ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»

Институт компьютерных наук и технологий

Высшая школа программной инженерии

ПОЛИТЕХ
Санкт-Петербургский
политехнический университет
Петра Великого

КУРСОВАЯ РАБОТА

Разработка игры тетрис на языке JAVA

по дисциплине «Объектно-ориентированное программирование»

Студенты гр. 3530202/80001 *А. Горячевская, И. Дац, Г. Кашин*

Руководитель

Ст. преподаватель С.М.Бойко

Санкт-Петербург

2019г

Содержание

Введение

В данной курсовой работе реализуется игра «Тетрис». При реализации программы используется объектно-ориентированный подход, важной частью которого является возможность формирования такой структуры программы, что добавление новых компонентов при ее дальнейшем развитии не будет влиять на существующие компоненты, или такое влияние будет сведено к минимуму.

Для написания кода игры используется язык Java, что способствует получению надежного программного продукта.

Цель работы

Целью данной работы является разработка игры "Тетрис". Она относится к логическим играм, которые помогают развить реакцию, тренируют логику. При длительной работе за компьютером возникает усталость и необходимость отвлечься на 5-10 минут. Для этой цели вполне подойдет именно «Тетрис». В стандартной поставке операционных систем данной игры нет, поэтому возникла необходимость в ее создании.

Задачами курсовой работы является изучение объектно-ориентированного языка программирования Java, рассмотрение игры Тетрис, и создание программного кода на языке Java.

Требования

К играм такого типа предъявляются следующие требования:

- использование простых средств управления;
- удобный графический интерфейс;
- постепенное усложнение игры при наборе определенного количества очков;
- корректное выполнение программы.

Смысл игры заключается в стремлении плотно занять игровое поле падающими геометрическими фигурами, изменяя их ориентацию в пространстве, добиваясь отсутствия пробелов в каждой строке.

Фигуры не должны выходить за пределы игрового поля, они перемещаться и переворачиваться. Для придания большей привлекательности внешнему виду игрового поля при написании игры нужно использовать яркую графику. Интерфейс программы, разрабатываемой должен быть удобен и понятен пользователю.

Программа должна реагировать на нажатие клавиш клавиатуры, выводя требуемое изображение на экран.

Введение в язык программирования Java

1. История развития языка программирования Java

Изначально появилась на свет как язык для создания небольших приложений для Интернета (апплетов), но со временем развилась как универсальная платформа для создания программного обеспечения, которое работает буквально везде - от мобильных устройств и смарт-карт до мощных серверов.

Данная курсовая начинается с изложения истории появления и развития Java. Такие знания позволят лучше понять особенности платформы и спектр существующих продуктов и технологий. Также создание Java является интересным примером истории одного из самых популярных и успешных проектов в компьютерном мире. Затем излагаются основные концепции ООП, необходимые для освоения объектно-ориентированного языка программирования Java. Ключевые понятия и конструкции языка описываются доступным языком, но, тем не менее, на достаточно глубоком уровне. Детально рассмотрены особенности лексики, системы типов данных, объектной модели. Уделяется особое внимание модификаторам доступа, соглашениям по именованию, преобразованию типов, работе с массивами, обработке ошибок (исключительных ситуаций). Курсовая завершается рассмотрением базовых библиотек Java, предоставляющих всю необходимую функциональность для создания самых разных приложений - коллекции объектов, работа с файлами, сетью, создание GUI приложений, построение многопоточной архитектуры и многое другое. Описание сетевой библиотеки предваряется изложением основ сетевых протоколов и технологий.

Казалось бы, на сегодняшний день изобретены уже все языки программирования, какие только можно придумать. Но нет - появился еще один, с названием Java. Этот язык сумел получить весьма заметную известность за последние несколько лет, так как он ориентирован на самую популярную компьютерную среду - сеть Internet и серверы Web.

Персональные компьютеры сделали информационные технологии частью массовой культуры. И тем не менее, уже довольно длительная история развития персональных компьютеров не знала ничего, подобного феномену Java. Что изменилось в мире в последние годы, почему этот феномен стал возможен?

Изменился Интернет. Он стал доступен миллионам людей, далеких от технических проблем. Число пользователей Интернет по порядку величины

уже не отличается от числа пользователей персональных компьютеров и продолжает взрывообразно расти. Одновременно Интернет обеспечил такую скорость распространения новинок информационных технологий, которую не могли и никогда не смогут дать традиционные каналы сбыта. Время спрессовалось. В Интернет, опоздав буквально на день, компьютерная компания, даже крупная, рискует серьезно ослабить свои позиции сразу во всем мире.

Язык Java произошел от языка программирования Oak (а не от C++, как думают многие). Oak был приспособлен для работы в Internet и затем переименован в Java.

Синтаксис Java близок к синтаксису языка C++. Унаследовав самое лучшее от языка программирования C++, язык Java при этом избавился от некоторых недостатков C++, в результате чего на нем стало проще программировать. В этом языке нет, например, указателей, которые сложны в использовании и потенциально могут послужить причиной доступа программы к не принадлежащей ей области памяти. Нет множественного наследования и шаблонов, хотя функциональные возможности языка Java от этого не пострадали.

Огромное преимущество Java заключается в том, что на этом языке можно создавать приложения, способные работать на различных платформах. К сети Internet подключены компьютеры самых разных типов - Pentium PC, Macintosh, рабочие станции Sun и так далее. Даже в рамках компьютеров, созданных на базе процессоров Intel, существует несколько платформ, например, MicrosoftWindows версии 3. 1, Windows 95, Windows NT, OS/2, Solaris, различные разновидности операционной системы UNIX с графической оболочкой XWindows. Между тем, создавая сервер Web в сети Internet, хотелось бы, чтобы им могло пользоваться как можно большее число людей. В этом случае выручают приложения Java, предназначенные для работы на различных платформах и не зависящие от конкретного типа процессора и операционной системы.

Программы, составленные на языке программирования Java, можно разделить по своему назначению на две большие группы.

К первой группе относятся приложения Java, предназначенные для автономной работы под управлением специальной интерпретирующей машины Java. Реализации этой машины созданы для всех основных компьютерных платформ. Вторая группа -- это так называемыеаплеты (applets). Аплеты представляют собой разновидность приложений Java,

которые интерпретируются виртуальной машиной Java, встроенной практически во все современные браузеры.

Приложения, относящиеся к первой группе (будем называть их просто приложениями Java), -- это обычные автономные программы. Так как они не содержат машинного кода и работают под управлением специального интерпретатора, их производительность заметно ниже, чем у обычных программ, составленных, например, на языке программирования C++. Однако не следует забывать, что программы Java без перетрансляции способны работать на любой платформе, что само по себе имеет большое значение в плане разработок для Internet.АплетыJava встраиваются в документы HTML, хранящиеся на сервере Web. С помощью аплетов можно сделать страницы сервера Web динамичными и интерактивными. Аплеты позволяют выполнять сложную локальную обработку данных, полученных от сервера Web или введенных пользователем с клавиатуры. Из соображений безопасности аплеты (в отличие от обычных приложений Java) не имеют никакого доступа к файловой системе локального компьютера. Все данные для обработки они могут получить только от сервера Web. Более сложную обработку данных можно выполнять, организовав взаимодействие между аплетами и расширениями сервера Web приложениями CGI и ISAPI.

Для повышения производительности приложений Java в современных браузерах используется компиляция "на лету"-- Just-In-Timecompilation (JIT). При первой загрузке аплета его код транслируется в обычную исполнимую программу, которая сохраняется на диске и запускается. В результате общая скорость выполнения аплетаJava увеличивается в несколько раз.

Язык Java является объектно-ориентированным и поставляется с достаточно объемной библиотекой классов. Так же как и библиотеки классов систем разработки приложений на языке C++, библиотеки классов Java значительно упрощают разработку приложений, представляя в распоряжение программиста мощные средства решения распространенных задач. Поэтому программист может больше внимания уделить решению прикладных задач, а не таких, как, например, организация динамических массивов, взаимодействие с операционной системой или реализация элементов пользовательского интерфейса.

Информационные перегрузки - характерная черта нашего времени. Созданы мощные механизмы, обеспечивающие производство огромного количества информации. Существенно меньше сделано для облегчения ее получения и усвоения. Типичной является ситуация, когда инициатива принадлежит поставщику, а не потребителю информации. Поставщик по

определенному поводу создает информацию и направляет ее всем, кто, по его мнению, в ней нуждается.

Так работают средства массовой информации, издательства, рекламные агентства. Как работает электронная почта. В большинстве случаев потребителю эта информация, может быть, и нужна, но не в данный момент, не сейчас. Потребитель вынужден архивировать полученную информацию. При этом в лучшем случае велика вероятность, что к моменту, когда информация действительно понадобится, она потеряет актуальность. Обычно же у потребителя просто накапливаются горы мусора, в котором отыскать нечто нужное почти невозможно.

Чтобы информация была актуальной для потребителя, она должна доставляться к нему по запросу -- в точности тогда, когда в ней возникла необходимость. Кроме того, поставщик должен сохранять возможность управления информацией, он должен не только создавать ее, но и вовремя обновлять и уничтожать. Централизованные компьютерные системы, доминировавшие еще 10 лет назад, позволяли пользователям сравнительно легко находить информацию в оперативном режиме, однако они затрудняли управление информацией, поскольку ее источники, как правило, разнородны и территориально разнесены.

Еще один важный недостаток централизованных систем -- их сложность и дороговизна. Сети персональных компьютеров существенно дешевле централизованных систем, они оставляют за поставщиком необходимую свободу управления информацией, однако потребителям приходится искать необходимые данные на множестве машин, среди большого числа приложений с различными интерфейсами. Рядовому пользователю работать в такой разнородной прикладной среде крайне неудобно.

Способ разрешения указанных проблем, к которому прибегают ведущие компании, состоит в построении информационной структуры организации по образу и подобию Интернет, с Web-сервисом в качестве концептуальной основы.

Возможность хранения данных различных типов (текст, графика, аудио, видео) в сочетании с механизмами связывания информации, расположенной в разных узлах компьютерной сети, позволяют рассредоточивать информацию в соответствии с естественным порядком ее создания и потребления, осуществлять единообразный доступ, отправляясь от небольшого числа известных "корней". Тем самым поставщик может

эффективно готовить и контролировать информацию, а потребитель в состоянии без труда найти необходимые данные именно тогда, когда они стали нужны.

Средства Web, помимо связывания распределенных данных, осуществляют еще одну очень важную функцию. Они позволяют рассматривать информацию с нужной степенью детализации, что существенно упрощает анализ больших объемов данных. Можно быстро отобрать самое интересное, а затем изучить выбранный материал во всех подробностях.

Таким образом, Web-серверы и Web-навигаторы могут и должны использоваться не только в "мировом масштабе". Web -- это инфраструктурный сервис, необходимый каждой организации со сколько-нибудь заметными информационными потоками. В то же время, Web-сервису присущи и определенные недостатки, вытекающие из отсутствия объектной ориентации и из природы HTTP-протокола. Во-первых, клиент по существу лишен средств управления внешним представлением объектов на просматриваемой WWW-странице.

Во-вторых, Web-страницы статичны. При использовании протокола HTTP, на клиентскую систему передаются только пассивные данные, но не методы объектов. Из общих соображений очевидна ограниченность подобного подхода. Данный недостаток, разумеется, связан с первым. Объект сам должен знать, как себя показывать - точнее говоря, он должен это выяснить, проанализировав клиентское окружение.

В-третьих, Web-сервис обладает весьма ограниченными интерактивными возможностями, которые сводятся к заполнению пользователем чисто текстовых форм с последующей отправкой на сервер. Сервер анализирует полученные данные, после чего формирует и возвращает клиенту новую WWW-страницу, которая нередко вновь оказывается формой. Такой стиль общения не всегда устраивает пользователей. Java-технология позволяет устранить все отмеченные недостатки. Как именно будет ясно из последующего изложения. В результате Web-сервис, и без того имевший огромную популярность, получил как бы новый импульс. Этот экспресс понесся вперед с удвоенной скоростью, увлекая за собой и Java. JAVA, JOE, NEO.

2. Язык Java - это объектно-ориентированный язык программирования

В узком смысле слова Java -- это объектно-ориентированный язык, напоминающий C++, но более простой для освоения и использования. В более широком смысле Java -- это целая технология программирования, изначально рассчитанная на интеграцию с Web-сервисом, то есть на использование в сетевой среде, Поскольку Web-навигаторы существуют практически для всех аппаратно-программных платформ, Java-среда должна быть как можно более мобильной, в идеале полностью независимой от платформы.

С целью решения перечисленных проблем были приняты, помимо интеграции с Web-навигатором, два других важнейших постулата:

Была специфицирована виртуальная Java-машина, на которой должны выполняться (интерпретироваться) Java-программы. Определены ее архитектура, представление элементов данных и система команд. Исходные Java-тексты транслируются в коды этой машины. Тем самым, при появлении новой аппаратно-программной платформы в портировании будет нуждаться только Java-машина; все программы, написанные на Java, пойдут без изменений.

Определено, что при редактировании внешних связей Java-программы и при работе Web-навигатора прозрачным для пользователя образом может осуществляться поиск необходимых объектов не только на локальной машине, но и на других компьютерах, доступных по сети (в частности, на WWW-сервере). Найденные объекты загружаются, а их методы выполняются затем на машине пользователя.

Несомненно, между двумя сформулированными положениями существует тесная связь. В компилируемой среде трудно как дистанцироваться от аппаратных особенностей компьютера, так и реализовать прозрачную динамическую загрузку по сети. С другой стороны, прием объектов извне требует повышенной осторожности при работе с ними, а, значит, и со всеми Java-программами. Принимать необходимые меры безопасности проще всего в интерпретируемой среде. Вообще, мобильность, динамизм и безопасность - спутники интерпретатора, а не компилятора. Принятые решения сделали Java-среду идеальным средством разработки клиентских компонентов Web-систем. Особо отметим прозрачную для пользователя динамическую загрузку объектов по сети. Из этого вытекает

такое важнейшее достоинство, как нулевая стоимость администрирования клиентских систем, написанных на Java. Достаточно обновить версию объекта на сервере, после чего клиент автоматически получит именно ее, а не старый вариант. Без этого реальная работа с развитой сетевой инфраструктурой практически невозможна. С другой стороны, при наличии динамической загрузки действительно возможно появление устройств класса Java-терминалов, изначально содержащих только WWW-навигатор, а все остальное (и программы, и данные) получающих по сети.

Здесь уместно отметить замечательную точность в выборе основных посылок проекта Java. Из минимума предположений вытекает максимум новых возможностей при сохранении практичности реализации.

В то же время, интеграция с WWW-навигатором и интерпретируемая природа Java-среды ставят вполне определенные рамки для реального использования Java-программ (хотя, конечно же, язык Java не менее универсален, чем, скажем, C++). Например, известно, что интерпретация, по сравнению с прямым выполнением, на 1-2 порядка медленнее. Применение компиляции "на лету" и специализированных Java-процессоров, несомненно, улучшит ситуацию, но пока использование Java на серверной стороне представляется проблематичным.

Хотя технология Интернет, основанная на использовании Web-сервиса в качестве информационной основы организации, является огромным шагом вперед, существуют и другие сервисы, как унаследованные, так и современные (например, реляционные СУБД), которые обязательно должны входить в состав корпоративной системы.

Если вся связь между клиентами и упомянутыми серверами будет осуществляться через сервер WWW, последний станет узким местом, а решения Интернет рискуют лишиться такого важнейшего достоинства, как масштабируемость. Значит, необходима прямая связь между клиентскими системами, написанными на языке Java, и произвольными сервисами. Как реализовать такую связь?

В общем виде ответ очевиден - нужны средства для полноценной интеграции Java в распределенную объектную среду. На серверной стороне компания SunMicrosystems имеет соответствующую технологию - NEO (NEtworkedObjects, сетевые объекты). Технология NEO удовлетворяет спецификациям OMG (ObjectManagementGroup), являющимся промышленным стандартом. При реализации корпоративных информационных систем с использованием NEO наиболее естественным

представляется использование трехуровневой архитектуры с серверами приложений, построенными на объектных принципах, на втором уровне и с базовыми и унаследованными серверами натретьем уровне.

К сожалению, столь общий ответ никак не помогает осуществлять прямую связь между Java-клиентом и NEO-сервером. Конечно, можно воспользоваться стандартными средствами программирования в сетевой среде (а Java допускает использование библиотек, написанных на C/C++, равно как и вставку машинных кодов), но если бы это было единственной возможностью, Java рисковала остаться на уровне "оживлялок".

В конце марта компания SunSoft объявила о появлении нового продукта с именем Joe, как раз и предназначенного для существенного облегчения встраивания Java-клиентов в информационные системы Интернет, построенные в трехуровневой архитектуре с использованием среды NEO.

Таким образом, сложилась полная и изумительно красивая картина организации современных Интернет-систем.

3. Java - язык и технология

Язык программирования Java является мобильным. Это нужно понимать в том смысле, что имеется принципиальная возможность переноса программ Java на различные платформы.

Однако следует отметить, что создание приложений, действительно работающих на разных платформах -- непростая задача. К сожалению, дело не ограничивается необходимостью перекомпиляции исходного текста программы для работы в другой среде. Много проблем возникает с несовместимостью программных интерфейсов различных операционных систем и графических оболочек, реализующих пользовательский интерфейс.

Вот хотя бы проблемы, связанные с переносом 16-разрядных приложений Windows в 32-разрядную среду Windows 95 и Windows NT. Даже если тщательно следовать всем рекомендациям, разрабатывая приложения так, чтобы они могли работать в будущих версиях Windows, едва ли удастся просто перекомпилировать исходные тексты, не изменив в них ни строчки. Ситуация еще больше ухудшается, если нужно, например, перенести исходные тексты приложения Windows в среду операционной системы OS/2 или в оболочку X-Windows операционной системы UNIX. А ведь есть еще другие компьютеры и рабочие станции!

Как нетрудно заметить, даже если стандартизовать язык программирования для всех платформ, проблемы совместимости с программным интерфейсом операционной системы значительно усложняют перенос программ на различные платформы. И, конечно, нужно, чтобы загрузочный модуль одной и той же программы мог работать без изменений в среде различных операционных систем и на различных платформах. Если программа подготовлена для процессора Intel, она ни за что не согласится работать на процессоре Alpha или каком-либо другом.

В результате создавая приложение, способное работать на различных платформах, необходимо фактически делать несколько различных приложений и сопровождать их по отдельности.

Покажем, как приложение, изначально разработанное для Windows NT, переносится на платформу AppleMacintosh. Вначале программист готовит исходные тексты приложения для платформы Windows NT и отлаживает их там. Для получения загрузочного модуля исходные тексты компилируются и редактируются. Полученный в результате загрузочный модуль может

работать на процессоре фирмы Intel в среде операционной системы Windows NT.

Для того чтобы перенести приложение в среду операционной системы компьютера Macintosh, программист вносит необходимые изменения в исходные тексты приложения. Эти изменения необходимы из-за различий в программном интерфейсе операционной системы Windows NT и операционной системы, установленной в Macintosh. Далее эти исходные тексты транслируются и редактируются, в результате чего получается загрузочный модуль, способный работать в среде Macintosh, но не способный работать в среде Windows NT.

Программа на языке Java компилируется в двоичный модуль, состоящий из команд виртуального процессора Java. Такой модуль содержит байт-код, предназначенный для выполнения Java-интерпретатором. На настоящий момент уже созданы первые модели физического процессора, способного выполнять этот байт-код, однако интерпретаторы Java имеются на всех основных компьютерных платформах. Разумеется, на каждой платформе используется свой интерпретатор, или, точнее говоря, свой виртуальный процессор Java.

Если приложение Java (или апплет) должно работать на нескольких платформах, нет необходимости компилировать его исходные тексты несколько раз. Можно откомпилировать и отладить приложение Java на одной, наиболее удобной для вас платформе. В результате вы получите байт-код, пригодный для любой платформы, где есть виртуальный процессор Java. Таким образом, приложение Java компилируется и отлаживается только один раз, что уже значительно лучше. Остается, правда, вопрос - как быть с программным интерфейсом операционной системы, который отличается для разных платформ?

Приложение Java не обращается напрямую к интерфейсу операционной системы. Вместо этого оно пользуется готовыми стандартными библиотеками классов, содержащими все необходимое для организации пользовательского интерфейса, обращения к файлам, для работы в сети и так далее.

Внутренняя реализация библиотек классов зависит от платформы. Однако все загрузочные модули, реализующие возможности этих библиотек, поставляются в готовом виде вместе с виртуальной машиной Java, поэтому программисту не нужно об этом заботиться. Для операционной системы

Windows, например, поставляются библиотеки динамической загрузки DLL, внутри которых запрятана вся функциональность стандартных классов Java.

Абстрагируясь от аппаратуры на уровне библиотек классов, программисты могут больше не заботиться о различиях в реализации программного интерфейса конкретных операционных систем. Это позволяет создавать по-настоящему мобильные приложения, не требующие при переносе на различные платформы перетрансляции и изменения исходного текста.

Еще одна проблема, возникающая при переносе программ, составленных на языке программирования С, заключается в том, что размер области памяти, занимаемой переменными стандартных типов, различный на разных платформах. Например, в среде операционной системы Windows версии 3. 1 переменная типа int в программе, составленной на С, занимает 16 бит. В среде Windows NT этот размер составляет 32 бита.

Очевидно, что трудно составлять программу, не зная точно, сколько имеется бит в слове или в байте. При переносе программ на платформы с иной разрядностью могут возникать ошибки, которые трудно обнаружить.

В языке Java все базовые типы данных имеют фиксированную разрядность, которая не зависит от платформы. Поэтому программисты всегда знают размеры переменных в своей программе.

Формулировка задания

1. Правила игры тетрис

Случайные фигурки тетрамино падают сверху в прямоугольный стакан шириной 10 и высотой 20 клеток. В полёте игрок может поворачивать фигурку и двигать её по горизонтали. Также можно «сбрасывать» фигурку, то есть ускорять её падение, когда уже решено, куда фигурка должна упасть. Фигурка летит, пока не наткнётся на другую фигурку либо на дно стакана. Если при этом заполнился горизонтальный ряд из 10 клеток, он пропадает и всё, что выше его, опускается на 1 клетку. Темп игры постепенно увеличивается. Название игры происходит от количества клеток, из которых состоит каждая фигура. Игра заканчивается, когда новая фигурка не может поместиться в стакан. Игрок получает очки за каждую фигурку, поэтому его задача - заполнять ряды, не заполняя сам стакан как можно дольше, чтобы таким образом получить как можно больше очков.

2. Начисление очков в игре тетрис

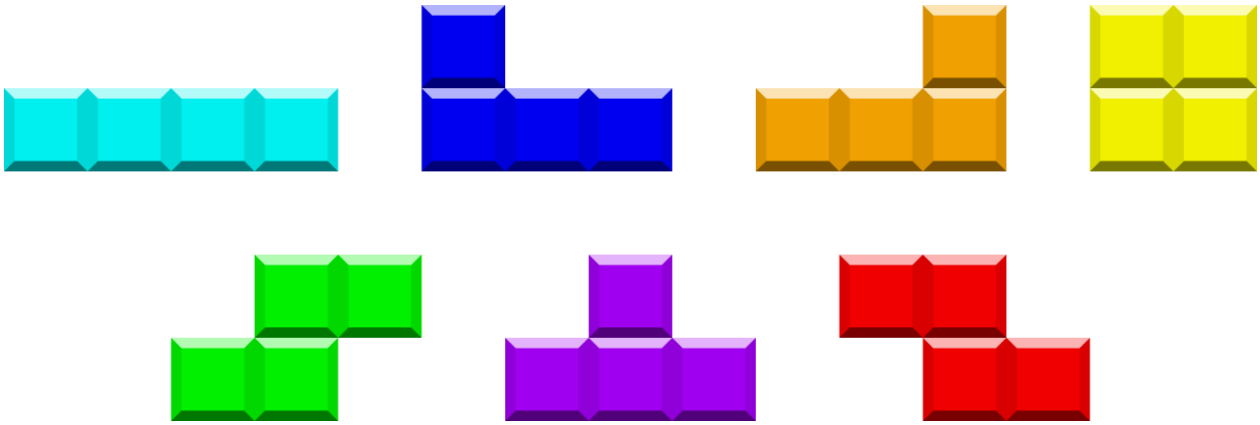
Начисление очков в разных версиях «Тетриса» довольно разнообразное. Очки могут начисляться за убранные линии, за сброшенные фигурки, за переход на новую скорость и тому подобное.

При начислении очков за линии количество очков обычно зависит от того, сколько линий убрано за один раз., популярных в СНГ в 1990-х годах, начисление очков обычно было таким: 1 линия - 100 очков, 2 линии - 300 очков, 3 линии - 700 очков, 4 линии (то есть, сделать Тетрис) - 1500 очков. То есть, чем больше линий убирается за один раз, тем больше отношение количества очков к количеству линий. Любопытно, что тетрисом во многих версиях игры также называется действие, после которого исчезает сразу 4 линии. Это можно сделать только одним способом - сбросить «палку» (фигурку, в которой все клетки расположены на одной линии) в «шахту» ширины 1 и глубины как минимум 4.

При начислении очков за сброшенные фигурки могут учитываться высота, на которой остановилась фигурка (например, чем ниже, тем лучше), расстояние, которое пролетела фигурка после «сбрасывания» (ускорения падения). Хотя обычно приоритетом являются линии, а за фигурки начисляется относительно небольшое количество очков.

3. Фигуры

Фигуры представляют собой тетрамино. Каждая фигура имеет свой размер, цвет и обозначение: голубой – I, синий – J, оранжевый – L, желтый – O, зеленый – S, фиолетовый – T, красный – Z.



Появляются фигуры случайным образом так, чтобы на поле в один момент была только одна летящая фигура.

4. Уровни

Переход на новый уровень осуществляется при достижении игроком 3000 очков, после чего скорость движения фигур увеличивается.

5. Управление

Реализация игры подразумевает наличие управляющих клавиш:

«Вниз» - ускорение движения фигуры вниз.

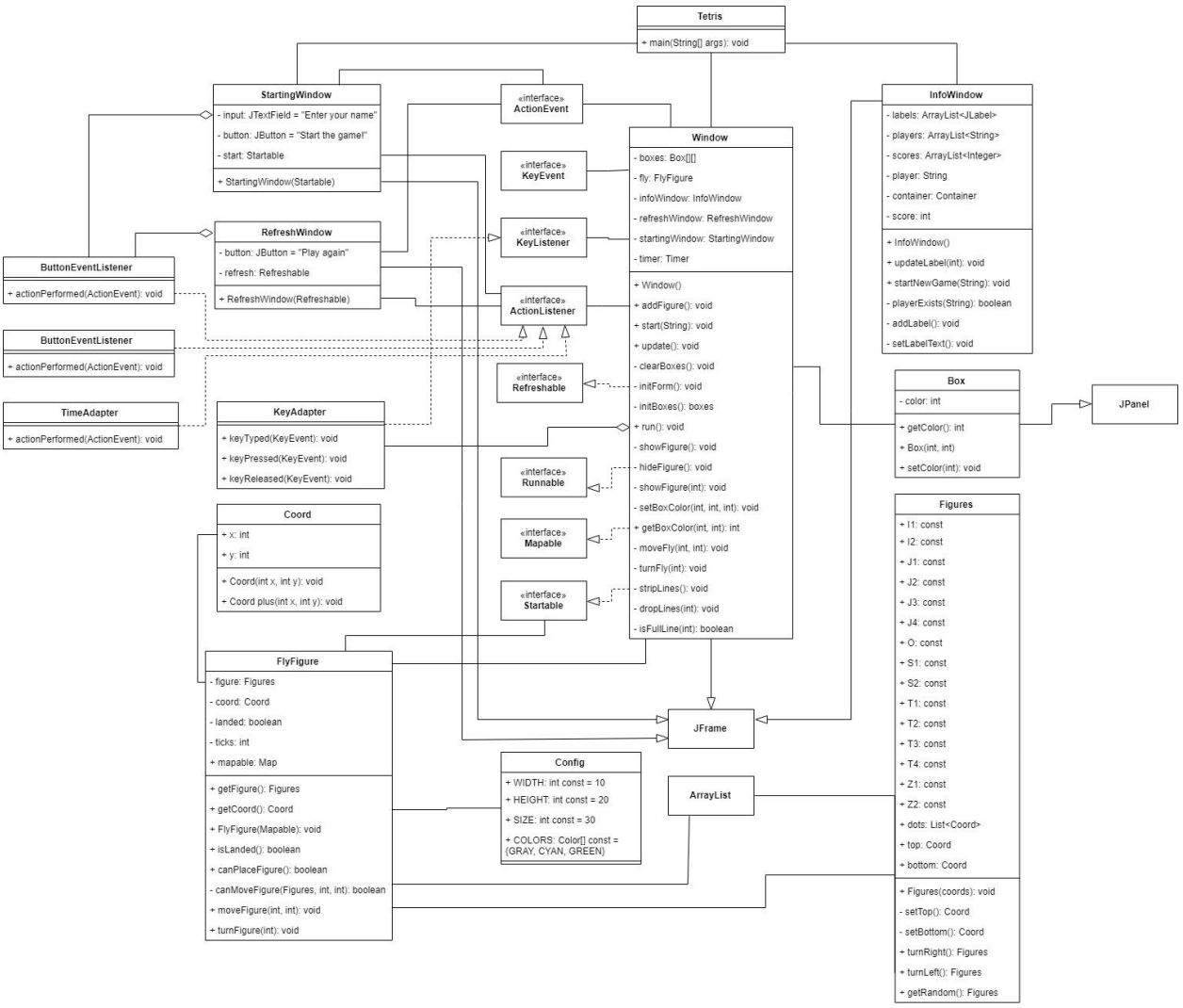
«Вправо» - перемещение фигуры вправо

«Влево» - перемещение фигуры влево

«Вверх» - поворот фигуры на 90 градусов по часовой стрелке.

6. Условие поражения

Игра заканчивается, когда очередная фигура не может вместиться на поле.


Tetris
+ main(String[] args): void
StartingWindow
- input: JTextField = "Enter your name"
- button: JButton = "Start the game!"
- start: Startable
+ StartingWindow(Startable)
«interface» ActionEvent
«interface» KeyEvent
«interface» KeyListener
«interface» ActionListener
«interface» Refreshable
«interface» Runnable
«interface» Mapable
«interface» Startable
Window
- boxes: Box[][]
- fly: FlyFigure
- infoWindow: InfoWindow
- refreshWindow: RefreshWindow
- startingWindow: StartingWindow
- timer: Timer
+ Window()
+ addFigure(): void
+ start(String): void
+ update(): void
- clearBoxes(): void
- initForm(): void
- initBoxes(): boxes
+ run(): void
- showFigure(): void
- hideFigure(): void
- showFigure(int): void
- setBoxColor(int, int, int): void
+ getBoxColor(int, int): int
- moveFly(int, int): void
- turnFly(int): void
- stripLines(): void
- dropLines(int): void
- isFullLine(int): boolean
InfoWindow
- labels: ArrayList<JLabel>
- players: ArrayList<String>
- scores: ArrayList<Integer>
- player: String
- container: Container
- score: int
+ InfoWindow()
+ updateLabel(int): void
+ startNewGame(String): void
- playerExists(String): boolean
- addLabel(): void
- setLabelText(): void
RefreshWindow
- button: JButton = "Play again"
- refresh: Refreshable
+ RefreshWindow(Refreshable)
ButtonEventListener
+ actionPerformed(ActionEvent): void
ButtonEventListener
+ actionPerformed(ActionEvent): void
TimeAdapter
+ actionPerformed(ActionEvent): void
KeyAdapter
+ keyTyped(KeyEvent): void
+ keyPressed(KeyEvent): void
+ keyReleased(KeyEvent): void
Coord
+ x: int
+ y: int
+ Coord(int x, int y): void
+ Coord plus(int x, int y): void
Box
- color: int
+ getColor(): int
+ Box(int, int)
+ setColor(int): void
JPanel
Figures
+ I1: const
+ I2: const
+ J1: const
+ J2: const
+ J3: const
+ J4: const
+ O: const
+ S1: const
+ S2: const
+ T1: const
+ T2: const
+ T3: const
+ T4: const
+ Z1: const
+ Z2: const
+ dots: List<Coord>
+ top: Coord
+ bottom: Coord
+ Figures(coords): void
- setTop(): Coord
- setBottom(): Coord
+ turnRight(): Figures
+ turnLeft(): Figures
+ getRandom(): Figures
FlyFigure
- figure: Figures
- coord: Coord
- landed: boolean
- ticks: int
+ mapable: Map
+ getFigure(): Figures
+ getCoord(): Coord
+ FlyFigure(Mapable): void
+ isLanded(): boolean
+ canPlaceFigure(): boolean
- canMoveFigure(Figures, int, int): boolean
+ moveFigure(int, int): void
+ turnFigure(int): void
JFrame
Config
+ WIDTH: int const = 10
+ HEIGHT: int const = 20
+ SIZE: int const = 30
+ COLORS: Color[] const = {GRAY, CYAN, GREEN}
ArrayList

Список источников

1. Н.А. Вязовик. Программирование на Java. Курс лекций, Интернет-университет информационных технологий, 2003 г., 592 стр.
2. Герберт Шилдт, Джеймс Холмс, Искусство программирования на Java, 2005 г., 336 стр.
3. http://ru.wikipedia.org/wiki/Тетрис
4. Хабибуллин И.Ш. Самоучитель Java 2. СПб.: БХВ-Петербург, 2007. - 720 с.: Ил.
5. Эккель Б. Философия Java. Библиотека программиста. СПб.: Питер, 2001.
6. Беркунський Е.Ю. Объектно-ориентированное программ-мирование на языке Java: Методические указания для студентов направления "Компьютерные науки". - М.: НУК, 2006. - 52 с.