение назначения клавиш, а затем отображаются изображения самой

игры. При дохождении программой до последнего кадра ветки пользователю предоставляется выбор последующего развития событий в зависимости от того, нажмёт пользователь на левую половину экрана или правую, при нажатии на одну из них осуществляется переход в подветку отображая изображения одной из двух папок прошлой ветки, таким образом реализуется развилка сюжета (табл. 1).

Таблица 1 - Взаимодействие с программой

Клавиши, вызывающие событие	Действие пользователя	Действие программы
ЛКМ (левая кнопка мыши)	Нажатие в любом месте окна	Переход к следующему кадру
ПКМ (правая кнопка мыши)	Нажатие в любом месте окна	Возврат к прошлому кадру
ЛКМ на последнем кадре ветки	Нажатие на левую/правую сторону экрана	Переход на следующий уровень вложенности
ЛКМ на кнопку «к последнему кадру»	Нажатие на кнопку	Переход к максимальному (последнему) кадру текущей ветки для моментального перехода к моменту развилки
ЛКМ на кнопку «сохранить»	Нажатие на кнопку	Сохранение текущей ветки и номера кадра в файла save.txt

Схема программы. Функция WinMain, выполняющая задание описания характеристик окна и его отображение (рис. 1)

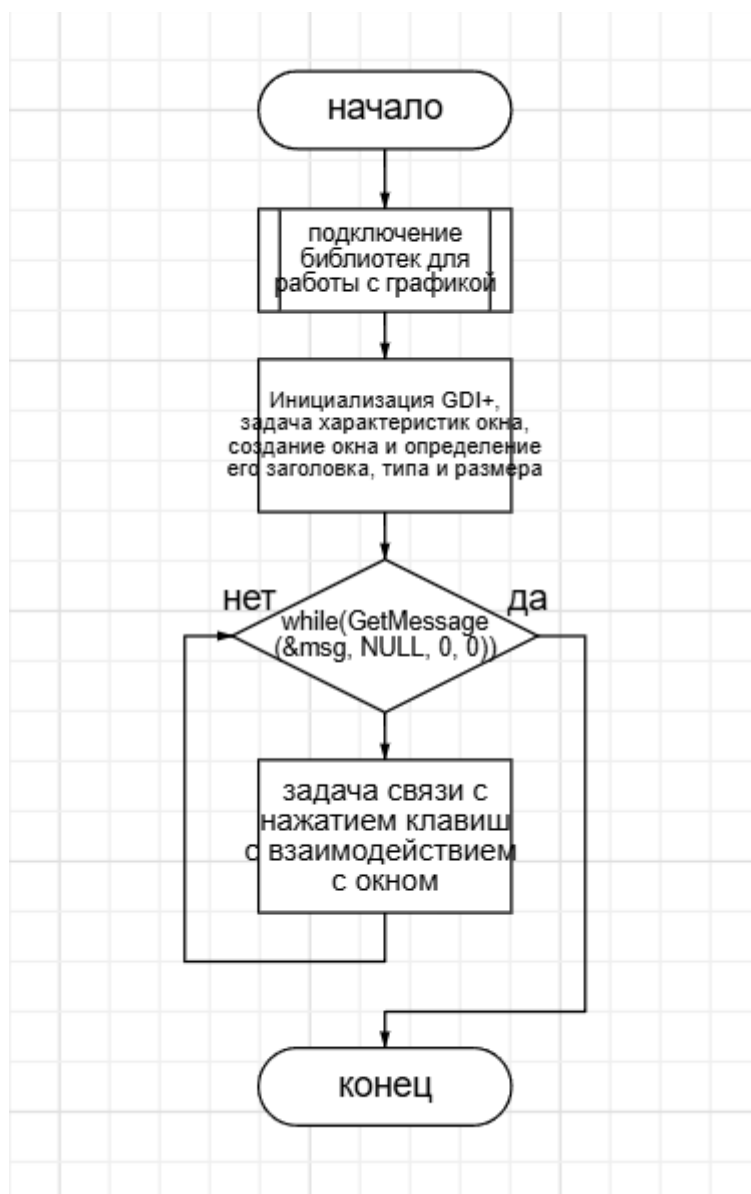


Рисунок 1 - Функция WinMain

Схема программы. Функции (рис. 2). Функция SaveGame() реализует сохранение адреса текущей ветки и нынешнего кадра в ней в файл save.txt, сохраняя соответствующие элементы массива. Функция LoadGame() реализует чтение и запись этих данных обратно в соответствующие элементы массивов, срабатывает при запуске программы(подробнее в описании следующей схемы). Функция DrawCurrent(HWND hwnd, Graphics& g) реализует создание пути к текущей ветке на основе действий пользователя. Функции IsButtonClick() и IsSaveButtonClick() проверяют, где находится курсор мыши по координатам x и y, и сравнивают с координатами относящимися к относящимся к ним кнопкам.

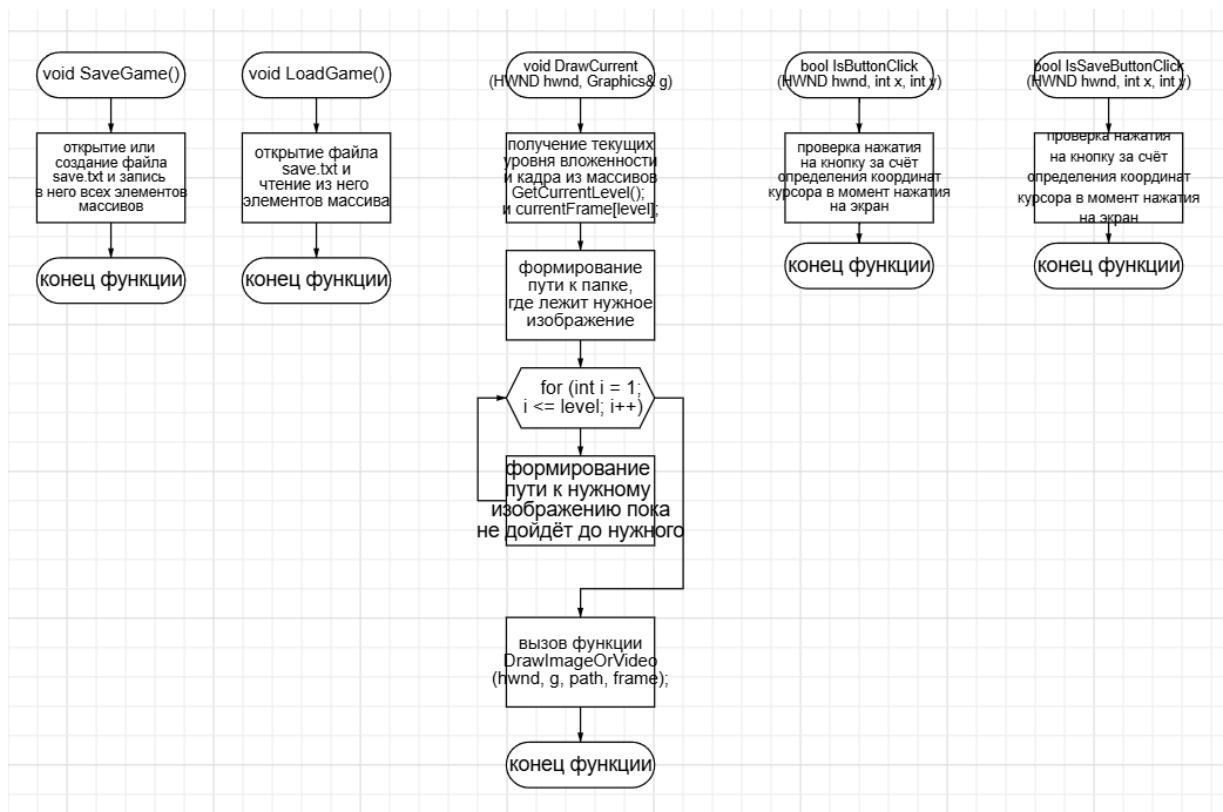


Рисунок 2 – Функции

Схема программы. Функция обратного вызова WndProc, реализованная за счёт конструкции switch, выполняющей конкретные задачи при конкретном действии пользователя, находящегося в одном из блоков case (рис. 3). case WM_CREATE реализует восстановление сохранения из функции LoadGame(), и выполняется автоматически при запуске программы. case WM_LBUTTONDOWN срабатывает при нажатии левой кнопки мыши, он обрабатывает нажатия пользователя на экран и в зависимости от того, где совершено нажатие либо выполняет переход на следующий кадр, либо в случае, если нажатие было произведено на одну из кнопок, выполняет переход на последний кадр или сохранение текущего состояния. case WM_RBUTTONDOWN работает аналогично, но выполняет возврат к прошлому кадру. case WM_PAINT срабатывает после выполнения case WM_LBUTTONDOWN или case WM_RBUTTONDOWN и выполняет отрисовку нужного кадра внутри окна. case WM_DESTROY выполняется в случае нажатия на кнопку закрытия окна, завершая отладку программы.

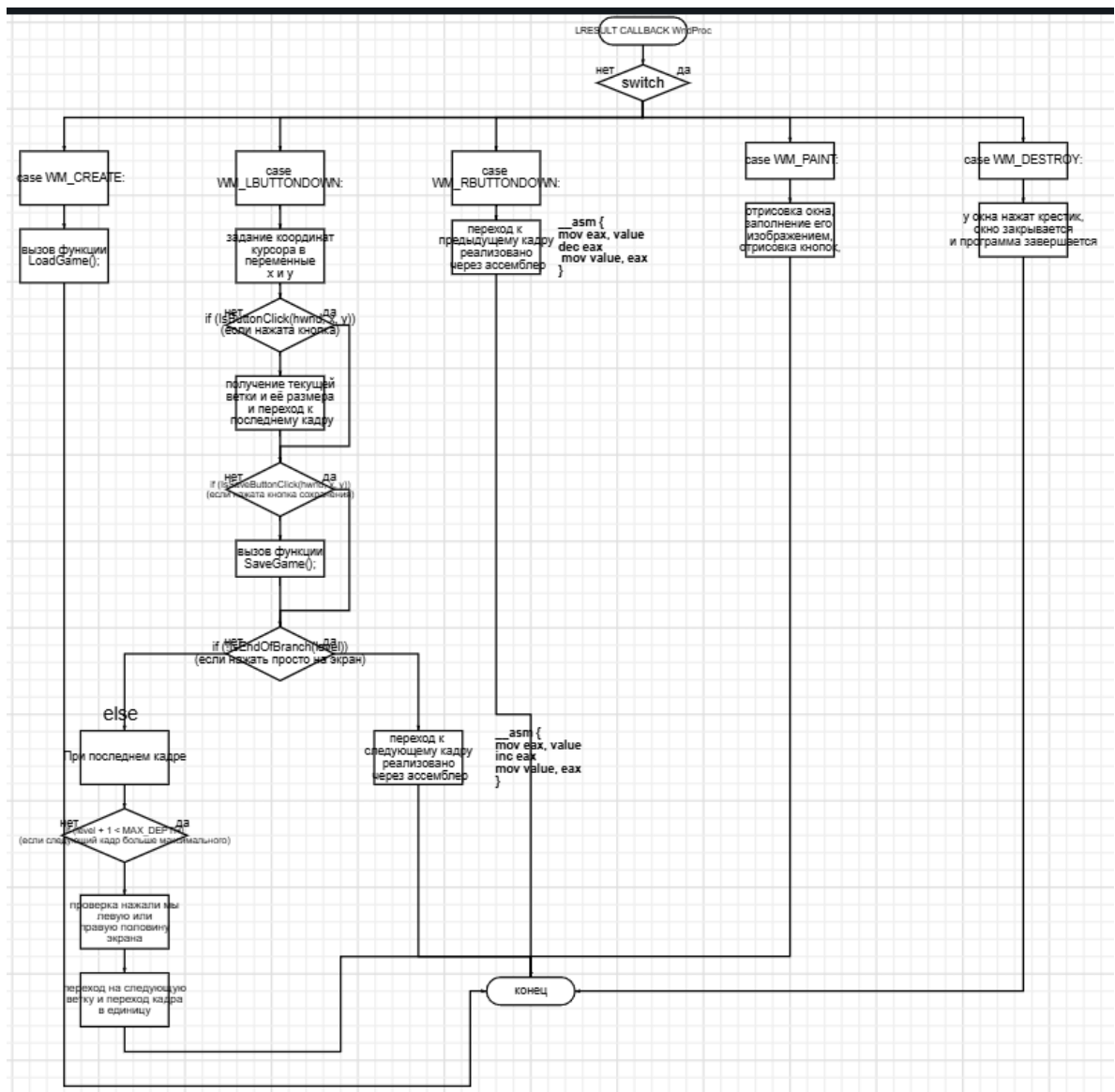


Рисунок 3 – Функция WndProc

Разработанная программа представляет собой полноценную систему для воспроизведения нелинейных сценариев с поддержкой ветвления сюжета и сохранения прогресса.

Руководство пользователя

Данная программа представляет собой игру в формате визуальной новеллы, а именно предоставляет возможность последовательно просматривать изображения, а также в определённые моменты развилки влиять на дальнейший сюжет, благодаря нажатию на левую либо правую часть экрана. Помимо этого, у пользователя есть возможность сохранить текущий прогресс, нажав на кнопку «Сохранить», и мгновенно перейти на момент следующего выбора, нажав кнопку «К последнему кадру».

В игре предоставлена возможность сделать 2 выбора, влияющих на сюжет, соответственно приводящих к 4 концовкам – 1 хорошей и 3 плохим.

Запуск программы реализуется либо через среду Visual Studio по нажатию на кнопку Локальный отладчик Windows, либо напрямую через файл с расширением .exe (приложение).

При запуске программы открывается заставка (рис. 4), объяснение взаимодействия с игрой (рис. 5) и затем отображается сама игра. Пользователь имеет возможность:

1. При всех кадрах кроме последнего
 - 1.1. Перейти к следующему кадру по нажатию левой кнопкой мыши.
 - 1.2. Вернуться к прошлому кадру по нажатию правой кнопки мыши
2. При последнем кадре (при разветвлении) (рис. 6)
 - 2.1. Перейти на первый кадр первой подветки по нажатию левой кнопки мыши на левую половину экрана
 - 2.2. Перейти на первый кадр второй подветки по нажатию левой кнопки мыши на правую половину экрана
3. Кнопки
 - 3.1. При нажатии на кнопку «перейти к последнему кадру» левой кнопкой мыши – мгновенный переход к моменту разветвления
 - 3.2. При нажатии на кнопку «сохранить» сохраняет текущую ветку и кадр в ней в файл save.txt

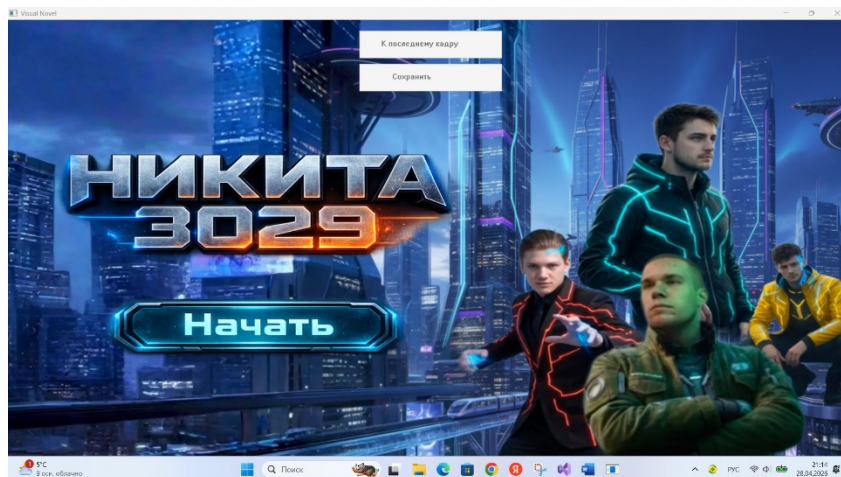


Рисунок 4 – Заставка игры

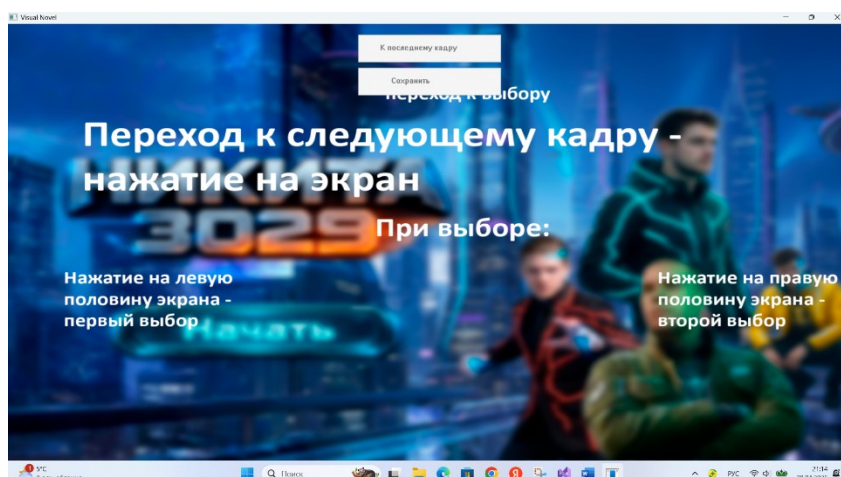


Рисунок 5 - Обучение

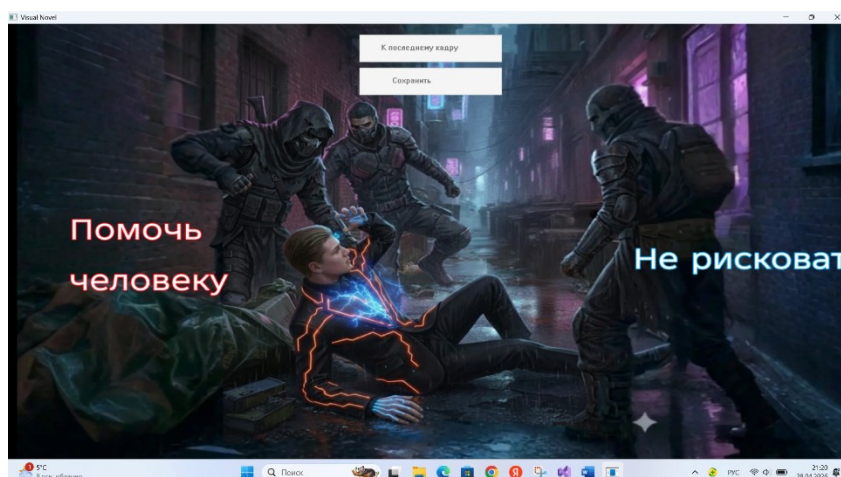


Рисунок 6 – Пример выбора

Продуманная система взаимодействия с пользователем делает программу доступной для широкого круга пользователей.

Заключение

При выполнении данной курсовой работы были получены навыки разработки многомодульных программ. Была освоена работа с файлами, были получены практические навыки разработки оконных приложений Windows с использованием WinAPI и GDI+. Также были получены основные навыки отладки и тестирования программ и программирования в среде VisualStudio 2022 на языках Си/C++ и Assembler.

В рамках работы был написан простой движок визуальной новеллы. Он поддерживает все заявленные функции: загрузку медиа, ветвление сюжета, сохранение и загрузку прогресса.

В дальнейшем программу можно улучшить, добавив поддержку звукового сопровождения или заменив воспроизведение видео через сторонний плеер на встроенный.

Список используемых источников

1. Керниган Б., Ритчи Д. Язык программирования Си. — 2-е изд. — СПб.: Вильямс, 2015. — 304 с.
2. Дейтел Х., Дейтел П. Как программировать на Си. — 8-е изд. — М.: Бином, 2017. — 944 с.
3. Прата С. Язык программирования С. Лекции и упражнения. — 6-е изд. — СПб.: Диалектика, 2019. — 928 с.
4. Подбельский В. В., Фомин С. С. Курс программирования на языке Си: учебник. — М.: ДМК Пресс, 2012. — 384 с.
5. Ирвин К. Язык ассемблера для процессоров Intel. — 6-е изд. — М.: Вильямс, 2013. — 912 с.
6. Пирогов В. Ассемблер. Учебный курс. — СПб.: БХВ-Петербург, 2008. — 640 с.
7. Хайд Р. Искусство программирования на языке ассемблера. — М.: Вильямс, 2003. — 1232 с.
8. Зубков С. Assembler. Для DOS, Windows и Unix. — СПб.: Питер, 2004. — 608 с.

Приложение А Листинги программы

Файл ConsoleApplication2.cpp

```
#include <windows.h>
#include <gdiplus.h>
#include "MediaFallback.h"
#include "Choices.h"

#pragma comment(lib, "gdiplus.lib")

using namespace Gdiplus;

ULONG_PTR token;

void SaveGame()
{
    FILE* f;
    if (fopen_s(&f, "save.dat", "wb") == 0)
    {
        fwrite(currentFrame, sizeof(int), MAX_DEPTH, f);
        fwrite(selectedBranch, sizeof(int), MAX_DEPTH, f);
        fclose(f);
    }
}

void LoadGame()
{
    FILE* f;

    if (fopen_s(&f, "save.dat", "rb") != 0)
    {
        ZeroMemory(currentFrame, sizeof(currentFrame));
        ZeroMemory(selectedBranch, sizeof(selectedBranch));

        currentFrame[0] = 1;
        currentLevel = 0;
        return;
    }
    fread(currentFrame, sizeof(int), MAX_DEPTH, f);
    fread(selectedBranch, sizeof(int), MAX_DEPTH, f);
    fclose(f);

    currentLevel = 0;

    for (int i = 0; i < MAX_DEPTH; i++)
    {
        if (currentFrame[i] == 0)
            break;
        currentLevel = i;
    }
}

void DrawCurrent(HWND hwnd, Graphics& g)
```

```

{
    int level = currentLevel;
    int frame = currentFrame[level];

    wchar_t path[256] = L"images";
    wchar_t temp[32];

    for (int i = 1; i <= level; i++)
    {
        wsprintfW(temp, L"\\v%d", selectedBranch[i] + 1);
        lstrcatW(path, temp);
    }

    DrawImageOrVideo(hwnd, g, path, frame);
}

bool IsButtonClick(HWND hwnd, int x, int y)
{
    RECT rc;
    GetClientRect(hwnd, &rc);

    int w = 260;
    int h = 50;

    int bx = (rc.right - w) / 2;
    int by = 20;

    return (x >= bx && x <= bx + w && y >= by && y <= by +
h);
}

bool IsSaveButtonClick(HWND hwnd, int x, int y)
{
    RECT rc;
    GetClientRect(hwnd, &rc);

    int w = 260;
    int h = 50;

    int bx = (rc.right - w) / 2;
    int by = 80;

    return (x >= bx && x <= bx + w && y >= by && y <= by +
h);
}

LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM
wParam, LPARAM lParam)
{
    switch (msg)
    {
        case WM_CREATE:
            LoadGame();
            break;

```

```

case WM_LBUTTONDOWN:
{
    int x = LOWORD(lParam);
    int y = HIWORD(lParam);

    int level = currentLevel;

    if (IsButtonClick(hwnd, x, y))
    {
        Node* node = GetNodeAtLevel(level);

        if (node)
            currentFrame[level] = node->maxFrames;

        InvalidateRect(hwnd, NULL, TRUE);
        break;
    }

    if (IsSaveButtonClick(hwnd, x, y))
    {
        SaveGame();
        break;
    }

    if (!IsEndOfBranch(level))
    {
        int value = currentFrame[level];

        __asm {
            mov eax, value
            inc eax
            mov value, eax
        }

        currentFrame[level] = value;
    }
    else
    {
        Node* node = GetNodeAtLevel(level);

        int index = (x < 640) ? 0 : 1;

        if (!node || !node->children[index])
            break;

        currentLevel++;

        selectedBranch[currentLevel] = index;
        currentFrame[currentLevel] = 1;

        // очистка хвоста

```

```

        for (int i = currentLevel + 1; i < MAX_DEPTH;
i++)
        {
            currentFrame[i] = 0;
            selectedBranch[i] = 0;
        }
    }

    InvalidateRect(hwnd, NULL, TRUE);
    break;
}

case WM_RBUTTONDOWN:
{
    int level = currentLevel;

    if (currentFrame[level] > 1)
    {
        int value = currentFrame[level];

        __asm {
            mov eax, value
            dec eax
            mov value, eax
        }

        currentFrame[level] = value;
    }

    InvalidateRect(hwnd, NULL, TRUE);
    break;
}

case WM_PAINT:
{
    PAINTSTRUCT ps;
    HDC hdc = BeginPaint(hwnd, &ps);

    Graphics g(hdc);
    g.Clear(Color(0, 0, 0));

    DrawCurrent(hwnd, g);

    RECT rc;
    GetClientRect(hwnd, &rc);

    int w = 260;
    int h = 50;

    int bx = (rc.right - w) / 2;
    int by = 20;

```

```

        HBRUSH brush = CreateSolidBrush( RGB(245, 245,
245));
        HBRUSH oldBrush = (HBRUSH)SelectObject(hdc, brush);

        HPEN pen = CreatePen(PS_SOLID, 1, RGB(200, 200,
200));
        HPEN oldPen = (HPEN)SelectObject(hdc, pen);

        Rectangle(hdc, bx, by, bx + w, by + h);

        SelectObject(hdc, oldBrush);
        SelectObject(hdc, oldPen);

        DeleteObject(brush);
        DeleteObject(pen);

        SetBkMode(hdc, TRANSPARENT);
        SetTextColor(hdc, RGB(120, 120, 120));
        TextOutW(hdc, bx + 40, by + 15, L"К последнему
кадру", 19);

        by = 80;

        brush = CreateSolidBrush( RGB(245, 245, 245));
        oldBrush = (HBRUSH)SelectObject(hdc, brush);

        pen = CreatePen(PS_SOLID, 1, RGB(200, 200, 200));
        oldPen = (HPEN)SelectObject(hdc, pen);

        Rectangle(hdc, bx, by, bx + w, by + h);

        SelectObject(hdc, oldBrush);
        SelectObject(hdc, oldPen);

        DeleteObject(brush);
        DeleteObject(pen);

        SetTextColor(hdc, RGB(120, 120, 120));
        TextOutW(hdc, bx + 60, by + 15, L"Сохранить", 10);

        EndPaint(hwnd, &ps);
        break;
    }

    case WM_DESTROY:
        PostQuitMessage(0);
        break;

    default:
        return DefWindowProc(hwnd, msg, wParam, lParam);
}

```

```

        return 0;
    }

    int WINAPI WinMain(HINSTANCE hInst, HINSTANCE, LPSTR, int
nCmdShow)
    {
        GdiplusStartupInput input;
        GdiplusStartup(&token, &input, NULL);

        WNDCLASSW wc = {};
        wc.lpfnWndProc = WndProc;
        wc.hInstance = hInst;
        wc.lpszClassName = L"VN";

        RegisterClassW(&wc);

        HWND hwnd = CreateWindowW(
            L"VN",
            L"Visual Novel",
            WS_OVERLAPPEDWINDOW,
            100, 100,
            1600, 900,
            NULL, NULL, hInst, NULL
        );

        ShowWindow(hwnd, nCmdShow);

        MSG msg;
        while (GetMessage(&msg, NULL, 0, 0))
        {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
        }

        GdiplusShutdown(token);
        return 0;
    }

```

Файл MediaFallback.h

```

#pragma once

#include <windows.h>
#include <gdiplus.h>
#include <shellapi.h>
#include <stdio.h>

#pragma comment(lib, "gdiplus.lib")
#pragma comment(lib, "shell32.lib")

using namespace Gdiplus;

```

```

inline void DrawImageOrVideo(HWND hwnd, Graphics& g, const
wchar_t* folder, int frame)
{
    wchar_t path[256];
    wsprintfW(path, L"%s\\%d.png", folder, frame);
    if (GetFileAttributesW(path) !=
INVALID_FILE_ATTRIBUTES)
    {
        Image img(path);
        g.DrawImage(&img, 0, 0, 1600, 900);
    }
    else
    {
        wsprintfW(path, L"%s\\%d.mp4", folder, frame);
        ShellExecuteW(NULL, L"open", path, NULL, NULL,
SW_SHOW);
    }
}

```

Файл Choices.h

```

#pragma once
#include <windows.h>

#define MAX_DEPTH 5
#define MAX_BRANCH 5

typedef struct Node
{
    int maxFrames;
    struct Node* children[MAX_BRANCH];
} Node;

// уровень 2
static Node v1_v1 = { 7, { 0 } };
static Node v1_v2 = { 6, { 0 } };
static Node v2_v1 = { 4, { 0 } };
static Node v2_v2 = { 5, { 0 } };

// уровень 1
static Node v1 = { 67, { &v1_v1, &v1_v2 } };
static Node v2 = { 56, { &v2_v1, &v2_v2 } };

// корень
static Node root = { 33, { &v1, &v2 } };

static int currentFrame[MAX_DEPTH] = { 0 };
static int selectedBranch[MAX_DEPTH] = { 0 };
static int currentLevel = 0;

static Node* GetNodeAtLevel(int level)
{
    Node* node = &root;

```

```

    for (int i = 1; i <= level; i++)
    {
        int branch = selectedBranch[i];

        if (branch < 0 || branch >= MAX_BRANCH)
            break;

        if (!node->children[branch])
            break;

        node = node->children[branch];
    }

    return node;
}

static BOOL IsEndOfBranch(int level)
{
    Node* node = GetNodeAtLevel(level);

    if (!node) return TRUE;

    return currentFrame[level] >= node->maxFrames;
}

```

Приложение В Результаты работы программы

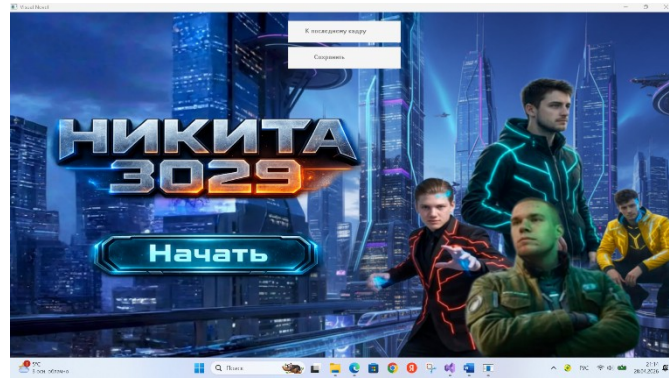


Рисунок 7 - Начальный экран

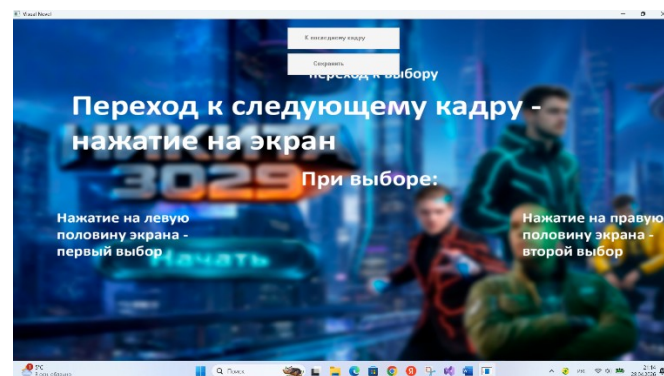


Рисунок 8 - Обучение



Рисунок 9 - Воспроизведение видео

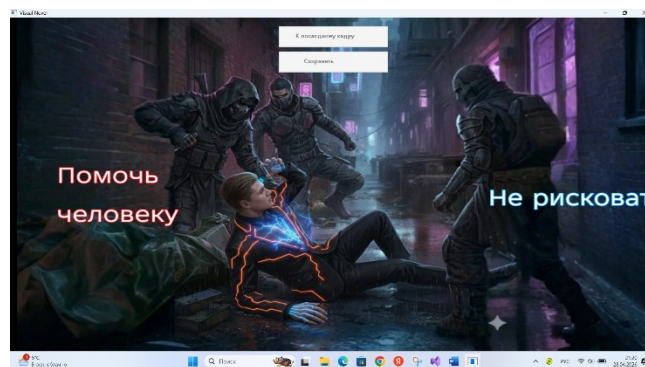


Рисунок 10 - Момент выбора

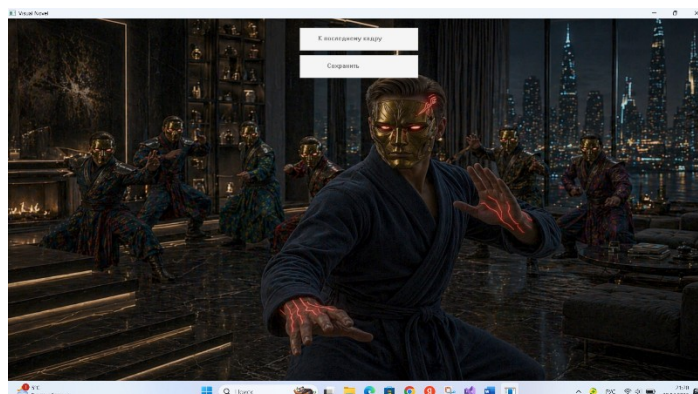


Рисунок 11 – пример кадра из игры 1

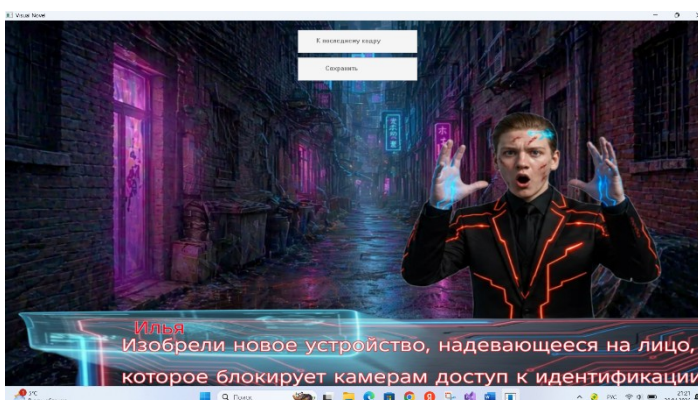


Рисунок 12 – пример кадра из игры 2

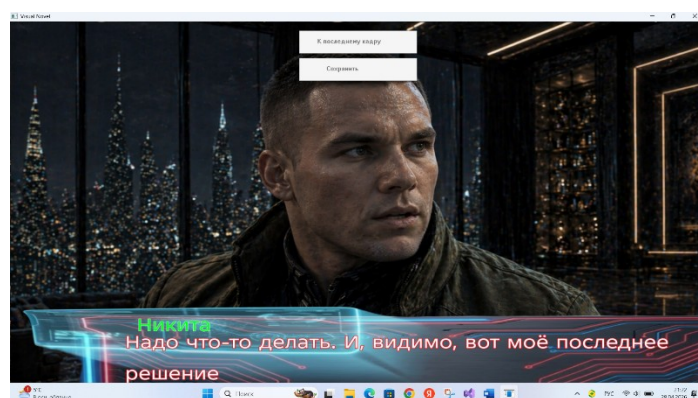


Рисунок 13 – пример кадра из игры 3