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

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

«Программное средство Тетрис»
БГУИР КР 1-40 01 01 609 ПЗ

Студент: гр. 851006 Забенько Т.И.
Руководитель: Жиденко А.Л.

Минск 2020

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

_____

(подпись)

_____2020 г.

ЗАДАНИЕ

по курсовому проектированию

Студенту Забенько Тамаре Игоревне, 851006

1. Тема работы Программное средство «Тетрис»

2. Срок сдачи студентом законченной работы 02.09.2020 г.

3. Исходные данные к работе Операционная система – Windows 10; среда разработки – Visual Studio 19; язык программирования – Си, интерфейс программирования приложений Windows API.

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

Введение.

1. Теоретические сведения;

2. Определение и анализ требований;

3. Разработка программного средства;

4.Тестирование программного средства;

5.Руководство пользователя.;

Заключение;

Список источников;

Приложение А;

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

1. «Программное средство Тетрис», А1, схема программы

6. Консультант по курсовой работе Жиденко А.Л.

7. Дата выдачи задания 10.09.2020 г.

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

раздел 1 к 28.09.2020 – 15 % готовности работы;

разделы 2 к 21.10.2020 – 30 % готовности работы;

разделы 3 к 18.11.2020 – 60 % готовности работы;

раздел 4, 5 к 01.12.2020 – 90 % готовности работы;

оформление пояснительной записки и графического материала к 10.12.2020 – 100 % готовности работы.

Защита курсовой работы с 01.12.2020 по 21.12.2020 г.

РУКОВОДИТЕЛЬ А.Л. Жиденко

(подпись)

Задание принял к исполнению Т.И. Забенько

(дата и подпись студента)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Технический прогресс в последние несколько десятков лет сильно изменил жизнь обычного человека. Цифровые технологии ускорили развитие различных областей науки, позволили повысить качество жизни. Немалое влияние развитие информационных технологий оказало и на индустрию развлечений: видеоблогинг, стриминг, кино и, конечно, видеоигры.

История видеоигр начинается в 1950-х годах, когда создавались первые простые игры и симуляции. Изобретателем компьютерных игр обычно считают одного из трёх людей: Ральфа Бауэра (инженер, выдвинувший идею интерактивного телевидения в 1951 году), Александра Дугласа (разработчик «OXO» – компьютерной реализации крестиков-ноликов) или Уильяма Хигинботама (создатель игры «Tennis for Two», 1958).

Врачи, психологи и исследователи со всей планеты на протяжении долгих лет докладывают о пользе тетриса. В 2009 году было доказано, что эта игра повышает работоспособность головного мозга. Позднее выяснилось, что с её помощью можно преодолеть посттравматический синдром. А не так давно тетрис назвали отличным средством от чувства беспокойства и состояния тревоги. Причём версия, в которой скорость фигурок увеличивается по мере уничтожения линий, оказалась эффективнее обычной – она как бы безмолвно хвалит играющего, подкидывая ему всё более сложные испытания и позволяя на время отвлечься от проблем реальной жизни.

Первым изобретателем игры «Тетрис» был советский программист Алексей Пажитнов. Игра была выпущена 6 июня 1984 года – в это время Пажитнов работал в Вычислительном Центре Академии наук СССР. «Тетрис» представляет собой головоломку, построенную на использовании геометрических фигур «тетрамино» (разновидности полимино, состоящих из четырёх квадратов). Полимино в том или ином виде использовались в настольных играх и головоломках задолго до создания «Тетриса». Идею Пажитнову подсказала игра в пентамино. Первоначальная версия игры была написана Пажитновым на языке Паскаль для компьютера «Электроника-60». Первая коммерческая версия игры была выпущена американской компанией Spectrum HoloByte в 1987 году. В последующие годы «Тетрис» в разных версиях был перенесён на множество устройств, включая все возможные компьютеры и игровые консоли, а также такие устройства, как графический калькуляторы, мобильные телефоны, медиаплееры, карманные персональные компьютеры. По количеству проданных коммерческих версий «Тетрис» превосходит любую другую компьютерную игру в истории. В 2007 году «Тетрис» вошел в число десяти важнейших компьютерных игр, принятых на сохранение в Библиотеку Конгресса. В 2014 году количество платных загрузок игры для мобильных телефонов превысило 425 миллионов.

Обычно игрок проигрывает из-за того, что не может справиться со слишком быстрым темпом игры, или потому, что данная реализация реагирует на клавиши слишком медленно по сравнению с ускоряющимся темпом падения фигурок, вследствие чего игрок уже не может в принципе приложить достаточное количество сдвигов к фигурке. Была опубликована статья, автор которой доказывает, что даже если бы игрок реагировал мгновенно и всегда принимал правильные решения, то и в этом случае он бы в конечном счёте проиграл. Проблемой являются S- и Z-образные фигурки. Достаточно большое количество S-фигурок заставит игрока оставить дырку в правом нижнем углу. Достаточно большое количество Z-фигурок после этого заставит игрока оставить дырку в левом углу следующего ряда, не заполнив предыдущую дырку. Если после этого опять выпадет достаточно много S-фигурок, достаточно много Z-фигурок, и так много раз, заполнится (с дырками по краям) всё поле, и для следующей фигурки места не останется. Если генератор случайных чисел идеален и выдает дискретное равномерное распределение, любая (в том числе и такая) комбинация рано или поздно выпадет. Однако среднее время, через которое выпадет такая комбинация, огромно и превышает время существования Вселенной. Тем не менее не исключено, что существует какая-то другая, более трудная для доказательства причина, по которой идеальный игрок должен проиграть намного раньше указанной верхней границы. Некоторые задачи, возникающие перед игроком в ходе игры, являются NP-полными.

В настоящее время популярность этой логической игры нисколько не уменьшилась. Даже наоборот – появились в интернете целые сообщества и форумы, посвящённые «Тетрису», периодически проводятся чемпионаты внутри многих стран и чемпионаты мира, а также существует негласное соревнование по времени прохождения, принять участие в котором может абсолютно любой пользователь.

Целью данного проекта является разработка игрового приложения «Тетрис».

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Обзор аналогов

На данный момент существует множество вариаций игры «Тетрис»: с клетками прямоугольной или непрямоугольной формы, в трёхмерном пространстве, многопользовательские варианты.

Одной из самых популярных реализаций игры является «Тетрис 99», выпущенный японской компанией Nintendo. Внешний вид данного приложения представлен на рисунке 1.1.

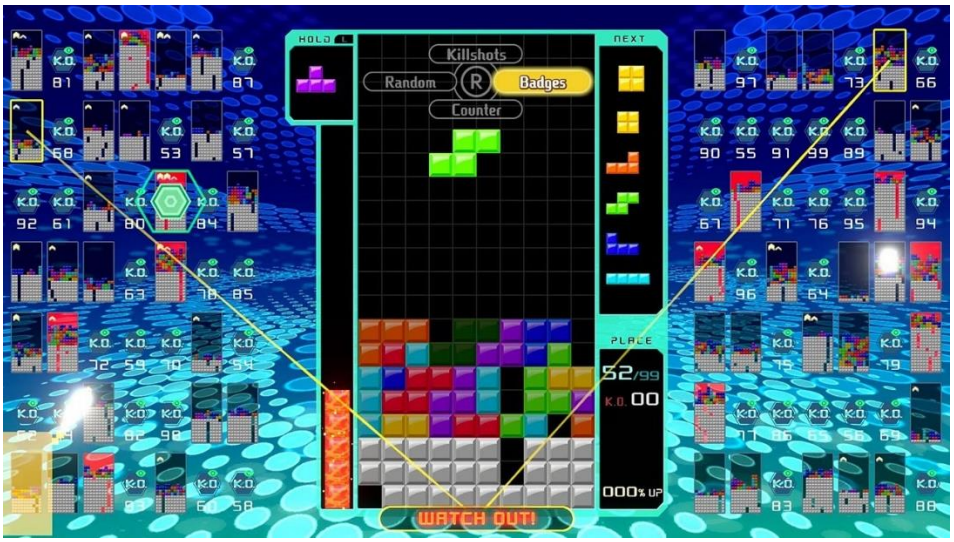


Рисунок 1.1 – Игровое приложение «Тетрис 99»

Визуально это самый обычный тетрис, в котором для удобства фигуры покрашены в разные цвета, а спецэффекты почти полностью отсутствуют, но на самом деле это настоящая королевская битва. Вы запускаете поиск оппонентов, за 20 секунд собирается лобби из 99 участников, и начинается матч. В самом центре экрана расположен ваш стакан, а слева и справа от него видны стаканы оппонентов. Вы играете, и вдруг внизу появляются странные бесцветные линии с одним отверстием в каждой из них. Связано это с тем, что другие игроки вас атакуют. То же делаете и вы: если уничтожены две линии, врагу вы передаёте одну, если четыре – то четыре. Очевидно, при переполненном стакане вы выбываете из игры, а игроку, чьи линии вас погубили, засчитывается убийство.

Ещё один аналог «Tetris Effect» проект японской студии Resonair. В основе лежит самый обыкновенный тетрис со стаканом 10 на 20, знакомыми фигурами, стирающимися линиями и прочими привычными элементами.

Однако это новый современный взгляд на классическую игру. Основной режим – это путешествие по разным мирам и эпохам. Игрока заносит в пустыню с прохаживающимися по песку верблюдами, на дно океана, где плавает какая-то живность или в космос с летающими искусственными спутниками. С окружением меняется и дизайн фигурок. Каждый уровень непохож на предыдущий. Игровой процесс и визуальная составляющая тесно связана с музыкой. Эта версия тетриса сохранила в себе нововведения, которые появились в подобных проектах за последние годы, в том числе возможность поворачивать фигуры даже после её соприкосновения с дном стакана или конструкцией. Для желающих просто поиграть в стандартный тетрис тоже есть соответствующие режимы – и обычная версия, в которой невозможно проиграть, и ускоренная, где фигурки не падают сверху, а сразу появляются в нижней части экрана. Внешний вид данного аналога представлен на рисунке 1.2.

Рисунок 1.2 – Игровое приложение «Tetris Effects»

1.2. Постановка задачи

В рамках данного курсового проекта планируется разработка игрового программного средства «Тетрис», адаптированного под запуск на современных операционных системах, в частности, версиях ОС Windows 8 и старше, на компьютерах с 4К-разрешением.

В процессе реализации будет разработан алгоритм заполнения и прорисовки игрового поля, игровой логики и взаимодействия с пользователем.

В игровом средстве планируется реализовать следующие функции:

- поворот выпавшей фигуры;
- временная остановка игры;
- выход из временной остановки;
- отображение текущего результата на экране;
- выход из игры;

Для разработки программного средства будет использоваться язык программирования Си, интерфейс Windows API (Application Programming Interfaces) и среда разработки Microsoft Visual Studio 2019.

Работа через Windows API – это наиболее близкий к операционной системе способ взаимодействия с ней из прикладных программ, представляющих собой множество функций, структур данных и числовых констант, следующих соглашениям вызова Си.

2. РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

2.1. Структура программы

В данном программном средстве использован модуль TOMA’STETRIS, который является основным логическим модулем. В данном модуле подключён заголовочный файл Shapes, в котором определён массив, задающий 18 фигуры. В заголовочном файле использована директива #ifndef и #endif, что в случае многократного включения данного файла в программу не вызовет ошибку компиляции связанной с переопределением массива. Каждая фигура определена матрицей 4 на 4. Проект представляет собой классическое приложение Windows.

2.2. Интерфейс программного средства

Внешний вид и удобность в использовании являются одними из главных критериев качества программного средства. Поэтому взаимодействие приложения с пользователем организовано максимально интуитивно и просто.

Программное средство имеет одно основное окно. Данное окно представлено на рисунке 2.1. В левой части окна располагается игровой стакан размером 15 на 30. Особенностью данного стакана является отсутствие разметки. Это не только делает вид интерфейса программного средства более современным, но и развивает глазомер пользователя. Над игровым стаканом пользователь может увидеть текущий счет. В правой части располагаются инструкции, благодаря которым можно понять по нажатию на какую клавишу можно осуществить желаемое действие. Над инстукциями выводится текущий рекорд пользователя в игре.

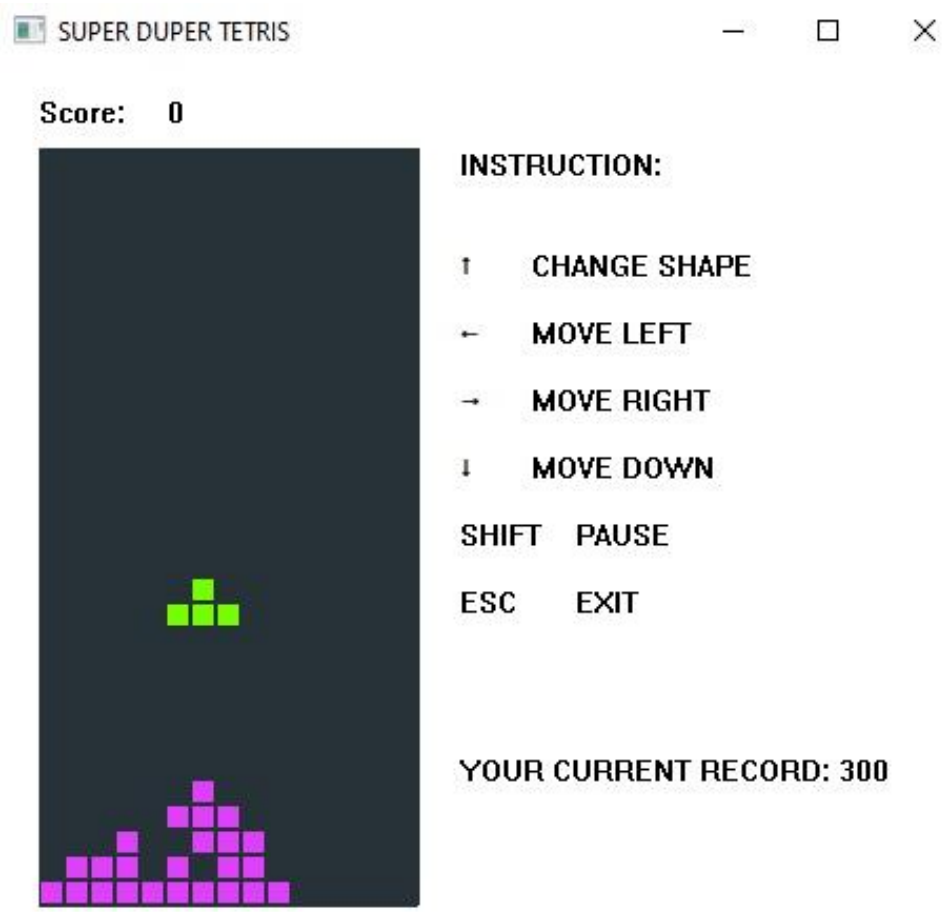


Рисунок 2.1 – Основное окно

Если фигура соприкасается с верхом игрового стакан, то игра завершается. На экране появляется диалоговое окно с результатом, а также вопросом к пользователю хочет ли он сыграть еще раз. Если пользователь установил новый рекорд, то в диалоговом окне это будет отображено. По нажатию на кнопку «да» игра, начинается сначала. По нажатию на кнопку «нет» программное средство закрывается.

2.3. Работа с фигурами

Изначально было реализовано 7 фигур. Однако для усложнения прохождения игры было принято решение увеличить число фигур и сделать его равным 18. Шаблоны фигур представлены на рисунках 2.2 и 2.3

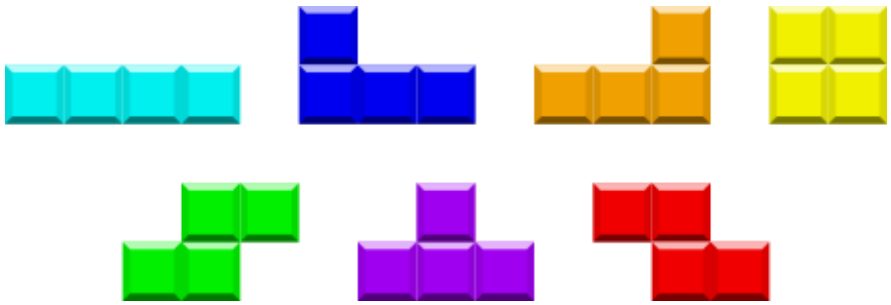


Рисунок 2.2 – Стандартные фигуры игры «Тетрис»

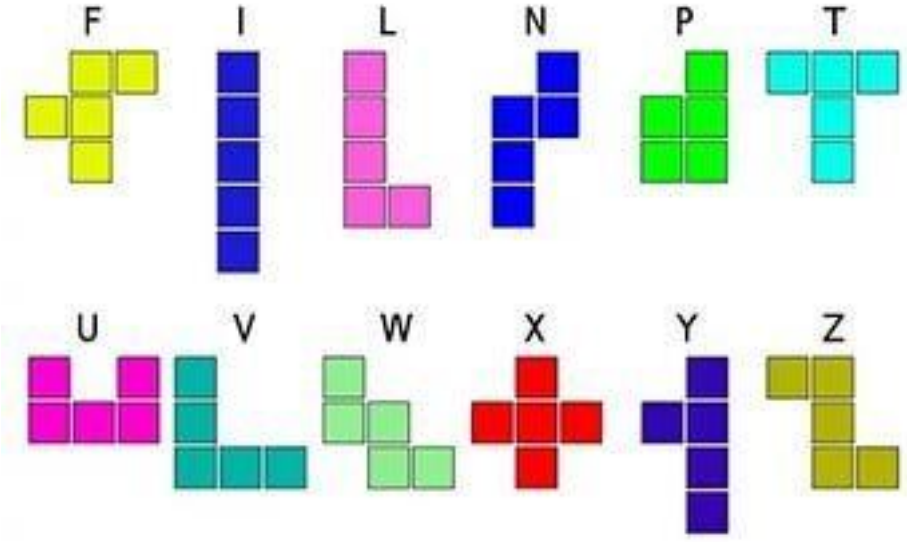


Рисунок 2.3 – Фигуры для усложнения игры «Тетрис»

Информация о фигурах хранится в трёхмерном целочисленном массиве shapes, расположенном в заголовочном файле Shapes.h. Какая фигура выпадет следующей определяется случайно с помощью генератора псевдослучайных чисел.

Информация о фигуре представляет собой матрицу размера 4 на 4 из 0 и 1. Фигура отрисовывается следующим образом: поочерёдно просматривается каждый элемент матрицы, если он 0, то происходит переход к следующему элементу, если он 1, вызывается функция PaintCell, которая отрисовывает 1клетку игрового стакана. В данном программном средстве реализован поворот фигуры на 90 градусов. Осуществляется это за счёт поворота и перерисовки соответствующей матрицы.

Фигура, еще не коснувшаяся дна стакана, имеет аквамариновый цвет. Как только она соприкасается со дном, цвет становится синим. Фигура отрисовывается с помощью объекта кисть. Создается логическая кисть с помощью функции CreateSolidBrush. Цвет задаётся в формате RGB.

2.4. Работа с рекордами

Текущий рекорд пользователя зафиксирован в файле record.dat. По завершению игры вызывается функция WorkWithRecords. В данной функции считывается текущее значение рекорда из файла и сравнивается с текущим счетом пользователя. Если текущий счет меньше рекорда, значение в файле обновляется, пользователь видит сообщение о том, что поставлен новый рекорд.

2.5. Работа с таймером

Движение фигуры вниз происходит за счет перерисовки по сообщению от таймера. Для создания виртуального таймера используется функция SetTimer. Период следования сообщений от таймера в миллисекундах в данном программном средстве равен 1000 миллисекунд, то есть 1 секунда, если строка результата показывает меньше 500, если же строка результата 500 и более, длина интервала уменьшается в 10 раз и становится равной 100 миллисекунд, если строка результата 1000 и более, длина интервала уменьшается в 100 раз и становится равной 10 миллисекунд, что существенно затрудняет процесс игры.

Пауза реализована с помощью уничтожения таймера, а затем его повторного создания. При нажатии на клавишу SHIFT вызывается функция PauseGame, в которой с помощью функции KillTimer уничтожается таймер. При возобновлении игры вызывается функция WakeGame, в которой повторно создается таймер с помощью функции SetTimer.

2.6. Работа с инструкциями

Для вывода инструкций используется функция TextOut, которая выводит строку символов содержащую инструкцию в заданном месте. Предварительно текст инструкции форматируется с помощью функции wsprintf.

2.7. Работа с игровым стаканом

Игровой стакан представляет собой прямоугольник. Отрисовка происходит в сложном цикле с помощью функции PaintCell по заданным значениям BOARD_TOP, BOARD_BOTTOM, BOARD_LEFT, BOARD_RIGHT.

2.8. Игровая логика

В начале каждой новой игры на экране пользователь видит пустое игровое поле. Строка результата устанавливается в 0. Затем появляется фигура, выбранная случайным образом. Новая фигура появляется только после соприкосновения текущей фигуры с дном стакана или с другой фигурой. После соприкосновения фигура окращивается в синий цвет.

Если строка полностью заполняется, то происходит ее очистка. Значение строки результата увеличивается на 100.

При нажатии на клавишу Esc происходит выход из игры. Управление фигурами осуществляется клавишами управления курсором. Работа клавиш описывается в функции RespondKey.

Если текущая фигура после соприкосновения с другой фигурой коснулась верха игрового стакана, игра завершается. На экране появляется сообщение с результатом. Также пользователю задаётся вопрос, хочет ли он продолжить игру, по нажатию на кнопку «да» игра возобновляется, по нажатию на кнопку «нет» программное средство закрывается.

После соприкосновения с дном стакана или с границами другой фигуры текущая фигура становится неподвижна и окрашивается в фиолетовый цвет. Это позволяет пользователю легко отследить уровень заполненности поля.

Окраска в фиолетовый цвет осуществляется за счет выбора цвета логической кисти, которая создаётся функцией CreateSolidBrush. Цвет задаётся в формате RGB.

3. ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

Перед выполнением тест кейсов необходимо запустить программное средство.

3.1. Тестирование функционала программы

Таблица 3.1 – Тестирование функционала программы

Заголовок тест кейса	Шаги	Ожидаемый результат	Полученный результат	Статус тест кейса пройден/провален
Выход из программного средства	1.Нажать клавишу esc	Программное средство закрывается	Программное средство закрылось	Пройден
Остановка игры	1.Нажать клавишу shift	Игра останавливается, значение строки результата сохраняется	Игра остановилась, значение строки результата сохранилось	Пройден
Возобновление игры после остановки	1.Пройти тест кейс «Остановка игры» 2.Нажать клавишу shift	Игра возобновляется с текущим результатом	Игра возобновилась с текущим результатом	Пройден
Поворот фигуры на 90 градусов	1.Нажать на верхнюю клавишу управления курсором	Текущая фигура поворачивается на 90 градусов	Фигура повернулась на 90 градусов, при повторном нажатии фигура ещё раз повернулась на 90 градусов	Пройден

Продолжение таблицы 3.1

Заголовок тест кейса	Шаги	Ожидаемый результат	Полученный результат	Статус тест кейса пройден/провален
Перемещение фигуры влево	1.Нажать на левую клавишу управления курсором	По одному нажатию фигура сдвигается влево на 1 игровую клетку	Фигура сдвинулась влево на 1 игровую клетку, при повторном нажатии сдвинулась влево ещё на 1 игровую клетку	Пройден
Перемещение фигуры вправо	1.Нажать на правую клавишу управления курсором	По одному нажатию фигура сдвигается на 1 игровую клетку вправо	Фигура сдвинулась на 1 игровую клетку вправо, при повторном нажатии ещё на 1 игровую клетку вправо	Пройден
Перемещение фигуры вниз	1.Нажать на нижнюю клавишу управления курсором	По одному нажатию фигура сдвигается на 1 игровую клетку вниз	Фигура сдвинулась на 1 игровую клетку вниз, при повторном нажатии ещё на 1 игровую клетку вниз	Пройден

Продолжение таблицы 3.1

Заголовок тест кейса	Шаги	Ожидаемый результат	Полученный результат	Статус тест кейса пройден/провален
Поведение программного средства при соприкосновении фигуры с дном игрового стакана (если это возможно, т.е. дно не полностью заполнено и к нему есть доступ)	1.С помощью клавиш управления курсором поместить текущую фигуру на дно игрового стакана	Фигура становится неподвижной, цвет текущей фигуры изменяется на фиолетовый, появляется новая фигура	Цвет фигуры изменился на фиолетовый, фигура не реагирует на нажатие клавиш управления курсора	Пройден
Поведение программного средства при соприкосновении фигуры с другой фигурой	1.С помощью клавиш управления курсором прикоснутьс я текущей фигурой к другой фигуре	Фигура становится неподвижной, цвет текущей фигуры изменяется на фиолетовый, появляется новая фигура	Цвет фигуры изменился на фиолетовый, фигура не реагирует на нажатие клавиш управления курсора	Пройден
Поведение программного средства при соприкосновении фигуры с верхом игрового стакана	1.С помощью клавиш управления курсором и фигур коснуться верха игрового стакана	Игра заканчивается, на экране появляется сообщение с результатом и вопросом о том, хочет ли пользователь сыграть ещё раз	Игра закончилась, на экране появилось сообщение с результатом и вопросом о том, хочет ли пользователь сыграть ещё раз	Пройден

Продолжение таблицы 3.1

Заголовок тест кейса	Шаги	Ожидаемый результат	Полученный результат	Статус тест кейса пройден/провален
Поведение программного средства при нажатии на кнопку «да» в диалоговом окне, появляющемся при завершении игры	1.Нажать на кнопку «да» в диалоговом окне	Игра начинается заново, строка результата устанавливается в 0	Игра началась заново, строка результата установилась в 0	Пройден
Поведение программного средства при нажатии на кнопку «нет» в диалоговом окне, появляющемся при завершении игры	1.Нажать на кнопку «нет» в диалоговом окне	Программное средство закрывается	Программное средство закрылось	Пройден
Поведение программного средства при заполнении всей строки игрового поля	1.С помощью клавиш управления курсором и фигур заполнить целую строку игрового поля	Строка очищается, фигуры на игровом поле сдвигаются на 1 клетку вниз, значение строки результата увеличивается на 100	Строка очистилась, фигуры на игровом поле сдвинулись на 1 клетку вниз, значение строки результата увеличилось на 100	Пройден

Продолжение таблицы 3.1

Заголовок тест кейса	Шаги	Ожидаемый результат	Полученный результат	Статус тест кейса пройден/провален
Поведение программного средства при достижении результата 1000 и более	1.Играть и достигнуть результата 1000 и более	Скорость движения фигур увеличивается, прохождение игры затрудняется	Скорость существенно увеличилась, играть практически невозможно	Пройден

3.2. Вывод из прохождения тестирования

Приложение успешно прошло все тесты, что показывает корректность работы разработанного программного средства и соответствие функциональным требованиям.

4. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

4.1. Правила игры

Правила полностью соответствуют классической концепции игры «Тетрис». Случайные фигурки тетрамино падают сверху в прямоугольный стакан шириной 15 и высотой 30 клеток. В полёте игрок может поворачивать фигурку на 90° и двигать её по горизонтали. Также можно «сбрасывать» фигурку, то есть ускорять её падение, когда уже решено, куда фигурка должна упасть. Фигурка летит до тех пор, пока не наткнётся на другую фигурку либо на дно стакана. Если при этом заполнился горизонтальный ряд из 30 клеток, он пропадает и всё, что выше него, опускается на одну клетку. Дополнительно показывается фигурка, которая будет следовать после текущей — это подсказка, которая позволяет игроку планировать действия. Темп игры постепенно ускоряется. Игра заканчивается, когда новая фигурка не может поместиться в стакан. Игрок получает очки за каждый заполненный ряд, поэтому его задача — заполнять ряды, не заполняя сам стакан (по вертикали) как можно дольше, чтобы таким образом получить как можно больше очков.

Если убирается одна линия, пользователю начисляется 100 очков.

4.2. Интерфейс программы

Главное окно программы, изображенное на рисунке 4.1, содержит следующие элементы:

– строка результата;
– игровое поле;
– инструкция к игре;

В программе присутствует визуальное сопровождение. Проверка и загрузка ресурсов происходит в момент запуска приложения. При этом поиск файлов осуществляется непосредственно в директории программы.

Для поворота фигуры на 90 градусов нужно нажать на верхнюю клавишу управления курсором. Для перемещения фигуры вниз – нижнюю клавишу управления курсором. Для перемещения фигуры влево – левую клавишу управления курсором. Для перемещения фигуры вправо – правую клавишу управления курсором. Для того, чтобы поставить игру на паузу, требуется нажать на клавишу Shift. Повторное нажатие на данную клавишу возобновляет игру. Для того, чтобы выйти из игры, требуется нажать на клавишу esc.

Результат можно увидеть в левом верхнем углу главного окна

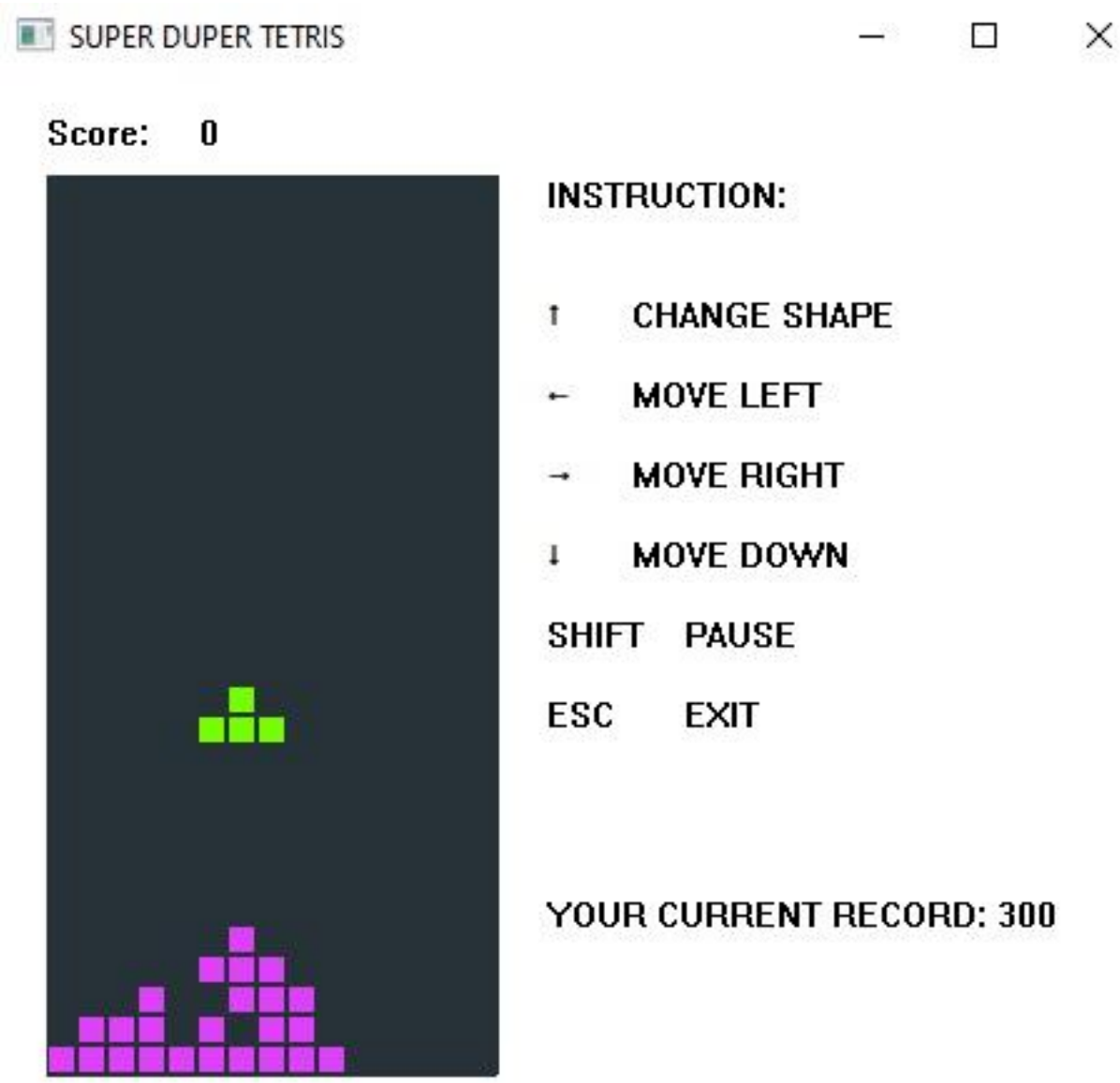


Рисунок 4.1 – Внешний вид главного окна программы

ЗАКЛЮЧЕНИЕ

В настоящее время популярность компьютерных игр с каждым днем становится все больше. Немалая часть из них не несет никакой полезной нагрузки, однако логический жанр не относится к таковым. Такие игры как «Cut the Rope», «2048» и подобные способствуют развитию внимания и логического мышления, если не проводить у экрана излишне много времени.

В рамках данного курсового проекта было разработано игровое программное средство «Тетрис», которое обеспечит веселое и увлекательное времяпрепровождение. Согласно поставленным задачам, в данном приложении были реализованы следующие функции:

– выбор режима игры;
– изменение уровня сложности;
– визуализация таблицы рекордов;
– звуковое сопровождение.

Для успешного выполнения всех поставленных целей потребовалось изучить объектно-ориентированные возможности языка Delphi, изучить основные принципы данной парадигмы, а также освоить создание собственных компонентов на основе уже существующих.

Существует много возможностей для дальнейшего улучшения приложения. Одним из самых простых направлений является разработка новых игровых алгоритмов и введение других режимов игры. Также возможна переработка визуальной концепции игры, как например, переход к 3D-изображениям. Ещё одним вариантом развития является адаптация проекта для запуска на устройствах с низкой разрешающей способностью экрана и переход к кроссплатформенной разработке.

Использование данного приложения позволит не только потратить время на игру, но также и развить свое внимание, логическое мышление и способность просчитывать действия на несколько шагов вперед.

СПИСОК ЛИТЕРАТУРЫ

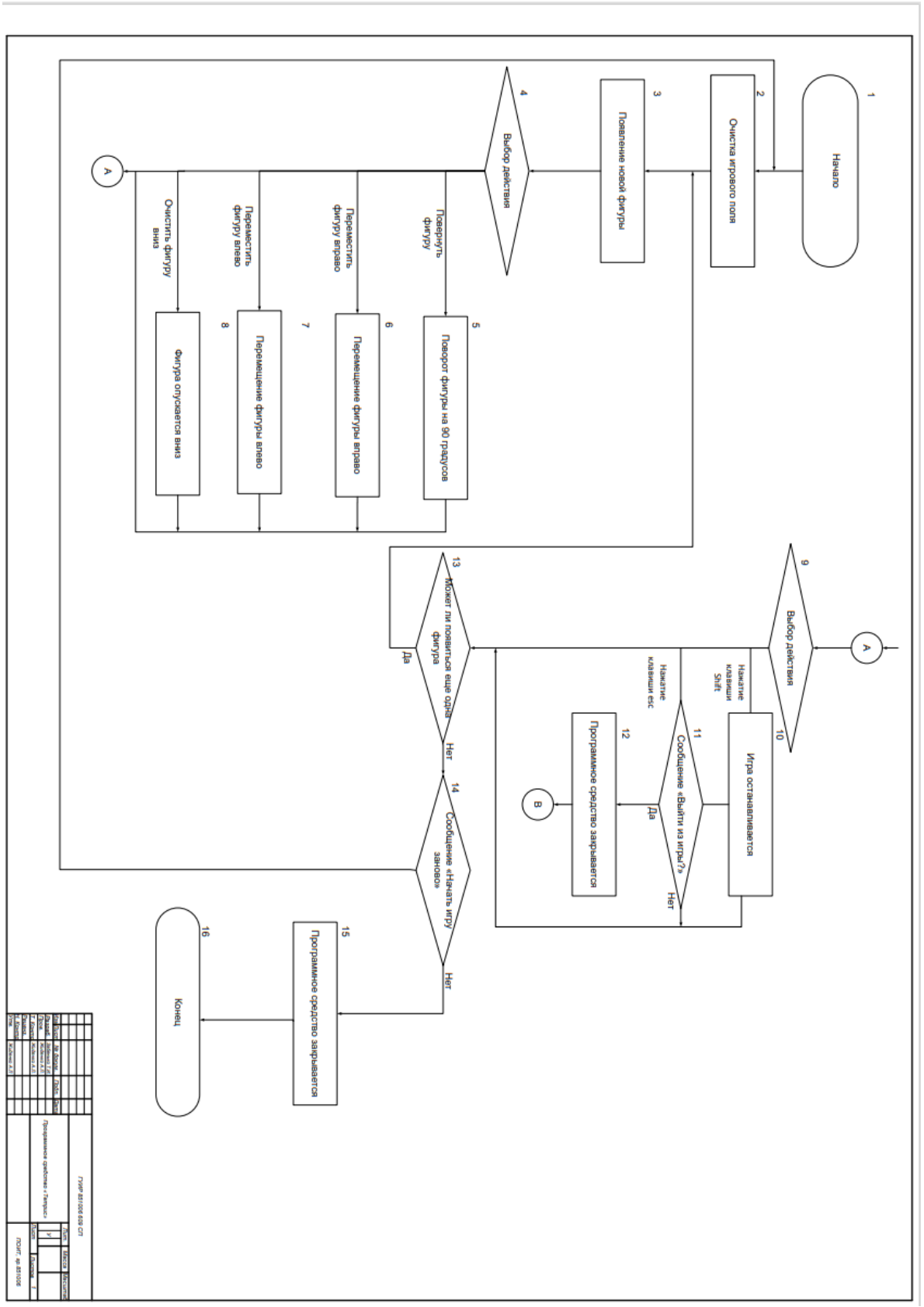
[1] ГОСТ 19.701–90 (ИСО 5807–85) [Текст]. – Единая система программной документации: Сб. ГОСТов. - М.: Стандартинформ, 2005 с.

[2] Библиотека MSDN

[3] https://en.wikipedia.org/wiki/Tetris - Tetris

ПРИЛОЖЕНИЕ 1

Общая схема работы программного средства «Тетрис»



ПРИЛОЖЕНИЕ 2

Исходный код программы

Файл Toma's Tetris.cpp:

```
#include "Toma'sTetris.h"
#include <windows.h>
#include <string.h>
#include <stdio.h>
#include <time.h>
#include "Shapes.h"

#define ID_TIMER  1
#define TIME_INTERVAL 1000

;LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
VOID CALLBACK TimerProc(HWND, UINT, UINT, DWORD);
#define BOARD_WIDTH 180
#define BOARD_HEIGHT 400
#define LONG_SLEEP 150

#define COLS 15
#define ROWS 30
#define EXTENDED_COLS 23
#define EXTENDED_ROWS 34

#define BOARD_LEFT 4
#define BOARD_RIGHT 18
#define BOARD_TOP 0
#define BOARD_BOTTOM 29

#define FIGURES_NUMBER 18

static int shape[4][4];
static int score = 0;

static int shape_row = 0;
```

```
static int shape_col = EXTENDED_COLS / 2 - 2;

static int** gBoard;
static int shape_num;

static int lattices_top = 40;
static int lattices_left = 20;
static int width = BOARD_WIDTH / COLS;
static int height = (BOARD_HEIGHT - lattices_top) / ROWS;

static HBRUSH white_brush = CreateSolidBrush(RGB(38, 50, 56));
static HBRUSH blue_brush = CreateSolidBrush(RGB(3, 169, 244));
static HPEN hPen = CreatePen(PS_SOLID, 1, RGB(38, 50, 56));
static HBRUSH turquoise_brush = CreateSolidBrush(RGB(24, 255, 255));
static bool gIsPause = false;

void InitGame(HWND);
void InitData();

void TypeInstruction(HWND);

void RandShape();

void AddScore();

void UpdateShapeRect(HWND hwnd);
void UpdateAllBoard(HWND hwnd);

void FallToGround();
void MoveDown(HWND hwnd);
void RePaintBoard(HDC hdc);
void PaintCell(HDC hdc, int x, int y, int color);
void ClearFullLine();

void RotateShape(HWND hwnd);
void MoveHori(HWND hwnd, int direction);
```

```
void RotateMatrix();
void ReRotateMatrix();
bool IsLegel();

void RespondKey(HWND hwnd, WPARAM wParam);

void PauseGame(HWND hwnd);
void WakeGame(HWND hwnd);

bool JudgeLose();
void LoseGame(HWND hwnd);
void ExitGame(HWND hwnd);
DWORD ReadRecord();
void WriteRecord(DWORD);
int WorkWithRecords(int);

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
       PSTR szCmdLine, int iCmdShow)
{
       static TCHAR szAppName[] = TEXT("SUPER DUPER TETRIS");
       HWND         hwnd;
       MSG          msg;
       WNDCLASS     wndclass;

       wndclass.style = CS_HREDRAW | CS_VREDRAW;
       wndclass.lpfnWndProc = WndProc;
       wndclass.cbClsExtra = 0;
       wndclass.cbWndExtra = 0;
       wndclass.hInstance = hInstance;
       wndclass.hIcon = LoadIcon(NULL, IDI_APPLICATION);
       wndclass.hCursor = LoadCursor(NULL, IDC_ARROW);
       wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
       wndclass.lpszMenuName = NULL;
       wndclass.lpszClassName = szAppName;

       if (!RegisterClass(&wndclass))
```

```
	{
		MessageBox(NULL, TEXT("Program requires Windows NT!"),
			szAppName, MB_ICONERROR);
		return 0;
	}

	hwnd = CreateWindow(szAppName, TEXT("SUPER DUPER TETRIS"),
		WS_OVERLAPPEDWINDOW | WS_THICKFRAME | WS_MAXIMIZEBOX,
		CW_USEDEFAULT, CW_USEDEFAULT,
		BOARD_WIDTH + 300, BOARD_HEIGHT + 100,
		NULL, NULL, hInstance, NULL);

	ShowWindow(hwnd, iCmdShow);
	UpdateWindow(hwnd);

	TypeInstruction(hwnd);

	while (GetMessage(&msg, NULL, 0, 0))
	{
		TranslateMessage(&msg);
		DispatchMessage(&msg);
	}
	return msg.wParam;
}

LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
	static HDC hdc;
	static HDC hdcBuffer;
	static HBITMAP hBitMap;
	static PAINTSTRUCT ps;

	switch (message)
	{
	case WM_CREATE:
		SetTimer(hwnd, ID_TIMER, TIME_INTERVAL, TimerProc);
```

```
			InitGame(hwnd);
			TypeInstruction(hwnd);
			return 0;
		case WM_SIZE:
			TypeInstruction(hwnd);
			return 0;
		case WM_KEYDOWN:
			RespondKey(hwnd, wParam);
			return 0;
		case WM_PAINT:
			hdc = BeginPaint(hwnd, &ps);
			if (score == 300) {
				KillTimer(hwnd, ID_TIMER);
				SetTimer(hwnd, ID_TIMER, TIME_INTERVAL / 10, TimerProc);
			}
			if (score == 1000) {
				KillTimer(hwnd, ID_TIMER);
				SetTimer(hwnd, ID_TIMER, TIME_INTERVAL / 100, TimerProc);
			}
			RePaintBoard(hdc);
			EndPaint(hwnd, &ps);
			return 0;
		case WM_DESTROY:
			KillTimer(hwnd, ID_TIMER);
			PostQuitMessage(0);
			return 0;
		}
		return DefWindowProc(hwnd, message, wParam, lParam);
}

VOID CALLBACK TimerProc(HWND hwnd, UINT message, UINT iTimerID, DWORD dwTime)
{
		MoveDown(hwnd);
}

void InitGame(HWND hwnd)
```

```
{
	gBoard = new int* [EXTENDED_ROWS];
	for (int i = 0; i < EXTENDED_ROWS; i++)
	{
		gBoard[i] = new int[EXTENDED_COLS];
	}
	srand(time(0));
	InitData();
	UpdateAllBoard(hwnd);
}

void InitData() {
	for (int i = 0; i < EXTENDED_ROWS; i++)
	{
		for (int j = 0; j < EXTENDED_COLS; j++)
		{
			gBoard[i][j] = 0;
		}
	}

	for (int i = 0; i < EXTENDED_ROWS; i++)
	{
		for (int j = 0; j < BOARD_LEFT; j++)
		{
			gBoard[i][j] = 1;
		}
	}

	for (int i = 0; i < EXTENDED_ROWS; i++)
	{
		for (int j = BOARD_RIGHT + 1; j < EXTENDED_COLS; j++)
		{
			gBoard[i][j] = 1;
		}
	}
```

```
	for (int i = BOARD_BOTTOM + 1; i < EXTENDED_ROWS; i++)
	{
		for (int j = 0; j < EXTENDED_COLS; j++)
		{
			gBoard[i][j] = 1;
		}
	}

	gIsPause = false;
	score = 0;
	RandShape();
	return;
}

void TypeInstruction(HWND hwnd)
{
	TEXTMETRIC  tm;
	int cxChar, cxCaps, cyChar, cxClient, cyClient, iMaxWidth;
	HDC hdc = GetDC(hwnd);
	GetTextMetrics(hdc, &tm);
	cxChar = tm.tmAveCharWidth;
	cxCaps = (tm.tmPitchAndFamily & 1 ? 3 : 2) * cxChar / 2;
	cyChar = tm.tmHeight + tm.tmExternalLeading;
	int startX = 180;
	int startY = 40;
	TCHAR Instruction[100];

	wsprintf(Instruction, TEXT("INSTRUCTION: "));
	TextOut(hdc, startX + 40, startY, Instruction, lstrlen(Instruction));

	wsprintf(Instruction, TEXT("↑       CHANGE SHAPE"));
	TextOut(hdc, startX + 40, startY + cyChar * 3, Instruction,
lstrlen(Instruction));

	wsprintf(Instruction, TEXT("←      MOVE LEFT"));
	TextOut(hdc, startX + 40, startY + cyChar * 5, Instruction,
lstrlen(Instruction));
```

```
	wsprintf(Instruction, TEXT("→	MOVE RIGHT"));
	TextOut(hdc, startX + 40, startY + cyChar * 7, Instruction, lstrlen(Instruction));

	wsprintf(Instruction, TEXT("↓	MOVE DOWN"));
	TextOut(hdc, startX + 40, startY + cyChar * 9, Instruction, lstrlen(Instruction));

	wsprintf(Instruction, TEXT("SHIFT	PAUSE "));
	TextOut(hdc, startX + 40, startY + cyChar * 11, Instruction, lstrlen(Instruction));

	wsprintf(Instruction, TEXT("ESC	EXIT "));
	TextOut(hdc, startX + 40, startY + cyChar * 13, Instruction, lstrlen(Instruction));

	DWORD dwRecord = ReadRecord();
	wsprintf(Instruction, TEXT("YOUR CURRENT RECORD: %d"), dwRecord);
	TextOut(hdc, startX + 40, startY + cyChar * 18, Instruction, lstrlen(Instruction));

	ReleaseDC(hwnd, hdc);
}

void RandShape()
{
	shape_num = rand() % FIGURES_NUMBER;

	for (int i = 0; i < 4; i++)
	{
		for (int j = 0; j < 4; j++)
		{
			shape[i][j] = shapes[shape_num][i][j];
		}
	}
}
```

```
void UpdateAllBoard(HWND hwnd)
{
	static RECT rect;
	rect.left = lattices_left;
	rect.right = lattices_left + COLS * width + width;
	rect.top = lattices_top - 30;
	rect.bottom = lattices_top + ROWS * height;
	InvalidateRect(hwnd, &rect, false);
}

void UpdateShapeRect(HWND hwnd)
{
	static RECT rect;
	rect.left = lattices_left;
	rect.right = lattices_left + COLS * width + width;
	rect.top = lattices_top + (shape_row - 1) * height;
	rect.bottom = lattices_top + (shape_row + 4) * height;
	InvalidateRect(hwnd, &rect, false);
}

void RePaintBoard(HDC hdc)
{
	SetBkColor(hdc, RGB(255, 255, 255));
	SelectObject(hdc, hPen);
	TCHAR score_str[50];
	wsprintf(score_str, TEXT("Score: %5d "), score);
	TextOut(hdc, 20, 15, score_str, lstrlen(score_str));
	for (int i = BOARD_TOP; i <= BOARD_BOTTOM; i++)
	{
		for (int j = BOARD_LEFT; j <= BOARD_RIGHT; j++)
		{
			PaintCell(hdc, i, j, gBoard[i][j]);
		}
	}
	for (int i = 0; i < 4; i++)
	{
```

```
			for (int j = 0; j < 4; j++)
			{
				if (shape[i][j] == 1)
					PaintCell(hdc, shape_row + i, shape_col + j, shape[i][j]);
			}
		}
}

void PaintCell(HDC hdc, int x, int y, int color)
{
		if (x < BOARD_TOP || x > BOARD_BOTTOM ||
			y < BOARD_LEFT || y > BOARD_RIGHT)
		{
			return;
		}

		x -= BOARD_TOP;
		y -= BOARD_LEFT;

		int _left = lattices_left + y * width;
		int _right = lattices_left + y * width + width;
		int _top = lattices_top + x * height;
		int _bottom = lattices_top + x * height + height;

		MoveToEx(hdc, _left, _top, NULL);
		LineTo(hdc, _right, _top);
		MoveToEx(hdc, _left, _top, NULL);
		LineTo(hdc, _left, _bottom);
		MoveToEx(hdc, _left, _bottom, NULL);
		LineTo(hdc, _right, _bottom);
		MoveToEx(hdc, _right, _top, NULL);
		LineTo(hdc, _right, _bottom);

		if (color == 0)
		{
			SelectObject(hdc, white_brush);
```

```
	}
	else if (color == 1)
	{
		SelectObject(hdc, turquoise_brush);
	}
	else if (color == 2)
	{
		SelectObject(hdc, blue_brush);
	}

	Rectangle(hdc, _left, _top, _right, _bottom);
}

void RespondKey(HWND hwnd, WPARAM wParam)
{

	if (wParam == VK_ESCAPE)
	{
		ExitGame(hwnd);
		return;
	}

	if (wParam == VK_SHIFT)
	{
		gIsPause = !gIsPause;
		if (gIsPause == true)
		{
			PauseGame(hwnd);
			return;
		}
		else
		{
			WakeGame(hwnd);
			return;
		}
	}
```

```
	if (!gIsPause)
	{
		if (wParam == VK_UP)
		{
			RotateShape(hwnd);
			return;
		}
		if (wParam == VK_DOWN)
		{
			MoveDown(hwnd);
			return;
		}
		if (wParam == VK_LEFT)
		{
			MoveHori(hwnd, 0);
			return;
		}
		if (wParam == VK_RIGHT)
		{
			MoveHori(hwnd, 1);
			return;
		}
	}
}

void PauseGame(HWND hwnd)
{
	KillTimer(hwnd, ID_TIMER);
}

void WakeGame(HWND hwnd)
{
	SetTimer(hwnd, ID_TIMER, TIME_INTERVAL, TimerProc);
}
```

```
void ExitGame(HWND hwnd)
{

	SendMessage(hwnd, WM_KEYDOWN, VK_SPACE, 0);

	int exit = MessageBox(NULL, TEXT("Do you want to exit?"), TEXT("EXIT"),
MB_YESNO);

	if (exit == IDYES)
	{
		SendMessage(hwnd, WM_DESTROY, NULL, 0);
	}

	else if (exit == IDNO)
	{
		return;
	}
}

void RotateShape(HWND hwnd)
{
	RotateMatrix();
	if (!IsLegel())
	{
		ReRotateMatrix();
	}
	UpdateShapeRect(hwnd);
	return;
}

bool IsLegel()
{
	for (int i = 0; i < 4; i++)
	{
		for (int j = 0; j < 4; j++)
		{
```

```
                    if (shape[i][j] == 1 && (gBoard[shape_row + i][shape_col + j] == 1
|| gBoard[shape_row + i][shape_col + j] == 2))
                    {
                           return false;
                    }
             }
      }
      return true;
}

void RotateMatrix()
{

      int(*a)[4] = shape;
      int s = 0;
      for (int n = 4; n >= 1; n -= 2)
      {
             for (int i = 0; i < n - 1; i++)
             {
                    int t = a[s + i][s];
                    a[s + i][s] = a[s][s + n - i - 1];
                    a[s][s + n - i - 1] = a[s + n - i - 1][s + n - 1];
                    a[s + n - i - 1][s + n - 1] = a[s + n - 1][s + i];
                    a[s + n - 1][s + i] = t;
             }
             s++;
      }
}

void ReRotateMatrix()
{
      int(*a)[4] = shape;
      int s = 0;
      for (int n = 4; n >= 1; n -= 2)
      {
             for (int i = 0; i < n - 1; i++)
             {
```

```
                int t = a[s + i][s];
                a[s + i][s] = a[s + n - 1][s + i];
                a[s + n - 1][s + i] = a[s + n - i - 1][s + n - 1];
                a[s + n - i - 1][s + n - 1] = a[s][s + n - i - 1];
                a[s][s + n - i - 1] = t;
            }
            s++;
        }
}

void MoveDown(HWND hwnd)
{
    shape_row++;

    if (!IsLegel())
    {
        shape_row--;
        if (JudgeLose())
        {
            LoseGame(hwnd);
            return;
        }

        FallToGround();
        ClearFullLine();
        UpdateAllBoard(hwnd);
        shape_row = 0;
        shape_col = EXTENDED_COLS / 2 - 2;
        RandShape();
    }
    UpdateShapeRect(hwnd);
}

bool JudgeLose()
{
    if (shape_row == 0)
```

```
	{
		return true;
	}
	return false;
}

void LoseGame(HWND hwnd)
{
	SendMessage(hwnd, WM_KEYDOWN, VK_SPACE, 0);
	TCHAR words[100];
	int IsRecord = WorkWithRecords(score);
	if (IsRecord == 1)
	{
		wsprintf(words, TEXT("Congratulations! \nYou set a new record %d. \nDo you want to try again?"), score);

	}
	else
	{
		wsprintf(words, TEXT("You have lost the Game. Total score: %d. \nDo you want to try again?"), score);
	}
	KillTimer(hwnd, ID_TIMER);
	int exit = MessageBox(NULL, words, TEXT("SUPER DUPER TETRIS"), MB_YESNO);

	if (exit == IDYES)
	{
		SendMessage(hwnd, WM_CREATE, NULL, 0);
		return;
	}
	else
	{
		SendMessage(hwnd, WM_DESTROY, NULL, 0);
		return;
	}
}
```

```
void FallToGround()
{
	for (int i = 0; i < 4; i++)
	{
		for (int j = 0; j < 4; j++)
		{
			gBoard[shape_row + i][shape_col + j] = shape[i][j] == 1 ? 2 : gBoard[shape_row + i][shape_col + j];
		}
	}
}

void ClearFullLine()
{
	for (int i = shape_row; i <= shape_row + 3; i++)
	{
		if (i > BOARD_BOTTOM)
		{
			continue;
		}

		bool there_is_blank = false;

		for (int j = BOARD_LEFT; j <= BOARD_RIGHT; j++)
		{
			if (gBoard[i][j] == 0) {
				there_is_blank = true;
				break;
			}
		}

		if (!there_is_blank)
		{
			AddScore();
			for (int r = i; r >= 1; r--) {
				for (int c = BOARD_LEFT; c <= BOARD_RIGHT; c++)
				{
```

```
                                gBoard[r][c] = gBoard[r - 1][c];
                        }
                }
        }
}

void AddScore()
{
        score += 100;
}

void MoveHori(HWND hwnd, int direction)
{
        int temp = shape_col;

        if (direction == 0)
                shape_col--;
        else
                shape_col++;

        if (!IsLegel()) {
                shape_col = temp;
        }

        UpdateShapeRect(hwnd);
        return;
}

const TCHAR szRecordsFileName[] = L"record.dat";

DWORD ReadRecord()
{
        DWORD dwRecord, dwTemp;
        HANDLE hFile = CreateFile(szRecordsFileName, GENERIC_READ, 0, NULL,
                OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
```

```
	if (INVALID_HANDLE_VALUE == hFile)
	{
		return 1;
	}
	ReadFile(hFile, &dwRecord, sizeof(dwRecord), &dwTemp, NULL);
	if (sizeof(dwRecord) != dwTemp)
	{
		CloseHandle(hFile);
		return 1;
	}
	CloseHandle(hFile);
	return dwRecord;
}

void WriteRecord(DWORD dwRecord)
{
	DWORD dwTemp;
	HANDLE hFile = CreateFile(szRecordsFileName, GENERIC_WRITE, 0, NULL,
		CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
	if (INVALID_HANDLE_VALUE == hFile)
	{
		return;
	}
	for (int i = 0; i < 3; i++)
	{
		WriteFile(hFile, &dwRecord, sizeof(dwRecord), &dwTemp, NULL);
	}

	CloseHandle(hFile);
}

int WorkWithRecords(int score) {
	DWORD dwRecord = ReadRecord();
	if (score > dwRecord)
	{
		WriteRecord(score);
```

```
                return 1;
        }
        return 0;
}
```

Файл Shapes.h:

```
#ifndef SHAPES_H
#define SHAPES_H

static int shapes[18][4][4] = {
        { { 0 , 0 , 0 , 0 },
      { 0 , 0 , 0 , 0 },
      { 0 , 0 , 0 , 0 },
      { 1 , 1 , 1 , 1 }
    },
    { { 0 , 0 , 0 , 0 },
      { 0 , 0 , 0 , 0 },
      { 0 , 1 , 0 , 0 },
      { 0 , 1 , 1 , 1 }
    },
    { { 0 , 0 , 0 , 0 },
      { 0 , 0 , 0 , 0 },
      { 0 , 0 , 0 , 1 },
      { 0 , 1 , 1 , 1 }
    },
    { { 0 , 0 , 0 , 0 },
      { 0 , 1 , 1 , 0 },
      { 0 , 1 , 1 , 0 },
      { 0 , 0 , 0 , 0 }
    },
    { { 0 , 0 , 0 , 0 },
      { 0 , 0 , 0 , 0 },
      { 0 , 1 , 1 , 0 },
      { 0 , 1 , 1 , 0 }
    },
    { { 0 , 0 , 0 , 0 },
      { 0 , 0 , 0 , 0 },
      { 0 , 0 , 1 , 0 },
      { 0 , 1 , 1 , 1 }
    },
    { { 0 , 0 , 0 , 0 },
      { 0 , 0 , 0 , 0 },
      { 0 , 1 , 1 , 0 },
      { 0 , 0 , 1 , 1 }
    },
    { { 0 , 0 , 0 , 0 },
      { 0 , 0 , 1 , 1 },
      { 0 , 1 , 1 , 0 },
      { 0 , 0 , 1 , 0 }
    },
    { { 1 , 0 , 0 , 0 },
      { 1 , 0 , 0 , 0 },
      { 1 , 0 , 0 , 0 },
      { 1 , 0 , 0 , 0 }
    },
    { { 1 , 0 , 0 , 0 },
      { 1 , 0 , 0 , 0 },
      { 1 , 0 , 0 , 0 },
      { 1 , 1 , 0 , 0 }
```

```
    },
    { { 0 , 1 , 0 , 0 },
      { 1 , 1 , 0 , 0 },
      { 1 , 0 , 0 , 0 },
      { 1 , 0 , 0 , 0 }
    },
    { { 0 , 0 , 0 , 0 },
      { 0 , 1 , 0 , 0 },
      { 1 , 1 , 0 , 0 },
      { 1 , 1 , 0 , 0 }
    },
    { { 0 , 0 , 0 , 0 },
      { 1 , 1 , 1 , 0 },
      { 0 , 1 , 0 , 0 },
      { 0 , 1 , 0 , 0 }
    },
    { { 0 , 0 , 0 , 0 },
      { 0 , 0 , 0 , 0 },
      { 1 , 0 , 1 , 0 },
      { 1 , 1 , 1 , 0 }
    },
    { { 0 , 0 , 0 , 0 },
      { 1 , 0 , 0 , 0 },
      { 1 , 0 , 0 , 0 },
      { 1 , 1 , 1 , 0 }
    },
    { { 0 , 0 , 0 , 0 },
      { 0 , 1 , 0 , 0 },
      { 1 , 1 , 1 , 0 },
      { 0 , 1 , 0 , 0 }
    },
    { { 0 , 0 , 0 , 0 },
      { 1 , 1 , 0 , 0 },
      { 0 , 1 , 0 , 0 },
      { 0 , 1 , 1 , 0 }
    },
    { { 0 , 1 , 0 , 0 },
      { 1 , 1 , 0 , 0 },
      { 0 , 1 , 0 , 0 },
      { 0 , 1 , 0 , 0 }
    }
}
#endif;
```

Обозначение	Наименование	Дополнительные сведения
	Текстовые документы	
БГУИР КП 1–40 01 01 609 ПЗ	Пояснительная записка	45 с.
	Графические документы	
ГУИР 851006 609 ПД	Блок-схемы работы приложения	Формат А1

					БГУИР КП 1-40 01 01 609 Д1		
Изм.	Л.	№ докум.	Подп.	Дата		Лист	Листов
Разраб.		Забенько Т.И.			Программное средство «Тетрис» Ведомость курсового проекта	Т	46 / 46
Пров.		Жиденко А.Л.					
						Кафедра ПОИТ гр. 851006	