**Министерство образования Республики Беларусь**

**Учреждение образования**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ**

**Факультет компьютерных систем и сетей**

**Кафедра программного обеспечения информационных технологий**

**Дисциплина: Компьютерные системы и сети(КСиС)**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту**

**на тему**

**Веб-сайт с RSS фидами**

**БГУИР КП 1-40 01 01 223 ПЗ**

**Выполнил:**

**студент: гр. 351002 Н.О.Сергушов**

**Проверил: Ф.И.Третьяков**

**Минск 2014**

Содержание

Введение 5

1 Анализ литературных источников 6

1.1 Обзор аналогов 6

2 Постановка задачи 7

3 Схема Алгоритма 8

3.1 Схема алгоритма 8

3.2 Схема вращения фигуры 9

4 Разработка программного средства 11

4.1 Обоснование выбора языка 11

4.2 Структуры данных 11

4.3 Используемые формы и их методы 12

4.3.1.1 Форма главного меню 12

4.3.1.2 Компоненты главного меню ………………… …12

4.3.1.3 Методы главного меню ………………… …12

4.3.2.1 Форма игрового поля Game ………………… …13

4.3.2.2 Компоненты поля Game ………………… …14

4.3.2.3 Методы формы Game ………………… …15

5 Руководство пользователя 20

5.1 Начало работы 20

5.2 Работа с приложением 21

Заключение 24

Приложение А 25

# Введение

Rss технология весьма простая и эффективная. Чтобы начать ей пользоваться не нужно никаких усилий. Прежде всего, нужно осознать, как работает эта система. Это позволит понять и оценить всю пользу и ценность новостных лент.

Rss – это xml-формат данных, который позволяет следить за обновлениями сайтов в интернете. Несколько лет назад любой серьезный сайт предоставлял возможность подписки на свой RSS-канал. Сейчас наблюдается тенденция «устаревания», на сайтах перестали делать rss-ленты и привычную оранжевую иконку можно обнаружить все реже.   Но сам RSS от этого не стал хуже.

На каждом сайте периодически публикуется какой-то контент — статьи, новости, комментарии. Подписка на RSS-ленту новостей сайта позволяет узнавать об обновлениях на этом сайте максимально оперативно, без необходимости посещать и просматривать сам сайт. Это сродни просмотру заголовков утренней газеты. Зачем идти на сайт и просматривать его в поисках новостей, если можно просматривать анонсы этих статей у себя на компьютере, и в случае находки интересного материала пройти по ссылке на свежую статью, заголовок которой обязательно будет в RSS-ленте.  Мало того, что это сберегает кучу времени, так еще и уменьшает трафик, ибо вы идете на сайт только если вам действительно что-то интересно, наверняка у вас есть любимые сайты, которые вы периодически посещаете, читаете статьи и новости, следите за обновлениями.

Ленты новостей обрабатываются специальными программами — RSS-агрегатами.  Агрегаты бывают двух видов — web-сервисы и программы на вашем компьютере. Какой из них выбрать — это дело вкуса. Онлайн-агрегаты по функциональности чем-то похожи на почтовые сервисы — вы создаете аккаунт в Сети на одном из сервисов, и все ваши ленты будут приходить в ящик на вашем аккаунте. Этот способ хорош тем, что вы всегда будете иметь доступ к вашим лентам, независимо от того где вы находитесь. Но у них есть недостаток, что все же нужно идти и проверять ящик. То есть, чтобы узнать о новостях или подписаться на новую ленту вам открывать в браузере страницу сервиса-агрегата.

Поэтому есть еще более удобные — локальные RSS-агрегаты, которые могут быть встроенными в браузер, почтовик или быть отдельным приложением.  Если вы знаете что такое почтовая программа, то вы поймете преимущество новостных лент. Локальный агрегат позволяет следить за новостями гораздо эффективнее — вы получаете обновления на сайтах практически моментально. Локальные RSS-агрегаты сами «бегают» на сайт и, если там появилось что-то новенькое «приносят» вам заголовок или анонс статьи, а иногда и всю статью — это зависит от того какую ленту предоставляет сайт.

Цель данной курсовой работы заключается в создании веб-сайта, который будет отображать новости интересующих rss-каналов Formula 1.

Данная пояснительная записка состоит из ряда разделов, которые охватывают как теоретические, так и практические аспекты разработки программного средства.

# Анализ литературных источников

## Обзор аналогов

На данный момент из популярных браузеров возможность обработки rss ленты новостей реализована только Mozilla.

Firefox умеет обрабатывать rss-ленту в xml-формате.    Если вы пользуетесь браузером Firefox, при попытке подписаться на RSS вам откроется страничка, где вы сможете выбрать онлайн-сервис новостных лент. Сам Mozilla Firefox поддерживает подписку на RSS только в виде закладок на новости.

Браузер Google Сhrome также не умеет работать c RSS-каналами, но такой функционал можно в него добавить в виде расширений. Например, вот так выглядит расширение Slick RSS для Chrome.

Онлайн-сервисы для подписки на RSS-каналы абсолютно бесплатны, и чем-то похожи на почтовые. Самые популярный это — Яндекс. Лента.  Для того чтобы воспользоваться сервисом нужно иметь Yandex аккаунт.   Если вы уже пользуетесь какими-либо сервисами Яндекса, то нужно лишь войти под своим логином и подписаться на ленту. После чего вы сможете читать новостную ленту через интерфейс Яндекс-почты.

# Постановка задачи

В рамках данной курсовой работы необходимо разработать веб-сайт на гоночную тематику. Данный сайт должен содержать информацию о командных и индивидуальных зачётах. Сайт должен содержать в себе информацию о последних гран-при и их квалификационных заездов, так же информацию о пилотах.

# Разновидности rss ФОРМАТОВ

## RSS форматы

На данный момент существует 7 различных форматов RSS. Как программисту, пишущему программу-агрегат, вам придется сражаться со всеми этими форматами. Ну а какой формат выбрать пользователю, публикующему свои новости в формате RSS?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Версии RSS и рекомендации | | | | |
| Версия | Владелец | За | Статус | Советы |
| 0.90 | Netscape |  | Отменен версией 1.0 | Не пользуйтесь |
| 0.91 | UserLand | Очень-очень простой | Официально отменен выходом версии 2.0. Но все еще популярен | Пользуйтесь для простых публикаций. Если вам понадобится большее, вы легко сможете перейти на 2.0 |
| 0.92, 0.93, 0.94 | UserLand | Больше возможностей, чем у 0.91 | Отменен с выходом версии 2.0 | Пользуйтесь версией 2.0 |
| 1.0 | RSS-DEV Working Group | Основан на языке RDF. Расширяется с помощью модулей. Не зависит от какой-либо одной компании | Стабилен. Ведется активная разработка модулей | Используется для приложений, где используется RDF, либо в том случае, если нужен какой-то определенный модуль |
| 2.0 | UserLand | Расширяется с помощью модулей. Прост при миграции с ветки форматов 0.9х | Стабилен. Ведется активная разработка модулей | Используется для публикации новостей общего назначения |

### **Структура RSS.**

Скажем, вы захотели написать программу, которая считывает новости в формате RSS, чтобы, например, публиковать заголовки новостей на своем сайте, или чтобы создать портал новостей и так далее. Как выглядит RSS-файл? Все зависит от того, о какой версии RSS идет речь. Вот пример файла в формате RSS 0.91 :

<rss version="0.91">  
  <channel>  
    <title>XML.com</title>  
    <link>http://www.xml.com/</link>   
    <description>XML.com features a rich mix of information and services for the XML community.</description>  
    <language>en-us</language>  
    <item>  
      <title>Normalizing XML, Part 2</title>  
      <link>http://www.xml.com/pub/a/2002/12/04/normalizing.html</link>  
      <description>In this second and final look at applying relational normalization techniques to W3C XML Schema data modeling, Will Provost discusses when not to normalize, the scope of uniqueness and the fourth and fifth normal forms.</description>  
    </item>  
    <item>  
      <title>The .NET Schema Object Model</title>  
      <link>http://www.xml.com/pub/a/2002/12/04/som.html</link>  
      <description>Priya Lakshminarayanan describes in detail the use of the .NET Schema Object Model for programmatic manipulation of W3C XML Schemas.</description>  
    </item>  
    <item>  
      <title>SVG's Past and Promising Future</title>  
      <link>http://www.xml.com/pub/a/2002/12/04/svg.html</link>  
      <description>In this month's SVG column, Antoine Quint looks back at SVG's journey through 2002 and looks forward to 2003.</description>  
    </item>  
  </channel>  
</rss>

Все выглядит достаточно просто. Блок новостей (channel) состоит из заголовка, ссылки, данных о языке новостей и описания. После этого идет список самих новостей, где в каждом пункте указывается заголовок, ссылка и краткое описание новости.

Теперь давайте взглянем, как та же самая информация выглядит в формате RSS 1.0:

<rdf:RDF  
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
  xmlns="http://purl.org/rss/1.0/"  
  xmlns:dc="http://purl.org/dc/elements/1.1/"  
>  
  <channel rdf:about="http://www.xml.com/cs/xml/query/q/19">  
    <title>XML.com</title>  
    <link>http://www.xml.com/</link>  
    <description>XML.com features a rich mix of information and services for the XML community.</description>  
    <language>en-us</language>  
    <items>  
      <rdf:Seq>  
        <rdf:li rdf:resource="http://www.xml.com/pub/a/2002/12/04/normalizing.html"/>  
        <rdf:li rdf:resource="http://www.xml.com/pub/a/2002/12/04/som.html"/>  
        <rdf:li rdf:resource="http://www.xml.com/pub/a/2002/12/04/svg.html"/>  
      </rdf:Seq>  
    </items>  
  </channel>  
  <item rdf:about="http://www.xml.com/pub/a/2002/12/04/normalizing.html">  
    <title>Normalizing XML, Part 2</title>  
    <link>http://www.xml.com/pub/a/2002/12/04/normalizing.html</link>  
    <description>In this second and final look at applying relational normalization techniques to W3C XML Schema data modeling, Will Provost discusses when not to normalize, the scope of uniqueness and the fourth and fifth normal forms.</description>  
    <dc:creator>Will Provost</dc:creator>  
    <dc:date>2002-12-04</dc:date>      
  </item>  
  <item rdf:about="http://www.xml.com/pub/a/2002/12/04/som.html">  
    <title>The .NET Schema Object Model</title>  
    <link>http://www.xml.com/pub/a/2002/12/04/som.html</link>  
    <description>Priya Lakshminarayanan describes in detail the use of the .NET Schema Object Model for programmatic manipulation of W3C XML Schemas.</description>  
    <dc:creator>Priya Lakshminarayanan</dc:creator>  
    <dc:date>2002-12-04</dc:date>      
  </item>  
  <item rdf:about="http://www.xml.com/pub/a/2002/12/04/svg.html">  
    <title>SVG's Past and Promising Future</title>  
    <link>http://www.xml.com/pub/a/2002/12/04/svg.html</link>  
    <description>In this month's SVG column, Antoine Quint looks back at SVG's journey through 2002 and looks forward to 2003.</description>  
    <dc:creator>Antoine Quint</dc:creator>  
    <dc:date>2002-12-04</dc:date>      
  </item>  
</rdf:RDF>

Те, кто знаком с RDF, сразу узнают, что этот файл - RDF-документ, сохраненный в XML. Остальные, разберутся, что в файле представлена вся та же информация, что и в первом примере. Однако тут добавлена еще некоторая дополнительную информацию, как, например, авторство каждой новости, и дата публикации, которых нет в RSS 0.91.

Несмотря на то, что RSS 1.0 является смесью RDF и XML, структурно он схож с предыдущими версиями RSS - схож достаточно, чтобы рассматривать его как обычный XML-файл. Следовательно, можно написать одну программу, которая умеет извлекать информацию из обоих форматов: и из RSS 0.91 и из RSS 1.0. Однако есть все-таки некоторые различия, о которых программа должна знать:

1. Корневым элементом в RSS 1.0 является rdf:RDF, а не rss. Вам либо придется явно обрабатывать оба этих элемента, либо просто игнорировать их и слепо извлекать только ту информацию, которая вам нужна.
2. В RSS 1.0 используются пространства имен (namespaces). Пространство имен для RSS 1.0 выглядит так http://purl.org/rss/1.0/. И это пространство имен принимается по умолчанию. Кроме того в файле используются пространства имен http://www.w3.org/1999/02/22-rdf-syntax-ns# для элементов, специфичных для RDF (мы их тоже можем игнорировать), и http://purl.org/dc/elements/1.1/ (Dublin Core) для дополнительных метаданных об авторах статей и датах публикаций.

Можно пойти двумя путями: если XML-парсер не понимает пространства имен, вы можете просто считать, что в файле используются элементы с префиксами и слепо искать в них элементы items и dc:creator. Такой способ сработает в большинстве случаев, так как в новостях формата RSS 1.0 чаще всего используется только пространство имен, принятое по умолчанию, и пространство имён Dublin Core. Конечно, данный способ - не элегантен, ведь нет никаких гарантий, что в каких-нибудь новостях не будет использовано какое-либо другое пространство имен (что вполне легально с точки зрения RDF и XML), и парсер пропустит все новости.

Если же XML-парсер понимает пространства имен, вы можете построить более изящное решение, которое сумеет разобрать новости и формате 0.91 и в формате 1.0.

1. Менее очевидный, но важный факт состоит в том, что в RSS 1.0 элементы item находятся вне элемента channel. В RSS 0.91 элементы item расположены внутри channel. В 0.90 они были снаружи. В 2.0 - они внутри.
2. В элементе channel есть один элемент items. Он нужен только для RDF-парсеров (задает порядок новостей).Можно его игнорировать и считать, что все новости идут в том порядке, в каком расположены элементы item.

А как выглядит формат RSS 2.0? К счастью, для программ, понимающих форматы RSS 0.91 и 1.0, формат RSS 2.0 будет довольно простым.

<rss version="2.0" xmlns:dc="http://purl.org/dc/elements/1.1/">  
  <channel>  
    <title>XML.com</title>  
    <link>http://www.xml.com/</link>  
    <description>XML.com features a rich mix of information and services for the XML community.</description>  
    <language>en-us</language>  
    <item>  
      <title>Normalizing XML, Part 2</title>  
      <link>http://www.xml.com/pub/a/2002/12/04/normalizing.html</link>  
      <description>In this second and final look at applying relational normalization techniques to W3C XML Schema data modeling, Will Provost discusses when not to normalize, the scope of uniqueness and the fourth and fifth normal forms.</description>  
      <dc:creator>Will Provost</dc:creator>  
      <dc:date>2002-12-04</dc:date>      
    </item>  
    <item>  
      <title>The .NET Schema Object Model</title>  
      <link>http://www.xml.com/pub/a/2002/12/04/som.html</link>  
      <description>Priya Lakshminarayanan describes in detail the use of the .NET Schema Object Model for programmatic manipulation of W3C XML Schemas.</description>  
      <dc:creator>Priya Lakshminarayanan</dc:creator>  
      <dc:date>2002-12-04</dc:date>      
    </item>  
    <item>  
      <title>SVG's Past and Promising Future</title>  
      <link>http://www.xml.com/pub/a/2002/12/04/svg.html</link>  
      <description>In this month's SVG column, Antoine Quint looks back at SVG's journey through 2002 and looks forward to 2003.</description>  
      <dc:creator>Antoine Quint</dc:creator>  
      <dc:date>2002-12-04</dc:date>      
    </item>  
  </channel>  
</rss>

Как показывает данный пример, в RSS 2.0 тоже используются пространства имен, как и в RSS 1.0. Но это не RDF. Как и в RSS 0.91, нет пространства имен, принятого по умолчанию, а новости (в элементах item) размещены опять в элементе channel, что доказывает преимущество формата rss 2.0.

Изучив все форматы от rss 0.9 до rss 2.0, пришли к выводу, что в данной курсовой работе лучше всего рассматривать формат 2.0, так как в данный момент большинство сайтов используют именно данный формат для формирования своего RSS канала.

## Конкуренция с форматом Atom

В июне 2006 года появился конкурент RSS — формат [Atom](https://ru.wikipedia.org/wiki/Atom). Стандарт Atom 1.0 определяет 21 элемент канала новостей, RSS 2.0 — 30 элементов. Большинство элементов RSS 2.0 и Atom 1.0 не пересекаются и не соответствуют аналогам. Часть элементов редко используется на практике или их функциональность достигается другими путями. В середине 2005 года, когда стандарт Atom был только принят, он использовался лишь на нескольких десятках сайтов. В то же время RSS 2.0 был широко распространен на различных социальных сервисах. К началу 2006 года Atom 1.0 стал получать распространение и стал проблемой для программ-агрегатов, которые не могли разобрать формат, не похожий на RSS. Разработчики агрегатов принялись патчить свои программы на основе баг-репортов. К середине 2006 года, благодаря активному продвижению W3C и поддержке со стороны Google, IBM и еще ряда корпораций, Atom стал основным форматом для многих крупных онлайн-сервисов. При этом популярность RSS 2.0 не снизилась, так как контент-провайдеры зачастую стали представлять новости в обоих форматах, на выбор.

Канал RSS 2.0 может содержать только текстовую, или только гипертекстовую (HTML-escaped, теги экранируются в CDATA) информацию, без возможности указания, какое предоставление используется. Экранированные HTML (например, строка AT&T будет представлена как AT&amp;T) привносят дополнительные сложности разработчикам. Модель содержания заголовков (<title>) не определена. Заголовки с угловыми скобками или амперсандами будут интерпретированы значительной частью программ чтения независимо от представления.

Модель содержания RSS 2.0 не допускает использования XML, что уменьшает возможности повторного использования содержимого.

В свою очередь, Atom имеет тщательно проработанный контейнер содержания, которое может быть следующих типов:

* Простой текст, без разметки (по умолчанию).
* Экранированный HTML.
* Правильно оформленная (well-formed) разметка XHTML.
* Некоторые другие XML словари.
* Двоичное содержание, закодированное в base64.
* Указатель на внешнее веб-содержимое, не включенное в канал.

Atom не дает гарантии, что получатель будет иметь возможность сделать что-нибудь полезное с внешними данными или бинарным содержанием. Тем не менее, это ограждает получателей от неправильного определения типа содержимого на основе предположений.

По результатам анализа и сравнения можно сделать следующие выводы:

* Стандарт Atom в силу своих возможностей является более универсальным средством синдикации и агрегирования информации. Этот формат следует использовать при разработке новых приложений, использующих каналы.
* Спецификация RSS 2.0 остается широко распространенной и в ближайшее время останется таковой. Следовательно, разработчикам агрегаторов и роботов-сборщиков не стоит пока отказываться от этого формата синдикации и агрегирования.

# Разработка программного средства

## Обоснование выбора языка

Для разработки данного программного средства был выбран язык программирования C#.

C# — Cи-образный, типобезопасный объектно-ориентированный язык, предназначенный для разработки разнообразных приложений, выполняемых в среде .NET Framework.

Язык C# разрабатывался "с нуля" и вобрал в себя много полезных свойств таких языков, как C++, Java, Visual Basic, а также Pascal, Delphi и др.

В частности, с помощью языка C# можно быстро и качественно создавать обычные приложения Windows. Стоит также отметить наличие встроенных обработчиков событий. Более того, данный язык включает в себя множество готовых классов, использование которых значительно облегчают разработку любого программного средства. Важным достоинством является простота и изящность работы с разнообразными динамическими структурами данных.

Все эти факторы обосновывают выбор языка C# для написания пользовательского приложения под OC Windows.

## Структуры данных

## Структура сайта

Структура сайта будет состоять из следующих страниц: главная страница, страница с индивидуальным зачётом пилотов формулы 1, страница с пилотами данного сезона и их индивидуальной информации, а также страница с расписанием всех гран-при текущего сезона и страницы с результатом последней гонки. Главная страница будет включать в себя информацию о текущем индивидуальном зачёте лучших десяти пилотов и кубке конструкторов данного года, а также список с RSS ссылками на интересующие новости. Страница с индивидуальным зачётом будет содержать в себе информацию о индивидуальном зачёте каждого пилота. Страница с расписанием содержит в себе дату проведения каждого гран-при, а также время и ссылку на более подробную информацию о данной трассе. Последняя страница меню содержит в себе информацию о последнем гран-при: победитель, стартовая позиция, финишная позиция, лучший круг, время, статус, команда, количество очков, заработанных за гонку.

**4.3.1 Master Page**

В основе каждой страницы лежит мастер-страница. Мастер-страница - это средство ASP.NET, разработанное специально для стандартизации компоновки веб-страниц. Мастер-страница представляет собой шаблоны веб-страниц, которые могут определять фиксированное содержимое и объявлять часть веб-страницы, куда можно помещать нестандартное содержимое. При использовании одной и той же мастер-страницы во всем веб-сайте компоновка гарантированно будет одинаковой. Более того, если изменить определение мастер-страницы после ее применения, то все использующие ее веб-страницы автоматически воспримут это изменение.

Данная мастер страница состоит из верхнего контитула, контента и нижнего контитула. Верхний контитул содержит в себе строку с меню в виде списка.

**4.3.2 Главная страница**

При запуске сайта открывается главная страница Default.aspx. Данная страница отображает информацию об индивидуальном и командном зачётах а также генерирует список новостей в случае выбора RSS ссылки из составленного списка. Данный список содержится в БД. Его можно считывать, а также добавлять и удалять информацию из него. В случае загрузки главной страницы, срабатывает метод Page\_Load, в котором формируется список зачётов пилотов - standingsDrivers при вызове метода MakeDriversStandings() класса DriversStandings. Также в результате срабатывания метода Page\_Load формируется список кубка конструкторов – standingsTeam при вызове метода MakeTeamStandings() класса TeamStandings. После чего создаём подключение к БД, в которой хранится информация о сайте и его RSS ссылке (рисунок 1).

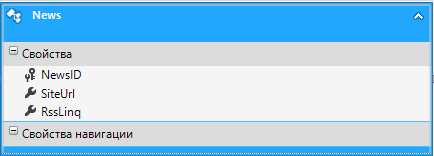


Рисунок 1 - Структура БД.

После создания подключения, считываем всю информацию с БД параллельно записывая в список DropDownListRssUrls url сайта и во временный список listRssUrls RSS ссылка сайта. Взаимодействие с БД выполнялось с использованием Entity Framework.

Серверный c# код главной страницы использует следующие классы логику которых рассмотрим подробнее: DriversStandings, TeamStandings.

**4.3.2.1 Класс DriversStandings**

Данный класс формирует список пилотов индивидуального зачёта - List<Driver> standingsDrivers, используя метод MakeDriversStandings().

public static void MakeDriversStandings();

Назначение: данный метод отправляет запрос по адресу - <http://ergast.com/api/f1/2015/driverStandings> и получает данные в формате xml (рисунок 2). В документе содержится информация о пилотах данного сезона ( рисунок 3).Прочитав всю необходимую информацию из документа, формируется список List<Driver> standingsDrivers. Другие основные методы:

public static string GetCurrentDriverNumber();

Назначение: Возвращает номер выбранного гонщика.

public static IEnumerable<Driver> GetDriverStandings();

Назначение: Возвращает список List<Driver> standingsDrivers, так как данный список объявлен с модификатором доступа private.

private static string GetImageCar(string str);

Назначение: Возвращает путь к картинке болида пилота относительно корневого элемента.

private static string GetImageDriver(string number);

Назначение: Возвращает путь к картинке пилота относительно корневого элемента.



Рисунок 2 – Пример Xml ответа, индивидуальный зачёт

public class Driver

{

public string Position { get; set; }

public string Points { get; set; }

public string Wins { get; set; }

public string PermanentNumber { get; set; }

public string GivenName { get; set; }

public string FamilyName { get; set; }

public string Nationality { get; set; }

public string Constructor { get; set; }

public string DateOfBirth { get; set; }

public string ImageCar { get; set; }

public string ImageDriver { get; set; }

public string NameSurname { get; set; }

}

Рисунок 3 – Структура класса Driver

**4.3.2.2 Класс TeamStandings**

Данный класс формирует список конструкторов - List<Team> standingsTeam, используя метод MakeTeamStandings().

public static void MakeTeamStandings();

Назначение: данный метод отправляет запрос по адресу - <http://ergast.com/api/f1/current/constructorStandings> и получает данные в формате xml (рисунок 4). В документе содержится информация о командах данного сезона (рисунок 5).Прочитав всю необходимую информацию из документа, формируется список List<Driver> standingsDrivers. Основные методы:

public static List<Team> GetTeamStandings();

Назначение: возвращает список конструкторов List<Driver> standingsDrivers.



Рисунок 4 – Пример Xml ответа, командный зачёт

public class Team

{

public string Position { get; set; }

public string Points { get; set; }

public string Wins { get; set; }

public string Name { get; set; }

public string Nationality { get; set; }

}

Рисунок 5 – Структура класса Team

**4.3.2.2 Компоненты формы Game**

Форма Game включает в себя следующие компоненты:

- Panel[,] – массив панелей, отображает игровое поле;

- Label – для отображения названия полей;

- TextBox – для отображения очков и уровня скорости падения;

- Timer – для прорисовки панелей и обработки логики игры.

Для отображения игрового поля выгодно использовать массив панелей, размером 20 на 10 элементов, которые создаются динамически в программе. В зависимости от значений полей логической матрицы, панели перекрашиваются в определённый цвет, тем самым отображая поле игры. Динамический массив панелей также удобно использовать для отображения следующей по очереди фигуры. В зависимости от значений логической матрицы панели принимают определённый цвет, создавая изображение фигуры. Все динамически созданные панели изменяют свой цвет по таймеру TimerDr.

**4.3.2.3 Методы формы Game**

В случае загрузки формы Game, таймеры TimerPr и TimerDr активируются, тем самым вызывая методы private void TimerPr\_Tick(object sender, EventArgs e) и private void TimerDr\_Tick(object sender, EventArgs e). Первый метод обрабатывает логику программы, вызывая методы с другого модуля. Второй метод прорисовывает игровое поле и изображение следующей фигуры в зависимости от полей матриц.

Другие методы перечислены ниже.

private void Game\_KeyDown(object sender, KeyEventArgs e)

Назначение: Вызывается при нажатии любой клавиши. Если нажатая клавиша отвечает за передвижение фигуры налево, направо, ускорение, поворот, паузу, выход, то вызывается метод соответствующий нажатой клавише.

private void Game\_Load(object sender, EventArgs e)

Назначение: Вызывается при загрузки формы Game. Данный метод инициализирует игровое поле, первую падающую фигуру , следующую фигуру , значение логического таймера, включает таймеры.

public void Draw()

Назначение: Изменяет цвет панелек в зависимости от значений полей матриц.

private void MapInitialize()

Назначение: Данный метод инициализирует значения полей логической матрицы Map.

public void NextFigurePanelCreate()

Назначение: Метод динамически создаёт панели, которые отрисовывают следующую фигуру.

public void GlassInitialize()

Назначение: Метод динамически создаёт панели, которые отображают игровое поле, на форме Game.

public void Tick(ref tetride[,] map, ref bool ground, ref bool finish, ref int scores,ref int CornI,ref int CornJ,ref int Figure,ref int FigureNext,ref int[,] NextFigArr)

Формальные параметры: map – логическая матрица, в которой хранится информация об игровом поле,NextFigArr – матрица, в которой хранятся значения для отрисовки следующей фигуры, FigureNext – следующая фигура, Figure – текущая фигура, CornI – номер строки, в которой находится верхний левый угол квадрата падающей фигуры, CornJ - номер столбца, в которой находится верхний левый угол квадрата падающей фигуры, scores – очки набранные пользователем, finish – флаг, принимает true – если игра закончена, ground – флаг, который отвечает за положение фигуры на поле, принимает true – если фигура завершила падение, false – если падает.

Назначение: Метод вызывается таймером TimerPr, через определённыый промежуток времени. Данный метод смещает падающую фигуру на одну позицию вниз, в случае, если фигура завершила падение, из этого метода вызывается другой, создающий новый элемент.

public void Rotate(ref tetride[,] map,ref int CornI,ref int CornJ,int Figure)

Формальные параметры: map – логическая матрица, CornI,CornJ – координата верхнего левого угла падающей фигуры, Figure – падающая фигура, которую необходимо повернуть.

Назначение: Метод вращает падающую фигуру против часовой стрелки относительно её геометрического центра. Схема вращения представлена на рисунках 2 и 3.

public void ElementRemoveRight(ref tetride[,] map,ref int CornI,ref int CornJ)

формальные параматры: map – логическая матрица, CornI,CornJ – координата верхнего левого угла падающей фигуры.

Назначение: Метод перемещает фигуру на одну позицию вправо, в случае если справа нет преграды или стенки.

public void ElementRemoveLeft(ref tetride[,] map, ref int CornI, ref int CornJ)

формальные параматры: map – логическая матрица, CornI,CornJ – координата верхнего левого угла падающей фигуры.

Назначение: Метод перемещает фигуру на одну позицию влево, в случае если справа нет преграды или стенки (Рисунок 6).



Рисунок 6 – Схема сдвига фигуры влево

private void ArrZero(ref tetride[,] tempshape)

Формальные параметры: tempshape – копия массива map;

Назначение: Метод обнуляет массив tempshape;

private bool CheckShifting(tetride[,] tempshape, tetride[,] map)

Формальные параметры: map - map – логическая матрица, в которой хранится информация об игровом поле, tempshape – копия массива map;

Назначение: Метод вызывается в методе Rotate. Он проверяет, наложилась ли облпсть массива tempshape на массив map.

private void ShiftLeft(ref tetride[,] tempshape, int posshift)

Формальные параметры: tempshape – копия массива map, posshift – количество позиций на которые нужно сдвинуть фигуру в массиве tempshape;

Назначение: Сдвигает фигуру в массиве tempshape влево на posshift позиций;

private void ShiftRight(ref tetride[,] tempshape, int posshift)

Формальные параметры: tempshape – копия массива map, posshift – количество позиций на которые нужно сдвинуть фигуру в массиве tempshape;

Назначение: Сдвигает фигуру в массиве tempshape вправо на posshift позиций;

private void NextFigureInitialize(ref int[,] NextFigArr,int NextFigure)

Формальные параметры: NextFigArr – массив для обрисовки следующей фигуры, NextFigure – следующая фигура;

Назначение: Данный метод инициализирует массив NextFigArr в зависимости от значения NextFigure.

private void ElementCreate(int Elem,ref tetride[,] map,ref int CornI,ref int CornJ)

Формальные параметры: Elem – текущая фигура, map - логическая матрица, в которой хранится информация об игровом поле, CornI,CornJ – координата верхнего левого угла падающей фигуры;

Назначение: Данный метод инициализирует значения CornI,CornJ, а также массив map в зависимости от значения Elem.

private void ElementRemoveDown(ref tetride[,] map,ref int CornI,ref int CornJ)

Формальные параметры: map - логическая матрица, в которой хранится информация об игровом поле, CornI,CornJ – координата верхнего левого угла падающей фигуры;

Назначение: Метод изменяет значение массива map, а именно перемещает поле с действующей фигурой на одну позицию ниже по вертикале (Рисунок 7);

private bool CrashRotateWallsRight(tetride[,] map)

Формальные параметры: map - логическая матрица, в которой хранится информация об игровом поле;

Назначение: Проверка на столкновение фигуры со стеной после поворота.



Рисунок 7 – Схема сдвига фигуры вниз

# Руководство пользователя

## Начало работы

После запуска программы (Tetris.exe) на экране появится главное меню игрового приложения (Рисунок 4).

Изначально работа с программным средством осуществляется через главное меню. Находясь в главном меню, пользователь может начать игру, выбрать для себя подходящий уровень сложности, а также покинуть игру.

Нажав на кнопку “Options”, откроется форма Options (Рисунок 8), где пользователь может выбрать свой уровень сложности. По умолчанию, уровень сложности в игре – средний.

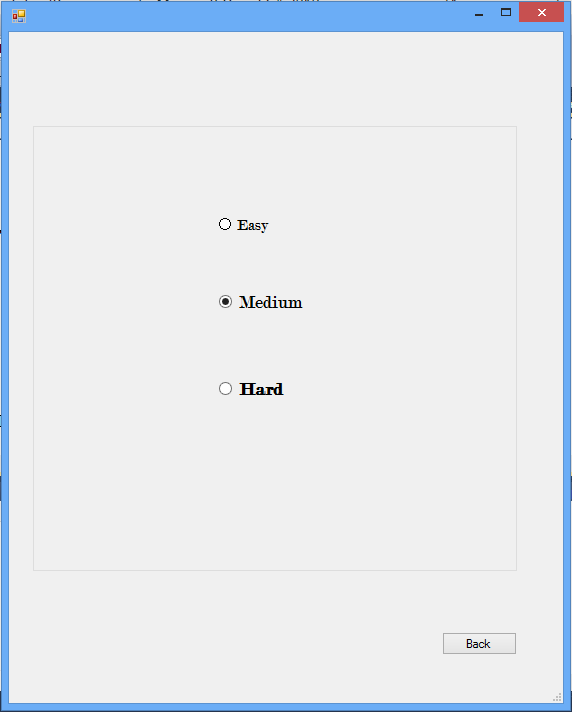


Рисунок 8 – Форма Options

Выход из формы Options в Главное меню осуществляется по клику на кнопку Back.

## Работа с приложением

В случае нажатия на кнопку главного меню “Start Game”, пользователю откроется форма Game, данная форма является игровым полем. Игра начинается сразу же после загрузки формы. Фигурки падают по очереди сверху вниз на поле в 20 линий высоту и 10 в ширину до самого низа. Их можно переворачивать во время падения, а также с ускорением отправлять вниз, когда вы уже решили ее положение. Она падает на дно если у нее нет препятствия в виде другой фигуры.

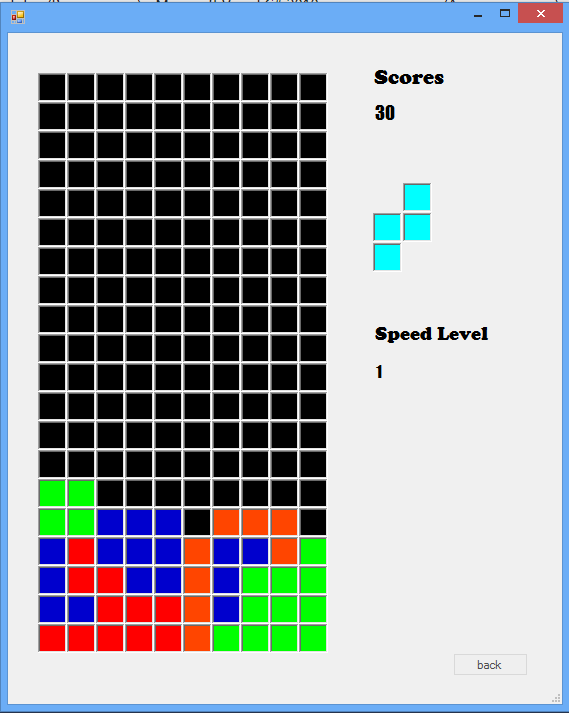


Рисунок 9 – Заполнение четырёх рядов

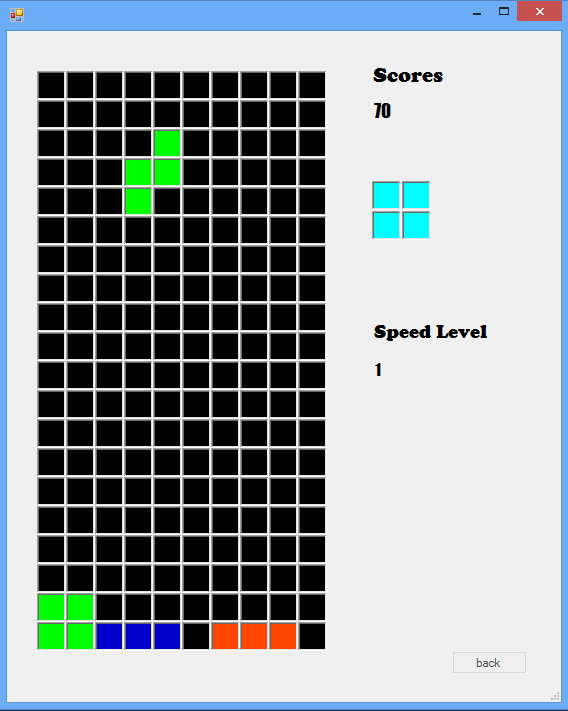


Рисунок 10 – Удаление заполненных рядов и начисление очков

Скорость падения фигур в начале игры зависит от уровня сложности. Когда игрок набирает определённое количество очков, темп игры увеличивается. На разных уровнях это количество очков, которые пользователь должен набрать для увеличения темпа игры, разное. На лёгком уровне через 200 очков, среднем – 100, трудном – 50 очков. Темп игры ускоряется на одну и ту же величину в независимости от уровня сложности. Текущий темп игры отображается в поле Speed Level.

Очки набираются в том случае если, ставить фигурки так, чтобы образовывалась одна сплошная линия по горизонтали. После чего она исчезнет и оставшиеся фигурки подвинуться на ее место. Одна заполненная линия даёт 10 очков.

Управление фигурами осуществляется через клавиши:

Left – перемещение фигуры влево;

Right – перемещение фигуры вправо;

Down – ускорение падения фигуры,Up – остановка ускоренного падения

Space – вращение фигуры;

P – пауза;

Escape – выход в главное меню;

Игра заканчивается если груда фигурок соберется до самого верха поля. В случае заполнения игрового поля пользователю выдаётся сообщение о результате игры (Рисунок 11).

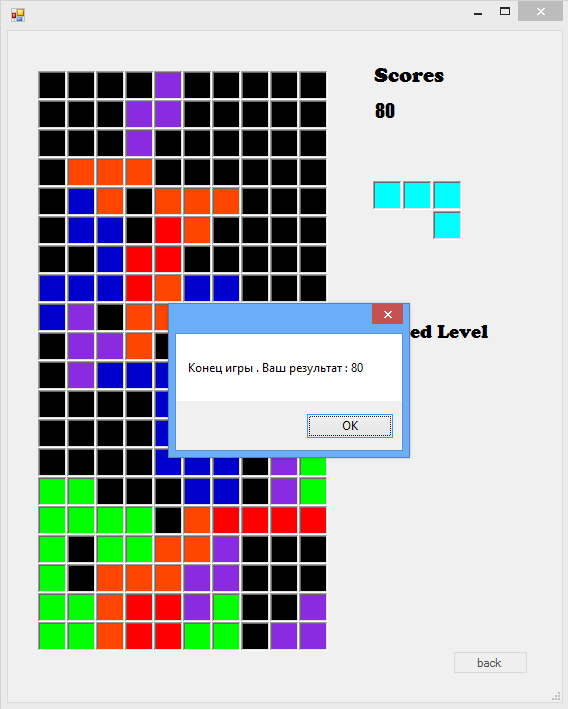


Рисунок 11 – Результат игры

Заключение

Результатом выполнения данной курсовой работы стало создание игрового приложения “Тетрис”. Эта головоломка является одной из самых знаменитых головоломок в мире.

В ходе выполнения курсовой работы были закреплены навыки разработки в визуальной среде Visual C#, работа с динамическими структурами и компонентами.

Разработанные возможности программы:

1. Удобный Интерфейс;
2. Возможность выбора уровня сложности;
3. Перемещение фигур влево;
4. Перемещение фигур вправо;
5. Ускорение падения фигур;
6. Остановка ускоренного падения фигур;
7. Вращение фигур;
8. Пауза;
9. Начисления очков;
10. Увеличения темпа игры;
11. Подсказка в виде поля со следующей фигурой;

В дальнейшей работе с данным проектом планируется добавление ряда функций:

1. Добавления музыкального сопровождения;
2. Усовершенствование интерфейса;
3. Добавления режима игры с двумя игроками;

Приложение А

(**обязательное)**

**Исходный код программы**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Threading;

namespace tetris

{

public struct tetride

{

public int field;

public int color;

}

public partial class Game : Form

{

static Menu menu;

public Game(Menu oMenu)

{

InitializeComponent();

menu = oMenu;

if (menu.EasyMod)

{

TimerInterval = 600;

SpeedInc = 20;

PointCut = 200;

PointInc = 200;

}

if (menu.MediumMod)

{

TimerInterval = 400;

SpeedInc = 20;

PointCut = 100;

PointInc = 100;

}

if (menu.HardMod)

{

TimerInterval = 300;

SpeedInc = 20;

PointCut = 50;

PointCut = 50;

}

TimInt = TimerInterval;

}

TimerProg TimPr = new TimerProg();

public tetride[,] Map = new tetride[20, 10];

public int[,] NextFigArr = new int[4, 4];

public Panel[,] FigPanel = new Panel[4, 4];

public bool ground = true;

public bool finish = false;

static int SpeedInc = 10;

static int TimerInterval ;

static int TimInt ;

static int PointCut ;

static int PointInc ;

const int SizePnl = 29, StartGlassX = 40, StartGlassY = 30, StartNextFigX = 75, StartNextFigY = 150 ;

public Panel[,] pnl = new Panel[20, 10];

public int Scores = 0, SpeedLvl = 1, FigureNext = 0;

public int CornerI = 0, CornerJ = 0, Figure;

private bool TimerStop = false;

public void GlassInitialize()

{

int i = 0, j = 0;

for (i = 0; i < 20; i++)

{

for (j = 0; j < 10; j++)

{

pnl[i, j] = new Panel();

pnl[i, j].BackColor = Color.Black;

pnl[i, j].Left = StartGlassY + j \* (SizePnl );

pnl[i, j].Top = StartGlassX + i \* (SizePnl );

pnl[i, j].Width = SizePnl;

pnl[i, j].Height = SizePnl;

pnl[i, j].BorderStyle = BorderStyle.Fixed3D;

this.Controls.Add(pnl[i, j]);

}

}

}

public void NextFigurePanelCreate()

{

int i, j;

for (i = 0; i < 4; i++)

{

for (j = 0; j < 4; j++)

{

FigPanel[i, j] = new Panel();

FigPanel[i, j].Visible = false;

FigPanel[i, j].Left = StartNextFigX + j \* (SizePnl + 1) + 10 \*SizePnl;

FigPanel[i, j].Top = StartNextFigY + i \* (SizePnl + 1) ;

FigPanel[i, j].Width = SizePnl;

FigPanel[i, j].Height = SizePnl;

FigPanel[i, j].BorderStyle = BorderStyle.Fixed3D;

this.Controls.Add(FigPanel[i, j]);

}

}

}

private void MapInitialize()

{

int i, j;

for (i = 0; i < Map.GetLength(0); i++)

{

for (j = 0; j < Map.GetLength(1); j++)

{

Map[i, j].field = 0;

Map[i, j].color = 0;

}

}

}

private void TimerPr\_Tick(object sender, EventArgs e) // падение по таймеру

{

if (ground)

{

TimerPr.Interval = TimInt;

}

if (Scores >= PointCut)

{

if (!(TimInt - SpeedInc <= 0))

TimInt = TimInt - SpeedInc;

PointCut = PointCut + PointInc;

SpeedLvl++;

}

TimPr.Tick(ref Map, ref ground, ref finish,ref Scores,ref CornerI,ref CornerJ,ref Figure,ref FigureNext,ref NextFigArr);

TextScores.Text = Convert.ToString(Scores);

TextSpeed.Text = Convert.ToString(SpeedLvl);

}

private void TimerDr\_Tick(object sender, EventArgs e) // перерисовка по таймеру

{

Draw();

if (finish)

{

TimerPr.Enabled = false;

TimerDr.Enabled = false;

MessageBox.Show("Конец игры . Ваш результат : " + Convert.ToString(Scores));

Form.ActiveForm.Close();

Form.ActiveForm.Dispose();

menu.Show();

}

}

public void Draw()

{

int i, j;

for (i = 0; i < Map.GetLength(0); i++)

{

for (j = 0; j < Map.GetLength(1); j++)

{

switch (Map[i, j].color)

{

case 0:

pnl[i,j].BackColor = Color.Black;

break;

case 1:

pnl[i,j].BackColor = Color.Lime;

break;

case 2:

pnl[i,j].BackColor = Color.Red;

break;

case 3:

pnl[i,j].BackColor = Color.OrangeRed;

break;

case 4:

pnl[i,j].BackColor = Color.MediumBlue;

break;

case 5:

pnl[i,j].BackColor = Color.BlueViolet;

break;

}

}

}

for (i = 0; i < NextFigArr.GetLength(0); i++)

{

for (j = 0; j < NextFigArr.GetLength(1); j++)

{

if (NextFigArr[i, j] == 2)

{

FigPanel[i, j].BackColor = Color.Aqua;

FigPanel[i, j].Visible = true;

}

else

FigPanel[i, j].Visible = false;

}

}

}

private void Game\_Load(object sender, EventArgs e)

{

Random rnd = new Random();

GlassInitialize();

NextFigurePanelCreate();

MapInitialize();

FigureNext = rnd.Next(7);

TimerPr.Enabled = true;

TimerDr.Enabled = true;

TextScores.Text = Convert.ToString(Scores);

TextSpeed.Text = Convert.ToString(SpeedLvl);

}

private void ButBack\_Click(object sender, EventArgs e)

{

Form.ActiveForm.Hide();

Menu mn = new Menu();

mn.ShowDialog();

}

private void Game\_KeyDown(object sender, KeyEventArgs e)

{

if (TimerStop == false)

{

if (e.KeyCode == Keys.Left) // влево

{

TimPr.ElementRemoveLeft(ref Map, ref CornerI, ref CornerJ);

}

if (e.KeyCode == Keys.Right) // вправо

{

TimPr.ElementRemoveRight(ref Map, ref CornerI, ref CornerJ);

}

if (e.KeyCode == Keys.Down) // сброс

{

TimerPr.Interval = TimInt / 10;

}

if (e.KeyCode == Keys.Up) // остановить сброс

{

TimerPr.Interval = TimInt;

}

if (e.KeyCode == Keys.Space) // вращение фигуры

{

if (ground == false)

TimPr.Rotate(ref Map, ref CornerI, ref CornerJ, Figure);

}

}

if (e.KeyCode == Keys.P)

{

TimerStop = !TimerStop;

if (TimerStop)

{

TimerPr.Enabled = false;

}

else

{

TimerPr.Enabled = true;

}

}

if (e.KeyCode == Keys.Escape)

{

TimerPr.Enabled = false;

Form.ActiveForm.Close();

Form.ActiveForm.Dispose();

menu.Show();

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Windows.Forms;

using System.Threading;

namespace tetris

{

public class TimerProg : Form

{

const int CountOfColors = 5;

public int Color;

Random rnd = new Random();

private void ElementCreate(int Elem,ref tetride[,] map,ref int CornI,ref int CornJ) // инициализация элементов

{

int Color;

CornI = 0;

CornJ = 3;

Color = rnd.Next(CountOfColors) + 1;

switch (Elem)

{

case 0: // OOOO

map[0, 3].field = 2;

map[0, 4].field = 2;

map[0, 5].field = 2;

map[0, 6].field = 2;

map[0, 3].color = Color;

map[0, 4].color = Color;

map[0, 5].color = Color;

map[0, 6].color = Color;

break;

case 1: // OO

map[0, 3].field = 2; // OO

map[0, 4].field = 2;

map[1, 3].field = 2;

map[1, 4].field = 2;

map[0, 3].color = Color; // OO

map[0, 4].color = Color;

map[1, 3].color = Color;

map[1, 4].color = Color;

break;

case 2: // OOO

map[0, 3].field = 2; // O

map[0, 4].field = 2;

map[0, 5].field = 2;

map[1, 4].field = 2;

map[0, 3].color = Color; // O

map[0, 4].color = Color;

map[0, 5].color = Color;

map[1, 4].color = Color;

break;

case 3: // O

map[0, 3].field = 2; // OO

map[1, 3].field = 2; // O

map[1, 4].field = 2;

map[2, 4].field = 2;

map[0, 3].color = Color; // OO

map[1, 3].color = Color; // O

map[1, 4].color = Color;

map[2, 4].color = Color;

break;

case 4: // O

map[0, 4].field = 2; // OO

map[1, 3].field = 2; // O

map[1, 4].field = 2;

map[2, 3].field = 2;

map[0, 4].color = Color; // OO

map[1, 3].color = Color; // O

map[1, 4].color = Color;

map[2, 3].color = Color;

break;

case 5: // OOO

map[0, 3].field = 2; // O

map[0, 4].field = 2;

map[0, 5].field = 2;

map[1, 3].field = 2;

map[0, 3].color = Color; // O

map[0, 4].color = Color;

map[0, 5].color = Color;

map[1, 3].color = Color;

break;

case 6: // OOO

map[0, 3].field = 2; // O

map[0, 4].field = 2;

map[0, 5].field = 2;

map[1, 5].field = 2;

map[0, 3].color = Color; // O

map[0, 4].color = Color;

map[0, 5].color = Color;

map[1, 5].color = Color;

break;

}

}

private void NextFigureInitialize(ref int[,] NextFigArr,int NextFigure)

{

int i, j;

for (i = 0; i < NextFigArr.GetLength(0); i++)

for (j = 0; j < NextFigArr.GetLength(1); j++)

NextFigArr[i, j] = 0;

switch (NextFigure)

{

case 0:

NextFigArr[0, 1] = 2;

NextFigArr[1, 1] = 2;

NextFigArr[2, 1] = 2;

NextFigArr[3, 1] = 2;

break;

case 1:

NextFigArr[0, 0] = 2;

NextFigArr[0, 1] = 2;

NextFigArr[1, 0] = 2;

NextFigArr[1, 1] = 2;

break;

case 2:

NextFigArr[0, 0] = 2;

NextFigArr[0, 1] = 2;

NextFigArr[0, 2] = 2;

NextFigArr[1, 1] = 2;

break;

case 3:

NextFigArr[0, 0] = 2;

NextFigArr[1, 0] = 2;

NextFigArr[1, 1] = 2;

NextFigArr[2, 1] = 2;

break;

case 4:

NextFigArr[0, 1] = 2;

NextFigArr[1, 1] = 2;

NextFigArr[1, 0] = 2;

NextFigArr[2, 0] = 2;

break;

case 5:

NextFigArr[0, 0] = 2;

NextFigArr[0, 1] = 2;

NextFigArr[0, 2] = 2;

NextFigArr[1, 0] = 2;

break;

case 6:

NextFigArr[0, 0] = 2;

NextFigArr[0, 1] = 2;

NextFigArr[0, 2] = 2;

NextFigArr[1, 2] = 2;

break;

}

}

private void ElementRemoveDown(ref tetride[,] map,ref int CornI,ref int CornJ) // сдвиг элемента вниз на 1 клетку

{

int i, j;

tetride temp;

//temp.field = 0;

//temp.corner = false;

for (i = map.GetLength(0) - 1; i >= 1; i--)

{

for (j = 0; j < map.GetLength(1); j++)

{

if (map[i - 1, j].field == 2)

{

temp = map[i,j];

map[i, j] = map[i - 1, j];

map[i-1,j] = temp;

}

}

}

CornI = CornI + 1;

}

private void DeleteOneStr(ref tetride[,] map, int pos) // удаление одной строки

{

int i, j;

for (j = 0; j < map.GetLength(1); j++)

{

map[pos, j].field = 0;

map[pos, j].color = 0;

}

for (i = pos; i >= 1; i--)

{

for (j = 0; j < map.GetLength(1); j++)

{

map[i, j].field = map[i - 1, j].field;

map[i, j].color = map[i - 1, j].color;

}

}

}

private void DeleteStrings(ref tetride[,] map,ref int scores) // удаление строк

{

int i, j;

bool flag;

for (i = 0; i < map.GetLength(0); i++)

{

flag = true;

for (j = 0; j < map.GetLength(1); j++)

{

if (map[i, j].field == 0)

flag = false;

}

if (flag)

{

DeleteOneStr(ref map, i);

scores = scores + 10;

}

}

}

private void RechangeMap(ref tetride[,] map) // замена элемента на кучу

{

int i, j;

for (i = 0; i < map.GetLength(0); i++)

{

for (j = 0; j < map.GetLength(1); j++)

{

if (map[i, j].field == 2)

{

map[i, j].field = 1;

}

}

}

}

private bool CrashGround(tetride[,] map) // проверка снизу элемента на преграда

{

bool flag = true;

int i, j;

for (j = 0; j < map.GetLength(1); j++)

{

if (map[19, j].field == 2)

flag = false;

}

for (i = map.GetLength(0) - 2; i >= 0; i--)

{

for (j = 0; j < map.GetLength(1); j++)

{

if (map[i, j].field == 2)

{

if (map[i + 1, j].field == 1)

{

flag = false;

}

}

}

}

return flag;

}

private bool CrashleftWallsBlocks(tetride[,] map) // проверка слева элемента на приграды

{

bool flag = true;

int i,j;

for (i = 0; i < map.GetLength(0); i++)

{

if (map[i, 0].field == 2)

flag = false;

}

for (i = 0; i < map.GetLength(0); i++)

{

for (j = 1; j < map.GetLength(1) - 1; j++)

{

if (map[i, j].field == 2)

{

if (map[i,j-1].field == 1)

{

flag = false;

}

}

}

}

return flag;

}

private bool CrashRotateWallsLeft(tetride[,] map)

{

int i, j;

for (i=0;i<map.GetLength(0);i++)

{

if (map[i, 0].field == 2)

return true;

}

return false;

}

private bool CrashRotateWallsRight(tetride[,] map)

{

int i, j;

for (i = 0; i < map.GetLength(0); i++)

{

if (map[i, map.GetLength(1) - 1].field == 2)

return true;

}

return false;

}

private bool CrashRightWallsBlocks(tetride[,] map) // проверка справа элемента на приграды

{

bool flag = true;

int i,j;

for (i = 0; i < map.GetLength(0); i++)

{

if (map[i, map.GetLength(1) - 1].field == 2)

flag = false;

}

for (i = 0; i < map.GetLength(0); i++)

{

for (j = map.GetLength(1) - 2 ; j >= 0; j--)

{

if (map[i, j].field == 2)

{

if (map[i, j + 1].field == 1)

{

flag = false;

}

}

}

}

return flag;

}

private void ShiftRight(ref tetride[,] tempshape, int posshift) // сдвиг вправо

{

int i, j;

tetride temp;

while (posshift > 0)

{

for (j = tempshape.GetLength(1) - 1; j > 0; j--)

{

for (i = 0; i < tempshape.GetLength(0); i++)

{

if (tempshape[i, j - 1].field == 2)

{

temp = tempshape[i, j];

tempshape[i, j] = tempshape[i, j - 1];

tempshape[i, j - 1] = temp;

}

}

}

posshift--;

}

}

private void ShiftLeft(ref tetride[,] tempshape, int posshift) // сдвиг влево

{

int i, j;

tetride temp;

while (posshift > 0)

{

for (j = 0; j < tempshape.GetLength(1) - 1; j++)

{

for (i = 0; i < tempshape.GetLength(0); i++)

{

if (tempshape[i, j + 1].field == 2)

{

temp = tempshape[i, j];

tempshape[i, j] = tempshape[i, j + 1];

tempshape[i, j + 1] = temp;

}

}

}

posshift--;

}

}

private void ArrCopy(ref tetride[,] tempshape,tetride[,] map) // копирование массива

{

int i,j;

for (i = 0; i < map.GetLength(0); i++)

for (j = 0; j < map.GetLength(1); j++)

tempshape[i, j] = map[i, j];

}

private bool CheckShifting(tetride[,] tempshape, tetride[,] map) // проверка на сдвиг

{

int i, j;

for (i = 0; i < map.GetLength(0); i++)

for (j = 0; j < map.GetLength(1); j++)

if ((tempshape[i, j].field == 2) && (map[i, j].field == 1))

return false;

return true;

}

private void ArrZero(ref tetride[,] tempshape) // обнуление массива

{

int i, j;

for (i=0;i<tempshape.GetLength(0);i++)

for (j = 0; j < tempshape.GetLength(1); j++)

{

tempshape[i, j].field = 0;

tempshape[i, j].color = 0;

}

}

public void ElementRemoveLeft(ref tetride[,] map, ref int CornI, ref int CornJ) // сдвиг элемента влево

{

// проверка на столкновение со стенами и кубиками с боку

int i, j;

tetride temp;

//temp.corner = false;

//temp.field = 0;

if (CrashleftWallsBlocks(map))

{

for (j = 0; j < map.GetLength(1)-1; j++)

{

for (i = 0; i < map.GetLength(0); i++)

{

if (map[i , j + 1].field == 2)

{

temp = map[i, j];

map[i, j] = map[i, j + 1];

map[i, j + 1] = temp;

}

}

}

CornJ = CornJ - 1;

}

}

public void ElementRemoveRight(ref tetride[,] map,ref int CornI,ref int CornJ) // сдвиг элемента вправо

{

// проверка на столкновение со стенами и кубиками сбоку

int i, j;

tetride temp;

//temp.corner = false;

//temp.field = 0;

if (CrashRightWallsBlocks(map)) // если справа нет преград , то сдвигаем

{

for (j = map.GetLength(1) - 1; j > 0; j--)

{

for (i = 0; i < map.GetLength(0); i++)

{

if (map[i, j - 1].field == 2)

{

temp = map[i, j];

map[i, j] = map[i, j - 1];

map[i, j - 1] = temp;

}

}

}

CornJ = CornJ + 1;

}

}

public void Rotate(ref tetride[,] map,ref int CornI,ref int CornJ,int Figure)

{

int i, j, c ,posshift = 0;

bool flag = true,RotFl = true; // flag - показывает разрешён ли поворот

tetride[,] tempshape = new tetride[20,10]; // RotFl - допуск к повороту элемента, находящегося возле стены

if ((CornJ < 0) || (CornJ > 6) || (CornI > 16) ) // проверка выхода CornJ за левую или правую границу массива и поворот в низу стакана

{

if (CornJ < 0) // выход за левую границу массива

{

switch (Figure)

{

case 0:

posshift = 0 - CornJ;

ArrCopy(ref tempshape, map);

ShiftRight(ref tempshape,posshift);

RotFl = CheckShifting(tempshape,map);

if (RotFl)

ShiftRight(ref map, 1);

CornJ = CornJ + posshift;

break;

default:

posshift = 0 - CornJ;

ArrCopy(ref tempshape, map);

ShiftRight(ref tempshape,posshift);

RotFl = CheckShifting(tempshape,map);

if (RotFl)

CornJ = CornJ + posshift;

break;

}

}

else // выход за правую границу массива

{

if (CornJ > 6)

{

switch (Figure)

{

case 0:

posshift = CornJ - 6;

ArrCopy(ref tempshape, map);

ShiftLeft(ref tempshape, posshift);

RotFl = CheckShifting(tempshape, map);

if (RotFl)

//ShiftLeft(ref map, posshift);

CornJ = CornJ - posshift;

break;

default:

posshift = CornJ - 7;

ArrCopy(ref tempshape, map);

ShiftLeft(ref tempshape, posshift);

RotFl = CheckShifting(tempshape, map);

if (RotFl)

CornJ = CornJ - posshift;

break;

}

}

else // отмена поворота внизу стакана

{

switch (Figure)

{

case 0:

if (CornI > 16)

RotFl = false;

break;

default:

if (CornI > 17)

RotFl = false;

break;

}

}

}

}

if (RotFl) // true - допуск к повороту элемента

{

ArrZero(ref tempshape);

switch (Figure)

{

case 0: // поворот палки

if (map[CornI, CornJ].field == 2)

{

for (i = 0; i < 4; i++)

{

tempshape[CornI + i, CornJ + 1].field = 2;

tempshape[CornI + i, CornJ + 1].color = map[CornI,CornJ].color;

}

}

else

{

for (i = 0; i < 4; i++)

{

tempshape[CornI, CornJ + i].field = 2;

if (CornJ ==0)

tempshape[CornI, CornJ + i].color = map[CornI, CornJ + 1 ].color;

else

tempshape[CornI, CornJ + i].color = map[CornI, CornJ + 1 + posshift].color;

}

}

break;

case 1: return; // поворот кубика

default: // поворот всех остальных элементов

for (j = 3 - 1, c = 0; j >= 0; j--, c++)

{

for (i = 0; i < 3; i++)

{

tempshape[CornI + c, CornJ + i] = map[CornI + i, CornJ + j];

}

}

break;

}

for (i = 0; i < map.GetLength(0); i++) // проверка на поворот элемента(когда поворачиваемый элемент оказывается на месте текстуры)

{

for (j = 0; j < map.GetLength(1); j++)

{

if ((tempshape[i, j].field == 0) && (map[i, j].field == 1))

{

tempshape[i, j].field = 1;

tempshape[i, j].color = map[i, j].color;

}

if ((tempshape[i, j].field == 1) && (map[i, j].field == 0)) // если в пробном массиве текстура а на карте пустота , то на пробный помещаем пустоту

{

tempshape[i, j].field = 0;

tempshape[i, j].color = 0;

}

if ((tempshape[i, j].field == 1) && (map[i, j].field == 2)) // если в пробном текстура а на карте элемент который надо повернуть , то в пробный записываем пустоту

{

tempshape[i, j].field = 0;

tempshape[i, j].color = 0;

}

if ((tempshape[i, j].field == 2) && (map[i, j].field == 1)) // если на пробном повёрнутый элемент а на карте текстура , то отменяем запись повёрнутого элемента на карту

flag = false;

}

}

if (flag) // true - записываем повёрнутый элемент в массив

{

switch (Figure)

{

case 0: // запись на карту элемент кубик

for (i = 0; i < 4; i++)

{

for (j = 0; j < 4; j++)

{

map[CornI + i, CornJ + j] = tempshape[CornI + i, CornJ + j];

}

}

break;

default: // запись на карту других элементов

for (i = 0; i < 3; i++)

{

for (j = 0; j < 3; j++)

{

map[CornI + i, CornJ + j] = tempshape[CornI + i, CornJ + j];

}

}

break;

}

}

}

}

public void Tick(ref tetride[,] map, ref bool ground, ref bool finish, ref int scores,ref int CornI,ref int CornJ,ref int Figure,ref int FigureNext,ref int[,] NextFigArr) // вызывается таймером. опускает кубик вниз

{

// проверка на столкновение с элемента с низом и с кучой , если сталкивается , то кубик фиксируется

if ( ground == true)

{

Figure = FigureNext;

FigureNext = rnd.Next(7);

NextFigureInitialize(ref NextFigArr, FigureNext);

ElementCreate(Figure, ref map, ref CornI, ref CornJ); // создание элемента,если предыдущий упал

if (CrashGround(map))

ground = false;

else

{

finish = true;

}

}

else

{

if (CrashGround(map)) // если ниже элемента нет земли или другого элемента , то сдвигаем его вниз на 1 клетку

ElementRemoveDown(ref map,ref CornI,ref CornJ);

else

{

RechangeMap(ref map); // иначе фиусируем движущийся кубик как стенку и проверяем стену на целые строки затем удаляем их

DeleteStrings(ref map,ref scores);

ground = true;

}

}

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace tetris

{

public partial class Menu : Form

{

static Menu menu = new Menu();

static Game game = new Game(menu);

static Options options = new Options(menu);

public bool EasyMod;

public bool MediumMod;

public bool HardMod;

public Menu()

{

EasyMod = false;

MediumMod = true;

HardMod = false;

InitializeComponent();

}

private void Start\_Game\_Click(object sender, EventArgs e)

{

Game game = new Game(menu);

Form.ActiveForm.Hide();

game.ShowDialog();

}

private void Menu\_Load(object sender, EventArgs e)

{

}

private void ButOptions\_Click(object sender, EventArgs e)

{

Form.ActiveForm.Hide();

options.ShowDialog(menu);

}

private void Exit\_Click(object sender, EventArgs e)

{

Application.Exit();

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace tetris

{

public partial class Options : Form

{

static Menu menu = new Menu();

public Options(Menu menustr)

{

menu = menustr;

InitializeComponent();

}

private void Options\_Load(object sender, EventArgs e)

{

RbEasy.Checked = menu.EasyMod;

RbMedium.Checked = menu.MediumMod;

RbHard.Checked = menu.HardMod;

}

private void ButBackOptions\_Click(object sender, EventArgs e)

{

menu.EasyMod = RbEasy.Checked;

menu.MediumMod = RbMedium.Checked;

menu.HardMod = RbHard.Checked;

Form.ActiveForm.Close();

menu.Show();

}

}

}