Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Операционные системы и системное программирование (ОСиСП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

Программное средство игра “Змейка”

БГУИР КП 1-40 01 01  05  ПЗ

Студент: гр. 051003 Гаркушенко С.С.

Руководитель: Комар Д.Л.

Минск 2022

Учреждение образования

«Белорусский государственный университет информатики и

радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Лапицкая Н.В. 2022г.

ЗАДАНИЕ

по курсовому проектированию

Студенту *Гаркушенко Станиславу Сергеевичу.*

1. Тема работы *Программное средство игра “Змейка”*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Срок сдачи законченной работы *17.12.2022г.*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Исходные данные к работе *­­­­­­­­­­­­­­­­­­ Язык программирования C++ с использованием стандартных функций интерфейсов Windows API. Реализация перемещения змейки по экрану, размещение еды для змейки, употребление еды змейкой. Ведение счёта, вывод результатов игры.*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

*Введение*

*1 Анализ литературных источников*

*2 Постановка задачи*

*3 Моделирование предметной области*

*4 Проектирование предметной области*

*5 Разработка программного средства*

*6 Тестирование и проверка работоспособности программного средства*

*7 Руководство по использованию программного средства*

*Заключение*

*Список использованных источников*

*Приложение А (обязательное). Текст исходного кода программы*

*Приложение Б (обязательное). Текст программного модуля обновления*

*Приложение В (обязательное). Текст программного модуля инициализации*

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*Схема алгоритма оконной процедуры, А1, чертёж.*

6. Консультант по курсовой работе *Комар Д.Л.*

7.Дата выдачи задания *15.09.2022г.*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего обьема работы):

*Раздел 1. Введение к 25.09.2022г. – 10 % готовности работы;*

*Раздел 2 к 10.10.2022г. – 20% готовности работы;*

*Раздел 3 к 25.10.2022г. – 35% готовности работы;*

*Раздел 4 к 10.11.2022г. – 50% готовности работы;*

*Раздел 5 к 25.11.2022г. – 65% готовности работы;*

*Раздел 6 к 1.12.2022г. – 80% готовности работы;*

*Раздел 7. Заключение. Приложения к 10.12.2022г. – 90% готовности работы;*

*оформление пояснительной записки и графического материала к 14.12.2022г. – 100% готовности работы.*

*Защита курсового проекта с 14.12.2022г. по 21.12.2022г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

РУКОВОДИТЕЛЬ *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Комар Д.Л.*

*(подпись)*

Задание принял к исполнению *\_\_\_\_\_\_\_\_\_\_\_\_\_Гаркушенко С.С. 15.09.2022г.*

*(дата и подпись студента)*

**СОДЕРЖАНИЕ**

[Введение 5](#_Toc122173523)

[1 Анализ существующих аналогов 6](#_Toc122173524)

[1.1 Snake Classics 6](#_Toc122173525)

[1.2 Snake 7](#_Toc122173526)

[1.3 Snake, Snake, Snake 8](#_Toc122173527)

[2 Постановка задачи 9](#_Toc122173528)

[3 Моделирование предметной области 10](#_Toc122173529)

[3.1 Windows API 10](#_Toc122173530)

[3.2 Основные сведения о жизненном цикле приложения 12](#_Toc122173531)

[3.3 Преимущества разработки игр на WinAPI 13](#_Toc122173532)

[3.4 Тестирование производительности кода 14](#_Toc122173533)

[3.5 Управление жизненным циклом игры 14](#_Toc122173534)

[4 Проектирование программного средства 16](#_Toc122173535)

[4.1 Разработка внутренних структур данных и определение пользовательского интерфейса программного приложения 16](#_Toc122173536)

[4.2 Выбор технологии, языка и среды программирования 17](#_Toc122173537)

[4.3 Проектирование структуры программы 18](#_Toc122173538)

[5 Разработка программного средства 19](#_Toc122173539)

[5.1 Генерация еды для змейки 19](#_Toc122173540)

[5.2 Рисование змейки 20](#_Toc122173541)

[5.3 Рисование сетки 21](#_Toc122173542)

[6 Тестирование и проверка работоспособности программного средства 22](#_Toc122173543)

[7 Руководство по использованию программного средства 23](#_Toc122173544)

[Заключение 24](#_Toc122173545)

[Приложение А 26](#_Toc122173546)

[Приложение Б 27](#_Toc122173547)

[Приложение В 29](#_Toc122173548)

Введение

«Змейка» считается одним из популярных игровых приложений, которые идут в стандартном комплекте для телефонов Nokia. Впервые была интегрирована в кнопочный телефон Nokia 6110 в 1997 году.

«Змейка» cтала популярной и любимой пользователями по всему миру благодаря своей относительной простоте, а также очень высокому проценту сходимости: в новых версиях мобильных телефонов она также была сохранена в коллекции игр.

В 2008 году вышла последняя версия «Змейки» – "Snakes Subsonic". После выхода «Snakes Subsonic» легендарная «Змейка» уже не развивалась – эра кнопочных телефонов прошла, а для сенсорных мобильных устройств игра не вполне подходила. Тем не менее в «Змейку» продолжают играть и сейчас.

Первое упоминание о «Змейке» датируется 1977 годом. Тогда эта игра эта игра была доступна на игровом автомате. Позже игра получила распространение и стала появляться в большинстве мобильных телефонов.

В разные периоды «Змейка» имела вариации и изменения, в зависимости от платформы, на которой она была отображена. Например, в последних версиях игры были добавлены музыкальные фрагменты и режимы мультиплеера. В некоторых версиях игры присутствовала возможность изменения скорости игры и другие интересные особенности.

С развитием мобильных телефонов и интернета возникли версии «Змейки», которые позволяют играть в эту игру онлайн. В этих версиях игроки могут проверять свои результаты и присоединяться к лидерам и другим игрокам по всему миру. В таких версиях игры появились возможности игры на аккаунты, наборы достижений, социальная интеграция и другие интересные приемы.

# Анализ существующих аналогов

На сегодняшний день существует многочисленное количество вариаций игры “Змейка”. При поиске и изучении аналогов было замечено, что пользователи часто пытаются найти программное средство, которое не только отвечает требованиям пользователя, но и имеет весьма приятный интерфейс.

## Snake Classics

Snake Classics – наиболее распространенная версия игры “Змейка” для ОС Windows.

Интерфейс программы представлен на рисунке 1.1.

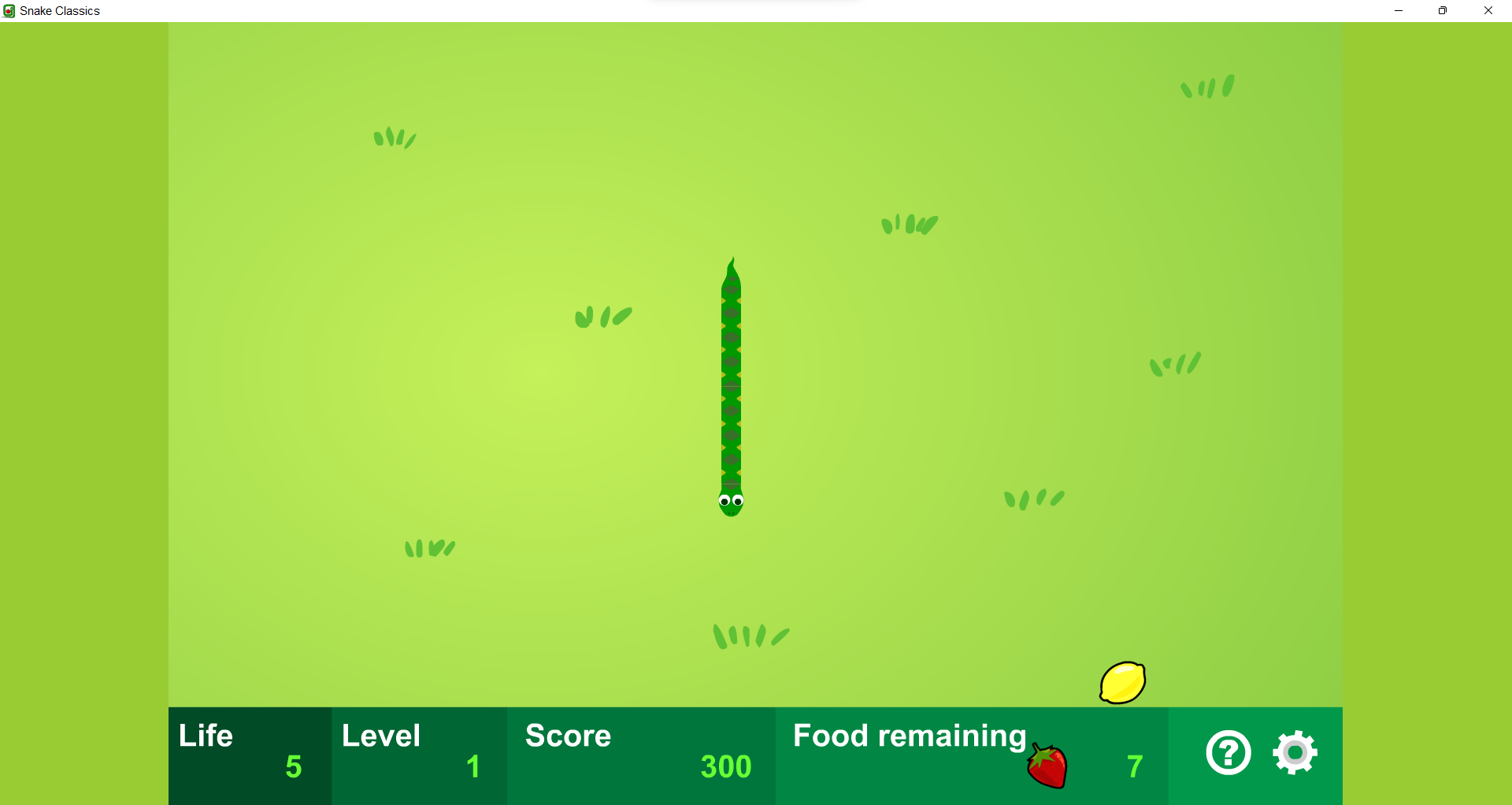


Рисунок 1.1 – Интерфейс программы Snake Classics

Плюсы:

* возникновение еды различных видов;
* наличие уровней;

Минусы:

* отсутствие плавной анимации.

## Snake

Snake – оконная версия игры “Змейка”.

Интерфейс программы представлен на рисунке 1.2.



Рисунок 1.2 – Интерфейс программы Snake

Плюсы:

* наличие паузы;
* наличие уровней
* наличие таблицы лидеров;

Минусы:

* возможность игры только в оконном режиме;
* устаревший пользовательский интерфейс.

## Snake, Snake, Snake

Snake, Snake, Snake – онлайн версия игры “Змейка”.

Интерфейс программы представлен на рисунке 1.3.

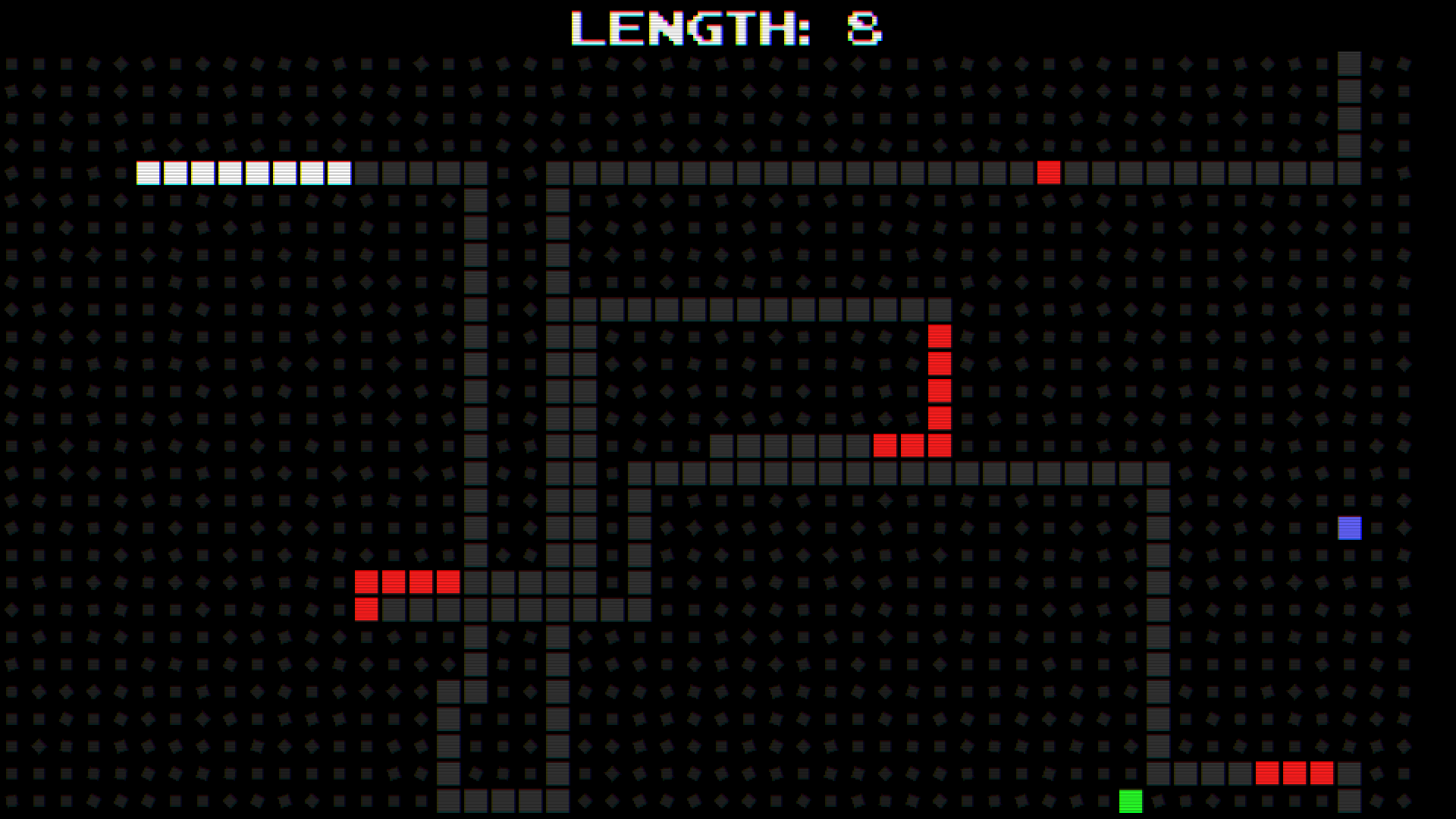


Рисунок 1.3 – Интерфейс программы Snake, Snake, Snake

Плюсы:

* наличие соперников;
* современный пользовательский интерфейс;

Минусы:

* неудобный для чтения счёт.

Таким образом, на сегодняшний день существует достаточно большое количество аналогов файловых менеджеров, которые отличаются как по набору функций, так и по внешнему виду. Но самым большим недостатком является то, что неопытному пользователю достаточно сложно ориентироваться в таком большом многообразии функций и возможностей работы с файловой системой.

# Постановка задачи

В задачу курсового проекта входит создание оконного приложения игра «Змейка» на языке C++ с использованием WinAPI в интегрированной среде разработки Microsoft Visual Studio 2022.

Для того, чтобы программное средство можно было считать игрой “Змейка”, на основе анализа популярных клиентов, в конечном приложении необходимо наличие следующих возможностей:

* движение змейки;
* функция генерации новой еды для змейки после потребления

предыдущей единицы;

* увеличение размера змейки при поедании;
* регистрация столкновения змейки со своим туловищем или хвостом с

последующим окончанием игры;

* возможность начать новую игру;
* определите количество очков, полученных игроком и отображение их

на экране.

Полученное программное средство должно быть портативным, иметь простой, интуитивно-понятный интерфейс, позволяющий неопытному пользователю легко взаимодействовать с игрой.

# Моделирование предметной области

## Windows API

Программное средство должно быть реализовано с использованием WinAPI. Далее будут приведены основные понятия.

Windows API – общее наименование набора базовых функций интерфейсов программирования приложений операционных систем семейств Microsoft Windows корпорации Microsoft.

Windows API спроектирован для использования в языке Си для написания прикладных программ, предназначенных для работы под управлением операционной системой Windows. Работа через WinAPI – это наиболее близкий к операционной системе способ взаимодействия с ней из прикладных программ [5].

У каждого окна, создаваемого программой, имеется соответствующая оконная процедура. Эта процедура является функцией, которая может находиться либо в самой программе, либо в динамически подключаемой библиотеке. Windows посылает сообщение окну путем вызова оконной процедуры, на основе этого сообщения окно совершает какие-то действия и затем возвращает управление Windows. Более точно, окно всегда создается на основе "класса окна". Класс окна определяет оконную процедуру, обрабатывающую поступающие окну сообщения. Использование класса окна позволяет создавать множество окон на основе одного и того же класса окна и, следовательно, использовать одну и ту же оконную процедуру. Например, все кнопки во всех программах для Windows созданы на основе одного и того же класса окна. Этот класс связан с оконной процедурой (расположенной в динамически подключаемой библиотеке Windows), которая управляет процессом передачи сообщений всем кнопкам всех окон.

Алгоритм работы простейшей программы с событийным управлением представлен на рисунке 3.1.

Оконная процедура обрабатывает сообщения, поступающие окну. Очень часто эти сообщения передают окну информацию о том, что пользователь осуществил ввод с помощью клавиатуры или мыши. Таким образом, например, кнопки "узнают" о том, что они нажаты. Другие сообщения говорят окну о том, что необходимо изменить размер окна или о том, что поверхность окна необходимо перерисовать. Когда программа для Windows начинает выполняться, Windows строит для программы очередь сообщений (message queue). В этой очереди хранятся сообщения для любых типов окон, которые могли бы быть созданы программой. Небольшая часть программы, которая называется циклом обработки сообщений (message loop), выбирает эти сообщения из очереди и переправляет их соответствующей оконной процедуре. Другие сообщения отправляются непосредственно оконной процедуре, минуя очередь сообщений [6].



Рисунок 3.1 – Программа с событийным управлением

Приложение должно быстро реагировать (перерисовывать) игровые кадры. Отрисовка в рабочей области окна должно происходить при обработке сообщения WM\_PAINT. Это сообщение крайне важно для программирования под Windows. Оно сообщает программе, что часть или вся рабочая область окна недействительна (invalid), и ее следует перерисовать. Но если реализовать отрисовку сразу на экран, то есть вероятность возникновения «мерцания». Картинка будет обновляться быстрее, чем происходит вывод изображения на экран. Чтобы избежать данной проблемы, обычно используют двойную буферизацию. Для этого сначала выделяют память нужного размера под изображение. Затем каждый раз происходит изменяется именно изображения в памяти, а при срабатывании сообщения WM\_PAINT картинка из памяти выводится на экран. Данный способ позволяет избавиться от мерцания в приложении при быстром изменении данных.

Как было описано ранее, приложение производит все обновления во время отсутствия сообщения в очереди. Таким образом, когда происходит выход из цикла, сначала проверяется, в каком состоянии находится игра, затем в зависимости от этого вызываются функция обновления для нужного состояния.

Организация таймеров в приложении происходит следующим образом: в начале игры происходит инициализация данных таймеров (текущего времени), затем при каждом вызове процедуры обновления производится проверка интервала прошедшего времени, если время истекло, выполняется обновление и сохранение нового времени, если нет – выход из процедуры. Функция GetTickCount возвращает количество миллисекунд, прошедших с момента запуска системы. Ее использование позволяет определять интервалы между обновлениями.

## Основные сведения о жизненном цикле приложения

Жизненный цикл приложения включает в себя шесть основных этапов: планирование, анализ, проектирование, реализация, тестирование и поддержка.

### Планирование

На этом этапе составляют план разработки приложения, определяют время и ресурсы, которые нужно вложить в проект. Также определяют цели и задачи приложения, определяют требования к функциональности, узнают потребности пользователя и принимают решения о бюджете проекта.

### Анализ

На этом этапе анализируют требования к приложению и подбирают соответствующие технологические решения. Исследуют потребности и предпочтения пользователей и определяют их профили. Также должны изучить конкурентные продукты и анализировать имеющиеся решения для понимания их преимуществ и недостатков.

### Проектирование

На этом этапе разрабатывают архитектуру приложения и проектируют интерфейс пользователя. Определяют используемые технологии, а также проектируют базу данных, интерфейсы и алгоритмы.

### Реализация

На этом этапе реализуют разработанные архитектуры, алгоритмы и интерфейсы. Также используют программное обеспечение для разработки приложения и добавляют функциональность.

### Тестирование

На этом этапе проводят тестирование приложения на соответствие требованиям, определяют ошибки и добавляют дополнительную функциональность. Также проводят тестирование пользовательского интерфейса, а также тестируют приложение на соответствие критериям безопасности и производительности.

### Поддержка

На этом этапе проводят постоянную поддержку приложения, обновляют и модифицируют код, а также проверяют и исправляют ошибки. Также добавляют новые функции и анализируют использование приложения для извлечения полезной информации.

## Преимущества разработки игр на WinAPI

Игровое программирование на WinAPI является ключевым инструментом для разработки игр под Windows. WinAPI предоставляет программистам мощные инструменты для создания графических интерфейсов, ввода устройств и сложных игровых механик. Основными аспектами WinAPI являются графические приложения, ввод устройств и игровая механика. Графика приложений позволяет разработчикам создавать реалистичные и красивые картинки для игр. Ввод устройств позволяет разработчикам использовать мышь, клавиатуру и даже геймпады для управления игрой. Игровая механика состоит из сложных правил, которые определяют поведение игрока и игровых объектов.

При разработке игр на WinAPI разработчики должны учитывать многие различные аспекты, такие как производительность, графика, аудио и игровая механика. Для успешной разработки игр на WinAPI разработчики должны иметь глубокое понимание API и быть в состоянии корректно его использовать. Использование WinAPI для разработки игр дает разработчикам большую гибкость и многофункциональность. Они могут использовать предоставленные API для настройки игры, добавления новых компонентов и даже изменения существующих функций и правил.

WinAPI предоставляет разработчикам мощные инструменты для создания увлекательных и интересных игр, имеющих многие различные возможности. WinAPI предоставляет разработчикам инструменты для создания мощных и гибких игровых движков. Это позволяет разработчикам создавать сложные игровые механики и динамические игровые среды. Разработчики игр могут использовать WinAPI для создания функциональных интерфейсов пользователя. Это позволяет игрокам иметь больше влияния на процесс создания игры и иметь более приятную игровую среду. WinAPI позволяет разработчикам реализовывать более плавную и эффективную отрисовку изображения. Это позволяет создавать более реалистичные и привлекательные игры.

## Тестирование производительности кода

Тестирование производительности на WinAPI является важным элементом процесса разработки приложения и представляет собой метод оценки производительности за счет использования интерфейсов WinAPI. Эти интерфейсы предоставляют информацию о времени исполнения приложения и других параметрах. Тестирование производительности на WinAPI обеспечивает прозрачность и позволяет вам оценить, насколько хорошо ваше приложение работает и какие меры можно предпринять, чтобы улучшить его производительность. Это может быть особенно полезно для администраторов систем, которые пытаются оптимизировать скорость работы их системы.

Тестирование производительности на WinAPI может быть проведено на нескольких уровнях. На низком уровне он может быть использован для измерения производительности отдельных процессов или приложений. На более высоком уровне он может использоваться для анализа производительности всей системы. Для проведения тестирования производительности на WinAPI необходимо использовать специальные инструменты, которые могут помочь вам анализировать приложение и просматривать информацию о производительности. Эти инструменты могут быть использованы для сравнения производительности нескольких приложений и для анализа производительности системы в целом.

Тестирование производительности на WinAPI также может включать анализ использования памяти и анализ утечек памяти. Это позволяет определить, насколько хорошо приложение управляет памятью и какие процессы или другие приложения могут потреблять больше ресурсов, чем необходимо. Тестирование производительности на WinAPI также может помочь идентифицировать проблемы с производительностью и предлагать решения для их решения. Это может быть особенно полезно для администраторов систем, которые пытаются оптимизировать производительность своей системы. Тестирование производительности на WinAPI может быть полезным для отладки приложений и устранения неполадок.

## Управление жизненным циклом игры

Управление жизненным циклом игры – это процесс, при котором разработчики игры планируют и реализуют релиз игры на рынок. Этот процесс состоит из шести этапов: планирование, разработка, продвижение, продажи, поддержка и подход к управлению.

Первым шагом в управлении жизненным циклом игры является планирование. На этом этапе необходимо определить цели и сроки запуска игры, целевую аудиторию, ресурсы, необходимые для создания игры, а также маркетинговые стратегии.

Далее начинается процесс разработки. Разработчики создают игровой движок, визуальное и аудио оформление, сценарий, геймплей и игровые механики, чтобы сделать игру привлекательной для игроков.

Затем идет процесс продвижения. В этом процессе разработчики используют различные маркетинговые инструменты для продвижения игры и привлечения к ней большего количества игроков.

После продвижения наступает процесс продаж. Этот этап включает в себя разработку продающей стратегии и распространение игры на рынке. Далее идет процесс поддержки. Разработчики предоставляют поддержку игрокам, отвечают на их вопросы и предоставляют дополнительные обновления и исправления ошибок.

После этого наступает процесс управления. На этом этапе разработчики оценивают результаты продаж и продвижения, анализируют данные и разрабатывают планы для улучшения игры.

Наконец, разработчики должны принять решение о продолжении жизненного цикла игры. На этом этапе они могут выбрать один из трех путей: добавление новых обновлений, удаление игры с рынка или продление жизненного цикла игры.

# Проектирование программного средства

## Разработка внутренних структур данных и определение пользовательского интерфейса программного приложения

Большинство алгоритмов зависит от того, каким образом организованы данные поэтому начинать проектирование программы следует не с алгоритмов, а с разработки структуры, необходимых для представления входных, выходных и промежуточных данных.

Структура данных – программная единица, позволяющая хранить и просматривать множество однотипных или логически связанных данных в вычислительной технике. Структура данных часто является реализацией, какого либо, абстрактного типа данных.

Какой смысл заключен в данных, какими символами они выражены – буквенными или цифровыми, что означает то или иное число – все это определяется программой обработки. Все данные необходимые для решения практических задач подразделяются на несколько типов, причем понятие тип связывается не только с представлением данных в адресном пространстве, но и со способом их обработки.

Любые данные могут быть отнесены к одному из двух типов: основному (простому), форма представления которого определяется архитектурой компьютера, или сложному, конструируемому пользователем для решения конкретных задач.

Данные простого типа это – символы, числа и т.п. элементы, дальнейшее дробление которых не имеет смысла. Из элементарных данных формируются структуры (сложные типы) данных.

В данном программном средстве будут использоваться структуры для создания змейки, еды, создания состояний игры. Из этого следует, что программа будет работать c сложными данными. Также будут использоваться данные простых типов символы, числа для промежуточных вычислений.

Для разработки данного программного продукта, необходимо спроектировать и разработать окно программы. Которое должно обеспечить удобное визуальное отображение игрового поля, змейки и еды, которая может появляться в любых точках игрового поля.

Определение пользовательского интерфейса программного приложения является неотъемлемой частью проектирования игры «Змейка».

В понятие пользовательского интерфейса входит не только картинка на экране, но и способы взаимодействия пользователя с системой.

Программный интерфейс – система унифицированных связей, предназначенных для обмена информацией между компонентами вычислительной системы. Программный интерфейс задает набор необходимых процедур, их параметров и способов обращения.

Интерфейс пользователя – это элементы и компоненты программы, которые способны оказывать влияние на взаимодействие пользователя с программным обеспечением.

В разрабатываемом программном средстве будут использоваться следующие сложные типы данных:

* структура, в которой будет хранится информация о текущих координатах;
* перечисление, хранящее все возможные состояния игры;
* перечисление, хранящее направления движения;
* структура, в которой будет хранится информация о змейке.

## Выбор технологии, языка и среды программирования

Технологией программирования называют совокупность методов и средств, используемых в процессе разработки программного обеспечения. Как любая другая технология, технология программирования представляет собой набор технологических инструкций, включающих:

* указание последовательности выполнения технологических операций;
* перечисление условий, при которых выполняется та или иная операция;
* описания самих операций, где для каждой операции определены исходные данные, результаты, а также инструкции, нормативы, стандарты, критерии и методы оценки и т. п.

Кроме набора операций и их последовательности, технология также определяет способ описания проектируемой системы, точнее модели, используемой на конкретном этапе разработки.

Различают технологии, используемые на конкретных этапах разработки или для решения отдельных задач этих этапов, и технологии, охватывающие несколько этапов или весь процесс разработки. В основе первых, как правило, лежит ограниченно применимый метод, позволяющий решить конкретную задачу. В основе вторых обычно лежит базовый метод или подход (парадигма), определяющий совокупность методов, используемых на разных этапах разработки, или методологию. Теперь рассмотрим основные технологии программирования.

Модульное программирование предполагает выделение групп подпрограмм, использующих одни и те же глобальные данные, в отдельно компилируемые модули (библиотеки подпрограмм), например, модуль графических ресурсов. Связи между модулями при использовании данной технологии осуществляются через специальный интерфейс, в то время как доступ к реализации модуля (телам подпрограмм и некоторым "внутренним" переменным) запрещен.

Объектно-ориентированное программирование (ООП) определяется как технология создания сложного программного обеспечения, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного типа (класса), а классы образуют иерархию с наследованием свойств. Взаимодействие программных объектов в такой системе осуществляется путем передачи сообщений.

При разработке данного программного продукта будем придерживаться технологии модульного программирования. Для написания будем использовать язык программирования C++, поскольку он позволяет напрямую взаимодействовать с функциями WinAPI. Для разработки будет использоваться среда программирования Visual Studio 2022, которая довольно сильно облегчает разработку программного средства за счет обширного количества возможностей, встроенных библиотек и модулей.

## Проектирование структуры программы

Так как данное программное средство должно быть довольно простым и удобным в использовании, то был сделан выбор в пользу того, чтобы разместить все визуальные компоненты в одном главном окне, в соответствии с концепцией разработки приложений на WinAPI, когда все прочие пользовательские элементы располагаются в дочерних окнах главного родительского окна.

Главное окно разбивается на 2 области:

* В левом верхнем углу располагается текущий счёт;
* В центральной находится игровая зона.

Структура главного окна представлена на рисунке 4.3.



Рисунок 4.3 – Структура главного окна приложения

# Разработка программного средства

## Генерация еды для змейки

Для того, чтобы увеличивать размер змейки, ей необходимо есть, а чтобы пользователю было интересно играть, еда появляется в случайных местах. Алгоритм генерации еды приведен на рисунке 4.1.



Рисунок 5.1 – Схема алгоритма generateFood

## Рисование змейки

Для начала и продолжения игры необходимо рисовать змейку. Данный алгоритм представлен на рисунке 4.2.



Рисунок 5.2 – Схема алгоритма Draw

## Рисование сетки

Для удобной игры необходимо отобразить сетку, по линиям которой будет двигаться змейка. Данный алгоритм представлен на рисунке 4.3.



Рисунок 5.3 – Схема алгоритма draw

# Тестирование и проверка работоспособности программного средства

Таблица 6.1 – Тесты

|  |  |  |
| --- | --- | --- |
| Идентификатор теста | Шаги для воспроизведения | Ожидаемый результат |
| Запуск программы | 1. запустить программу. | Начало игры |
| Управление змейкой | 1. запустить программу; 2. нажать любую из стрелок   управления. | Изменение направления движения змейки |
| Увеличение счёта | 1. запустить программу; 2. направить змейку в   направлении еды. | Увеличение счёта на 1 с каждой съеденной единицей еды |
| Столкновение змейки с своим хвостом | 1. запустить программу; 2. увеличить длину змейки до   5;   1. замкнуть змейку на своё   хвосте. | Вывод результатов и завершение игры |
| Выход змейки за границы экрана | 1. запустить программу; 2. направить змейку в сторону   границы экрана. | Появление змейки из другого конца экрана |

Все выбранные функции выполнили свою работу корректно. Такого набора функций достаточно для полного цикла моделирования неисправностей, их анализа и сохранения для передачи по сети или последующего использования.

# Руководство по использованию программного средства

Программное средство спроектировано таким образом, что процесс его установки и использования не должен вызвать никаких затруднений у пользователя. Установка программного средства не требуется, потому что оно состоит всего из одного исполняемого файла. Достаточно лишь скопировать исполняемый файл в любую удобную для пользователя директорию компьютера.

Для успешного запуска и комфортной работы программного средства необходимо соответствие минимальным системным требованиям:

* процессор: 1 ГГц и выше;
* оперативная память: 128 Мб и выше;
* SSD: 100 Мб;
* операционная система: Windows 8 и выше.

После открытия игры на экране открывается окно, представленное на рисунке 6. Здесь пользователь начинает свою игру.

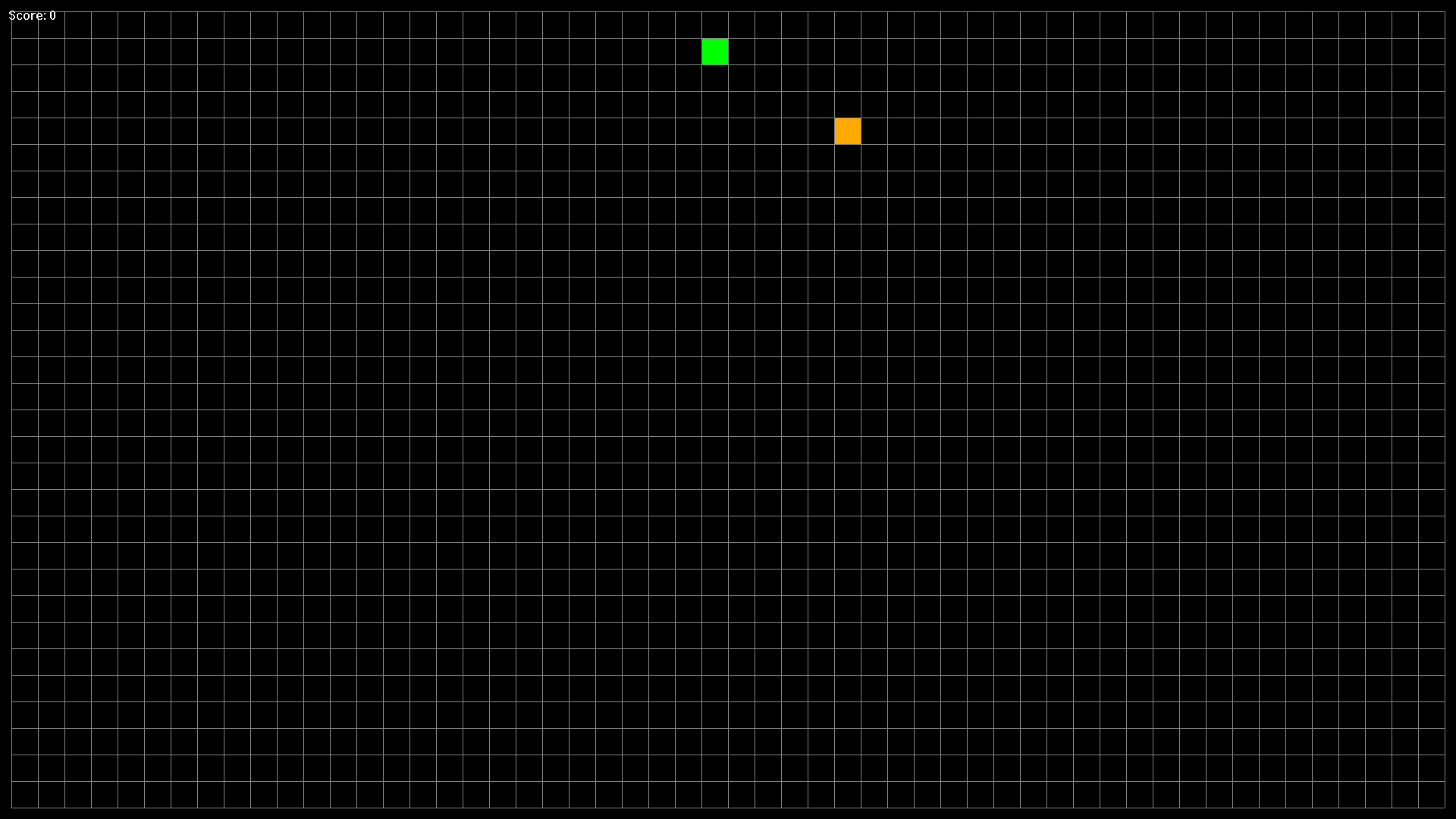


Рисунок 6 – Начальное окно программы

Для управления змейкой можно использовать стрелки на клавиатуре, “W”, “A”., “S”, “D”, клавиши “Num2”, “Num4”, “Num6”, “Num8”.

Заключение

Ни один пользователь не может обойтись без каких-либо игр, ведь они помогает пользователю отдохнуть от повседневных задач. Игры помогают развивать мышление и улучшать навыки анализа. Также игры предоставляют пользователям возможность поиграть с друзьями или даже против других игроков по всему миру.

Современные файловые менеджеры могут напугать неопытного пользователя большим многообразием функций для работы с объектами файловой системы, что в свою очередь ухудшает понимание того, что делает данного программное средство. Поэтому целью данного курсового проекта было создание простого, но в то же время функционального файлового менеджера, который предоставит набор стандартных функций для работы с каталогами и файлами.

В рамках данного курсового проекта был произведен анализ предметной области и реализовано программное обеспечение игра "Змейка". Согласно поставленным задачам, в данном приложении были реализованы такие возможности, как движение змейки, случайное появление еды, возможность ведения счёта.

Для успешного выполнения всех поставленных задач потребовалось изучить основы C++, ознакомиться с возможностями среды Microsoft Visual Studio.

При тестировании и отладке не было выявлено случаев некорректной работы программы, нестабильной работы, появления сторонних ошибок и т.д.

Написанный код легко модифицируется для добавления новых функций. В дальнейшем возможны улучшения и доработки, вносящие новый функционал в данную игру.

Среди возможных расширений данного программного средства существует такой вариант, как добавление уровней игры для того, чтобы заинтересовать пользователя.

Еще одним вариантом возможного расширения будет многопользовательский и онлайн режимы игры.

Спроектированная игра имеет интуитивно понятный интерфейс для пользователя, проста в использовании, не требует серьезных аппаратных затрат.

Список использованной литературы

[1] Windows API – Academic [Электронный ресурс] Режим доступа: [https://dic.academic.ru/dic.nsf/ruwiki/43165 –](https://dic.academic.ru/dic.nsf/ruwiki/43165%20–) Дата доступа: 13.09.2022.

[2] Чтение данных о событиях входного буфера [Электронный ресурс] Режим доступа: https://learn.microsoft.com/ru-ru/windows/console/reading-input-buffer-events – Дата доступа: 15.09.2022.

[3] Генерация псевдослучайных чисел C++ [Электронный ресурс] Режим доступа: https://en.cppreference.com/w/cpp/numeric/random/rand – Дата доступа: 18.09.2022.

[4] Центр разработки для Windows [Электронный ресурс] Режим доступа: https: <https://learn.microsoft.com/ru-ru/windows/uwp/develop/> – Дата доступа: 21.09.2022.

[5] Литвиненко Н. А. Технология программирования на С++. Win32API приложения. – СПб.: БХВ-Петербург,2010. – 288 с.

[6] Щупак Ю. А. Win32 API. Разработка приложений для Windows – СПб.: Питер, 2008. – 592 с.

[7] Безруков В. А. Win32 API. Программирование / учебное пособие – СПб.: СПбГУ ИТМО, 2009. – 90 с.

[8] Руководство по разработке игр для Windows [Электронный ресурс] Режим доступа: https://learn.microsoft.com/ru-ru/windows/uwp/gaming/e2e – Дата доступа: 28.09.2022.

[9] Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. – СПб.: Питер, 2015. – 1120 с.

[10] Рихтер Джеффри. Создание эффективных WIN32-приложений с учетом специфики 64-разрядной версии Windows. 4-е изд. – СПб.: Питер, 2008. – 720 с.

Приложение А

(обязательное)

Исходный код программы

#include "Pch.h"

#include "Application.h"

using namespace CPPSnake;

Int32 WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,

LPSTR lpCmdLine, Int32 nCmdShow)

{

\_app = new Application();

if (!\_app->initialize())

{

MessageBox(NULL, TEXT("Failed to initialize application.

The program will now exit."),

TEXT("Error"), MB\_OK | MB\_ICONERROR);

delete \_app;

return 0;

}

\_app->run();

delete \_app;

return 0;

}

Приложение Б

(обязательное)

Исходный код программы

#include "Pch.h"

#include "ResetSequence.h"

#include "Timer.h"

#include "Grid.h"

#include "ApplicationWindow.h"

namespace CPPSnake

{

Void ResetSequence::start()

{

if (\_isPlaying) return;

\_isPlaying = true;

\_isFadingOut = true;

\_startTime = \_timer->getCurrentTime();

}

Void ResetSequence::draw()

{

if (!\_isPlaying) return;

const Float fadeDuration = 2.0f;

BGRA32 startColor = BGRA32(0, 0, 0, 0);

BGRA32 endColor = BGRA32(0, 0, 0, 255);

UInt32 currentColor;

if (\_isFadingOut)

{

Float fadeTime = \_timer->getCurrentTime() - \_

startTime;

Float t = fadeTime / fadeDuration;

if (t > 1.0f)

{

\_isFadingOut = false;

\_willStartToFadeIn = true;

t = 1.0f;

}

HDC backbufferDC = \_gfxDevice->getBackbufferDC();

std::string info = "Game Over";

SetTextColor(backbufferDC, 0x00FFFFFF);

TextOut(backbufferDC, 900, 550, info.c\_str(),

(UInt32)info.length());

info = "Your Score is: " +

std::to\_string(\_snake->getNumFoodEaten());;

SetTextColor(backbufferDC, 0x00FFFFFF);

TextOut(backbufferDC, 890, 570, info.c\_str(),

(UInt32)info.length());

currentColor = BGRA32::lerp(startColor, endColor, t);

}

else

{

\_willStartToFadeIn = false;

Float fadeTime = \_timer->getCurrentTime() - \_

startTime - fadeDuration;

Float t = fadeTime / fadeDuration;

if (t > 1.0f)

{

\_isPlaying = false;

t = 1.0f;

}

currentColor = BGRA32::lerp(endColor, startColor, t);

}

\_gfxDevice->drawTransparentQuad({0, 0},

\_appWindow->getClientWidth(),

\_appWindow->getClientHeight(), currentColor);

}

}

Приложение В

(обязательное)

Исходный код программы

#include "Pch.h"

#include "Application.h"

namespace CPPSnake

{

Application\* \_app{};

Application::~Application()

{

delete \_snake;

delete \_grid;

delete \_gfxDevice;

delete \_appWindow;

delete \_timer;

}

Bool Application::initialize()

{

srand(timeGetTime());

\_timer = new Timer();

if (!\_timer->initialize()) return false;

\_appWindow = new ApplicationWindow();

if (!\_appWindow->initialize()) return false;

\_gfxDevice = new GfxDevice();

if (!\_gfxDevice->initialize()) return false;

\_grid = new Grid();

if (!\_grid->initialize(\_gridSettings)) return false;

\_snake = new Snake();

if (!\_snake->initialize(\_snakeSettings)) return false;

ShowCursor(FALSE);

return true;

}

Void Application::run()

{

SNK\_ASSERT(\_appWindow);

MSG msg{};

while (true)

{

if (PeekMessage(&msg, NULL, 0, 0, PM\_REMOVE))

{

if (msg.message == WM\_QUIT) break;

TranslateMessage(&msg);

DispatchMessage(&msg);

}

else processFrame();

}

}

Void Application::quit()

{

PostQuitMessage(0);

}

Void Application::processFrame()

{

\_gfxDevice->clearColor(0x00000000);

if (\_gameState == GameState::Snaking)

{

\_snake->update();

\_snake->draw();

\_grid->draw();

drawScore();

if (\_snake->bitHisOwnTail())

{

\_gameState = GameState::Reset;

\_resetSequence.start();

}

}

else

if (\_gameState == GameState::Reset)

{

\_snake->draw();

\_grid->draw();

drawScore();

\_resetSequence.draw();

if (\_resetSequence.willStartToFadeIn())

{

\_grid->initialize({});

\_snake->initialize({});

}

if (!\_resetSequence.isPlaying())

\_gameState = GameState::Snaking;

}

\_gfxDevice->present();

}

Void Application::drawScore()

{

HDC backbufferDC = \_gfxDevice->getBackbufferDC();

std::string info = "Score: " + std::to\_string

(\_snake->getNumFoodEaten());

SetTextColor(backbufferDC, 0x00FFFFFF);

TextOut(backbufferDC, 10, 10, info.c\_str(),

(UInt32)info.length());

}

Void Application::drawPause()

{

HDC backbufferDC = \_gfxDevice->getBackbufferDC();

std::string info = "Pause. Tap 'Space' to continue";

SetTextColor(backbufferDC, 0x00FFFFFF);

TextOut(backbufferDC, 900, 550, info.c\_str(),

(UInt32)info.length());

}

}