Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

по дисциплине «Технология разработки программных систем»

НА ТЕМУ:

Игровая программа «Тетрис»

Студент	ИУ6-56		Т.А. Тищенко
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Руководитель			Т.Н. Ничушкина
		(Подпись, дата)	(И.О. Фамилия)

2019 г.

РЕФЕРАТ

Расчетно-пояснительная записка на __ с., __ рис., __ источников

ИГРОК, ФИГУРА, ПОЛЕ, ТЕТРИС, КЛЕТКА, ДВИЖЕНИЕ, ПОВОРОТ, МИНИМАЛИЗМ.

Программным продуктом, разрабатываемым в ходе данного курсового проекта, является игровая программа «Тетрис», предназначенная для организации игрового процесса.

Цель работы – проектирование программы, способной выполнять следующие функции:

– инициализация игры;

– поворот фигур;

– выбор направления движения фигуры;

– изменение уровня сложности игры;

– завершение игрового процесса;

– выход из игры.

В результате разработки была спроектирована и реализована программа, позволяющая создать игровую сцену и организовать игровой процесс в «Тетрис».

Пользователями данного продукта могут быть как люди, интересующиеся «Тетрисом», так и обычные пользователи планшетных компьютеров, заинтересованные данной игрой.

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

Полимино - плоские геометрические фигуры, образованные путём соединения нескольких одноклеточных квадратов по их сторонам.

Портировать – переносить с одного устройства на другое

Прокрастинация – это в психологии склонность к постоянному откладыванию даже важных и срочных дел, приводящая к жизненным проблемам и болезненным психологическим эффектам.

ТЗ – техническое задание

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Игровая индустрия предлагает широкий выбор различных игровых приложений, которые поражают великолепной графикой, первоклассным звуковым оформлением, увлекательным игровым процессом. Однако такие программы часто сложны в освоении, требуют много времени на установку, подключения к интернету, используют слишком много оперативной памяти компьютера, используют слишком много места на жестком диске, а также очень требовательны к системе, как в целом, так и в частности. Эти и другие сложности в использовании крупных игровых приложений заставляют пользователей отказываться от них в пользу более простых программ. Количество несложных игровых программ для планшетных компьютеров в последнее время увеличилось, однако игровое приложение «Тетрис», известное каждому, осталось без внимания разработчиков, которые не усовершенствовали первоначальную версию исходя из тенденций современного мира.

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. Первоначальная версия игры была написана Пажитновым на языке Паскаль для компьютера «Электроника-60». В 1987 году была выпущена первая коммерческая версия игры. В последующие годы Тетрис во множестве различных версий был портирован на великое множество устройств, включая все возможные компьютеры и игровые консоли, а также такие устройства, как графические калькуляторы, мобильные телефоны, медиаплееры, карманные персональные компьютеры, в том числе и осциллографы. По количеству проданных коммерческих версий Тетрис превосходит любую другую компьютерную игру в истории.

Из-за такой сильной популярности этой игры, было решено разработать собственный вариант игры в виде игровой программы «Тэтрис».

В разрабатываемой программе сохранится основополагающие механики классического Тетриса, такие как: поворот фигуры, смещение фигуры вправо или влево, формы фигур и постепенное увеличение скорости. Однако, в игровой программе «Тетрис» будут только основные особенности игры. Отсутствие лишних функций и анимаций позволит игроку погрузиться в мир прокрастинации, а не нагружать мозг лишними раздражителями.

1. Анализ предметной области и уточнение спецификаций

1.1 Анализ задания, выбор технологии, языка и средств

Как показал анализ технического задания, предметной областью задачи является игровой процесс. Анализ предметной области показал, что в ней можно выделить две сущности, которые будут непосредственно участвовать в игровом процессе: фигура и поле. Эти сущности обладают определенными свойствами. Так, поле хранит в себе все игровые клетки, а сами клетки могут быть заполнены или не заполнены фигурой. Кроме того, эти сущности обладают определенным поведением. Игровое поле проверяет состояние фигуры на столкновение. Поэтому для реализации приложения было решено выбрать объектный подход.

В соответствие с выбранным подходом, указанные сущности следует реализовать в виде классов со своими методами и полями.

По результатам решения об использовании объектного подхода к разработке игровой программы «Тетрис», была выделена группа языков программирования, которая отвечает требованиям объектно-ориентированного программирования.

Были проанализированы такие объектно-ориентированные языки программирования, как C++, Java, Turbo Delphi Pascal. Из этих трех языков был выбран Java, потому язык Java значительно быстрее своего аналога на языке Pascal, и имеет больше полезных функций которых нет в Turbo Delphi. Язык C++ не был выбран по той причине, что он проигрывает Java в производительности и скорости. В результате проведенного анализа был выбран язык Java, как наиболее знакомый и более приоритетный.

В качестве среды разработки приложения было решено выбрать Android Studio. Как показал анализ, эта среда поддерживает удобное графическое представление форм интерфейса, которое упрощает разработку приложения и обеспечивает быстрый доступ ко всем файлам приложения, что значительно ускоряет разработку приложения. Кроме того, среда поддерживает язык Java, выбранный в качестве языка разработки.

При разработке программного продукта целесообразно использовать спиральную модель жизненного цикла. В соответствии с данной схемой программное обеспечение создается не сразу, а итерационно с использованием метода прототипирования, базирующегося на создании прототипов. Прототип – действующий программный продукт, реализующий отдельные функции и внешние интерфейсы разрабатываемого программного обеспечения. Основным достоинством данной схемы является то, что, начиная с некоторой итерации, на которой обеспечена определенная функциональная полнота, продукт можно продемонстрировать пользователю.

В соответствие со спиральной моделью, на первом этапе было решено разработать интерфейсную часть. На следующем спроектировать основную часть программы, предназначенную для визуализации игрового пространства на экране и обработки действий пользователя. Далее будут реализованы функции движения фигур на игровом поле, ускорение движения фигур, а также проверка на условие столкновения фигуры с неподвижной частью поля.

1.2 Проектирование структуры программного продукта

1.2.1 Анализ игрового процесса и разработка вариантов использования

Как видно из ТЗ, функцией игровой программы является игровой процесс. Он состоит в том, что игрок должен двигать фигуру, которая будет случайным образом выбрана, до тех пор, пока не будет столкновения с неподвижной частью поля.

Разработку программы целесообразно начать с определения вариантов использования, которые выявляют внешних пользователей приложения и их взаимодействие с игровой программой.

Из ТЗ видно, что пользователь у приложения один – игрок. В процессе анализа основных функций были выделены следующие аспекты поведения

приложения при взаимодействии с пользователем:

Типичный ход событий.

Таблица 1 – Основной вариант использования

Действие исполнителя	Отклик системы
1. Игрок нажимает на кнопку «New Game».	2. Система запускает игру и генерирует первую фигуру.
3. Игрок управляет фигурой путём нажатия на кнопки, пока она не столкнётся с неподвижной частью поля.	4. Система переписывает границы неподвижной части поля и генерирует новую фигуру.
5. Игрок проигрывает.	6. Система завершает игру и выдает сообщение о проигрыше, предлагает начать новую игру (кнопка «New Game»).

Альтернатива 1:

5. Пользователь выходит из программы
6. Программа прекращает работу.

Диаграмма вариантов использования приведена на рисунке 1.

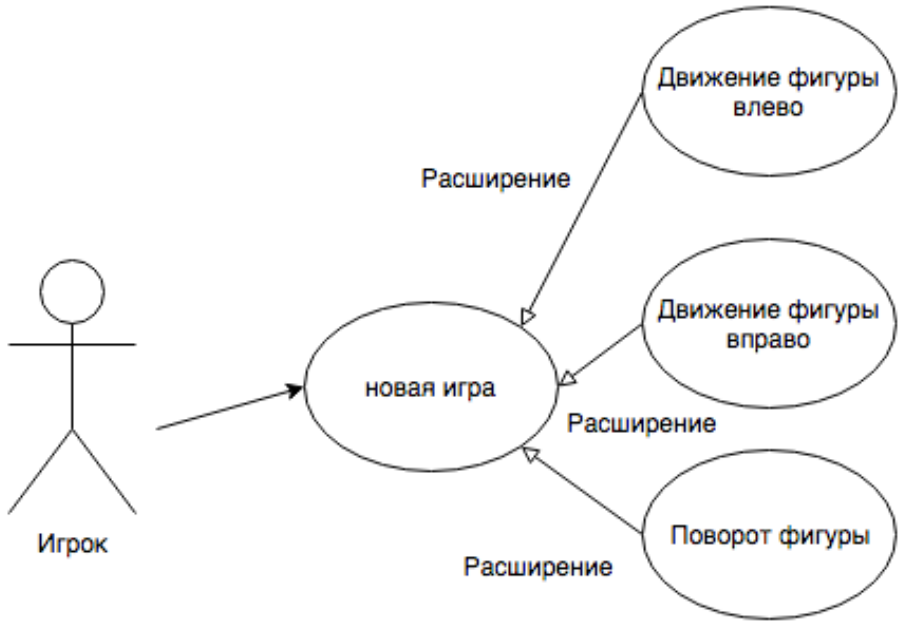

Рисунок 1 – Диаграмма вариантов использования

1.2.2 Проектирование структурной схемы меню

После анализа предметной области и вариантов использования можно составить структурную схему меню программы. Нужно выделить главное меню, как центр всех действий пользователя. Из главного меню пользователь может начать игру.

Структурная схема меню программного продукта представлена на Рисунке 2.

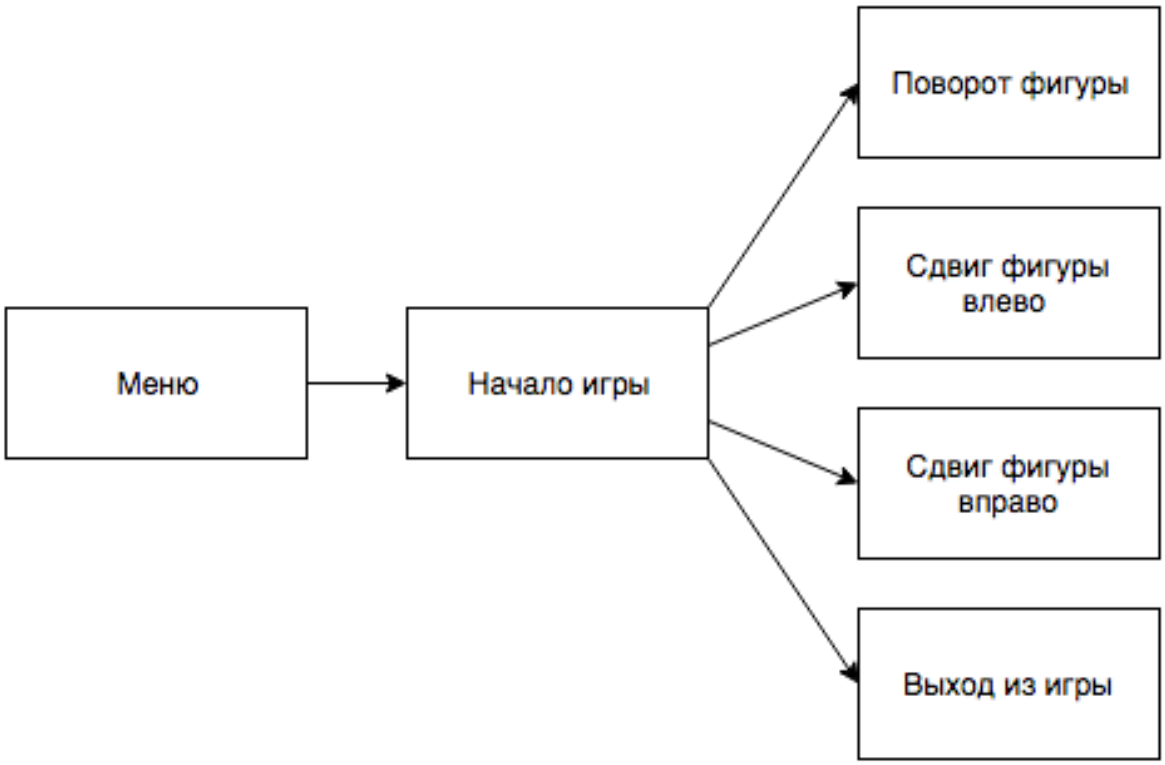

Рисунок 2 – Структурная схема меню

1.2.3 Разработка концептуальной диаграммы классов предметной области

Так как для проектирования разрабатываемого продукта используется объектный подход, актуально представить предметную область в виде взаимодействия объектов. Тем самым можно показать, каким образом нужно выстраивать иерархию классов для правильного установления отношений между ними.

Сущность «Поле» должна обрабатывать нажатия кнопок, опознавать клетку, на которой находится фигура, и в результате либо двигать фигуру,

либо перезаписывать границы неподвижной части поля. Сама сущность «Фигура» в результате этой операции будет менять свое состояние с подвижной на неподвижную.

На рисунке 3 приведена объектная декомпозиция программы.

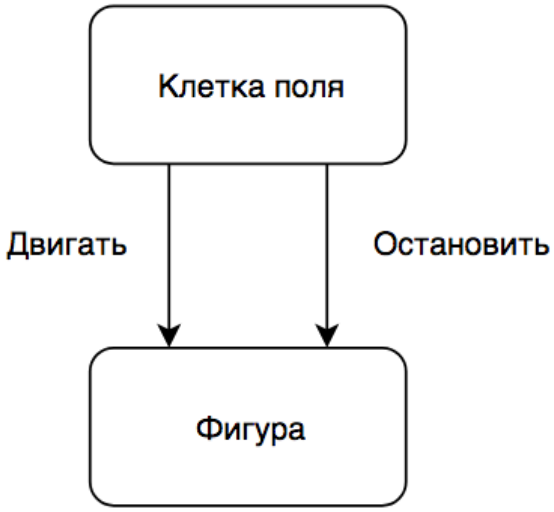

Рисунок 3 – Объектная декомпозиция

2. Проектирование структуры и компонентов программного продукта

2.1 Разработка интерфейса пользователя

2.1.1 Построение графа состояний интерфейса пользователя

Как следует из ТЗ, разрабатываемая программа предназначена для пользователей, имеющих небольшой навык работы с планшетным компьютером, и к интерфейсу предъявляются особенные требования. А так как программа является развлекательной, следовательно, интерфейс должен быть хорошо продуман.

На основе анализа ТЗ, можно выделить два основных состояния интерфейса пользователя:

1) Игра. В этом режиме пользователь может различным образом двигать фигуры.

2) Ожидание. В этом режиме пользователь может начать игру заново и выйти.

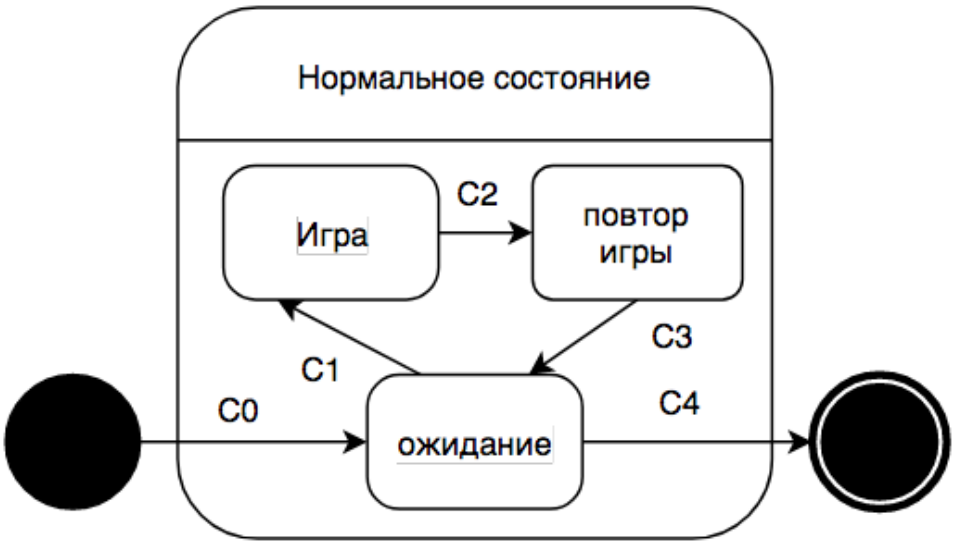

Рисунок 4 – Диаграмма состояний интерфейса.

На диаграмме представлены следующие события:

С0 – Активация главного окна программы;

С1 – Запуск игры;

С2 –Завершение игры;

С2 –Активация дополнительного окна программы;

С4 – Выход из игры.

2.1.2 Разработка форм интерфейса

Далее, для каждого из состояний интерфейса следует разработать формы ввода и вывода информации. Поэтому, следующий этап – проектирование форм интерфейса. В контексте решаемой задачи, приложение удобно представить в виде главного окна, задачей которого будет обеспечение пользователю доступа к основной функции приложения.

Для визуального представления интерфейса игры пользователя следует разработать текстуры. Каждое графическое изображение объекта на игровой сцене предпочтительно, чтобы было нарисовано с помощью закрашивания определённых областей поля. Этот вариант позволит сэкономить место и время, а также позволит избежать ошибок в будущем. Вид всех фигур приложения приведен на рисунке 5.

Главное окно должно содержать приветствие и кнопку для перехода к игровому полю. Оно нужно для лёгкого перехода с данной версии игровой программы на следующую, предполагается, что в следующих версиях будет добавлена кнопка «Settings». Изображение главного окна приложения представлено на рисунке 6. При нажатии на кнопку «New Game» открывается окно с полем для игры (см. Рисунок 7) с набором кнопок «Right», «Left», «Rotate» и «Exit», функционал которых описаны в приложении А технического задания. На рисунке 8 показан вид конечного окна, на которое можно перейти в случае проигрыша или нажатия на кнопку «Exit». Оно носит такой же функционал, как и главное окно. На этом окне так же есть кнопка «New Game», функционал которой описаны в приложении А технического задания.

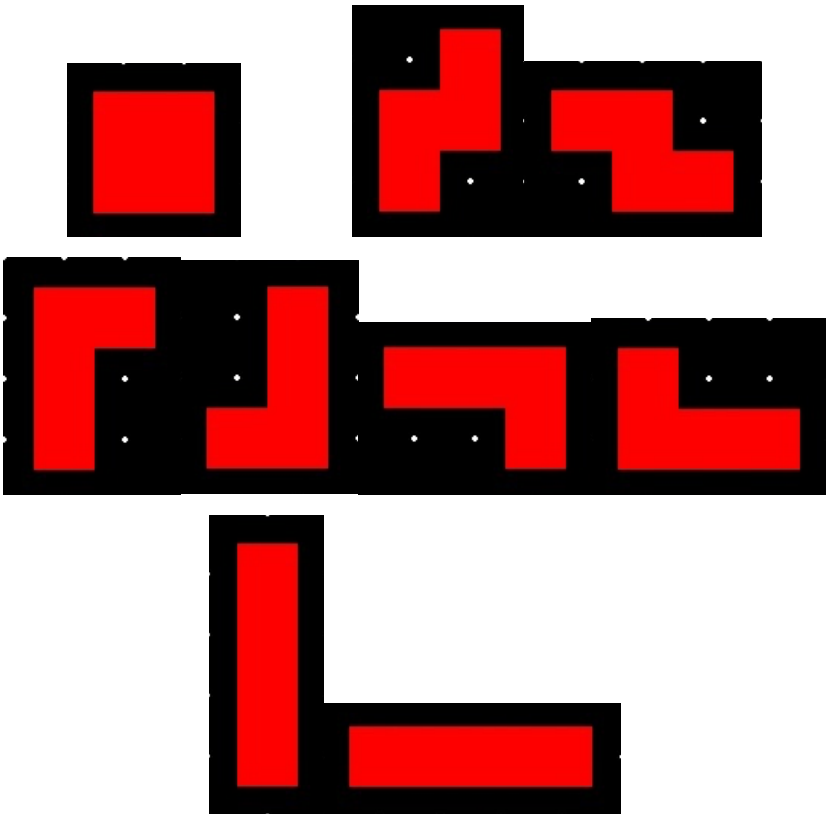

Рисунок 5 – Вид всех фигур и их состояний(разделить)

Рисунок 6 – Вид главного окна

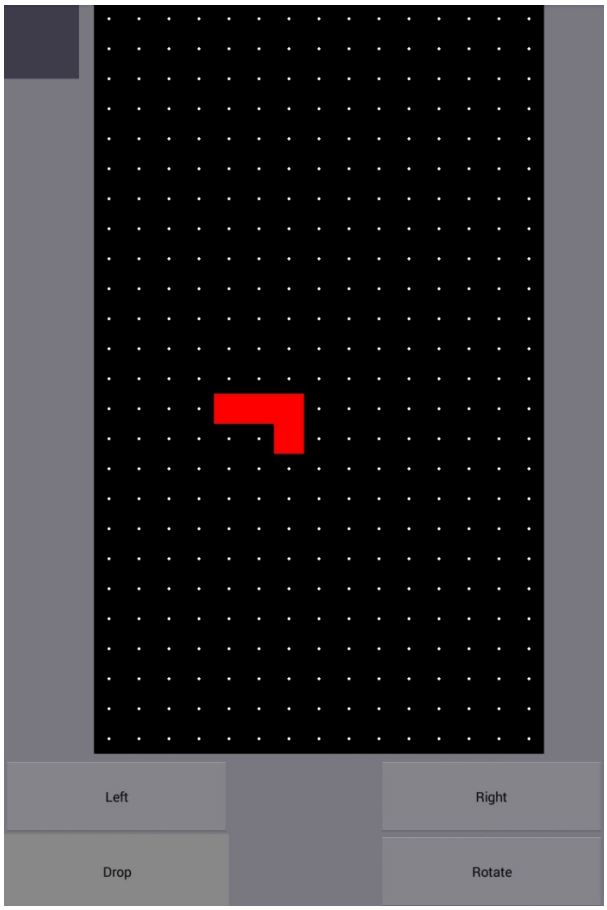

Рисунок 7 – Вид игрового окна.

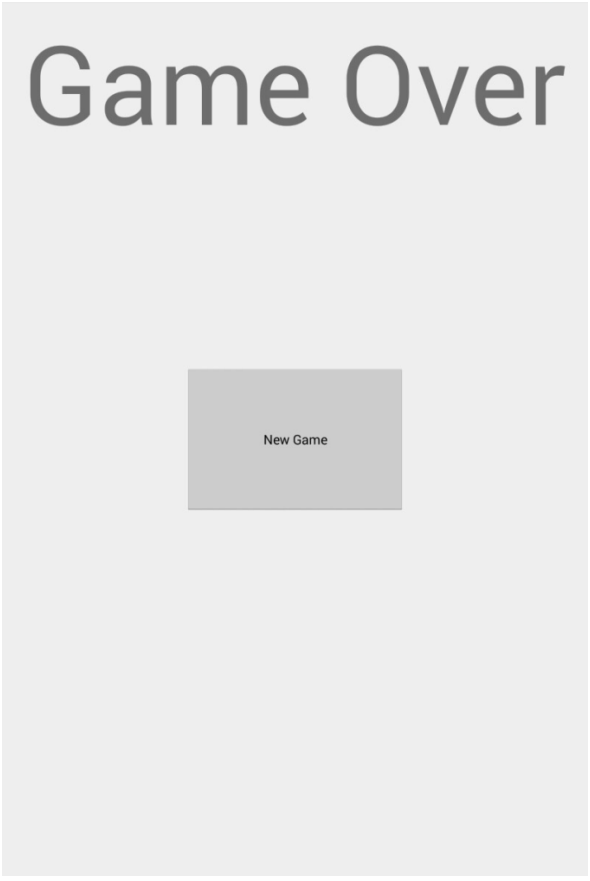

Рисунок 8 – Вид конечного окна.

2.2 Проектирование классов

На диаграмме классов показаны отношение классов между собой. Теперь стоит уточнить поля и методы данных классов.

Класс «Фигура» (ActiveFigure). У этого класса будет три поля - целочисленные переменные: id, pos, weight. Первый атрибут будет иметь значение номера фигуры. Второе поле будет хранить значение позиции фигуры, то есть как она повёрнута. Последнее поле – это вес фигуры, в последствии он будет прибавлятся к общему счёту.

В классе необходимо предусмотреть ряд методов, обеспечивающих различные манипуляции с этими фигурами. Для начала, кроме инициализации полей, нужно прописать сами фигуры и их виды поворотов. При помощи оператора выбора switch-case было решено реализовать эти действия. Так как всё поле поделено на квадраты, просто указываем четыре координаты фигуры. Далее эти координаты отправляются классу «FieldBlock», описываемый далее. Так же нужен метод changePos() для поворота фигур. В нём просто переключается с одного положения фигуры на следующее, координаты которого беруться из предыдущего метода.

Класс «Поле» (FieldBlock). У этого класса будет так же три воля – целочисленные переменные: x, y, state. Первые две переменные – это координаты, они понадобятся для отрисовки фигур и перезаписи границ неподвижной фигуры. Последняя используется для определения состояния поля в определённом месте. То есть, либо один цвет – это значит, что фигуры там нет, пустое поле, либо другой цвет – это значит, что в этом месте находится двигующаяся фигура, и либо третий цвет – это значит, что это неподвижная фигура. У этого класса будет только один метод, кроме инициализации полей, который будет отрисовывать поле и отвичать за цвет поля в каждом кадврате.

Класс «Форма» (tetrisDraw). Этот класс нужен для активизации в нужные моменты два предыдущих класса. У него будет десять полей: размеры стакана (int glassX, glassY), размер клетки поля (int blockWidth),

начальная скорость (int startDelay, постепенно уменьшается с 500 до 100, так как при скорости меньше ста фигуры будут «падать» слишком быстро, и это будет противоречить заявленому функционалу), координаты стакана на экране дивайса (int leftMargin, topMargin), массив клеток (FieldBlock[][] fields), активная фигура (ActiveFigure activeF), размеры экрана (int widthDisp; float halfWidthDisp).

В этом классе для откисовки игрового поля изначально нужно получить размер экрана, для этого следует предусмотреть отдельный метод, где так же будет вычисляться отступ слева от края экрана. Далее, нам нужен метод, который будет активизировать другие классы и формировать саму игру. В этом методе будет сначало активизироваться класс «Поле», который будет создавать поле для игры, затем активизироваться класс «Фигура», от которого приходят данные о новой фигуре, и в конце вызывается метод таймер, который и запускает игру. Метод таймер будет отрисовывать передвижение фигур с заданной частотой. После каждого изменения будет вызываться следующий метод с проверкой границ. В методе ifNotStuck_Move() будет проверяться фигура на столкновение с границами. Из этого метода будет вызываться последний метод – это проверка на заполнение всей строки, в случае, если строка заполнена, то при следующем изменении положения активной фигуры исчезнит и сама заполненая строчка, а не доконца заполненые строчки выше этой линии сместятся вниз, после чего будет перезапись границ неподвижной фигуры.

Детализирующая диаграмма классов уровня проектирования приведена на рисунке 10.

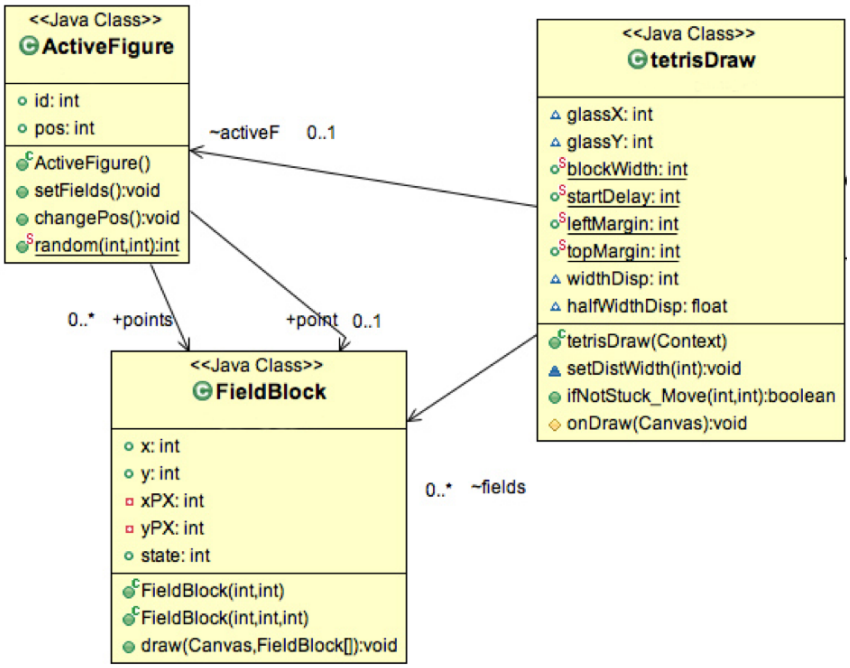

Рисунок 10 – Детализирующая диаграмма классов уровня проектирования

2.3 Разработка алгоритма игрового процесса

Для реализации игровой программы наиболее важным аспектом является реализация самого игрового процесса. Игровой процесс в программе было решено реализовать посредством обработчика события нажатия игроком кнопок на поле.

Сначала, при запуске игры визуализируется само поле, далее генерируется фигура. Фигура выбирается случайным образом из четырёх представленных. Движение фигур происходит благодаря смене положения этой фигуры, вызываемой в классе Таймер.

Далее, программа ожидает нажатия на одну из возможных кнопок. Когда игрок будет нажимать на кнопку «Rotate», программа должна вызвать метод смены положения фигуры. При нажатии кнопок «Left» и «Right» вызывается проверка возможности столкновения или с границами поля, или с границами неподвижной фигуры, затем, если смещение возможно, фигура смещается на одну клетку влево или вправо соответственно.

Схема алгоритма приведена на рисунке 11.

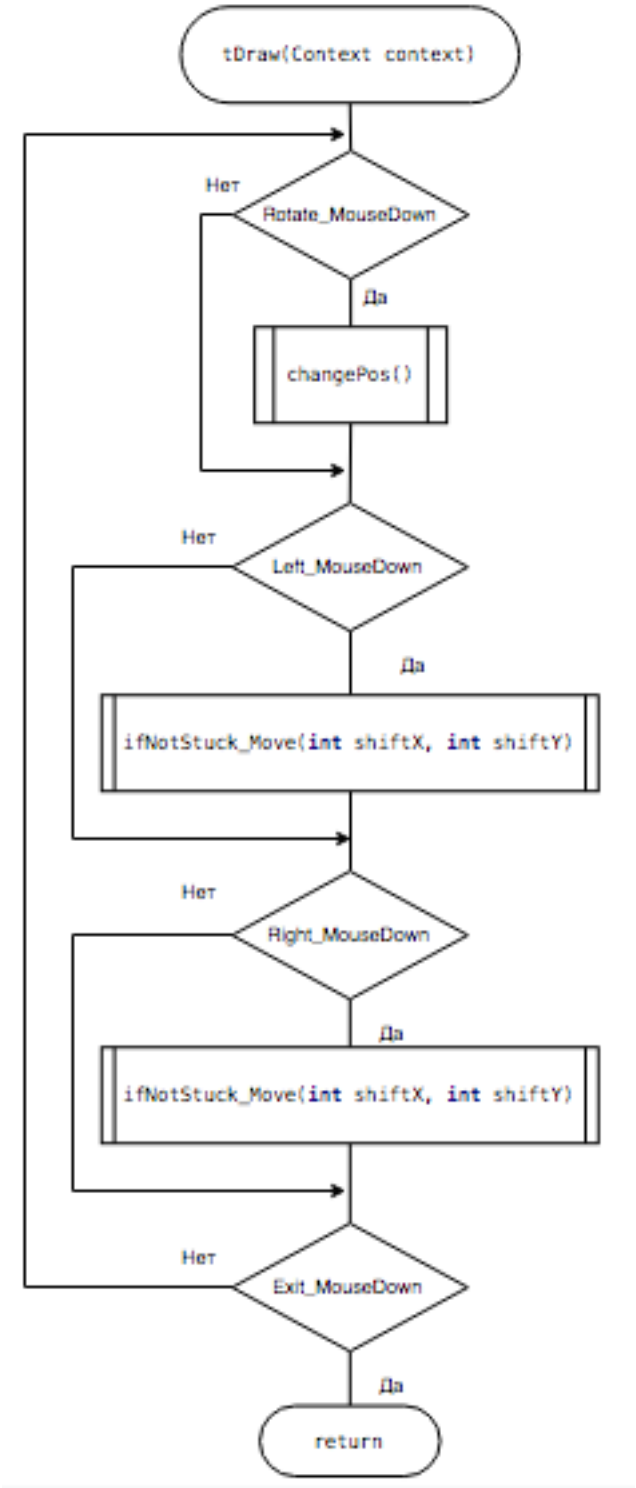

Рисунок 11 – Схема алгоритма игрового процесса

2.4 Реализация приложения

Согласно спроектированным и разработанным элементам, была реализована игровая программа. Код программы, согласно требованиям технического задания, написан на языке программирования Java. Написанный самодокументированный код снабжен комментариями. С

учетом всех задействованных в программе модулей была разработана диаграмма компоновки, которая приведена на рисунке 12.

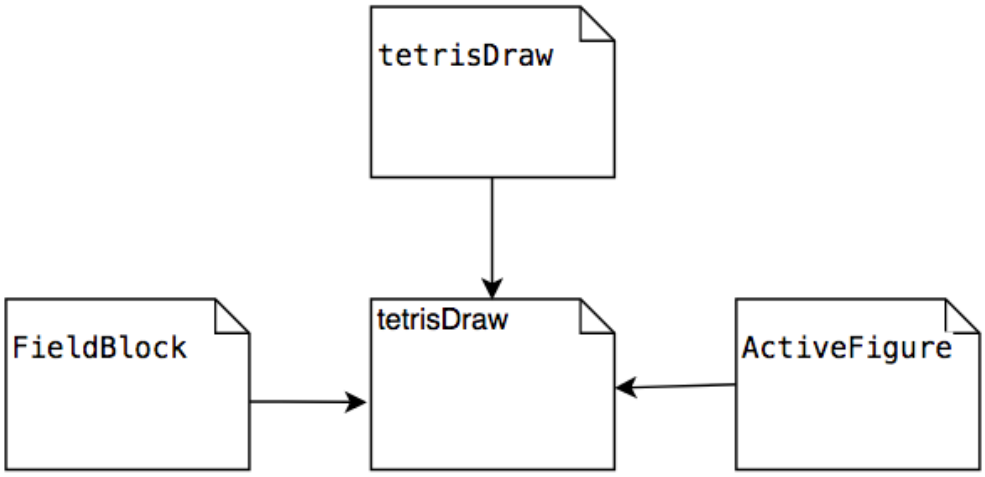

Рисунок 12 – Диаграмма компоновки.

3. Выбор стратегии тестирования и разработка тестов

Существует множество методов тестирования программных продуктов, которые необходимы для выявления ошибок, но никакое тестирование не может доказать отсутствие ошибок в программе. Можно лишь выбрать наиболее подходящий метод тестирования, исходя из анализа технического задания и объектной декомпозиции. Было решено остановиться на тестировании по принципу «черного ящика», поскольку вычислений в самой программе практически нет и классы четко структурированы. Это исключает такие распространенные ошибки, как выдача неверных результатов вычисления, изменения глобальных переменных и ряда других. Так, принцип «черного ящика» позволяет выявить ошибки в ходе разработки, проектировании и непосредственно отладке программы, а оценочное тестирование – мнение потенциальных пользователей по конечному продукту.

3.1 Тестирование по принципу «Черного ящика»

Для выяснения обстоятельств, в которых поведение программы не соответствует ожидаемому, следует провести тестирование по принципу «черного ящика». Для этого необходимо воспользоваться одним из методов такого тестирования – методом «эквивалентное разбиение». Основная идея метода состоит в том, чтобы исходные данные программы разбить на конечное число классов эквивалентности, так чтобы можно было предположить, что каждый тест, являющийся представителем некоторого класса, эквивалентен любому другому тесту этого класса. Иными словами, если тест какого-либо класса обнаруживает ошибку, то предполагается, что все другие тесты этого класса эквивалентности (классы эквивалентности представлены в таблице 2) тоже обнаружат эту ошибку и наоборот. Так же каждый тест должен включать по возможности максимальное количество различных входных условий, что позволяет минимизировать общее число необходимых тестов. Результаты тестирования приведены в таблице 3.

Таблица 2 – Классы эквивалентности

Входное условие	Правильные классы эквивалентности	Неправильные классы эквивалентности
Нажатие на кнопку «Left»	Сдвиг в лево во всех ситуациях кроме ситуаций, когда с лево препятствие	Сдвиг в лево, когда с лева находится препятствие
Нажатие на кнопку «Right»	Сдвиг в право во всех ситуациях кроме ситуаций, когда с право препятствие	Сдвиг в право, когда с права находится препятствие
Нажатие на кнопку «Rotate»	Поворот фигуры, если повёрнутая фигура находится на свободных клетках поля, в случае, если повёрнутая фигура находится на занятой клетке, то фигура не поворачивается	Поворот фигуры при находящемся рядом препятствии (только в том случае, если повёрнутая фигура частично или полностью совпадает с неподвижной фигурой или заходит за границу поля)

Таблица 3 – Результаты тестирования методом «черного ящика»

Номер теста	Событие	Ожидаемый результат	Полученный результат	Метод решения
1	Фигура «врезается» в нижнею грань поля.	Остановка фигуры и смена цвета с красного на серый.	Остановка фигуры и смена цвета с красного на серый.	Результат совпадает с ожиданием.
2	Фигура «врезается» в левую грань поля.	Ничего не происходит. Фигура продолжает движение вниз.	Ничего не происходит. Фигура продолжает движение вниз.	Результат совпадает с ожиданием.
3	Фигура «врезается» в правую грань поля.	Ничего не происходит. Фигура продолжает движение вниз.	Ничего не происходит. Фигура продолжает движение вниз.	Результат совпадает с ожиданием.

Таблица 3 – Продолжение

4	Фигура «врезается» в неподвижную фигуру снизу.	Фигура остановилась и поменяла свой цвет с красного на серый.	Фигура останавливается, но не меняет своего цвета с красного на серый.	В метод проверки на столкновение добавлена смена цвета при остановке фигуры.
5	Фигура «врезается» в неподвижную фигуру слева.	Ничего не происходит. Фигура продолжает двигаться вниз.	Фигура пропадает.	В метод проверки на столкновение с неподвижной фигурой добавлена проверка на левое столкновение с неподвижной фигурой.
6	Фигура «врезается» в неподвижную фигуру справа.	Ничего не происходит. Фигура продолжает двигаться вниз.	Фигура пропадает.	В метод проверки на столкновение с неподвижной фигурой добавлена проверка на левое столкновение с неподвижной фигурой.

Таблица 3 – Продолжение

7	Фигура поворачивается около неподвижной фигуры (есть возможность столкновения).	В случае возможности столкновения с неподвижной фигурой, программа не поворачивает фигура пока не будет достаточного места для этого манёвра.	Фигура поворачивается и пропадает во время столкновения	Во время вызова метода поворота фигуры добавлена проверка. Сначала проверяется возможность поворота (получаются новые координаты фигуры, их и проверяют на возможность столкновения), если поворот фигуры возможен, то фигура поворачивается.

После проведённого тестирования, были выявлены ошибки путём сравнения полученного результата с ожидаемым. Все ошибки были исправлены.

3.2 Оценочное тестирование

После завершения комплексного тестирования приступают к оценочному тестированию, целью которого является тестирование программы на соответствие основным требованиям. Эта стадия тестирования особенно важна для программных продуктов, предназначенных для продажи на рынке.

Для осуществления данного вида тестирования привлекается автор программы и некоторое количество потенциальных пользователей данного программного продукта. Тестирование проводилось 7 пользователями на разных стадиях разработки с оценкой программы по 4 графам по 10-бальной

шкале в порядке возрастания. Результаты оценочного тестирования представлены ниже в таблице 3.

Таблица 3 – Результаты оценочного тестирования.

Пользователь	Оценка интерфейса	Удобство интерфейса	Полнота и доступность основных функций	Понятность происходящего на экране
1	7	5	7	10
2	6	7	8	10
3	8	9	9	10
4	7	9	9	10
5	10	10	10	10
6	7	10	9	10
7	9	10	10	10

Тестирование программы первым пользователем показало, что интерфейс недостаточно удобен. Интерфейс был переделан в сторону косметических изменений, исходя из основной цели и категории разработанной программы. В последующих версиях были внесены изменения, позволяющие получать более высокие оценки от других пользователей.

Заключение

В результате выполнения курсовой работ был разработан программный продукт. В процессе работы были пройдены следующие стадии:

- Анализ предметной области
- Проектирование программного продукта
- Реализация программного продукта
- Тестирование готового программного продукта
- Разработка документации

Разработанная игровая программа предназначена для развития логического мышления, навыков решения проблем, навыков концентрации внимания, способности быстро переключаться между разными задачами, и используемой для получения нового опыта от игры «Тетрис».

Разработка велась с использованием «Спиральной модели» жизненного цикла – сначала была поставлена задача создания обыкновенного тетриса с основными параметрами и проведен ее анализ, затем были проведены проектирование и реализация первой версии программного продукта и цикл повторялся. Во второй цикл реализации программного продукта поставленной задачей были дополнительные параметры игры: поворот фигуры, выход из игры и два вида меню. Также программный продукт имеет потенциал на будущее обновление версий.

Приложение успешно прошло комплексное и оценочное тестирование.

В дальнейших версиях программного продукта целесообразно расширить набор режимов игры и добавить возможность просмотра счета. А также создать форму с возможностью просмотра рекордов игры.

Список использованной литературы

- Иванова Г.С. Технология программирования: Учебник для вузов. – М.: Изд-во МГТУ им. Н.Э.Баумана, 2006. – 336 с.
- Иванова Г.С., Ничушкина Т.Н., Пугачев Е.К. Методические указания по выполнению курсовой работы по курсу «Технология программирования»: Методическое пособие. – М. 2003.
- Роберт Лигуори, Патрисия Лигуори Java 8. Карманный справочник.: Пер. с англ. – М.: ООО «И.Д. Вильямс», 2016. – 256 с.
- Патрик Нимейер, Дэниэл Леук Программирование на Java. Исчерпывающее руководство для профессионалов.: пер. с англ. М. А. Райтмана. - 4-е изд. - Москва : Эксмо, 2014. - 1215 с.
- Аллен Б. Доуни Алгоритмы и структуры данных. Извлечение информации на языке Java.: пер. с англ. К. Синица. - Санкт-Петербург: Питер, 2018. - 237 с.
- Tetris [электронный ресурс]. Режим доступа: https://en.wikipedia.org/wiki/Tetris (дата обращения: 13.11.2019)

Приложение А

Техническое задание

на 5 листах

Приложение Б

Руководство пользователя

на 6 листах

Приложение В

Фрагмент кода программы

на 2 листах