**УО «Белорусский государственный технологический университет»**

Факультет **информационных технологий**

Кафедра **информационных систем и технологий**

Специальность **1-40 05 01-03 «Информационные системы и технологии (издательско-полиграфический комплекс)»**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОЙ РАБОТЫ**

**по дисциплине «**Защита информации и надежность информационных систем»

**тема «**Реализация подстановочного алгоритма шифрования «EV-GEN» на основе таблицы Виженера»

|  |  |  |
| --- | --- | --- |
| **Исполнитель** |  | |
| Студент 3 курса группы1 | подпись, дата | Е. В. Гончаревич  инициалы и фамилия |
|  |  |  |
| **Руководитель** |  |  |
| Ассистент | подпись, дата | *Д. В. Сазонова*  инициалы и фамилия |

|  |  |
| --- | --- |
| **Курсовая работа защищена с оценкой** | **\_\_\_\_\_\_\_\_\_\_** |
| **Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  (подпись) | *Д. В. Сазонова*  инициалы и фамилия |

Минск 2023

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра информационных систем и технологий

Утверждаю

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Смелов

подпись инициалы и фамилия

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

**ЗАДАНИЕ**

**к курсовому проектированию по дисциплине**"Защита информации и надёжность информационных систем"

Специальность 1-40 05 01 03 «Информационные системы и технологии (издательско-полиграфический комплекс)»

Группа: 1

Студент: Гончаревич Евгений Витальевич

**Тема: «**Реализация подстановочного алгоритма шифрования «EV-GEN» на основе таблицы Виженера»

**1. Срок сдачи студентом законченной работы**: «10» мая 2023г.

**2. Исходные данные к проекту:**

**2.1**. Функционально должны быть выполнены следующие задачи:

* Разработать криптографический метод шифрования/расшифрования сообщения, основываясь на алгоритме таблицы Виженера;
* Разработать программное средство реализации данного метода или алгоритма;
* Реализовать в программе 3D-демонстрацию работы алгоритма;
* Составить руководство пользователя.

**2.2. Требования:**

* Необходимо провести аналитический обзор литературы по теме проекта
* Необходимо описать сферу применимости метода
* Программное средство может быть разработано на любом языке
* Архитектура приложения выбирается разработчиком
* Листинги проекта должны содержать комментарии

**3. Содержание расчётно-пояснительной записки:**

* Введение
* Постановка задачи
* Описание метода
* Описание программного средства
* Тестирование программного средства
* Руководство пользователя
* Заключение
* Список используемых источников
* Приложения

**4. Форма представления выполненного курсового проекта:**

* + Теоретическая часть курсового проекта должна быть представлена в формате MS Word.
  + Оформление записки должно быть согласно правилам.
  + Необходимые схемы, диаграммы и рисунки допускается делать в MS Office Visio или копии экрана (интерфейс).
  + Полные листинги проекта представляются в приложении.
  + К записке необходимо приложить CD (DVD), который должен содержать: пояснительную записку, листинги и файлы базы данных.

#### Календарный план

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 17.02.2023 |  |
| 2 | Аналитический обзор литературы по теме проекта | 23.02.2023 |  |
| 3 | Разработка метода | 14.03.2023 |  |
| 4 | Разработка прототипа программного средства | 21.03.2023 |  |
| 5 | Разработка программного средства | 27.03.2023 |  |
| 6 | Тестирование программного средства | 12.04.2023 |  |
| 7 | Написание руководства пользователя | 17.04.2023 |  |
| 8 | Оформление пояснительной записки | 25.04.2023 |  |
| 9 | Сдача проекта | 10.05.2023 |  |

**5. Дата выдачи задания** «15» февраля 2023г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Д. В. Сазонова*

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата и подпись студента)

[Оглавление 5](#_Toc27730)

[Введение 7](#_Toc17545)

[1 Аналитический обзор литературы 8](#_Toc12794)

[1.1 Обзор аналогов 8](#_Toc15705)

[1.2 Вывод по разделу 12](#_Toc15958)

[2 Проектирование программного средства 13](#_Toc14209)

[2.1 Выбор технологий и средств 13](#_Toc26447)

[2.2 Исследование алгоритма 13](#_Toc5863)

[2.3 Оценка скорости работы 14](#_Toc22602)

[2.4 Вывод по разделу 15](#_Toc15374)

[3 Разработка программного средства 16](#_Toc29717)

[3.1 Описание программного средства 16](#_Toc21834)

[3.2 Создание интерфейса приложения 16](#_Toc23058)

[3.3 Структура проекта в среде Unity 17](#_Toc3863)

[3.4 Структура проекта в среде Visual Studio 18](#_Toc28875)

[3.5 Вывод по разделу 19](#_Toc6107)

[4 Тестирование программного средства 20](#_Toc4819)

[4.1 Тестирование функций приложения 20](#_Toc4965)

[4.2 Вывод по разделу 23](#_Toc14196)

[5 Руководство пользователя 24](#_Toc11109)

[5.1 Вывод по разделу 28](#_Toc31460)

[Заключение 29](#_Toc26465)

[Список используемых источников 30](#_Toc14896)

[ПРИЛОЖЕНИЕ А 31](#_Toc15713)

[ПРИЛОЖЕНИЕ Б 33](#_Toc13222)

[ПРИЛОЖЕНИЕ В 35](#_Toc14650)

[ПРИЛОЖЕНИЕ Г 36](#_Toc30593)

Введение

Шифр Виженера - это классический метод шифрования, который использует ключевое слово для зашифровки исходного текста. В отличие от простой замены символов, этот шифр использует повторяющиеся ключевые слова для создания сложных шифрованных сообщений, что делает его более стойким к взлому. Сегодня, в эпоху быстрого развития технологий и массового доступа к интернету, шифр Виженера становится все более популярным методом защиты информации. В связи с этим, развитие новых методов криптоанализа и создание более продвинутых версий этого шифра становятся актуальными задачами.

Целью данного курсового проекта является модификация классического шифра Виженера для обеспечения дополнительной стойкости и безопасности передаваемых данных. Это будет достигнуто путем добавления дополнительных параметров для зашифровки и расшифровки сообщения, чтобы сделать атаки по статистике более сложными. Кроме того, будет разработано программное средство, которое демонстрирует работу модифицированного шифра и позволяет проводить тестирование его стойкости и эффективности.

На основе поставленной цели были выделены следующие задачи:

* разработать криптографический метод шифрования/расшифрования сообщения, основываясь на алгоритме таблицы Виженера;
* разработать программное средство реализации данного метода или алгоритма;
* реализовать в программе 3D-демонстрацию работы алгоритма;
* составить руководство пользователя.

# Аналитический обзор литературы

## Обзор аналогов

В данной главе приведены аналоги шифра, основанные на использовании подстановок.

Шифр Цезаря – это классический метод шифрования, который был использован в Древнем Риме. Он основывается на замене каждой буквы в сообщении на букву, находящуюся на несколько позиций дальше в алфавите. Шифр Цезаря легко взламывается, но может быть использован в качестве базового алгоритма для более сложных методов шифрования. Пример использования шифра Цезаря представлен на рисунке 1.1.



Рисунок 1.1 – Пример использования шифра Цезаря.

Преимущества шифра Цезаря:

* Простота использования: Шифр Цезаря очень прост в реализации и использовании. Он не требует сложных математических операций или специального оборудования;
* Быстрота шифрования и дешифрования: Процесс шифрования и дешифрования с помощью шифра Цезаря происходит быстро, так как требуется только сдвиг букв в алфавите;
* Интуитивность: Шифр Цезаря легко понять и запомнить. Для шифрования или дешифрования текста не требуется сложная логика или ключевые фразы.

Недостатки:

* Низкая стойкость к криптоанализу: Шифр Цезаря имеет очень низкую стойкость к криптоанализу. Из-за небольшого количества возможных ключей (сдвигов), он легко подвержен методам перебора и частотного анализа;
* Низкая стойкость к статистическому анализу: Шифр Цезаря не меняет частотность символов в зашифрованном тексте. Это делает его уязвимым к статистическому анализу, основанному на распределении символов в языке.

Шифр Плейфера – это метод шифрования, который использует таблицу, заполненную случайными буквами, для зашифровки сообщения. Он заменяет каждую букву сообщения на пару букв, определенных ее позицией в таблице. Шифр Плейфера является более стойким, чем Шифр Цезаря, но может быть взломан методами частотного анализа. Пример шифрования представлен на рисунке 1.2.

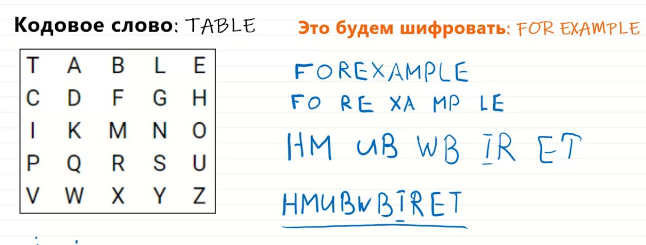


Рисунок 1.2 – Пример использования шифра Плейфера.

Преимущества:

* Устойчивость к частотному анализу: Шифр Плейфера заменяет символы исходного текста на пары символов (диграммы) из шифровальной таблицы. Это затрудняет проведение частотного анализа, так как каждый символ представлен парой;
* Эффективное использование ключа: Шифр Плейфера требует ключа в виде шифровальной таблицы, что позволяет использовать более длинные ключи и повышает степень стойкости шифра;
* Относительная простота реализации: Шифр Плейфера не требует сложных математических операций. Он может быть реализован с помощью таблицы и правил замены, что делает его относительно простым в использовании.

Недостатки:

* Зависимость от шифровальной таблицы: Шифр Плейфера полностью зависит от использования правильной шифровальной таблицы. Если таблица потеряется или будет использована неправильная таблица, то процесс расшифрования станет затруднительным или невозможным;
* Ограниченное пространство символов: Шифр Плейфера оперирует парами символов (диграммами), что ограничивает количество возможных символов, которые могут быть зашифрованы. Это может привести к потере некоторых символов, например, знаков пунктуации или специальных символов.

Шифр Бэкона – это шифр, который использует алфавит, состоящий из 24 символов, для шифрования и расшифрования сообщения. Этот метод шифрования был использован в прошлом для передачи секретных сообщений, но сейчас считается устаревшим. Таблица кодирования представлена на рисунке 1.3.

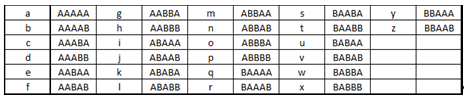


Рисунок 1.3 – Таблица кодирования шифра Бэкона.

Преимущества:

* Простота использования - не требуется никаких специальных знаний для шифрования и дешифрования сообщений;
* Эффективность - шифр Бэкона может быть быстро выполнен вручную;
* Высокая степень секретности - использование групп символов вместо отдельных букв усложняет процесс расшифровки сообщений.

Недостатки:

* Шифр Бэкона является очень уязвимым для частотного анализа - злоумышленник может легко определить, что каждая группа символов соответствует конкретной букве, что сильно упрощает процесс расшифровки сообщений;
* Шифр Бэкона может занимать много места для хранения и передачи больших сообщений, так как каждая буква заменяется на пять символов.

Шифр Виженера - это метод полиалфавитного шифрования, в котором каждая буква открытого текста заменяется на букву шифрованного текста с использованием ключевого слова или фразы. Шифр Виженера является криптографическим шифром, который обеспечивает дополнительный уровень безопасности по сравнению с простыми моноалфавитными шифрами замены. На рисунке 1.4 пример использования шифра.



Рисунок 1.4 – Пример зашифрования шифром Виженера

Преимущества:

* Ключевое слово обеспечивает повышенную стойкость - использование ключевого слова или фразы влияет на последовательность применяемых шифровальных алфавитов, что делает шифр Виженера более устойчивым к частотному анализу и другим методам криптоанализа;
* Периодичность шифровальной последовательности - шифр Виженера обладает периодичностью в своей шифровальной последовательности, что добавляет дополнительный слой защиты от анализа статистики;
* Относительная простота реализации - шифр Виженера можно реализовать сравнительно просто и эффективно, используя стандартные алгоритмы и программные инструменты.

Недостатки:

* Восстановление ключа - если длина ключа известна, то шифр Виженера можно легко взломать с использованием методов криптоанализа, основанных на длине периода шифровальной последовательности;
* Зависимость от длины ключа - для достижения максимальной стойкости, длина ключа должна быть не короче длины шифруемого сообщения. Это может создавать проблемы, особенно при передаче или хранении больших объемов данных;



## Вывод по разделу

Таким образом, результатом данного раздела стал анализ подстановочных методов шифрования. На основе выделенных преимуществ и недостатков, был разработан метод «EV-GEN» (Encoded Vector Generation), сочетающий в себе преимущества других аналогов. Проектирование, разработка и тестирование программного средства будет описана в следующих главах.

# Проектирование программного средства

## Выбор технологий и средств

Unity – это кроссплатформенный игровой движок, который позволяет разрабатывать игры и визуализации для различных платформ, включая компьютеры, мобильные устройства, консоли и виртуальную реальность. С помощью этого движка будет создаваться интерфейс и остальная визуальная часть. Так же движок будет инкапсулировать работу с кодом, такую как генерацию событий, рендер графики и т.д. В конце, движок соберет весь проект в одно приложение, запускаемое через exe-файл.

Blender – профессиональное свободное и открытое программное обеспечение для создания трёхмерной компьютерной графики, включающее в себя средства моделирования, скульптинга, анимации, симуляции, рендеринга, постобработки и монтажа видео со звуком, компоновки с помощью «узлов», а также создания 2D-анимаций. Будет использоваться для создания моделей букв и цифр, которые будут использоваться в качестве демонстаций алфавитов в 3D.

C# – это объектно- и компонентно-ориентированный язык программирования. C# предоставляет языковые конструкции для непосредственной поддержки такой концепции работы. Благодаря этому C# подходит для создания и применения программных компонентов.

IDE Visual Studio – это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Интегрированная среда разработки (IDE) представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения.

Git – это распределенная система контроля версий (version control system – VCS). Контроль версий означает что вы храните все версии редактируемых документов и можете вернуться к любой сохраненной версии в любой момент времени.

## Исследование алгоритма

Подстановочный алгоритм шифрования «EV-GEN» основывается на шифре Виженера, но имеет ряд модификаций. Стандартный шифр использует один ключ для шифрования, тогда как модификация добавляет ещё три параметра:

* шаг, s (step), число;
* направление шифрования, D (direction). Принимает значения “R”, ”L”, ”T”, ”B”, которые влияют на знак шага s (делают положительным или отрицательным).
* глубина шифрования, d (depth), число.

Для зашифрования используется формула 1.1:

(1.1)

В данном выражении присутствуют следующие параметры:

* сi соответствует i-й букве шифртекста;
* mi соответствует i-й букве сообщения;
* ki соответствует i-й букве ключа;
* d соответствует глубине;
* s соответствует шагу (зависит от параметра D. Если D равен “R” или “B”, то s число положительное. Иначе число отрицательное);
* N соответствует длине алфавита.

Формула расшифрования (1.2) выглядит следующим образом:

(1.2)

В данном выражении присутствуют следующие параметры:

* сi соответствует i-й букве шифртекста;
* mi соответствует i-й букве сообщения;
* ki соответствует i-й букве ключа;
* d соответствует глубине;
* s соответствует шагу (зависит от параметра D. Если D равен “R” или “B”, то s число отрицательное. Иначе число положительное);
* N соответствует длине алфавита.

Шифртекст имеет структуру, которую можно представить набором символов CKdDs, где:

* С соответствует зашифрованному тексту;
* К соответствует ключу;
* d соответствует глубине кодирования;
* D соответствует направлению кодирования;
* S соответствует шагу кодирования.

## Оценка скорости работы

Замеры будут производиться с текстами размером 5, 50, 250, 1000, 5000 символов. Для визуализации результатов был построен график в Microsoft Excel.

Итоги оценки скорости алгоритма представлены в таблице 2.1.

Таблица 2.1 – Оценка скорости работы алгоритма

|  |  |  |
| --- | --- | --- |
| Количество символов в тексте | Время расшифрования (сек) | Время зашифрования (сек) |
| 1500 | 0,06 | 0,01 |
| 2000 | 0,07 | 0,012 |
| 2500 | 0,012 | 0,016 |
| 3000 | 0,016 | 0,017 |
| 3500 | 0,017 | 0,02 |

Результаты представим в виде графика (рисунок 2.1):

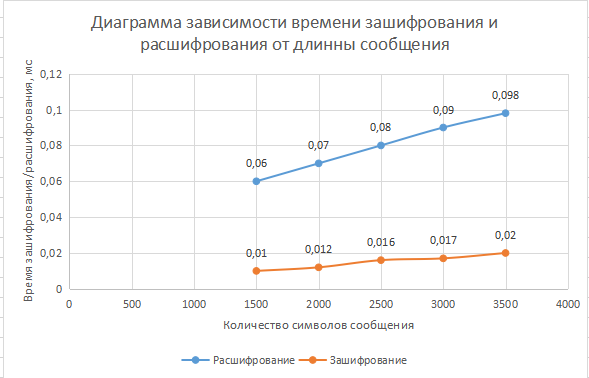


Рисунок 2.1 – График зависимости времени шифрования от размера входных данных

Видно, что производительность алгоритма очень слабо зависит от размера входных данных. Чем больше символов нужно зашифровать, тем медленнее происходит их шифрование.

Стоит так же отметить, что по сравнению с классическим шифром Виженера, алгоритм «EV-GEN» повышает криптостойкость сообщения и при этом практически не увеличивает длину шифртекста.

## Вывод по разделу

## Проведенный анализ алгоритма показал, что, несмотря на незначительное увеличение времени шифрования и дешифрования, модифицированный алгоритм обладает более высокой криптостойкостью. Это связано с тем, что к шифртексту добавляются дополнительные параметры.

# Разработка программного средства

## Описание программного средства

## Данное программное средство представляет собой оконное приложение (визуализацию), созданное с помощью игрового движка Unity и среды разработки Visual Studio. В проекте использовался язык программирования C#. В дополнение к этому, в приложении реализована демонстрация работы алгоритма, использующая трехмерные матрицы символов. Символы для матриц были смоделированы в ПО Blender. Древо проекта показана на рисунке 3.1

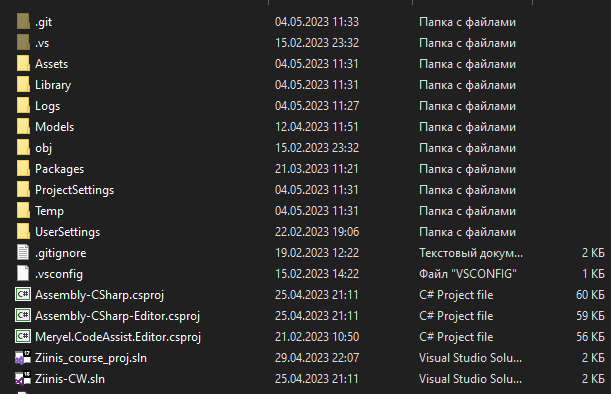


Рисунок 3.1 – Древо проекта

Следует также отметить, что при разработке крупных приложений неизбежны ситуации, когда в процессе работы возникают ошибки. Например, файлы могут быть повреждены, что приводит к невозможности продолжения работы, или разработчик может написать код, который не работает из-за запутанности. Поэтому, во избежание подобных ошибок, в проекте была задействована система контроля версий Git, позволяющая в любой момент откатить изменения к предущему стабильному состоянию.

## Создание интерфейса приложения

Движок Unity имеет обширные возможности для создания интерфейса приложения. В частности, элемент Canvas, который хранит в себе все элементы, начиная кнопками и заканчивая текстовыми полями. Благодаря ему, интерфейс остается статичной картинкой даже при движении внутриигровой камеры.

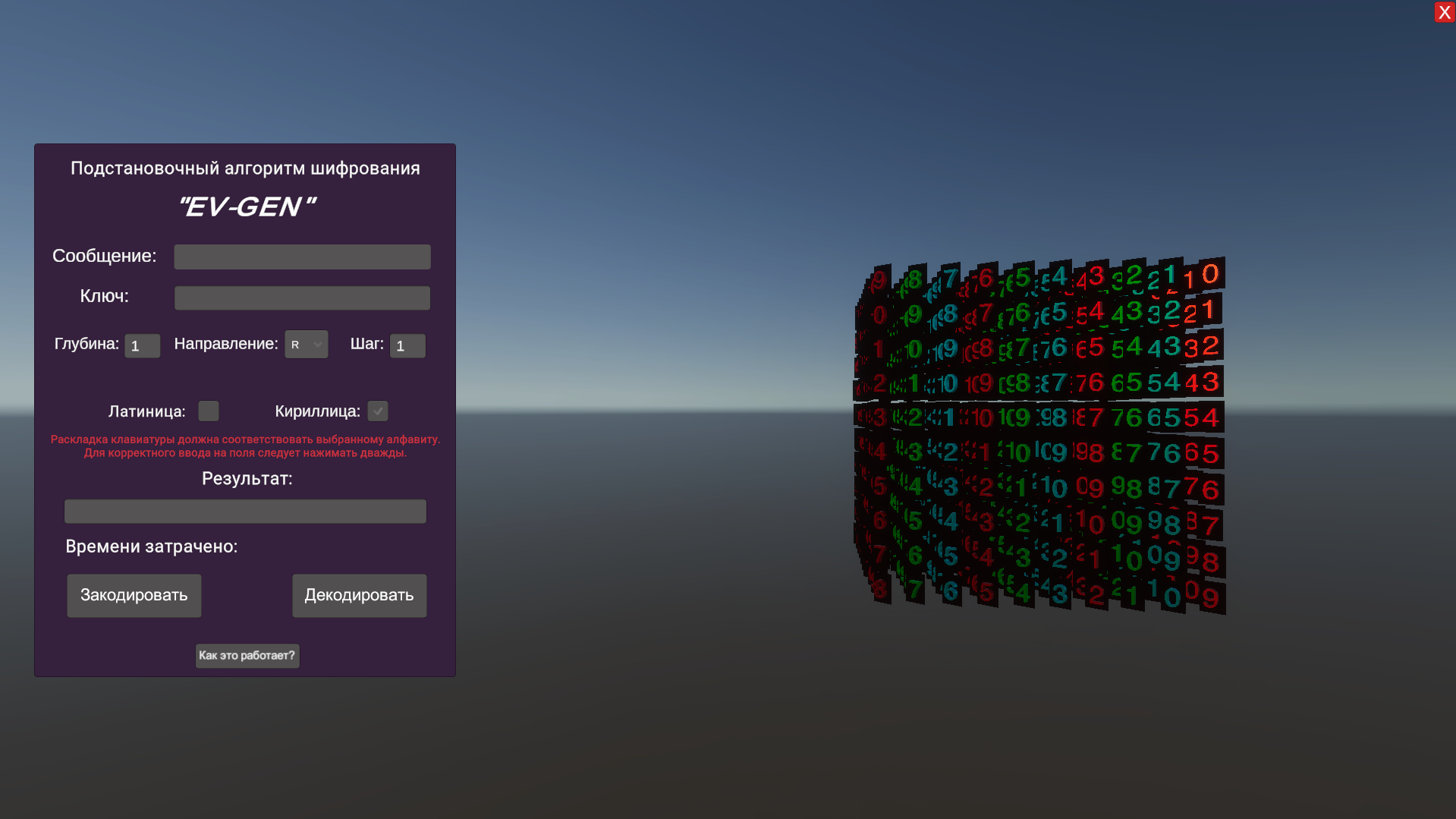


Рисунок 3.2 – Интерфейс приложения

## Структура проекта в среде Unity

Разработка приложения состоит из двух этапов: проектирование интерфейса в среде Unity и написание программного кода в среде Visual Studio. У каждой среды своя структура и иерархия элементов. На рисунке 3.3 показана структура проекта в среде Unity.

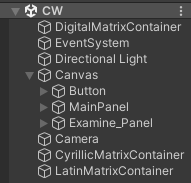


Рисунок 3.3 – Структура проекта Unity

В данной иерархии есть несколько элементов, на которых стоит остановиться. Элемент Canvas хранит в себе три элемента:

* Button – кнопка для выхода из приложения;
* MainPanel – панель, которая хранит все элементы для работы с шифрованием и расшифрованием текста, а также кнопку для перехода на демонстрационный режим;
* Examine\_Panel – панель, которая хранит элементы, необходимые для работы с демонстрацией работы алгоритма, а также кнопку для возврата в рабочий режим.

Элементы Digital-, Cyrillic- и LatinMatrixContainer используются для хранения элементов из алфавитов, соответствующих префиксам матриц.

На рисунке 3.4 показано, как выглядит элемент MainPanel в редакторе.

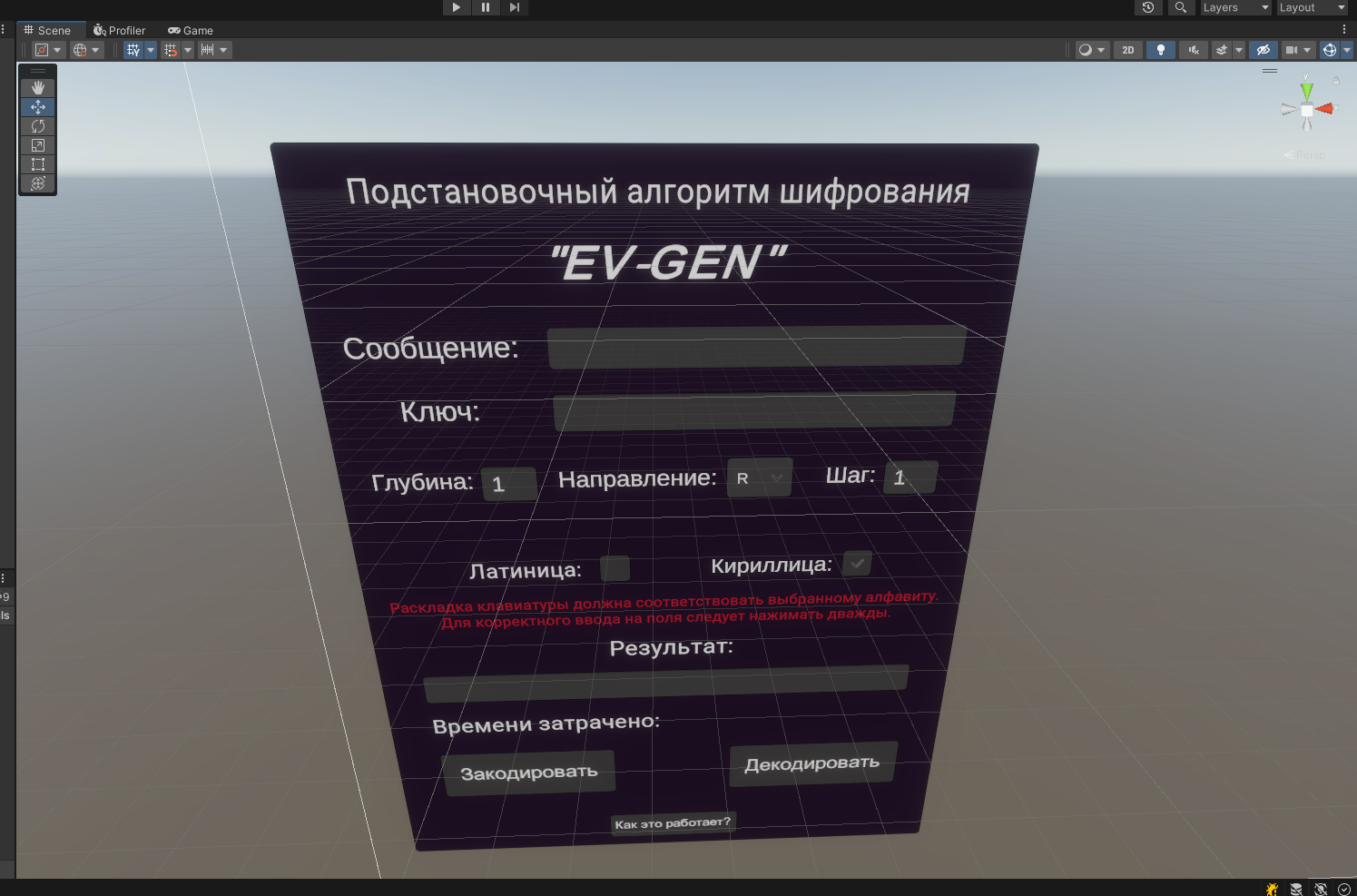


Рисунок 3.4 – Элемент MainPanel в редакторе

## Структура проекта в среде Visual Studio

На рисунке 3.5 показана структура проекта в среде Visual Studio.

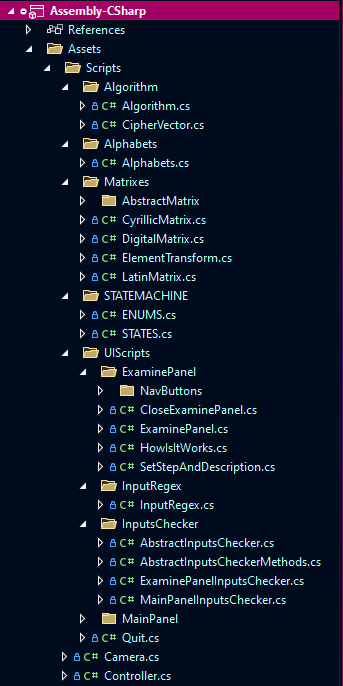


Рисунок 3.5 – Структура проекта Unity

Самыми важными элементами в иерархии являются классы Algorithm, Alphabets и CipherVector. Класс Algorithm хранит в себе логику по кодированию текста. Alphabets – класс, хранящий два алфавита: русский и английский. CipherVector – класс, объекты которого хранят информацию о шифртексте. Листинги классов Algorithm, Alphabets и CipherVector приведены в приложениях А, Б и В соответственно.

## Вывод по разделу

Таким образом, результатом данного раздела стала разработка приложения, которое шифрует и расшифровывает текст, а так же демонстрирует работу алгоритма на примере трехмерных матриц.

# Тестирование программного средства

## Тестирование функций приложения

На данном этапе курсового проекта проведем тестирование разработанного приложения, чтобы удостовериться в правильности выполнения функция, а также проверить программу на отказоустойчивость.

В проекте есть несколько мест, где пользователь может ввести некорректные данные. Это поля с сообщением, ключом, глубиной, направлением и шагом. На рисунке 4.1 эти места показаны.

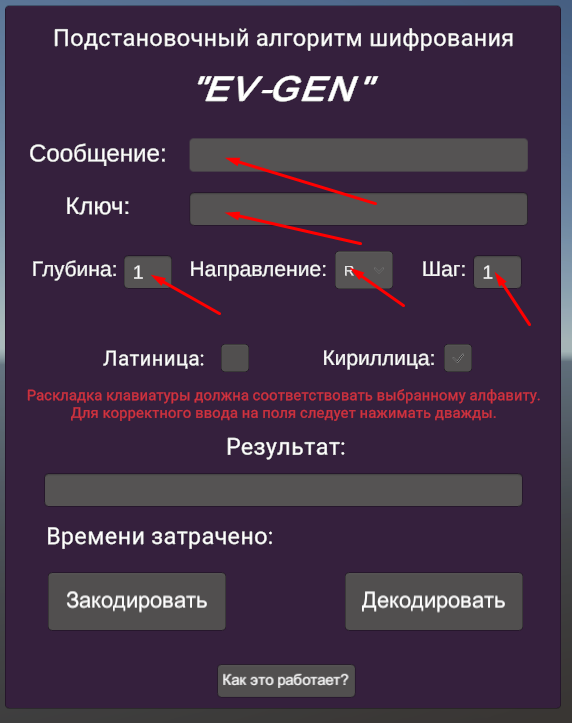


Рисунок 4.1 – Уязвимые места

Во избежание ошибок было принято решение использовать регулярные выражения, которые будут, при случае ввода невалидных данных, игнорировать ввод пользователя. Код класса с регулярными выражениями приведен в приложении Г.

Регулярные выражения применяются ко всем полям, исключая направление, так как оно является списком с заранее заданными параметрами.

Ещё одно действие, способное сломать программу, это ввод текста на языке, противоположном выбранному. Поэтому, были приняты следуюшие решения:

* Ввод сообщения и ключа проверяется регулярными выражениями, и если введенный символ не соответствует выбранному алфавиту, то он не вводится в поля;
* При смене алфавита поля ключа и сообщения очищаются, что можно увидеть на рисунках;
* Для информирования пользователя добавлена подсказка, информирующая о том, что раскладка клавиатуры должна соответствовать алфавиту.

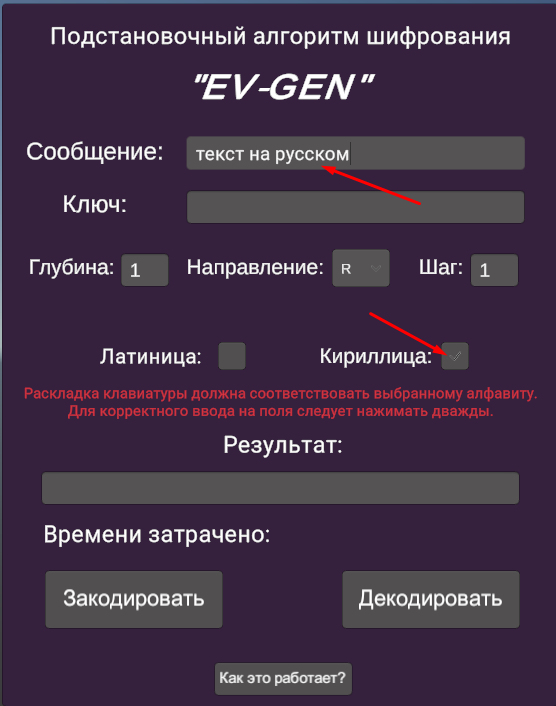


Рисунок 4.2 – Поле с сообщением до смены алфавита

Рисунок 4.3 показывает состояние приложения при смене алфавита.

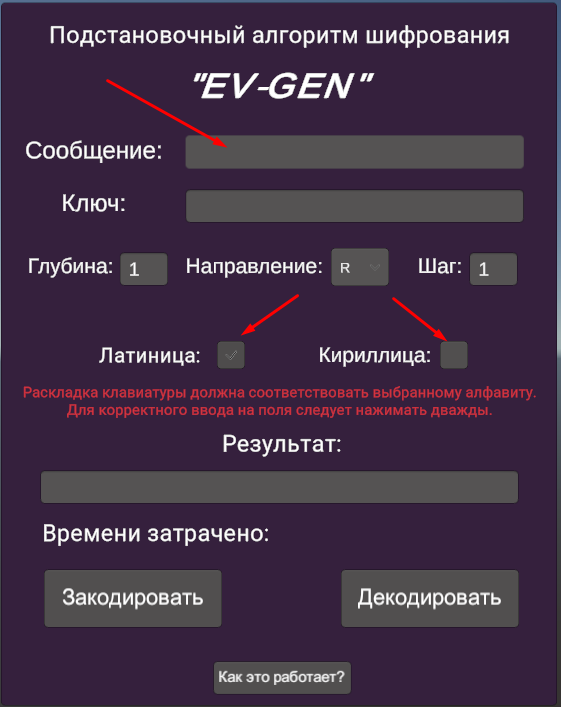


Рисунок 4.3 – Поле с сообщением после смены алфавита

На рисунке 4.4 показана подсказка о том, что раскладка клавиатуры должна соответствовать выбранному алфавиту.

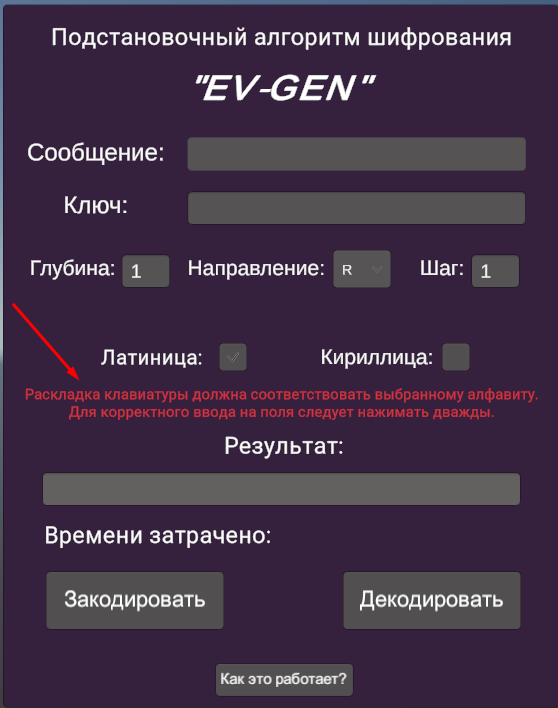


Рисунок 4.4 – Подсказка о раскладке клавиатуры

Ещё одна уязвимость программы в том, что пользователь, помимо ввода некорректных данных, может отправить пустое сообщение или ключ.. Отправка пустого сообщения, в отличие от пустого ключа, не является ошибкой. Но для оптимизации программы, как для ключа, так и для сообщения был придуман механизм, который блокирует отправку пустого сообщения и ключа. Состояние программы при отправке пустых сообщений показано на рисунке 4.5.

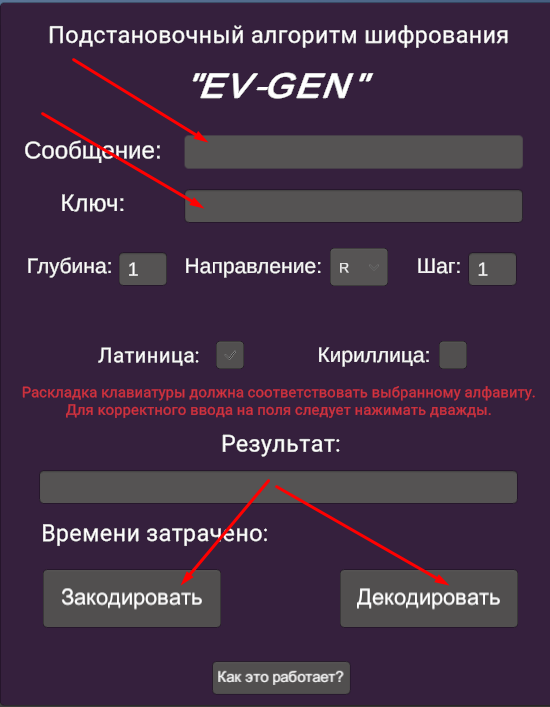


Рисунок 4.5 – Состояние приложения при отправке пустых полей

## Вывод по разделу

Таким образом, результатом данного раздела стало тестирование работоспособности разработанных функций приложения. Были проверены функции ввода данных. Каждая функция работает, как и задумывалось. При вводе некорректных данных механизм, в виде регулярных выражений, обрабатывает ошибки.

# Руководство пользователя

Основной вид приложения изображен на рисунке 5.1.

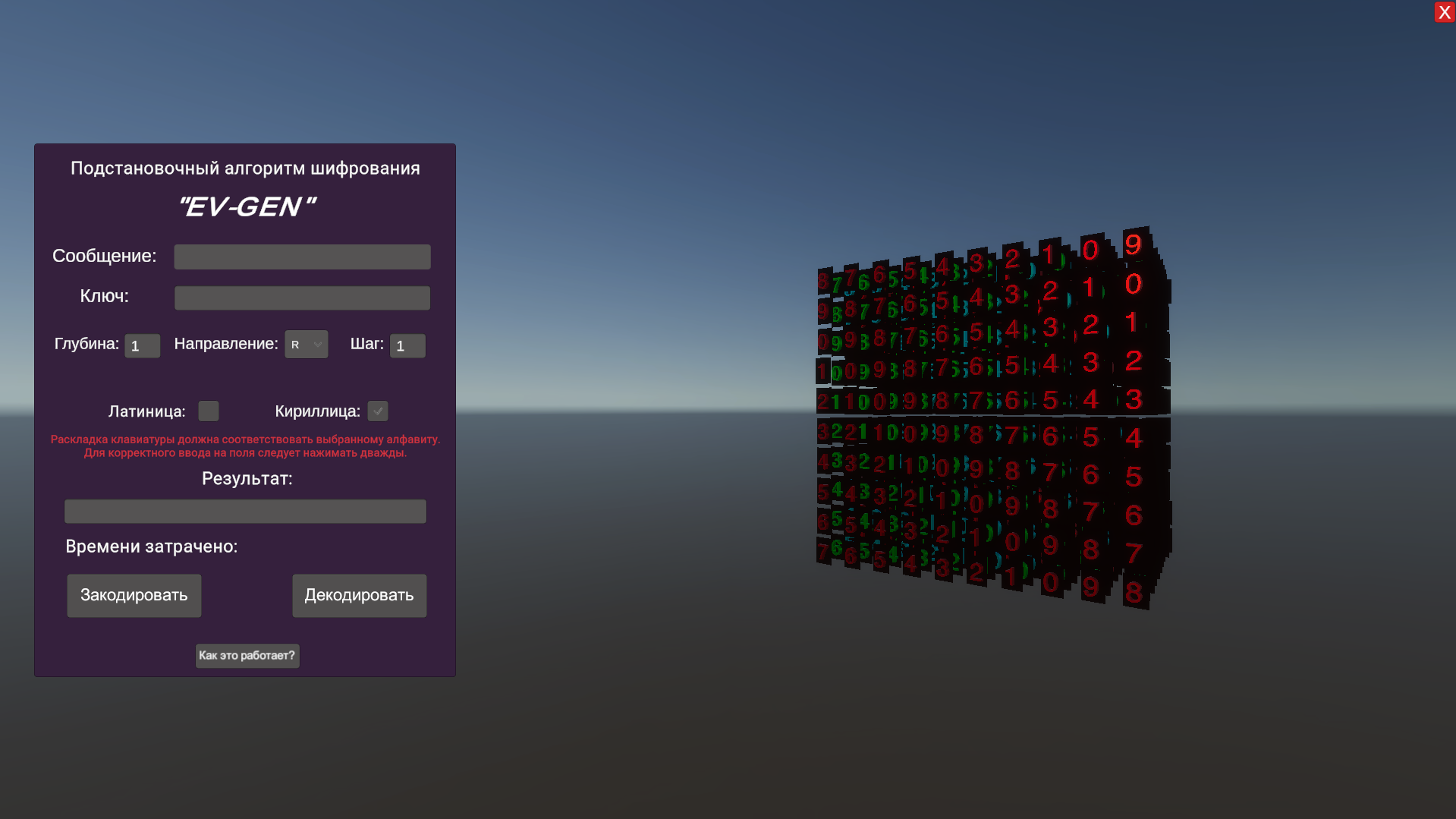


Рисунок 5.1 – Основной вид приложения

В рабочем режиме, для работы с текстом следует выбрать алфавит (например, русский, он же кириллица), указать сообщение, ключ, глубину, направление и шаг. Пример данных показан на рисунке 5.2.

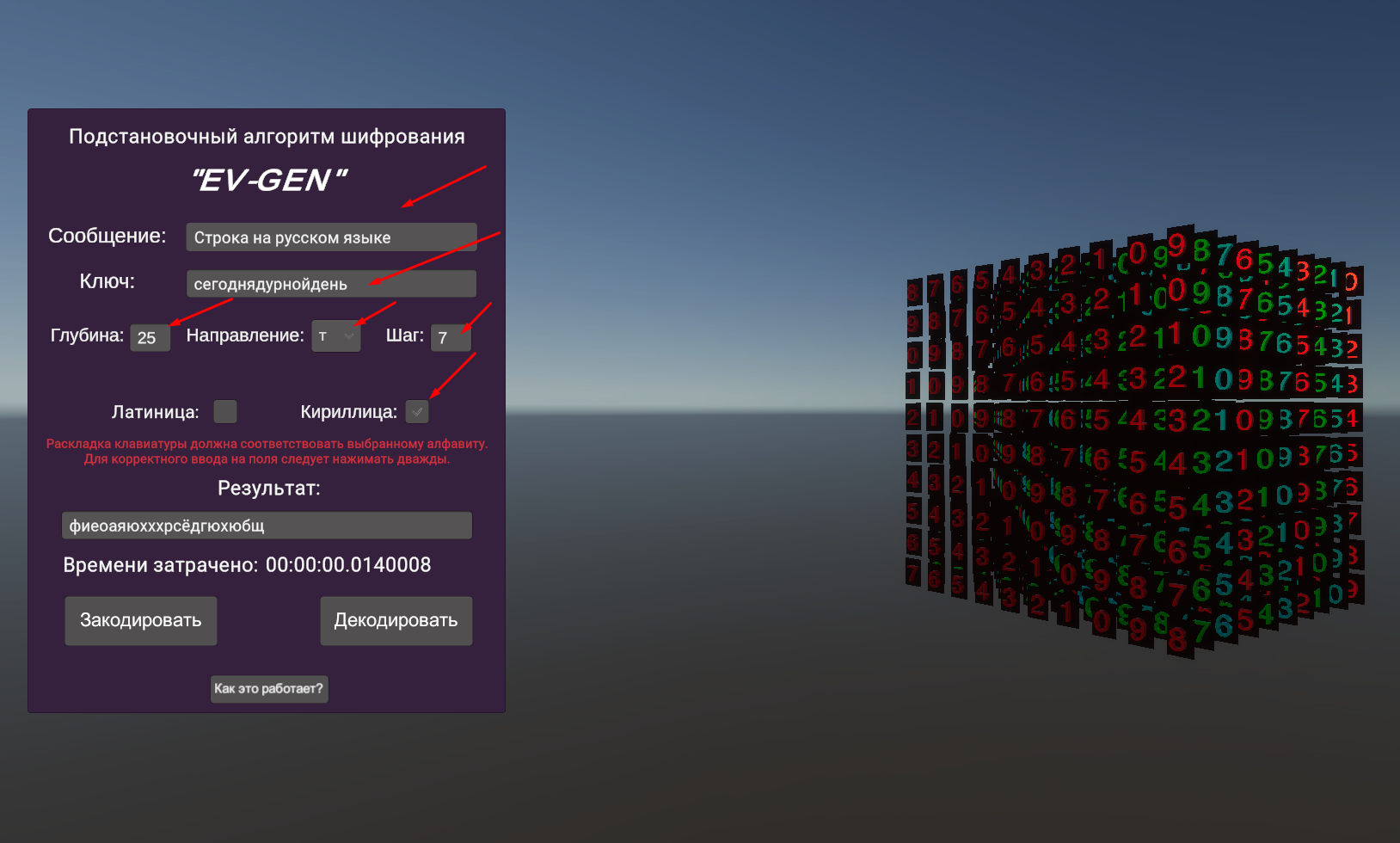


Рисунок 5.2 – Пример данных для обработки

Далее следует выбрать что делать с сообщением. Для кодирования и декодирования нужно нажать соответствующие кнопки. Пример результата кодирования приведен на рисунке 5.3.

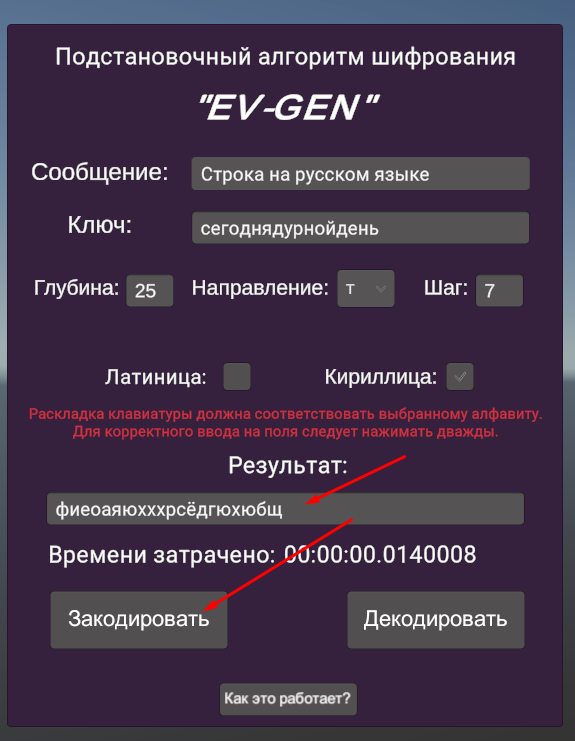


Рисунок 5.3 – Пример результата декодирования

При нажатии на кнопку «Как это работает» программа переходит в режим демонстрации работы алгоритма на выбранном алфавите. Вид интерфейса при переходе в данный режим показан на рисунке 5.4.

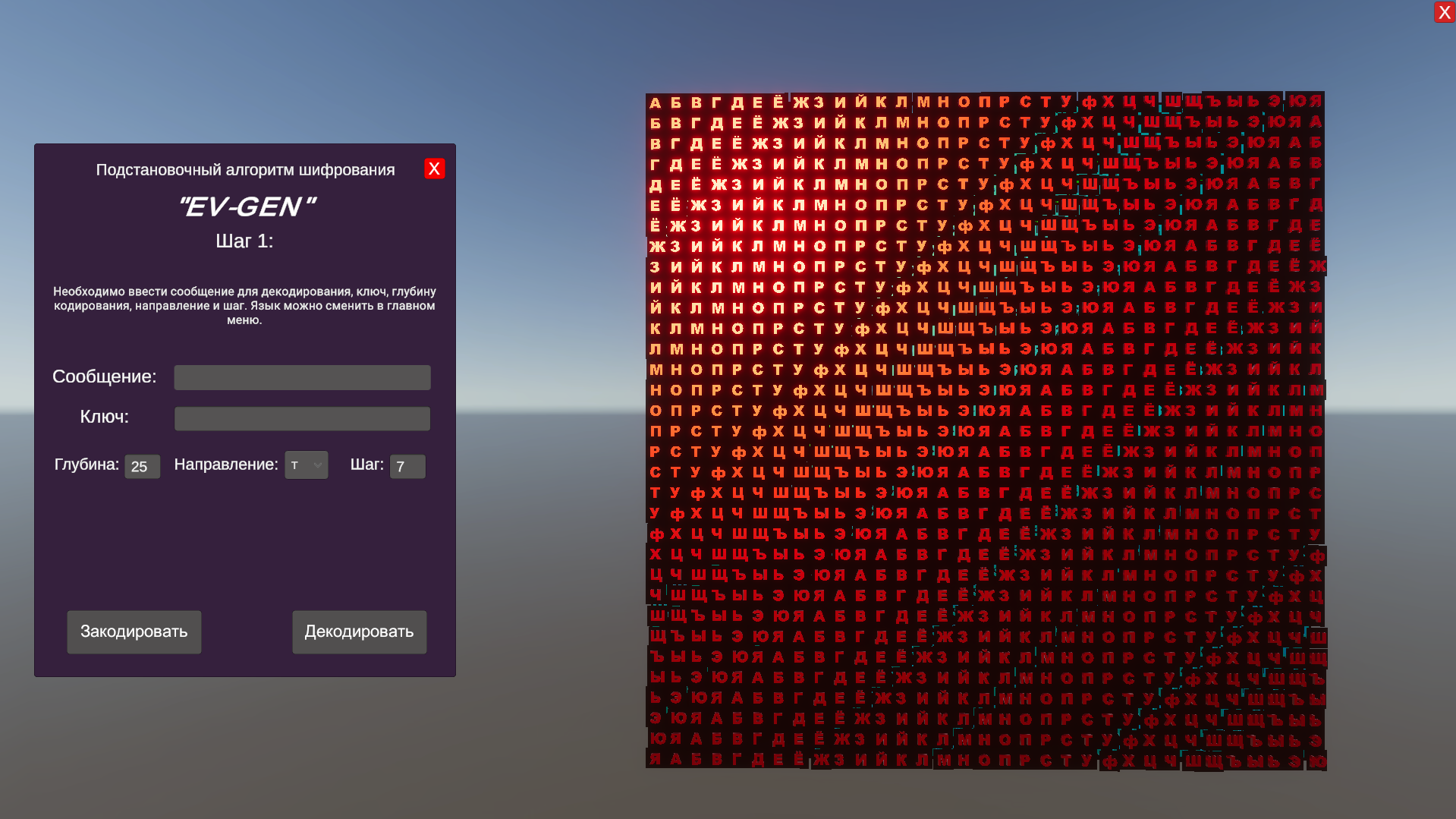


Рисунок 5.4 – Интерфейс при переходе в режим демонстрации

Демонстрационный режим разбит на шаги. Первый шаг не отличается от действий в рабочем режиме, но при нажатии кнопки кодирования или декодирования программа переходит на следующий шаг. В общем случае, шагов 5. Но шаги 2–5 повторяются столько раз, сколько символов содержит сообщение. Также, на шагах 2–5 на матрице символов, отображенной справа, подсвечивается текущая буква шифртекста. При этом, в зависимости от глубины, меняется слой матрицы. Пример подсветки буквы показан на рисунке 5.5.

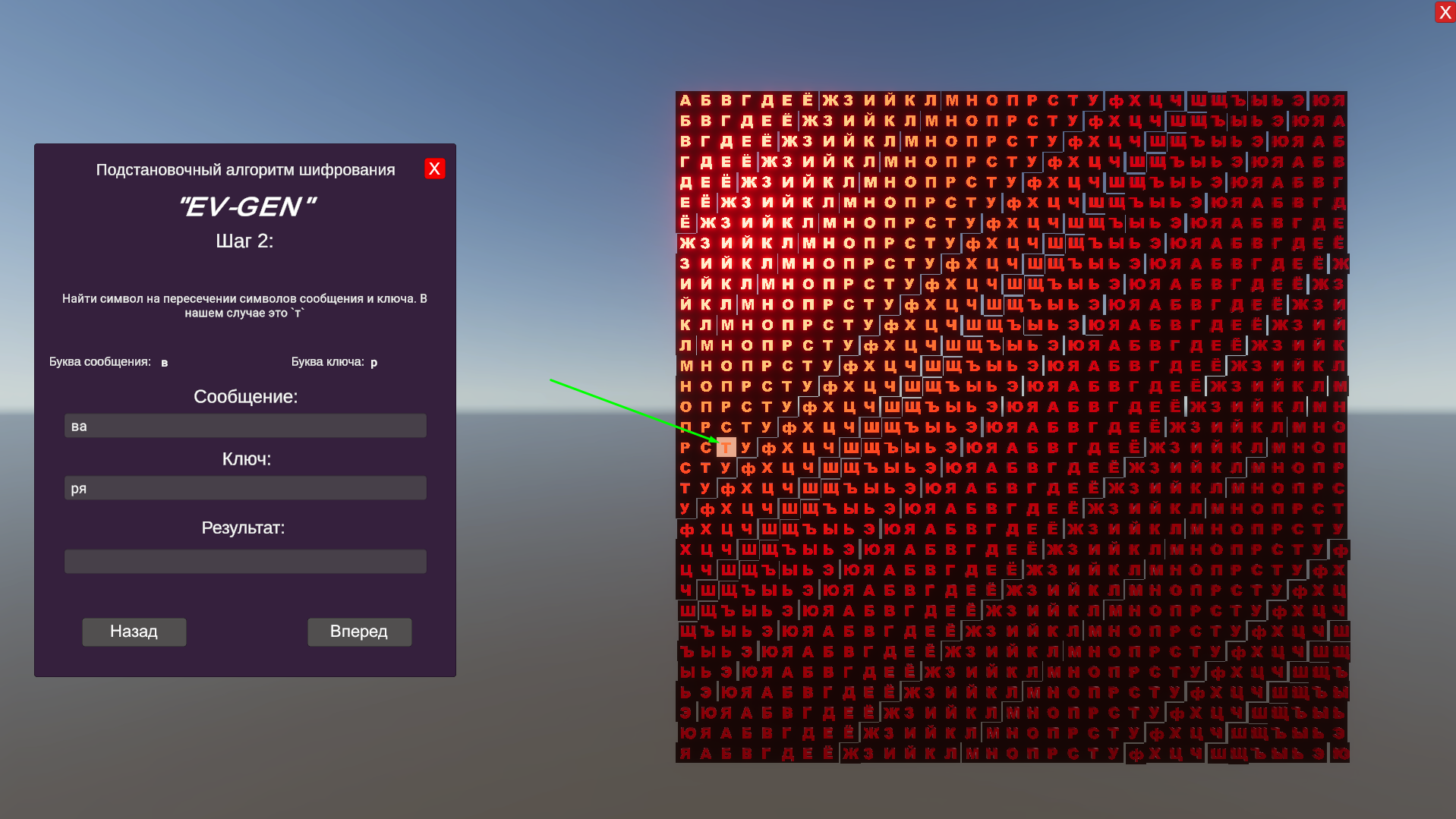


Рисунок 5.5 – Подсветка текущей буквы шифртекста

На рисунке 5.6 показано то, как выглядит матрица при изменении глубины кодирования.

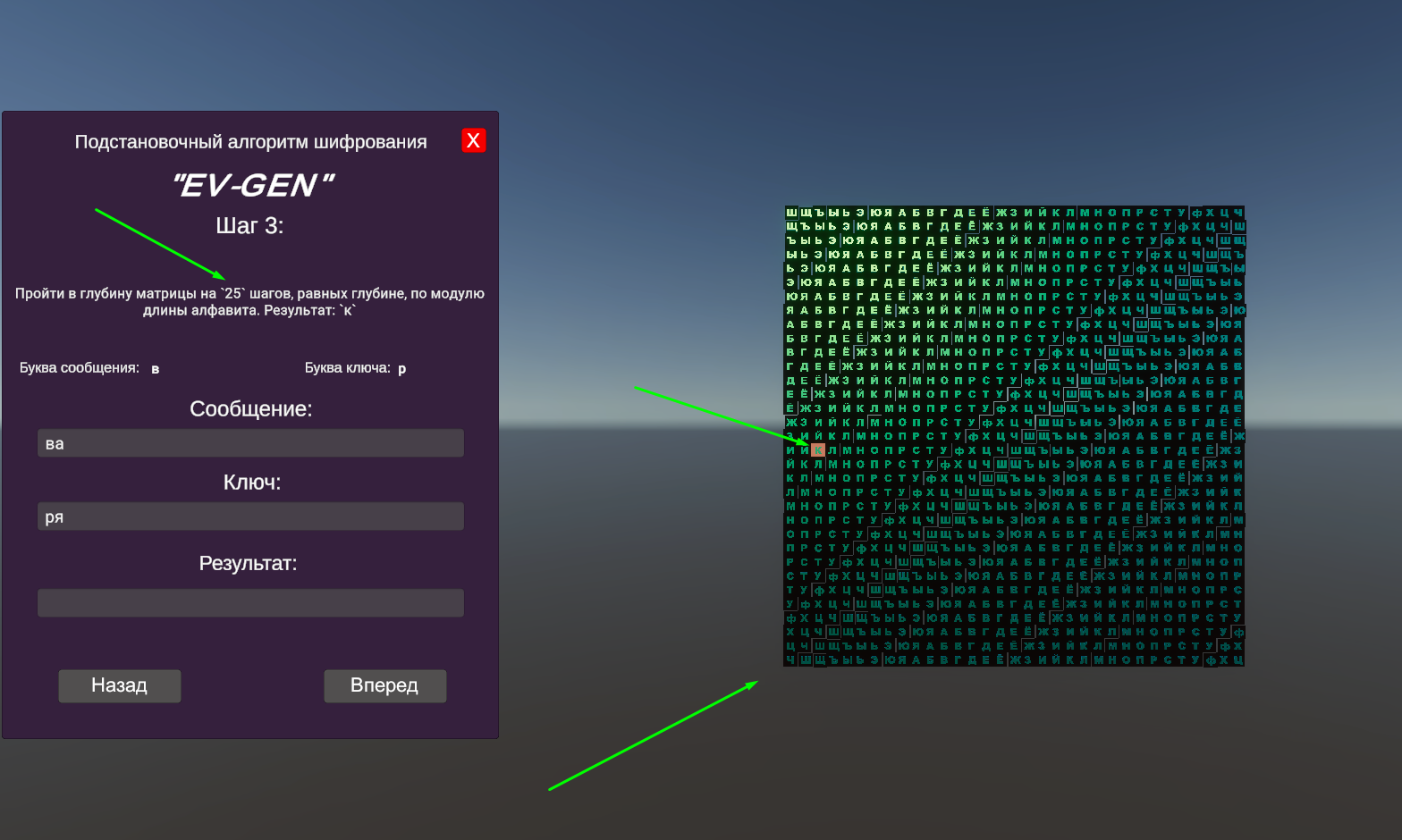


Рисунок 5.6 – Вид матрицы при изменении глубины

Каждый шаг программы приводит пояснение к действиям, поэтому, в паре с подсветкой букв, достаточно легко понять, что происходит на каждом шаге.

При выборе английского алфавита все указанные действия аналогичны указанным, но матрица состоит из английских букв. Вид матрицы на рисунке 5.7.

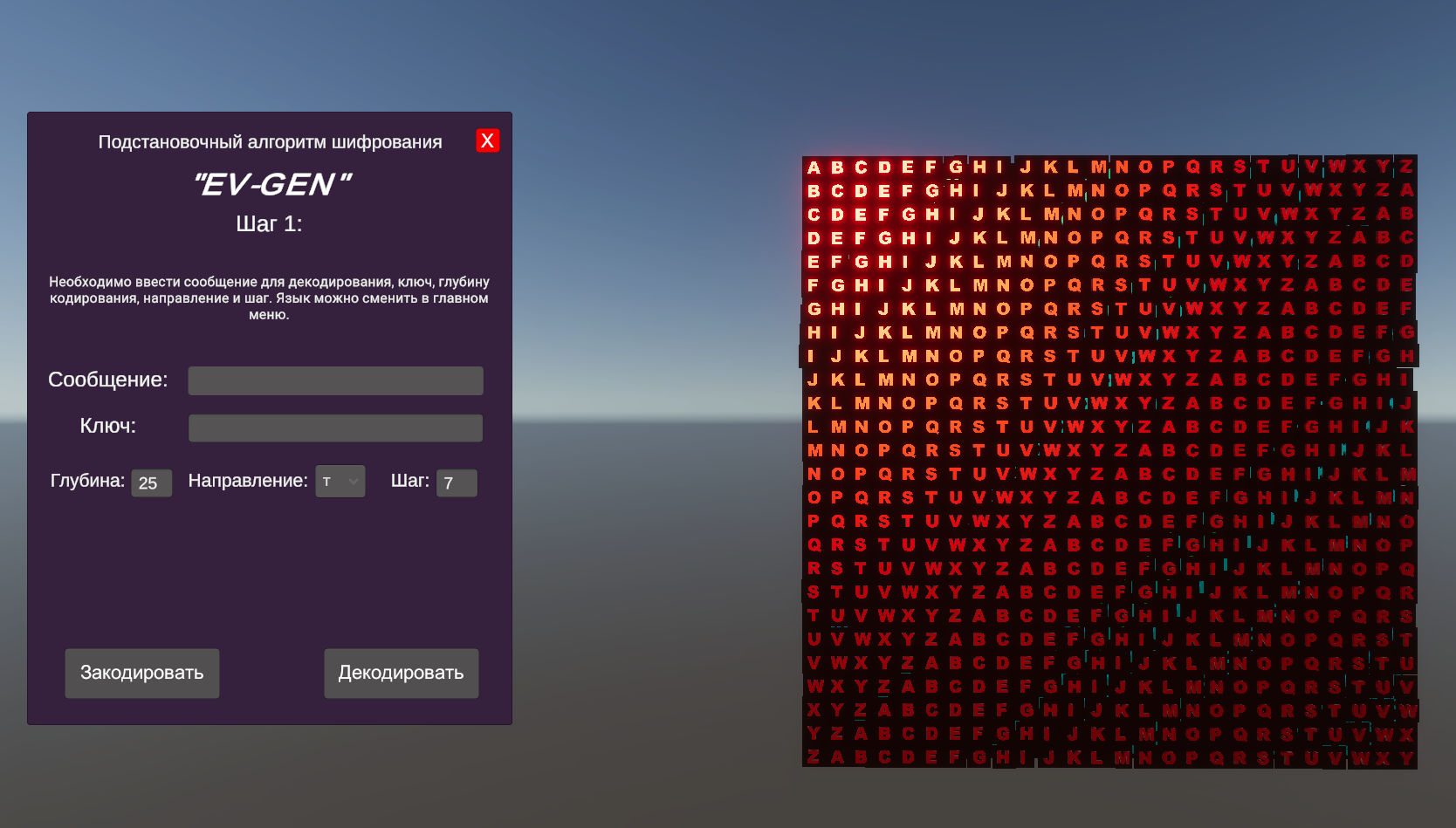


Рисунок 5.7 – Вид матрицы при выборе английского алфавита

## Вывод по разделу

Таким образом, результатом данного раздела стало создание руководства пользователя, в котором описано, как пользоваться приложением. Было показано как пользоваться приложением в рабочем режиме и в режиме демонстрации.

Заключение

В ходе выполнения курсового проекта была достигнута поставленная цель по модификации классического шифра Виженера для обеспечения дополнительной стойкости и безопасности передаваемых данных. Были созданы необходимые классы, функции и пользовательский интерфейс.

При разработке проекта выполнены следующие задачи:

* разработать криптографический метод шифрования/расшифрования сообщения, основываясь на алгоритме таблицы Виженера;
* разработать программное средство реализации данного метода или алгоритма;
* реализовать в программе 3D-демонстрацию работы алгоритма;
* составить руководство пользователя.

В процессе выполнения проекта были рассмотрены различные подстановочные шифры. Был проведен их анализ и сравнение.

На следующем этапе было спроектировано приложение, а также выбраны технологии для его реализации. Этими технологиями стали движок Unity для визуализации, IDE Visual Studio для разработки функций приложения, C# как язык программирования, Blender для моделирования букв алфавитов и система Git для контроля версий.

Результатом третьего этапа стало приложение, рассчитанное для использования в ОС Windows. Приложение предоставляет все регламентированные функции, а также графический интерфейс.

По результатам тестирования на следующих этапах были устранены и исправлены всевозможные ошибки, а также проверена работоспособность, и отказоустойчивость программного средства.

Было составлено руководство пользователя.

По результатам выполнения программы можно сделать вывод о том, что разработанная программа работает должным образом и требования технического задания полностью выполнены.

Список используемых источников

1. Простейшие методы шифрования текста [Электронный ресурс]. Режим доступа: https://infojournal.ru/wp-content/uploads/2020/04/mir\_info-4-2020.pdf. Дата доступа: 09.03.2023

2. Душкин, Р. В. Математика и криптография. Тайны шифров и логическое мышление. / Р. В. Душкин – Москва: АСТ, 2017. (http://maxima-library.org/knigi/genre/b/423167?format=read)

3. Урбанович, П. П. Лабораторный практикум по дисциплинам «Защита информации и надежность информационных систем» и «Криптографические методы защиты информации». В 2 ч. Ч. 2. Криптографические и стеганографические методы защиты информации / П. П. Урбанович, Н. П. Шутько. – Минск: БГТУ, 2020. – 40 с. (https://elib.belstu.by/handle/123456789/37338)

4. Библиотеки класса .NET [Электронный ресурс]. Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/standard/class-libraries. Дата доступа: 28.03.2023.

ПРИЛОЖЕНИЕ А

Файл Algorithm.cs

|  |
| --- |
| using System;  using System.Collections.Generic;  using TMPro;  using static ENUMS;  public class Algorithm  {  public static string Encode(CipherVector cipherText, TMP\_Text elapsed = null)  {  var startTime = DateTime.Now;  var (message, key, depth, direction, step, alphabetType) = UnwrapCipherVector(cipherText);  var (charToIndexDict, alphabet, alphabetLen) = GetAlphabetParams(alphabetType);  string encodedMessage = "";  for (int i = 0; i < message.Length; i++)  {  var (messageIndex, keyIndex, P) = (  charToIndexDict[$"{message[i]}"],  charToIndexDict[$"{key[i % key.Length]}"],  (direction == DIRECTIONS.RIGHT || direction == DIRECTIONS.BOTTOM) ? step : -step);  var indexOfEncodedLetter = (messageIndex + keyIndex + depth + P) % alphabetLen;  if (indexOfEncodedLetter < 0)  indexOfEncodedLetter = (indexOfEncodedLetter + alphabetLen) % alphabetLen;  encodedMessage += alphabet[indexOfEncodedLetter];  }  if (elapsed != null)  elapsed.text = (DateTime.Now - startTime).ToString();  return encodedMessage;  }  public static string Decode(CipherVector cipherText, TMP\_Text elapsed = null)  {  var startTime = DateTime.Now;  var (message, key, depth, direction, step, alphabetType) = UnwrapCipherVector(cipherText);  var (charToIndexDict, alphabet, alphabetLen) = GetAlphabetParams(alphabetType);  string decodedMessage = "";  for (int i = 0; i < message.Length; i++)  {  var (messageIndex, keyIndex, P) = (  charToIndexDict[$"{message[i]}"],  charToIndexDict[$"{key[i % key.Length]}"],  (direction == DIRECTIONS.RIGHT || direction == DIRECTIONS.BOTTOM) ? -step : step);  var indexOfDecodedLetter = (messageIndex + P - keyIndex - depth) % alphabetLen;  if (indexOfDecodedLetter < 0)  indexOfDecodedLetter = (indexOfDecodedLetter + alphabetLen) % alphabetLen;  decodedMessage += alphabet[indexOfDecodedLetter];  }  if (elapsed != null)  elapsed.text = (DateTime.Now - startTime).ToString();  return decodedMessage;  }  private static (string message, string key, int depth, DIRECTIONS direction, int step, ALPHABETS alphabetType) UnwrapCipherVector(CipherVector cipherText)  => (cipherText.Message, cipherText.Key, cipherText.Depth, cipherText.Direction, cipherText.Step, cipherText.AlphabetType);  private static (Dictionary<string, int> charToIndexDict, string[] alphabet, int alphabetLen) GetAlphabetParams(ALPHABETS alphabetType)  {  return alphabetType switch  {  ALPHABETS.LATIN => (Alphabets.LatinDictionary, Alphabets.LatinAlphabet, Alphabets.LatinAlphabet.Length),  \_ => (Alphabets.CyrillicDictionary, Alphabets.CyrillicAlphabet, Alphabets.CyrillicAlphabet.Length),  };  }  } |

ПРИЛОЖЕНИЕ Б

Файл Alphabets.cs

|  |
| --- |
| using System.Collections.Generic;  public static class Alphabets  {  public static string[] CyrillicAlphabet { get; } = new string[]  {  "а", "б", "в", "г", "д", "е", "ё", "ж", "з", "и",  "й", "к", "л", "м", "н", "о", "п", "р", "с", "т",  "у", "ф", "х", "ц", "ч", "ш", "щ", "ъ", "ы", "ь",  "э", "ю", "я"  };  public static Dictionary<string, int> CyrillicDictionary { get; } = new()  {  {"а", 0}, {"б", 1}, {"в", 2}, {"г", 3}, {"д", 4},  {"е", 5}, {"ё", 6}, {"ж", 7}, {"з", 8}, {"и", 9},  {"й", 10}, {"к", 11}, {"л", 12}, {"м", 13}, {"н", 14},  {"о", 15}, {"п", 16}, {"р", 17}, {"с", 18}, {"т", 19},  {"у", 20}, {"ф", 21}, {"х", 22}, {"ц", 23}, {"ч", 24},  {"ш", 25}, {"щ", 26}, {"ъ", 27}, {"ы", 28}, {"ь", 29},  {"э", 30}, {"ю", 31}, {"я", 32}  };  public static Dictionary<string, int> LatinDictionary { get; } = new()  {  {"a", 0}, {"b", 1}, {"c", 2}, {"d", 3}, {"e", 4},  {"f", 5}, {"g", 6}, {"h", 7}, {"i", 8}, {"j", 9},  {"k", 10}, {"l", 11}, {"m", 12}, {"n", 13}, {"o", 14},  {"p", 15}, {"q", 16}, {"r", 17}, {"s", 18}, {"t", 19},  {"u", 20}, {"v", 21}, {"w", 22}, {"x", 23}, {"y", 24},  {"z", 25}  };  public static string[] LatinAlphabet { get; } = new string[]  {  "a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m",  "n","o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"  };  } |

ПРИЛОЖЕНИЕ В

Файл CipherVector.cs

|  |
| --- |
| using static ENUMS;  public class CipherVector  {  public string Message { get; set; }  public string Key { get; set; }  public int Depth { get; set; }  public DIRECTIONS Direction { get; set; }  public int Step { get; set; }  public ALPHABETS AlphabetType { get; set; }  public CipherVector() { }  public CipherVector(string message, string key, int depth, DIRECTIONS direction, int step, ALPHABETS alphabetType)  {  Message = message;  Key = key;  Depth = depth;  Direction = direction;  Step = step;  AlphabetType = alphabetType;  }  public override string ToString()  {  return Message + " " + Key + " " + Depth + " " + Direction + Step + " " + AlphabetType;  }  } |

ПРИЛОЖЕНИЕ Г

Файл InputRegex.cs

|  |
| --- |
| using System.Text.RegularExpressions;  internal static class InputRegex  {  internal static readonly Regex latMessage = new(@"^[az]\*$");  internal static readonly Regex cyrMessage = new(@"^([а-я]|ё)\*$");  internal static readonly Regex latKey = new(@"^[a-z]\*$");  internal static readonly Regex cyrKey = new(@"^([а-я]|ё)\*$");  internal static readonly Regex Depth = new(@"^[0-9]\*$");  internal static readonly Regex Step = new(@"^[0-9]\*$");  internal static bool IsMatch(string text, Regex regex) => regex.IsMatch(text);  } |