Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Языки программирования(ЯП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

ИГРОВОЕ ПРИЛОЖЕНИЕ “SNAKE”

Студент: гр. 551006 Жизневский В.С.

Руководитель: Марина И. М.

Минск 2017

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc469528394)

[1 СРАВНЕНИЕ АНАЛОГОВ 8](#_Toc469528395)

[1.1 Slither.io 8](#_Toc469528396)

[1.2 Snake Rewind 9](#_Toc469528397)

[2 ПОСТАНОВКА ЗАДАЧИ 11](#_Toc469528398)

[3 ПРОЕКТИРОВАНИЕ. РАЗРАБОТКА 12](#_Toc469528399)

[3.1 Описание ресурсов 12](#_Toc469528400)

[3.1.1 C++ 12](#_Toc469528401)

[3.1.2 Microsoft Visual Studio 12](#_Toc469528402)

[3.1.3 Bass.dll 13](#_Toc469528403)

[3.1.4 openGL. Freeglut 13](#_Toc469528404)

[3.2 Структура приложения 14](#_Toc469528405)

[3.3 Основной игровой процесс 15](#_Toc469528406)

[3.3.1 Класс mainGame 16](#_Toc469528407)

[3.3.2 Класс TSnake 19](#_Toc469528408)

[3.3.3 Класс TFood 21](#_Toc469528409)

[3.4 Музыкальное и звуковое сопровождение игрового процесса 23](#_Toc469528410)

[3.4.1 Класс TSound 23](#_Toc469528411)

[3.5 Уровни сложности 25](#_Toc469528412)

[3.6 Генерация случайных препятствий на игровой сцене 26](#_Toc469528413)

[3.6.1 Корректирование стен 27](#_Toc469528414)

[3.6.2 Класс TWall 28](#_Toc469528415)

[3.7 Таблица рекордов 30](#_Toc469528416)

[3.7.1 Сортировка 31](#_Toc469528417)

[3.7.2 Обновление рекорда 32](#_Toc469528418)

[3.8 Управление игровым процессом 32](#_Toc469528419)

[3.9 Идентификация игрока 33](#_Toc469528420)

[4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 34](#_Toc469528421)

[4.1 Системные требования 34](#_Toc469528422)

[4.2 Запуск программы. Начало игры 34](#_Toc469528423)

[4.3 Правила игры 34](#_Toc469528424)

[4.4 Управление 34](#_Toc469528425)

[ЗАКЛЮЧЕНИЕ 35](#_Toc469528426)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 36](#_Toc469528427)

[ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ ТЕКСТ ПРОГРАММЫ 37](#_Toc469528428)

# **ВВЕДЕНИЕ**

В современной практике образования достаточно большое значение имеет время, затрачиваемое на выполнение типичных процессов, таких как проведение самостоятельных и контрольных работ, проверка посещаемости и ведение журнала с состоянием сдачи лабораторных работ.   
Данные процедуры — основная часть документационной работы всех преподавателей университета. Так как такие мероприятия необходимы для корректной организации процесса создания систем и техник образования, однако большая часть из них, такие как самостоятельные работы, могут отнимать достаточно большие объёмы времени, так как помимо самого проведения самостоятельных работ, необходимо также в дальнейшем проверить написанные студентами ответы на предмет корректности.   
 В современном мире каждая область нашей жизни проходит процесс перехода к использованию программных и аппаратных средств. Яркими примерами процесса перехода к автоматизированным процессам могут служить такие службы и учреждения, как банки, общественный транспорт, дизайн и т.д., что свидетельствует о широком распространении использования информационных технологий.   
 В процессе получения образования в учреждении образования БГУиР довольной низкий процент имеет использование тех или иных средств автоматизации, соответственно, процессы проведения и организации образовательных мероприятий затрачивает достаточно большой объём времени.   
 Целью данной курсовой работы является создание программного средства, которое будет обладать простотой в использовании, и которое будет способствовать автоматизации образовательных процессов, а также будет иметь возможность дальнейшего переиспользования.   
 Программное средство будет состоять из нескольких частей:

* мобильное приложение для студентов, в котором будут находиться работы, которые студенты могут пройти;
* RESTful API ( в дальнейшем, API ), предоставляемый для использования остальным частям программного средства;
* веб-приложение, которое будет служить для создания шаблонов для самостоятельных работ.

В данной части курсового проекта будет рассмотрено создание мобильного приложения.

# **СРАВНЕНИЕ АНАЛОГОВ**

Особенностями мобильных приложений систем опроса является то, что практически во все они предназначены для создания опросов, и очень мало мобильных клиетнов имеют в себе возможность прохождения данных опросов. Также все они предназначены для решения большого круга задач, и мало из низ заточены под какую-либо предметную область.

## **Surveymonkey**

Данная платформа включает в себ как сайт, так и клиент для мобильных устройств (рисунок 1.1). В частности мобильный клиент позволяет создавать опросы, рассылать по электронной почте, в SMS-сообщениях или через социальные сети. Также мобильный клиент позволяет просматривать результаты опроса в виде всяких графиков и диаграмм.

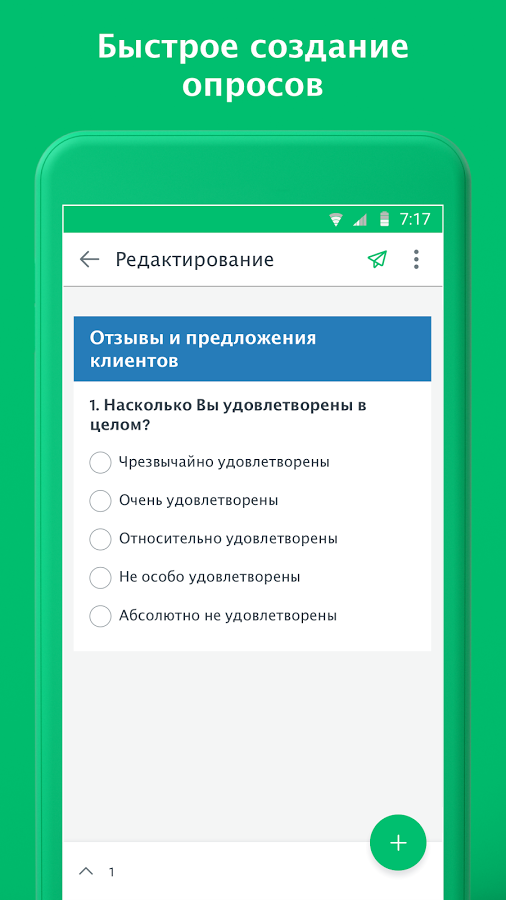


Рисунок 1.1 – Surveymonkey.

Однако данное приложение имеет лишь часть функционала, которое возможно будет реализовано в дальнейшем в последующих версиях курсового проекта, а именно просмотр статистики пользователя. Возможность прохождения опросов данное приложение не имеет.

## **Simpoll**

Данный клиент позволяет создавать вопросы, рассылать данные вопросы и вести статистику ответов (рисунок 1.2).

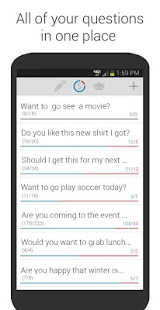
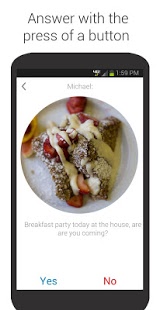


Рисунок 1.2 – Simpoll.

Недостатком данного клиента является отсутствие возможности группировки вопросов в опросы.

На основе анализа аналогов было решено реализовать мобильный клиент для системы автоматизации прохождения опросов.

# **ПОСТАНОВКА ЗАДАЧИ**

Разработать клиент-серверное игровое приложение “Bomberman”, реализующее основные механики одноименной игры, необходимые для игры друг против друга. Приложение будет состоять из клиента и сервера.

Сервер игрового приложения “Bomberman” должен реализовывать следующий функционал:

* основной игровой процесс;
* генерация карты;
* просчёт состояиня карты;
* прослушивание входящих подключений;
* идентификация клиента на сервере;
* обработка сообщений о действиях клиентов;
* рассылку текущего сострояния карты с целью дальнейшей отрисовки его клиентом.

Клиент игрового приложения “Bomberman” должен реализовывать следующий функционал:

* идентификация пользователя;
* подключение к серверу;
* получение состояния карты от сервера;
* отрисовка состояния карты;
* управление и отправка серверу сообщений о действиях пользователя.

# **ПРОЕКТИРОВАНИЕ. РАЗРАБОТКА**

## **Описание ресурсов**

Разработка данного приложения будет происходить в IDE Microsoft Visual Studio. Основным языком будет C#, а также будет использоваться язык разметки XAML. Основным фреймворком, на котором будет вестись разработка приложения, будет Xamarin, базируемый на платформе Mono. Также будут использоваться фрэймворки: Xamarin Forms, Prism, XLabs.

# C#

**C#** — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework и впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников — языков C++, Pascal, и, в особенности, Java — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественное наследование интерфейсов).

# Microsoft Visual Studio

Microsoft Visual Studio — продукт компании Microsoft, включающий интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данный продукт позволяет разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

# Xamarin

Xamarin — это фреймворк для кроссплатформенной разработки мобильных приложений (iOS, Android, Windows Phone) с использованием языка C#. Xamarin позволяет писать с применением всех основных возможностей С# и платформы .Net, наприер LINQ, лямбда-выражения, Generic типы и async. При имеется полный доступ ко всем возможностям SDK платформы и нативному механизму создания UI, получая на выходе приложение, которое, практически ничем не отличается от нативных.  
  
 Фреймворк состоит из нескольких основных частей:

* Xamarin.IOS — библиотека классов для C#, предоставляющая разработчику доступ к iOS SDK;
* Xamarin.Android — библиотека классов для C#, предоставляющая разработчику доступ к Android SDK;
* компиляторы для iOS и Android;
* IDE Xamarin Studio;
* плагин для Visual Studio.

Xamarin основан на open-source реализации платформы .NET — Mono. Эта реализация включает в себя собственный компилятор C#, среду выполнения, а так же основные .NET библиотеки. Основное назначение — позволить запускать программы, написанные на C#, на операционных системах, отличных от ОС Windows — Unix-системах, операционной системы Mac OS и других. Разработчиками Mono и Xamarin является компания Xamarin.

Xamarin содержит в себе компиляторы для ОС Android и операционной системы iOS, которые имеют разные принципы работы. С точки зрения исполнения приложений между ОС iOS и ОС Android есть одно ключевое различие — способ их предварительной компиляции. Для выполнения приложений в Android используется виртуальная Java-машина Dalvik. Нативные приложения, которые пишутся на Java, компилируются в некий промежуточный байт-код, который интерпретируется Dalvik`ом в команды процессора в момент исполнения программы. Это так называемая Just-in-time компиляция. В iOS используется другая модель компиляции — Ahead-of-Time (рисунок 3.1).

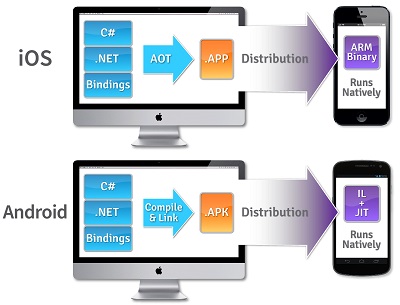


Рисунок 3.1 - Схема создания и функционирования приложений.

В случае ОС iOS принцип выполнения следующий — программный код заранее скомпилирован в машинный. Для этой цели используется AOT компилятор Mono.

Принцип выполения в ОС Android другой. При компиляции приложения происходит перевод кода на C# в промежуточный байт-код, понятный виртуальной машине Mono и сама эта виртуальная машина также добавляется в упакованное приложение. И Mono и Dalvik написаны на языке Си и работают поверх ядра ОС Linux. При запуске приложения на ОС Android обе виртуальные машины начинают работать бок о бок и обмениваются данными через специальный механизм wrapper`ов (рисунок 3.2).

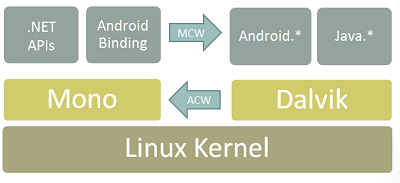
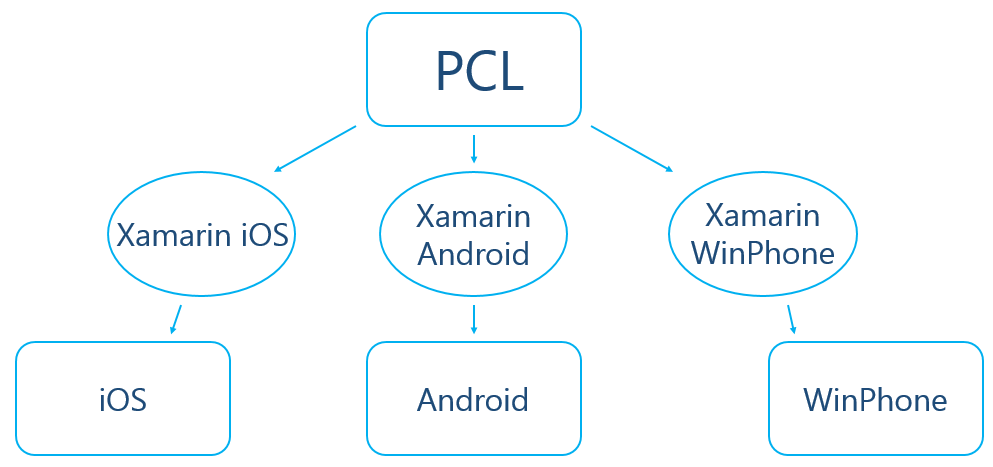


Рисунок 3.2 - Схема взаимодействия Mono и Dalvik.

# Xamarin Forms

Xamarin.Forms — позволяет использовать одну библиотеку элементов (текстовые поля, кнопки, переключатели, надписи, списочные представления и т.д.) для всех систем (Android, iOS, Windows).   
 Xamarin.Forms позволяет создавать элементы в Xamarin приложении, а на конкретной платформе(iOS или Android или Windows) каждого вашего элемента вызывается генерация и рисуются родные элементы(на iOS свои кнопки, для Android свои кнопки и т.д.).   
 Добавлять элементы можно как в Xamarin c# коде так и в XAML-формате.

Структура приложения Xamarin.Forms представлена на рисунке 3.3.



Русунок 3.3 – Структура приложения Xamarin.Forms.

Вверху схемы расположена PCL(Portable class library. По сути это и есть Xamarin.Forms. В общих чертах он представляет собой набор редакторов, навигационных панелей, layout-панелей и так далее. При разработке пользовательского интерфейса большую часть времени работаешь как раз с ними. Однако, данные контролы — это всего лишь абстракция внутри PCL части. Чтобы они смогли как-то отобразиться на устройстве, существуют так называемые рендереры. Располагаются они на следующей ступени иерархии в Xamarin Platform частях.  
 Под PCL частью у нас расположены Xamarin.iOS, Xamarin.Android и Xamarin.WinPhone. Это по сути и есть тот Xamarin, который уже существовал до Xamarin.Forms. Xamarin.iOS, Xamarin.Android и Xamarin.WinPhone содержат в себе C# обертки над нативными классами для каждой платформы. Так вот рендереры — это обертки над соответствующими визуальными компонентами, но которые дополнительно внутри себя содержат ссылки на PCL объекты, умеют читать у них выставленные свойства и применять их у себя.  
 В дальнейшем эти рендереры уже разворачиваются в нативные контролы, которые находятся в модулях, расположенных на нижних уровнях схемы.

Преимущества:

* использование в качетве языка C# и Mono;
* подход к созданию и работе с пользовательским интерфейсом близок WPF. Xamarin Forms поддерживает работу с XAML, биндинги, темплейты, стили и прочие концепции WPF. Однако они несколько урезаны по сравнению с WPF;
* из того, что Xamarin.Forms схож с WPF, вытекает следующий плюс этой платформы: MVVM(Model View ViewModel). Xamarin.Forms имеет XAML, визуальные элементы имеют BindingContext (аналог DataContext в WPF), есть BindableProperty (аналог DependencyProperty). Таким образом, можно связывать View с ViewModel аналогично тому, как в WPF;
* еще одно преимущество данной платформы в том, что так как UI описывается только в одном месте, то приложения под разными системами будут выглядеть очень похоже. Что может быть важно, например, в корпоративных разработках.

Недостатки:

* неполная реализация функционала WPF;
* различное поведение на разных платформах;
* производительность.

# Xlabs

Xamarin Forms Labs (XLabs) - библиотека с открытым исходным кодом, целью которого является создание мощного и кросс-платформенного набора элементов управления, адаптированных для работы с Xamarin Forms.

# Prism

Данная библиотека предназначена для упрощения реализации патттерна MVVM в приложении для XAML-based платформ, таких как WPF, Silverlight, Windows Phone, или Windows Store, Xamarin.Forms. Также библиотека предоставляет другие сервисы, как Container, Navigation servise т.д.

# MVVM

Паттерн MVVM (Model-View-ViewModel) позволяет отделить логику приложения от визуальной части (представления). Данный паттерн является архитектурным, то есть он задает общую архитектуру приложения.

Данный паттерн был представлен Джоном Госсманом (John Gossman) в 2005 году как модификация шаблона Presentation Model и был первоначально нацелен на разработку приложений в WPF. И хотя сейчас данный паттерн вышел за пределы WPF и применяется в самых различных технологиях, в том числе при разработке под Android, iOS, тем не менее WPF является довольно показательной технологией, которая раскрывает возможности данного паттерна.

MVVM состоит из трех компонентов (рисунок 3.4): модели (Model), модели представления (ViewModel) и представления (View).

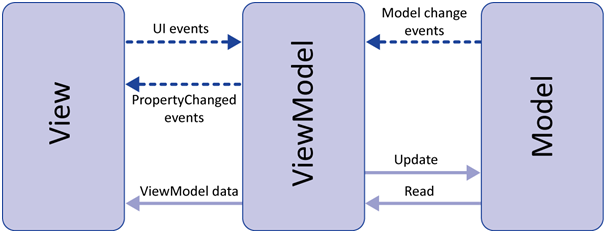


Рисунок 3.4 – схема паттерна MVVM.

Модель (Model) описывает используемые в приложении данные. Модели могут содержать логику, непосредственно связанную этими данными, например, логику валидации свойств модели. В то же время модель не должна содержать никакой логики, связанной с отображением данных и взаимодействием с визуальными элементами управления.

Нередко модель реализует интерфейсы INotifyPropertyChanged или INotifyCollectionChanged, которые позволяют уведомлять систему об изменениях свойств модели. Благодаря этому облегчается привязка к представлению, хотя опять же прямое взаимодействие между моделью и представлением отсутствует.

View или представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Применительно к WPF представление - это код в xaml, который определяет интерфейс в виде кнопок, текстовых полей и прочих визуальных элементов.

Хотя визуальный элемент в Xamarin.Forms может содержать как интерфейс в xaml, так и привязанный к нему код C#, однако в идеале код C# не должен содержать какой-то логики, кроме разве что конструктора, который вызывает метод InitializeComponent и выполняет начальную инициализацию окна. Вся же основная логика приложения выносится в компонент ViewModel.

Однако иногда в файле связанного кода все может находиться некоторая логика, которую трудно реализовать в рамках паттерна MVVM во ViewModel.

Представление не обрабатывает события за редким исключением, а выполняет действия в основном посредством команд.

ViewModel или модель представления связывает модель и представление через механизм привязки данных. Если в модели изменяются значения свойств, при реализации моделью интерфейса INotifyPropertyChanged автоматически идет изменение отображаемых данных в представлении, хотя напрямую модель и представление не связаны.

ViewModel также содержит логику по получению данных из модели, которые потом передаются в представление. И также VewModel определяет логику по обновлению данных в модели.

Поскольку элементы представления, то есть визуальные компоненты типа кнопок, не используют события, то представление взаимодействует с ViewModel посредством команд.

Например, пользователь хочет сохранить введенные в текстовое поле данные. Он нажимает на кнопку и тем самым отправляет команду во ViewModel. А ViewModel уже получает переданные данные и в соответствии с ними обновляет модель.

Итогом применения паттерна MVVM является функциональное разделение приложения на три компонента, которые проще разрабатывать и тестировать, а также в дальнейшем модифицировать и поддерживать.

## **Структура приложения**

Данная система, часть которой будет являться мобильный клиент, будет состоять из сайта insert: used languages and frameoworks, клиента бля мобильного телефона (Xamarin + Xamarin.Forms) и сервера(ASP.NET MVC?).

В рамках данной курсовой работы будет рассмотрена разработка клиента для мобильных платформ.

Данное приложение будет состоять из следеющих модулей:

* CourceProject.MC;
* CourceProject.MC.UI;
* CourceProject.MC.DomainModel;
* CourceProject.MC.Common;
* CourceProject.MC.Resources;
* CourceProject.MC.Foundation;
* CourceProject.MC.Android;
* CourceProject.MC.IOS.

CourceProject.MC

Является стартововй точкой приложения. Содержит в себе начальную инициализацию приложения, регистрацию сервисов в контейнере, регистрацию страниц в сервисе навигаций.

CourceProject.MC.UI

Содержит в себе часть, отвечающую за UI: страницы, связанные с ними ViewModel, конструктор элементов управления, собственные елементы управления, расширяющие базовые. Также содержит модуль, отвечающий за создание представления для опроса и привязки этого представления к соответствующим ViewModels.

CourceProject.MC.DomainModel

Содержит в себе модели, свзяванные с опросом и аутентификацией пользователя.

CourceProject.MC.Common

Содержит константы, общие данные, используемые в остальных модулях (например, используемые регулярные выражения и т.д.).

CourceProject.MC.Resources

Содержит общие ресурсы, используемые в остальных модулях (например, надписи приложения, иконки и т.д.).

CourceProject.MC.Foundation

Содержит сервисы, не связанные с UI. Например, сервис, отвечающий за взаимодействие с сервером, получение от него опросов а также аутентификацию пользователя.

CourceProject.MC.Android

Содержит в себе классы, отвечающие за преобразование и рендер страниц и управляющих элементов под платформу Andoid.

CourceProject.MC.IOS

Содержит в себе классы, отвечающие за преобразование и рендер страниц и управляющих элементов под платформу IOS.

## **Модуль CourceProject.MC**

1. Описание функциональности

Как было описано ранее, модуль CourceProject.MC является стартововй точкой приложения, который содержит в себе начальную инициализацию приложения, регистрацию сервисов в контейнере, регистрацию страниц в сервисе навигаций.

1. Контейнер. Dependensy injection. Inversion of control

Todo: write about DI, IoC and container in Prism.

1. Navigation service в Xamarin и Prism

Xamarin Forms предоставляет инструменты для навигации в виде свойства Navigation класса Page. Это свойство предтавляет интерфейс INavigation, в котором есть следующие методы:

Task PushAsync(Page page)

Task PushModalAsync(Page page)

В качестве параметра здесь передается объект Page - страница, на которую надо осуществить переход.

Второй метод имеет в своем названии слово "Modal" и осуществляет переход на модальную страницу.Обычно модальные страницы используются, когда приложению нужно получить некоторую информацию от пользователя. При этом до получения информации нельзя возвращаться на предыдущую страницу.

Также имеются методы для возврата на предыдущую страницу:

Task<Page> PopAsync()

Task<Page> PopModalAsync()

Исходя из вышеописанного, можно сделать вывод, что организация навигации осуществляется по принципу стека LIFO (Last in – first out).

Отсюда вытекают следующие примущества:

* для навигации не нужно запоминать полный адрес страницы.

Недостатки:

* для навигации на несколько уровней назад нужно последовательно сделать несколько операций Pop, усложняестя логика работы навигации страницы, увеличивается связанность страниц.

Навигация в приложении Prism концептуально отличается от стандартной навигации в Xamarin.Forms. В то время как навигационная система Xamarin.Forms полагается на экземпляр класса страницы для навигации, Prism удаляет все зависимости от типов страниц для достижения слабосвязанной навигации из ViewModel. В Prism концепция навигации к View или навигации к ViewModel не существует. Вместо этого переход осуществляется идентификаторам страниц. В стандартном случае используется система идентификации согласно URI.

Преимущества такого подхода:

* Меньшая связанность страниц, позволяет организовывать быструю навигацию мжеду страницами, т.к. каждая страница независима.

Недостатки данного подхода:

* Для навигации используется URI, которые в некоторых случаях могут иметь сложную структуру, отсутствует проверка валидности URI.

Навигация по страницам в Prism осуществляется при помощи службы INavigationService.

Регистрация страницы NaviagtionService в Prism происходит в App.Xaml.cs в методе RegisterTypes при помощи метода класса Container RegisterTypeForNavigation<T>.

Для последующего использования сервиса он передается в конструктор ViewModel или xaml.cs файл Containerом. Навигация происходит при помощи метода NavigateAsync(URI uri).

Также имеется возможность возврата на предыдущую страницу при помощи метода GoBackAsync.

Также удобной возможностью NavigationService в Prism является возможность передачи параметров на страницу при навигации через NavigationParameters, который представляет собой словарь, содержащий элементы типа: ключ – значение.

Таким образом инструменты навигации в Prism позволяют создавать более гибкие системы и взаимосвязи между страницами.

1. Привязка страниц к ViewModel в XamarinForms и Prism

Xamarin.Forms Привязка данных (data binding) является одним из ключевых моментов платформы Xamarin Forms.

Привязка данных состоит из двух компонентов: источника (source) и цели (target). Привязка осуществляется от свойства источника к свойству цели. И когда происходит изменение источника, механизм привязки автоматически обновляет также и цель.

Цель привязки должна представлять объект BindableObject, а свойство, к которому осуществляется привязка, должно быть свойством BindableProperty. Поскольку большинство визуальных элементов в Xamarin Forms наследуются от класса BindableObject, то в качестве цели привязки будут, как правило, выступать визуальные элементы.

А вот источником привязки может выступать любой объект языка C#. Однако, надо понимать, что цель привязки должна автоматически изменяться при изменении источника, поэтому нам нужно извещать систему о изменении свойств источника привязки. В Xamarin, да и вообще на платформе .NET, в качестве подобного механизма извещения выступает интерфейс INotifyPropertyChanged. То есть нужно реализовать данный интерфейс в объекте-источнике.

Объект BindableObject как раз реализует INotifyPropertyChanged. Поэтому если источником привязки является стандартный визуальный элемент из Xamarin Forms, то автоматически будет изменяться и цель привязки. Но если в качестве источника выступает не BindableObject, а какой-нибудь объект простого класса C#, то, как писалось выше, этот класс должен реализовать INotifyPropertyChanged.

Для установки привязки у объекта цели устанавливается свойство BindingContext. В качестве значения оно принимает источник привязки.

Привязку можно устанавливать как в XAML, так и cs файлах.

Однако Prism предоставляет более удобные инструменты для привязки, основанные на механизмах привязки в Xaramin. Также в Prism имеется возможноть привязки не только к свойствам, но и к событиям при помощи механизма комманд.

В основном последовательность привязки страницы к ViewModel в Prism состоит в следующем:

1. Создание страницы (xaml + xaml.cs файлы);
2. Создание соответствующей данной страницы ViewModel;
3. Регистрация в классе ViewModelLocationProvider зависимости при помощи метода Register<Page, ViewModel>. Данное действие предпочтительно производить в методе RegisterTypes класса App файла App.xaml.cs;
4. Далее добавить свойство: prism:ViewModelLocator.AutowireViewModel="True" к тэгу страницы в xaml файле.

После выполнения данных действий можно устанавливать привязку как в xaml файле, например Text=”{Binding TextPropertyViewModel, Mode=OneWayToSource}”, так и в xaml.cs файле при помощи метода SetBinding, например SetBinding(TextProperty, nameof(ViewModel. TextPropertyInViewModel), Mode = TwoWay).

В качестве Mode может выступать одно из значений перечисления BindingMode:

* OneWay. Указывает, что изменения должны распространяться только от источника (обычно View Model) до целевого объекта (BindableObject). Это режим по умолчанию для большинства значений BindableProperty;
* OneWayToSource. Указывает, что изменения должны распространяться только от целевого объекта (BindableObject) к источнику (обычно это View Model);
* TwoWay. Изменения распространяются в оба направления;
* Default. Значение по умолчанию, обычно равно OneWay.

Также Xamarin.Forms и Prism имеют в наличии механизмы комманд.

Механизм комманд позволяет избежать описания обработчиков событий в классе визуального компонента, таким образом достигается большее соответствие паттерну MVVM. Т.е. логика обработчиков выносится в ViewModel.

Т.о. механизм привязки позволяет реализовать паттерн MVVM. Т.о. образом отделяется представление от логики и данных, что обесбечивает большую взаимозаменяемость модулей и частей приложения а также увеличивает повторную используемость кода.

В рамках данной курсовой работы будут использованы инструменты, предоставляемые библиотекой Prism.

## **CourceProject.MC.DomainModel**

1. Описание функциолаьности

## Модуль CourceProject.MC.DomainModelв себе модели, свзяванные с опросом и аутентификацией пользователя.

1. Модель опроса

Данная модель представлена классом Survey. Опрос содержит в себе Id опроса, заголовок, Id создателя, время создания, время последней модификации, анонимный ли опрос, случайный ли порядок вопросов, показывается ли количество вопросов, показывать ли прогресс выполнения теста, помечать ли обязательные поля, и коллекцию страниц (рисунок 3.5).

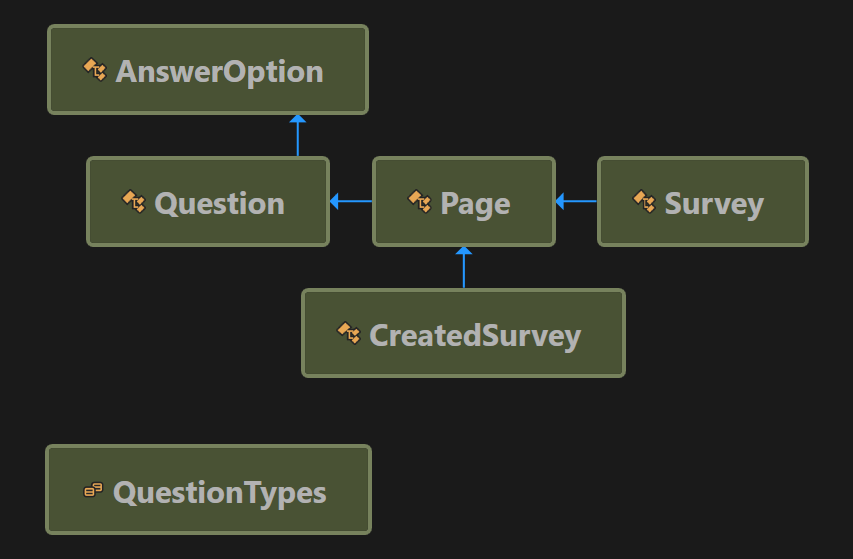


Рисунок 3.5 – Диаграмма модели опроса.

В свою очередь модель страницы представлена классом Page. Модель содержит в себе Id страницы, заголовок страницы и коллекцию вопросов.

Модель вопроса представлена классом Question. Модель содержит в себе Id вопроса, его тип, обязателен ли вопрос и коллекцию ответов. Тип представлен идексом, подробнее обозначение индексов содержится в перечислении QuestionTypes.

Типы вопроса зависят от вида управляющего элемента, предназначенного для ответа. Поддерживаются следующие типы:

* SingleLine(однострочное поле для ввода);
* MultiLine(многострочное поле для ввода);
* Date(выбор даты);
* RadioButtonGroup(связанная группа RadioButtonов);
* CheckBoxesGroup(связанная группа CheckBoxов).

Модель ответа представлена классом AnswerOption. Модель содержит в себе Id ответа и текстовое представление ответа.

1. Модель пользователя

Todo: describe User and models, related to authentification.

## **Модуль CourceProject.MC.UI**

1. Описание функциональности

Модуль CourceProject.MC.UI в себе часть, отвечающую за пользовательский интерфейс: страницы, связанные с ними ViewModel, конструктор элементов управления, собственные елементы управления, расширяющие базовые. Также содержит модуль, отвечающий за создание представления для опроса и привязки этого представления к соответствующим ViewModels.

1. Описание страниц

Данная версия приложения содержит в себе 3 страницы: страница аутентификации, представленная классом SingUpPage (рисунок 3.6), страница получения опроса, представленная классом GetSurveyPage (рисунок 3.7) и страница отображения опроса, представленная классом SelectedSurveyPage (рисунок 3.8). Также каждая страница имеет связанная с ней ViewModel: SignUpViewModel, GetSurveyViewModel, SelectedSurveyViewModel соответственно.

Todo: insert SingUpPage Android

Рисунок 3.6 – Страница входа для Android.

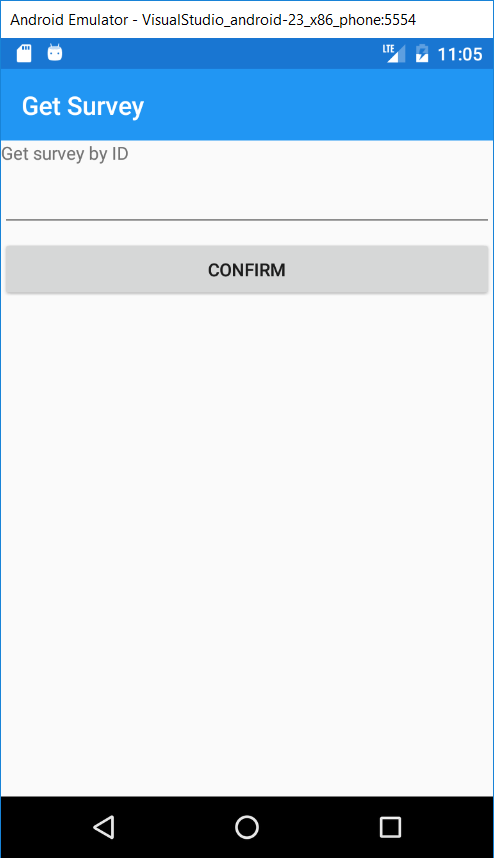


Рисунок 3.7 – Страница получения опроса для Android.

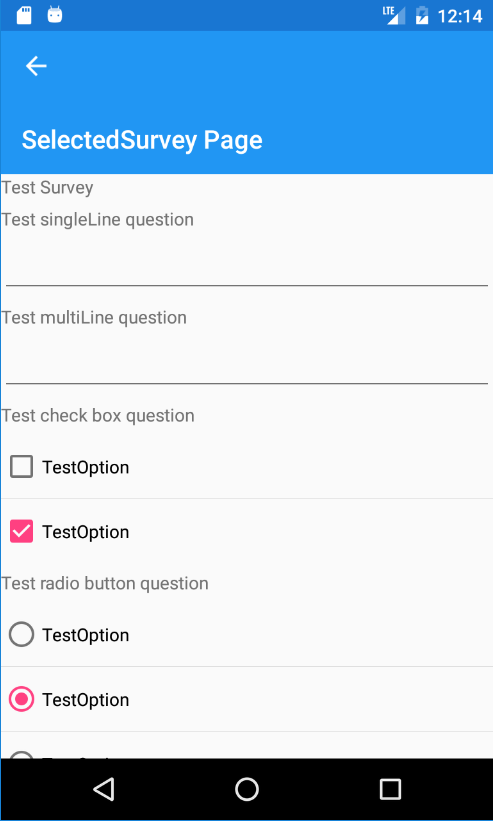


Рисунок 3.8 – Страница опроса для Android.

Далее будут подробно рассмотрены реализации данныз страниц.

1. SingUpPage

Todo: describe SingUpPage and its ViewModel.

1. GetSurveyPage

Данная страница (рисунок 3.7) предназначена для получения опроса от сервера по Id данного опроса. В последующих версиях планируется использование Push-нотификаций для предоставления опросов пользователям.

Данная страница содержит поле для ввода Id, кнопку, по нажанию на которую происходит проверка проверка правильности формата введенного Id и, при корректном Id происходит обращение к серверу для получения опроса с соответствующим Id. Если формат введенного Id является некорректным, то пользователю выводится соответствующее сообщение (рисунок 3.9). При неудачной попытке получения опроса от сервера выводится соответствующее сообщение (рисунок 3.10). При успешном получении опроса идёт переход на страницу опроса (рисунок 3.8).

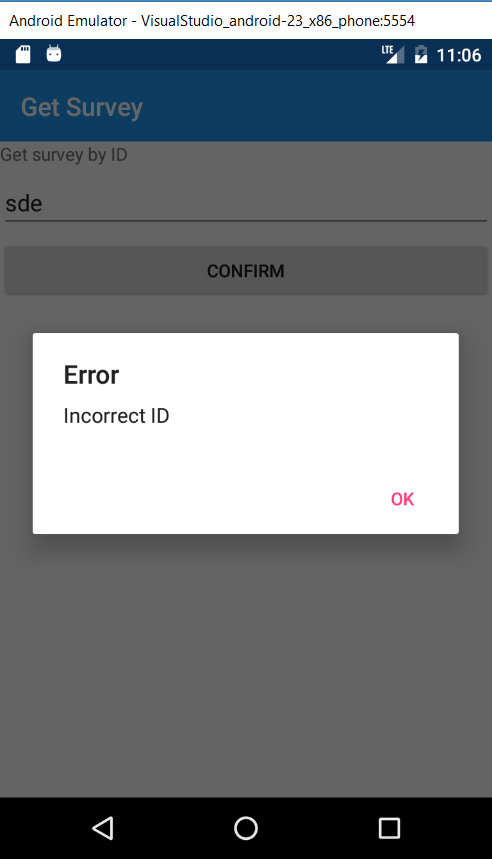


Рисунок 3.9 – Ошибка: “Некорректный Id”, Android.

Todo: error “cannot get survey”, iOS.

Рисунок 3.10 – Ошибка: “Не удалось получить опрос”, Android.

1. SelectedSurveyPage

Данная страница (рисунок 3.8) предназначена для отображения опроса пользователю с целью дальнейшего прохождения.

Данная страница имеет преставление опроса, сконструированное согласно пришедшему от сервера опросу, также данная страница имеет кнопку для отправки пройденного опроса. При попыте отправки невалидного ответа на опрос будут подсвечены соответствующие невалидные ответы (рисунок 3. …).

1. Конструктор представления опроса

При получении опроса со страницы GetSurvey необходимо, исходя из полученного опроса, посроить его представление, также неоходимо организовать изменение модели при вводе информации пользователем в сгенерированные упраляющие элементы.

Для решения данной задачи был разработан следующий механизм (рисунки 3.11, 3.12, 3.13, 3.14):

Сначала достаются все вопросы из модели опроса, далее для каждой модели вопроса создаётся своё представление. При построении предстваления для каждого управляющего элемента создаётся своя ViewModel, которая содержит в себе модель вопроса и связывается с соответствующим упраляющим элементом. Далее идёт связывание полей ввода с соответствующими свойствами ViewModel, при обращении к которым идёт взаимодействие с моделью вопроса, хранимой в ViewModel.

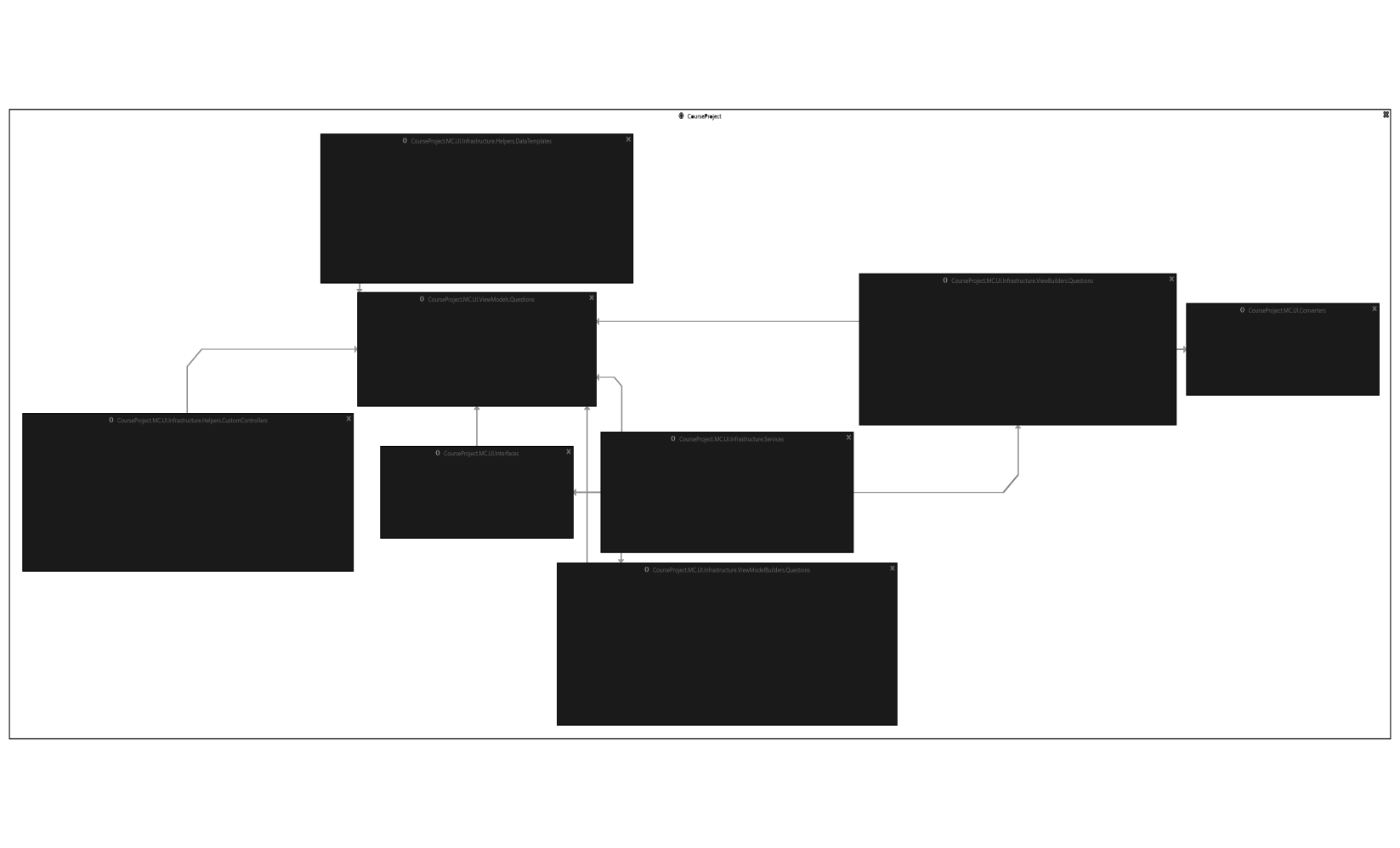


Рисунок 3.11 – Схема взаимосвязи элементов механизма формировния представления опроса.

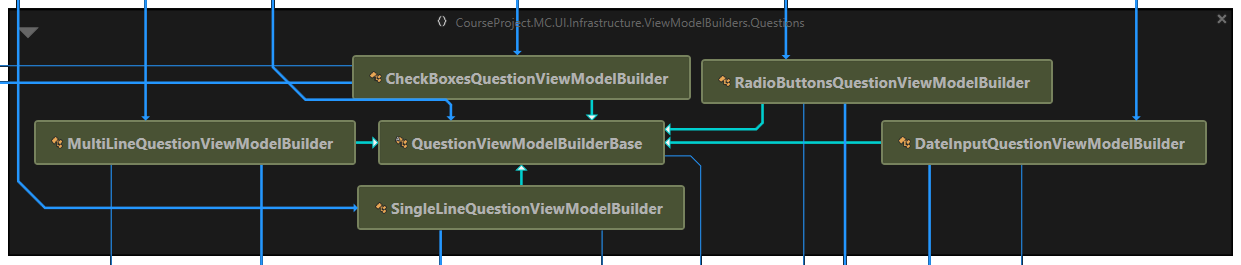


Рисунок 3.12 – Схема взаимосвязи элементов механизма формировния представления опроса. Иерархия ViewModelBuilders.

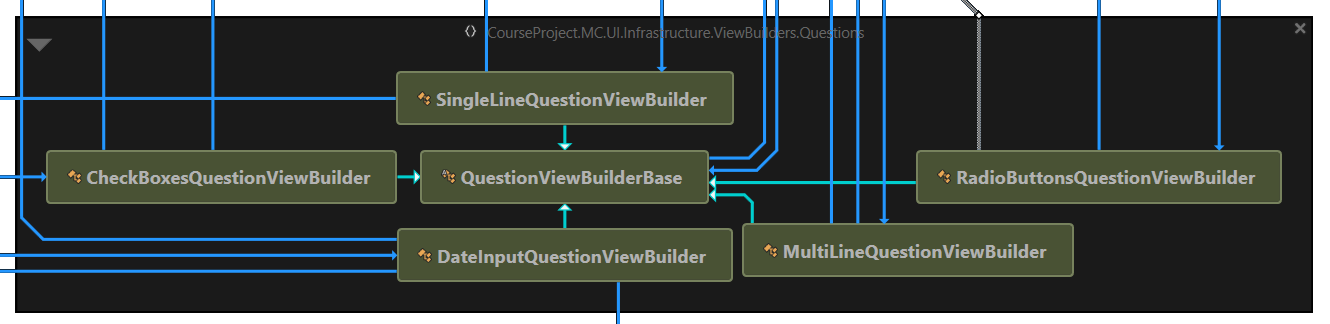


Рисунок 3.13 – Схема взаимосвязи элементов механизма формировния представления опроса. Иерархия ViewBuilders.

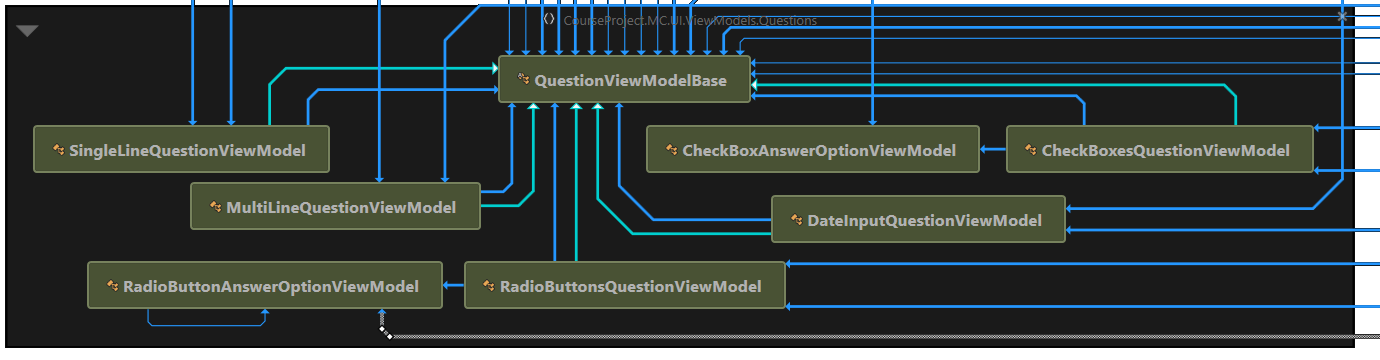


Рисунок 3.14 – Схема взаимосвязи элементов механизма формировния представления опроса. Иерархия ViewModels.

Построение представления для вопроса происходит при помощи класса QuestionViewBuilderService, который реализует интерфейс IQuestionViewBuilderService. Данный класс представляет собой реализацию паттерна “фабрика”.

Создание ViewModel происходит при помощи класса QuestionViewModelBuilder, который реализует интферфейс IQuestionViewModelBuilder. Данный класс представляет собой реализацию паттерна “фабрика”.

Далее будут рассмотрены принципы создания представления и ViewModel для каждогот типа вопросов.

1. Базовые классы QuestionViewBuilderBase и QuestionViewModelBase

Класс QuestionViewBuilderBase представляет собой класс, который является базовым для всех ViewBuilders для вопросов. Данный класс содержит метод создания представления, куда передаётся ViewModel, к которой будет происходить привязка.

Класс QuestionViewModelBase представляет собой класс, который является базовым для всех Models для вопросов. Данный класс, содержит себе модель вопроса, который пришёл от сервера. Также он содержит свойство ответа, через которое будет изменяться модель ответа, которая расположена в модели вопроса. Также он содерижит остальные свойства, через которые будет осущенствляться взаимодействие ViewModel и соответствующего управляющего элемента (например, цвет надписи для вопроса и т.д.).

1. SingleLine

В качестве представления был выбран стандартный элемент управления Entry (рисунок 3.15) из библиотеки Xamarin.Forms.

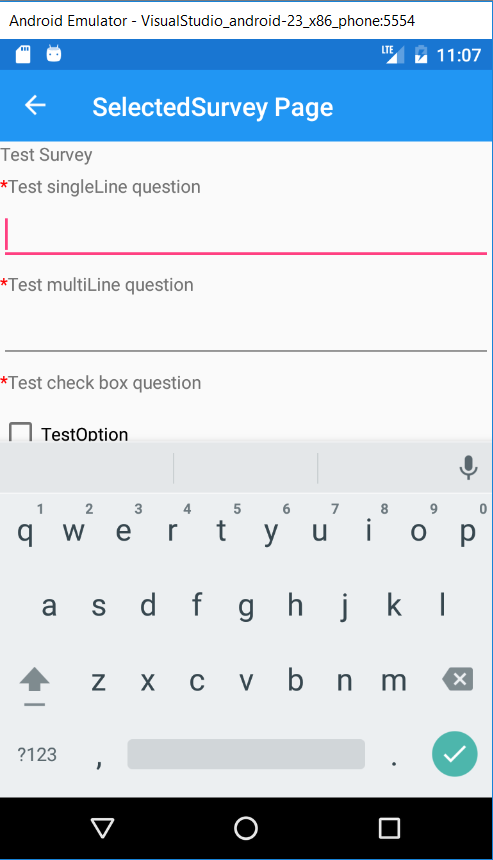


Рисунок 3.15 – Визульное представление SingleLine. Android.

Логика привязки следующая:

Т.к. по сути в данном случае фигурирует один управляющий элемент, то свойство Text элемента Entry будет привязываться к свойству Answer ViewModel. В свою очередь оно будет обращаться к нулевому элементу коллекции AnswerOptions, которая содержится в модели вопроса.

1. MultiLine

В качестве представления был выбран стандартный элемент управления Editor(рисунок 3.16) из библиотеки Xamarin.Forms.

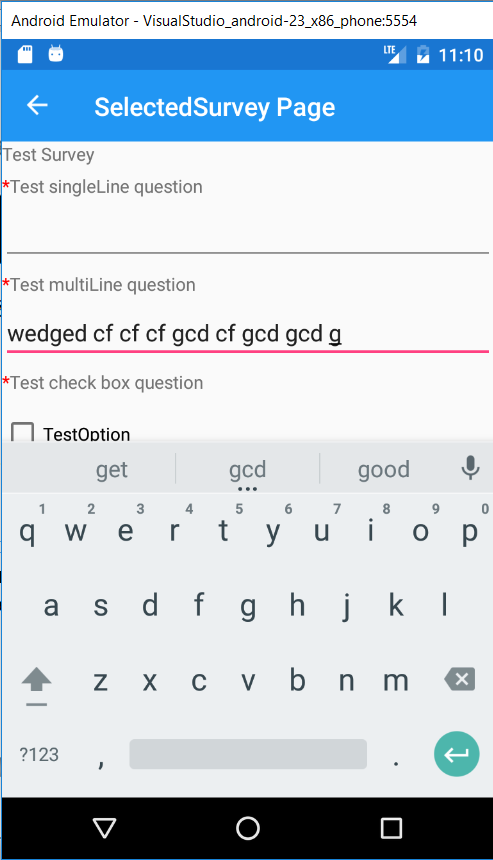


Рисунок 3.16 – Визульное представление MultiLine. Android.

Логика привязки следующая:

Т.к. по сути в данном случае фигурирует один управляющий элемент, то свойство Text элемента Editor будет привязываться к свойству Answer ViewModel. В свою очередь оно будет обращаться к нулевому элементу коллекции AnswerOptions, которая содержится в модели вопроса.

1. CheckBoxGroup

В качестве представления был выбран элемент CheckBox из библиотеки Xlabs, т.к. в стандартной библиотеки нету реализации CheckBox (рисунок 3.17).

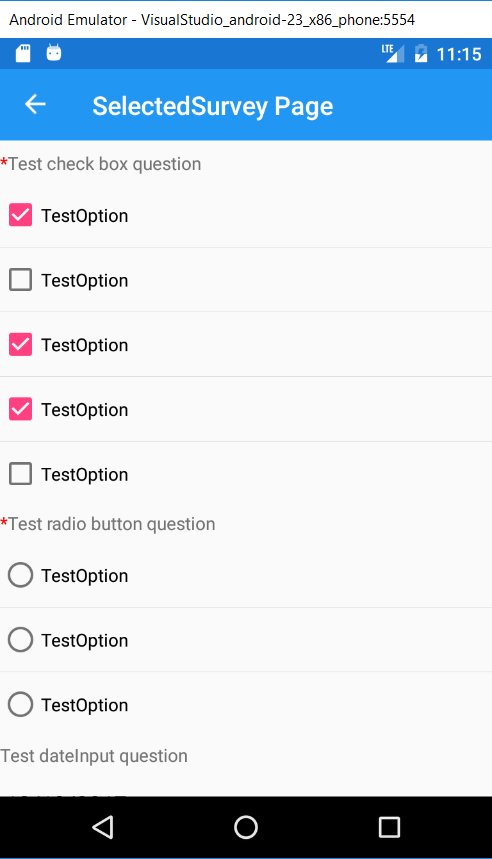


Рисунок 3.17 – Визульное представление CheckBoxGroup. Android.

Логика привязки следующая:

Вопрос представляет собой общее описание вопроса и несколько опций ответов. Соответственно представлением данного типа вопросов будет коллекция CheckBoxов, каждый из которых будет привязан к соответствующей ему ViewModel, которая содержит в себе модель AnswerOption. Ответ будет предсталять собой текстовое представление пременной логического типа: true или false, что будет обозначать отмечена ли вариант ответа или нет.

В качетсве ViewModel, соответствующая модели вопроса, представлен класс CheckBoxesViewModel, которая содержит в себе коллекцию CheckBoxViewModel, которая соответствует модели AnswerOption.

В качестве визуального элемента, содержащего в себе коллекцию CheckBoxов выступает стандартный элемент ListView из пространства имён Xamarin.Forms. В качестве ItemTemplate выступает CheckBoxDataTemplate, который представляет собой визуальный элемент Cell. Который содержит в себе CheckBox.

1. RadioButtonGroup

В качестве представления был выбран элемент CustomRadioButton из библиотеки Xlabs, т.к. в стандартной библиотеки нету реализации RadioButton (рисунок 3.18).

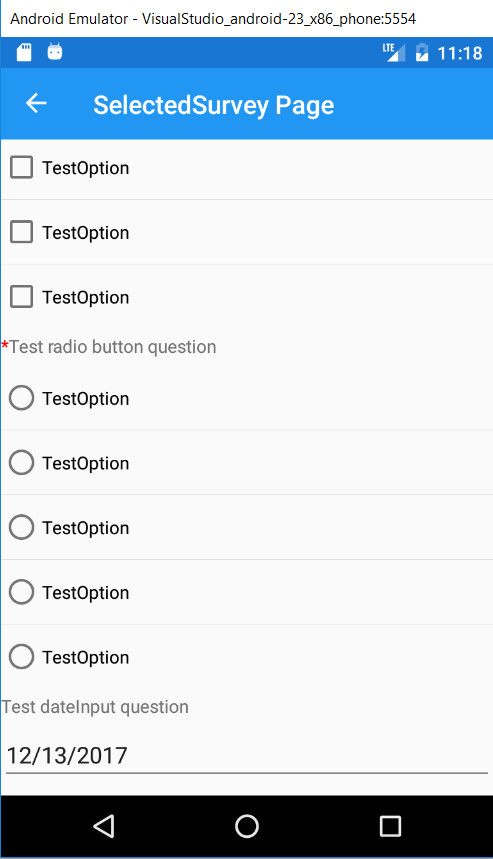


Рисунок 3.18 – Визульное представление RadioButtonGroup. Android.

Логика привязки следующая:

Вопрос представляет собой общее описание вопроса и несколько опций ответов. Соответственно представлением данного типа вопросов будет коллекция RadioButtonов, каждый из которых будет привязан к соответствующей ему ViewModel, которая содержит в себе модель AnswerOption. Ответ будет предсталять собой текстовое представление пременной логического типа: true или false, что будет обозначать отмечена ли вариант ответа или нет.

В качетсве ViewModel, соответствующая модели вопроса, представлен класс RadioButtonsViewModel, которая содержит в себе коллекцию RadioButtonViewModel, которая соответствует модели AnswerOption.

В качестве визуального элемента, содержащего в себе коллекцию RadioButtonов выступает стандартный элемент ListView из пространства имён Xamarin.Forms. В качестве ItemTemplate выступает RadioButtonDataTemplate, который представляет собой визуальный элемент Cell. Который содержит в себе CustomRadioButton.

При изменении значения RadioButtона происходит перерасчёт значения других RadioButtonов, т.к. в группе RadioButtonов значение “отмечен” может одновременно иметь только один элемент.

1. DateInput

В качестве представления был выбран стандартный элемент управления DatePicker (рисунок 3.19) из библиотеки Xamarin.Forms. Логика привязки следующая:

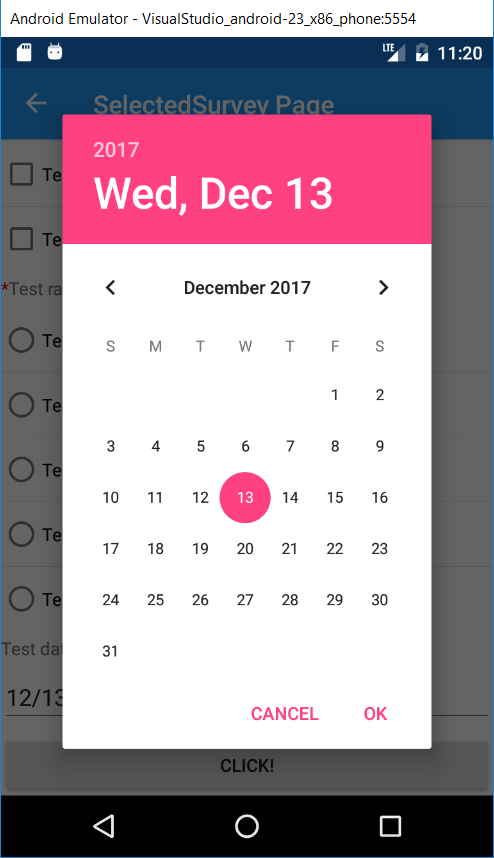


Рисунок 3.19 – Визульное представление DateInput. Android.

Т.к. по сути в данном случае фигурирует один управляющий элемент, то свойство Date элемента DatePicker будет привязываться к свойству Answer ViewModel. В свою очередь оно будет обращаться к нулевому элементу коллекции AnswerOptions, которая содержится в модели вопроса.

1. Custom controllers

Для реализации дополнительной логики взаимодействия с пользователем были разработаны следующие элементы управления:

* QuestionDescriptionLabel. Наследуется от класса Label из пространства имён Xamarin.Forms, в качестве дополнительного Bindable свойства имеет свойство цвета текста вопроса. В дальнейшем это свойство будет использоваться в том случае, когда пользователю нужно будет подсветить те вопросы, которые обязательны и которые он оставил пустыми (рисунок 3.20). Также при формировании данного элемента учитывается, обязательный ли вопрос, и в зависимости от этого помечает его или нет.

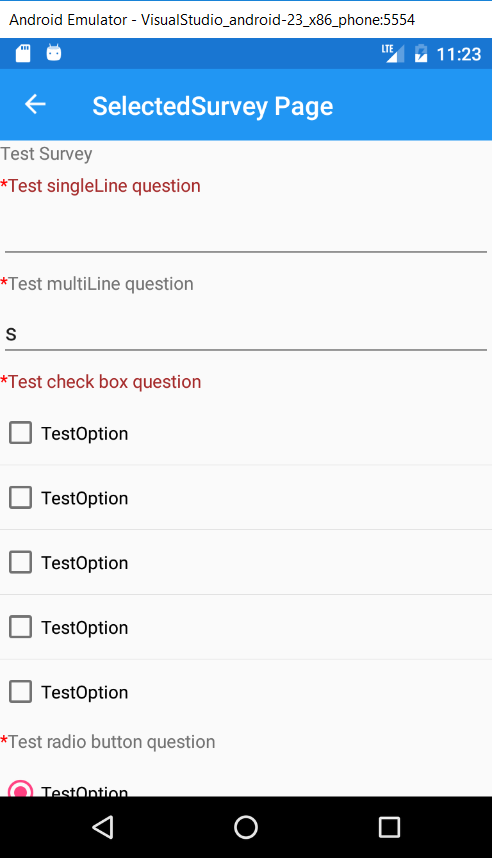


Рисунок 3.20 – Подсветка обязательных неотвеченных вопросов. Android.

1. Валидация ответов перед отправкой

Перед тем, как отправить данные опроса на сервер, нужно проверить, на все ли обязательные вопросы ответил пользователь. Для этого нужно определить критерии для каждого типа вопроса, по которым будет определяться, является ли ответ валидным:

* Для SingleLine ответ будет являться валидным, если он не пустой;
* Для MultiLine ответ будет являться валидным, если он не пустой;
* Для CheckBoxGroup ответ будет являться валидным, если хотя бы одна опция выбрана;
* Для RadioButtonGroup ответ будет являться валидным, если хотя бы одна опция выбрана;
* Todo: choose validation rules for DateInput.

Подход решения данной задачи:

Перед отправлением запроса анализиуется коллекция, содержащая в себе ViewModel вопросов. Если вопрос обязательный и ответ является не валидным, согласно типу вопроса и выше описанных правил, то операция отправление запроса отклоняется, а соответствующие невалидные вопросы подсвечиваются для пользователя (рисунок 3.21).

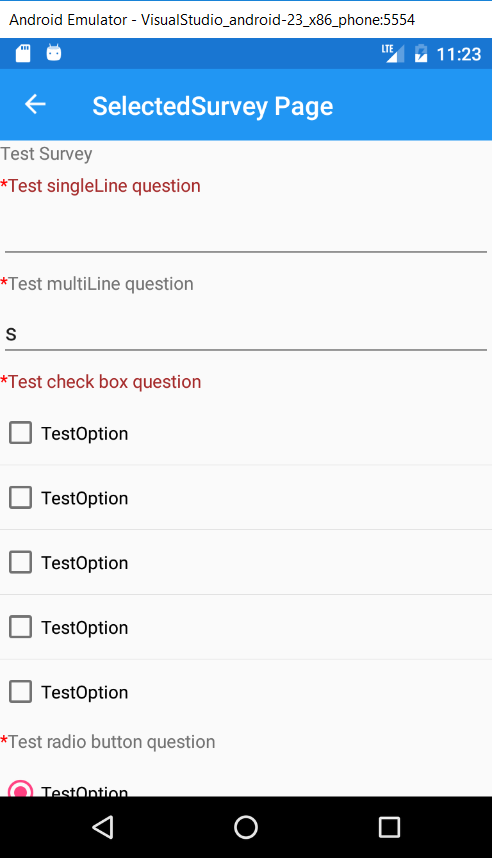


Рисунок 3.21 – подсветка невалидных ответов. Android.

## **Модуль CourceProject.MC.Common**

1. Описание функциональности

Содержит константы, общие данные, используемые в остальных модулях (например, используемые регулярные выражения, атрибуты и т.д.).

1. Регулярные выражения

Регулярные выражения - формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов. Для поиска используется строка-образец, состоящая из символов и метасимволов и задающая правило поиска. Для манипуляций с текстом дополнительно задаётся строка замены, которая также может содержать в себе специальные символы.

1. Атрибуты

Атрибуты представляют специальные инструменты, которые позволяют встраивать в сборку дополнительные метаданные. Атрибуты могут применяться как ко всему типу (классу, интерфейсу и т.д.), так и к отдельным его частям (методу, свойству и т.д.). Основу атрибутов составляет класс System.Attribute, от которого образованы все остальные классы атрибутов.

## **Модуль CourceProject.MC.Resources**

1. Описание функциональности

Содержит общие ресурсы, используемые в остальных модулях (например, надписи приложения, иконки и т.д.).

## **Модуль CourceProject.MC.Android**

1. Описание функциональности
2. Класс MainActivity

## **Модуль CourceProject.MC.iOS**

## **Модуль CourceProject.MC. Foundation**

# **ПРОВЕРКА КОРРЕКТНОСТИ РАБОТЫ ПРОГРАММНОГО СРЕДСТВА**

Ввиду небольшого размера проекта и небольшого функционала и количетсва рабочих окон данного проекта, было решено описать проверку корректности работы данного средства.

# **РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

1. **Системные требования**

Для успешного запуска игры нужно наличие поддержки видеокартой OpenGL и DX9. Системные требования OpenGL:

* процессор с частотой 1,2 ГГц или выше;
* 192 МБ оперативной памяти.

1. **Запуск программы. Начало игры**

Для запуска программы необходимо открыть исполняемый файл Snake.exe. Откроется начальное окно программы (рисунок 3.1). Для того чтобы начать играть нужно ввести своё имя и выбрать сложность, после чего начнётся игра (рисунок 3.2).

1. **Правила игры**

Цель игры: набрать как можно больше очков. Для того чтобы набирать очки, нужно поедать блоки еды. При этом змейка увеличивается в размерах.

Управлять можно только головой. Конец игры наступает, когда змейка врезается либо в свой хвост, либо в стену, либо пересекает край экрана.

1. **Управление**

Для управления программой во время игрового процесса предоставлены следующие клавиши:

* стрелка вниз, вверх, влево, вправо (управление змейкой);
* пробел (остановка/продолжение игры);
* 1, 2 (увеличение/уменьшение громкости музыки);
* 3, 4 (увеличение/уменьшение громкости звука);
* F1, F2 (вызов справки по управлению (рисунок 3.8)/игре (рисунок 3.9)).

Для закрытия таблицы рекордов нужно ввести ‘q’в поле ввода, после чего приложение закроется.

**ЗАКЛЮЧЕНИЕ**

Безусловно, игра “Змейка” является классикой в области компьютерных игр. Несмотря на возраст данной игры, на сегодняшний день находятся современные интерпретации данной идеи и появляются новые популярные проекты на базе “Змейки” (например, slither.io), что говорит об актуальности данной игры и в наше время.

В рамках данного курсового проекта было разработано игровое приложение “Snake”. В данной программе реализован основной игровой процесс, а также его дополнения и расширения:

* генерация случайных препятствий на игровой сцене;
* музыкальное и звуковое сопровождение игрового процесса;
* таблица рекордов;
* уровни сложности.

При разработке программы в качестве основы использовалась библиотека freeglut, которая взаимодействует с OpenGL, а также Bass.

Поставленная цель выполнена. Программа реализует все поставленные задачи.

В будущем планируется программу усовершенствовать, а именно:

* добавить больше параметров уровней сложности (размеры игрового поля, постепенное ускорение игры и т.д.);
* усовершенствовать графическую составляющую;
* расширить плейлист и звуковой ассортимент;
* добавить новые уровни сложности.

Список использованных источников

[1]  [Шилдт Г.: Самоучитель С++: Пер. с англ. – 3-е изд. – СПб.: БХВ- Питербург, 2007. – 688 с.](http://www.piter.com/collection/yazyk-programmirovaniya-ss-s/product/clr-via-c-programmirovanie-na-platforme-microsoft-net-framework-45-na-yazyke-c-4-e-izd-2)

[2] Каталог API и справочных материалов [Электронный ресурс]. –https://msdn.microsoft.com/library

[3] Дейтел, Х. М. Как программировать на С / Х. М Дейтел, П. Дж. Дейтел. – М.: ЗАО «Издательство БИНОМ», 2000. – 1008 с.: ил.

[4] Электронный учебно-методический комплекс по дисциплине «Языки программирования. Часть 1» [Электронный ресурс]. – Электрон. текстовые дан и прог.(6,1 Мб). – Минск: Бахтизин В.В., 2013.

[5] Конструирование программ и языки программирования: Учено-методическое пособие для студ. спец. 40 01 01 “Программное обеспечение информационных технологий” дневной формы обуч. В 2 ч./ В.В.Бахтизин [и др.] – Минск: БГУИР 2006.

[6] Шупляк В.И.: С++. Практический курс: учеб. пособие / Шупляк В.И – Минск: Новое знание, 2008. – 576 с.

**ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ ТЕКСТ ПРОГРАММЫ**

#define \_CRT\_SECURE\_NO\_WARNINGS

#pragma once

#include "targetver.h"

#include <stdio.h>

#include <tchar.h>

#include <GL\freeglut.h>

#include <time.h>

#include <stdlib.h>

#include <iOStream>

#include <vector>

#include <algorithm>

#include <string>

#include <windows.h>

#include "bass.h"

#include "stdafx.h"

#include "commonTypes.h"

#include "scoreBoard.h"

#include "TSound.h"

using namespace std;

typedef float\* TColor;

#define RED 1, 0, 0

#define GREEN 0, 1, 0

#define BLUE 0, 0, 1

#define WHITE 1, 1, 1

enum TTypeOfCell {NONE, SNAKE, FOOD, WALL};

TInfo scoreBoard;

TSound sound("eat.wav", "move.wav", "", "endSong.mp3");

class mainGame{

public:

int delay;

int width;

int height;

int startPosWinX;

int startPosWinY;

float scale;

float scaleWin;

int startPosSnakeX;

int startPosSnakeY;

int startSizeSnake;

int countOfWalls;

TTypeOfCell \*\*field;

char playerName[20];

TDifficult difficult;

long int score;

int windowIndex;

bool timeToShowRecords;

bool isGameStopped;

HWND descrCon;

mainGame(int delay, int width, int height, int startPosWinX, int startPosWinY, float scale, float scaleWin,

int startPosSnakeX, int startPosSnakeY, int startSizeSnake, int countOfWalls, bool isGameStopped) :

delay(delay), width(width), height(height), startPosWinX(startPosWinX), startPosWinY(startPosWinY),

scale(scale), scaleWin(scaleWin), startPosSnakeX(startPosSnakeX), startPosSnakeY(startPosSnakeY),

startSizeSnake(startSizeSnake), countOfWalls(countOfWalls), score(0), timeToShowRecords(false), isGameStopped(isGameStopped)

{

field = (TTypeOfCell\*\*)malloc(sizeof(TTypeOfCell\*) \* height);

for (int j = 0; j < height; j++)

{

field[j] = (TTypeOfCell\*)malloc(sizeof(TTypeOfCell) \* width);

for (int i = 0; i < width;i++)

field[j][i] = NONE;

}

}

mainGame();

void drawBlock(int x, int y);

void drawBlock(int x, int y, float R, float G, float B);

void fillField();

void fillSpace();

bool isGameOver();

bool isSnakeBumpintoItsSelf();

bool isSnakeBumpintoWalls();

char \*getTitle();

void leaveGame();

void stop\_continueGame();

void stopGame();

void continueGame();

void showRecords();

#define HINTMSG 0

#define ENDOFGAMEMSG 1

#define CONTROLMSG 2

void mainGame::showMessage(HWND handle, int messageIndex, bool isFirstTime);

private:

void fillSpaceBorders();

void fillSpaceCorners();

void fillSpaceIntoField();

}settings;

typedef struct \_TPieceBody{

int x;

int y;

} TPieceBody;

class TFood{

public:

int x;

int y;

TFood() :x(-1), y(-1) {};

void drop();

} food;

class TSnake{

public:

int direction;

bool isSnakeTurn;

int sizeSnake;

vector<TPieceBody> snakeBody;

TSnake(int direction, int sizeSnake) : direction(direction), sizeSnake(sizeSnake), isSnakeTurn(false)

{

snakeBody.push\_back({ settings.startPosSnakeX, settings.startPosSnakeY });

for (int i = 1; i < sizeSnake; i++)

{

snakeBody.push\_back({-1, -1});

}

};

void drawBody();

void makeAction();

private:

void makeMove();

bool eat();

} snake(GLUT\_KEY\_UP, 3);

typedef enum \_TTypeWall {line3, corner, quad} TTypeWall;

#define COUNTOFTYPEOFWALL 3

typedef struct \_TWallBlock

{

int x;

int y;

} TWallBlock;

class TWall

{

public:

vector<TWallBlock> walls;

void spawnWalls(int count);

bool placeBlock();

bool placeLine();

bool placeCorner();

bool placeQuad();

void drawWalls();

} wall;

mainGame::mainGame()

{

delay = 200;

width = 60;

height = 35;

startPosWinX = 200;

startPosWinY = 200;

scale = 1;

scaleWin = 5;

startPosSnakeX = width / 2;

startPosSnakeY = height / 2;

startSizeSnake = 3;

countOfWalls = 100;

score = 0;

timeToShowRecords = false;

isGameStopped = true;

field = (TTypeOfCell\*\*)malloc(sizeof(TTypeOfCell\*)\* height);

for (int j = 0; j < height; j++)

{

field[j] = (TTypeOfCell\*)malloc(sizeof(TTypeOfCell)\*width);

for (int i = 0; i < width; i++)

field[j][i] = NONE;

}

}

void mainGame::drawBlock(int x, int y)

{

glColor3f(1, 1, 1);

glVertex2f(x \* settings.scale, y \* settings.scale);

glVertex2f(x \* settings.scale + settings.scale, y \* settings.scale);

glVertex2f(x \* settings.scale + settings.scale, y \* settings.scale + settings.scale);

glVertex2f(x \* settings.scale, y \* settings.scale + settings.scale);

}

void mainGame::drawBlock(int x, int y, float R, float G, float B)

{

glColor3f(R, G, B);

glVertex2f(x \* settings.scale, y \* settings.scale);

glVertex2f(x \* settings.scale + settings.scale, y \* settings.scale);

glVertex2f(x \* settings.scale + settings.scale, y \* settings.scale + settings.scale);

glVertex2f(x \* settings.scale, y \* settings.scale + settings.scale);

}

bool mainGame::isSnakeBumpintoItsSelf()

{

for (int i = 1; i < snake.snakeBody.size(); i++)

if ((snake.snakeBody[0].x == snake.snakeBody[i].x) && (snake.snakeBody[0].y == snake.snakeBody[i].y))

return true;

return false;

}

bool mainGame::isSnakeBumpintoWalls()

{

for (int i = 0; i < wall.walls.size(); i++)

if ((snake.snakeBody[0].x == wall.walls[i].x) && (snake.snakeBody[0].y == wall.walls[i].y))

return true;

return false;

}

bool mainGame::isGameOver()

{

if (((snake.snakeBody[0].x < 0) || (snake.snakeBody[0].y < 0)) || isSnakeBumpintoItsSelf()

|| ((snake.snakeBody[0].x >= settings.width) || (snake.snakeBody[0].y >= settings.height)) || isSnakeBumpintoWalls())

{

return true;

}

return false;

}

void mainGame::fillField()

{

for (int i = 0; i < wall.walls.size(); i++)

{

if (((wall.walls[i].x < width) && (wall.walls[i].y < height)) && ((wall.walls[i].x >= 0) && (wall.walls[i].y >= 0)))

field[wall.walls[i].y][wall.walls[i].x] = WALL;

}

}

void mainGame::fillSpaceCorners()

{

if ((field[0][1] == WALL) || (field[1][0] == WALL))

{

field[0][0] = WALL;

wall.walls.push\_back({ 0, 0 });

}

if ((field[0][width - 2] == WALL) || (field[1][width - 1] == WALL))

{

field[0][width - 1] = WALL;

wall.walls.push\_back({ width - 1, 0 });

}

if ((field[height - 1][1] == WALL) || (field[height - 2][0] == WALL))

{

field[height - 1][0] = WALL;

wall.walls.push\_back({ 0, height - 1 });

}

if ((field[height - 1][width - 2] == WALL) || (field[height - 2][width - 1] == WALL))

{

field[height - 1][width - 1] = WALL;

wall.walls.push\_back({ width - 1, height - 1 });

}

}

void mainGame::fillSpaceBorders()

{

for (int i = 1; i < width - 1; i++)

{

int count = 0;

if (field[0][i - 1] == WALL)

count++;

if (field[0][i + 1] == WALL)

count++;

if (field[1][i] == WALL)

count++;

if (count > 1)

{

field[0][i] = WALL;

wall.walls.push\_back({ i, 0 });

}

}

for (int i = width - 2; i > 0; i--)

{

int count = 0;

if (field[0][i - 1] == WALL)

count++;

if (field[0][i + 1] == WALL)

count++;

if (field[1][i] == WALL)

count++;

if (count > 1)

{

field[0][i] = WALL;

wall.walls.push\_back({ i, 0 });

}

}

for (int i = 1; i < width - 1; i++)

{

int count = 0;

if (field[height - 1][i - 1] == WALL)

count++;

if (field[height - 1][i + 1] == WALL)

count++;

if (field[height - 2][i] == WALL)

count++;

if (count > 1)

{

field[height - 1][i] = WALL;

wall.walls.push\_back({ i, height - 1 });

}

}

for (int i = width - 2; i > 0; i--)

{

int count = 0;

if (field[height - 1][i - 1] == WALL)

count++;

if (field[height - 1][i + 1] == WALL)

count++;

if (field[height - 2][i] == WALL)

count++;

if (count > 1)

{

field[height - 1][i] = WALL;

wall.walls.push\_back({ i, height - 1 });

}

}

for (int j = 1; j < height - 1; j++)

{

int count = 0;

if (field[j - 1][0] == WALL)

count++;

if (field[j + 1][0] == WALL)

count++;

if (field[j][1] == WALL)

count++;

if (count > 1)

{

field[j][0] = WALL;

wall.walls.push\_back({ 0, j });

}

}

for (int j = height - 2; j > 0; j--)

{

int count = 0;

if (field[j - 1][0] == WALL)

count++;

if (field[j + 1][0] == WALL)

count++;

if (field[j][1] == WALL)

count++;

if (count > 1)

{

field[j][0] = WALL;

wall.walls.push\_back({ 0, j });

}

}

for (int j = 1; j < height - 1; j++)

{

int count = 0;

if (field[j - 1][width - 1] == WALL)

count++;

if (field[j + 1][width - 1] == WALL)

count++;

if (field[j][width - 2] == WALL)

count++;

if (count > 1)

{

field[j][width - 1] = WALL;

wall.walls.push\_back({ width - 1, j });

}

}

for (int j = height - 2; j > 1; j--)

{

int count = 0;

if (field[j - 1][width - 1] == WALL)

count++;

if (field[j + 1][width - 1] == WALL)

count++;

if (field[j][width - 2] == WALL)

count++;

if (count > 1)

{

field[j][width - 1] = WALL;

wall.walls.push\_back({ width - 1, j });

}

}

}

void mainGame::fillSpaceIntoField()

{

for (int j = 1; j < height - 1; j++)

{

for (int i = 1; i < width - 1; i++)

{

if (field[j][i] == NONE)

{

int count = 0;

if ((field[j][i + 1] == WALL))

count++;

if ((field[j][i - 1] == WALL))

count++;

if ((field[j - 1][i] == WALL))

count++;

if ((field[j + 1][i] == WALL))

count++;

if (count > 2)

{

field[j][i] = WALL;

wall.walls.push\_back({ i, j });

}

}

}

}

for (int j = 1; j < height - 1; j++)

{

for (int i = width - 2; i > 0; i--)

{

if (field[j][i] == NONE)

{

int count = 0;

if ((field[j][i + 1] == WALL))

count++;

if ((field[j][i - 1] == WALL))

count++;

if ((field[j - 1][i] == WALL))

count++;

if ((field[j + 1][i] == WALL))

count++;

if (count > 2)

{

field[j][i] = WALL;

wall.walls.push\_back({ i, j });

}

}

}

}

for (int j = height - 2; j > 0; j--)

{

for (int i = 1; i < width - 1; i++)

{

if (field[j][i] == NONE)

{

int count = 0;

if ((field[j][i + 1] == WALL))

count++;

if ((field[j][i - 1] == WALL))

count++;

if ((field[j - 1][i] == WALL))

count++;

if ((field[j + 1][i] == WALL))

count++;

if (count > 2)

{

field[j][i] = WALL;

wall.walls.push\_back({ i, j });

}

}

}

}

for (int j = height - 2; j > 0; j--)

{

for (int i = width - 2; i > 0; i--)

{

if (field[j][i] == NONE)

{

int count = 0;

if ((field[j][i + 1] == WALL))

count++;

if ((field[j][i - 1] == WALL))

count++;

if ((field[j - 1][i] == WALL))

count++;

if ((field[j + 1][i] == WALL))

count++;

if (count > 2)

{

field[j][i] = WALL;

wall.walls.push\_back({ i, j });

}

}

}

}

}

void mainGame::fillSpace()

{

fillSpaceIntoField();

fillSpaceBorders();

fillSpaceCorners();

fillSpaceBorders();

fillSpaceIntoField();

}

char \*mainGame::getTitle()

{

char temp[50] = "Snake. Player name: ";

strcat(temp, playerName);

strcat(temp, ". Score: ");

char buffScore[40];

char \*tempScore = \_itoa(score, buffScore, 10);

strcat(temp, tempScore);

return temp;

}

void mainGame::stop\_continueGame()

{

if (isGameStopped)

{

isGameStopped = false;

}

else

{

isGameStopped = true;

}

}

void mainGame::stopGame()

{

isGameStopped = true;

}

void mainGame::continueGame()

{

isGameStopped = false;

}

void mainGame::leaveGame()

{

TPlayerInfo temp;

temp.playerName[0] = '\0';

strcat(temp.playerName, playerName);

temp.countOfScore = score;

temp.difficult[0] = '\0';

temp.diffIndex = difficult.index;

strcat(temp.difficult, difficult.name);

scoreBoard.addDataToList(temp);

showMessage(FindWindowA(NULL, settings.getTitle()), ENDOFGAMEMSG, false);

glutHideWindow();

sound.stopPlayMusic();

sound.playEndSong();

scoreBoard.printData();

timeToShowRecords = true;

}

void mainGame::showRecords()

{

LPCSTR title = "SNAKE";

glutDestroyWindow(windowIndex);

glutLeaveMainLoop();

GetConsoleWindow();

ShowWindow(settings.descrCon, SW\_SHOW);

char answ;

getchar();

cin >> answ;

while (answ != 'q')

cin >> answ;

}

#define CONTROLTEXT "Control:\n\rF1 - Help\n\rF2 - rules of game\n\rspace - pause\n\r1 - increase music volume\n\r2 - decrease music volume\n\r3 - increase sound volume\n\r4 - decrease sound volume\n\rPRESS SPACE TO CONTINUE"

#define HINTTEXT "The rules is simple: you need eat green block to increase your size and to get more score. The end of the game is when you bump in your body of in the wall or into the borders of the field. Walls is red.\n\rPRESS SPACE TO CONTINUE"

#define ENDOFTHEGAMEMSG "Game Over"

#define CONTROLTEXTCON "Control:\n\rF1 - Help\n\rF2 - rules of game\n\rspace - pause\n\r1 - increase music volume\n\r2 - decrease music volume\n\r3 - increase sound volume\n\r4 - decrease sound volume\n\rPRESS SPACE TO START"

#define HINTTEXTCON "The rules is simple: you need eat green block to increase your size and to get more score. The end of the game is when you bump in your body of in the wall or into the borders of the field. Walls is red."

#define FIRSTTIME true

#define NOTFIRSTTIME false

void mainGame::showMessage(HWND handle, int messageIndex, bool isFirstTime)

{

switch (messageIndex)

{

case HINTMSG:

if (isFirstTime)

MessageBoxA(handle, HINTTEXTCON, "Snake", MB\_OK);

else

MessageBoxA(handle, HINTTEXT, "Snake", MB\_OK);

break;

case CONTROLMSG:

if (isFirstTime)

MessageBoxA(handle, CONTROLTEXTCON, "Snake", MB\_OK);

else

MessageBoxA(handle, CONTROLTEXT, "Snake", MB\_OK);

break;

case ENDOFGAMEMSG:

MessageBoxA(handle, ENDOFTHEGAMEMSG, "Snake", MB\_OK);

break;

default:

break;

}

}

void TFood::drop()

{

bool isOver = false;

while (!isOver)

{

isOver = true;

x = (rand() % settings.width);

y = (rand() % settings.height);

for (int i = 0; i < snake.sizeSnake; i++)

{

if ((x == snake.snakeBody[i].x) && (y == snake.snakeBody[i].y))

{

isOver = false;

break;

}

}

if (isOver)

{

for (int i = 0; i < wall.walls.size(); i++)

{

if ((x == wall.walls[i].x) && (y == wall.walls[i].y))

{

isOver = false;

break;

}

}

}

if (isOver)

{

settings.field[y][x] = FOOD;

}

}

}

bool TWall::placeBlock()

{

walls.push\_back({ rand() % settings.width, rand() % settings.height });

bool isPlaced = false;

while (!isPlaced)

{

if ((walls[walls.size() - 1].x == snake.snakeBody[0].x) && (walls[walls.size() - 1].y == snake.snakeBody[0].y))

{

walls[walls.size() - 1].x = rand() % settings.width;

walls[walls.size() - 1].y = rand() % settings.height;

}

else

{

settings.field[walls[walls.size() - 1].y][walls[walls.size() - 1].x] = WALL;

isPlaced = true;

}

}

return isPlaced;

}

#define LINEPOS 2

#define HOR 0

#define VERT 1

bool TWall::placeLine()

{

walls.push\_back({ rand() % settings.width, rand() % settings.height });

bool isPlaced = false;

while (!isPlaced)

{

if ((walls[walls.size() - 1].x > (snake.snakeBody[0].x - settings.startSizeSnake)) && (walls[walls.size() - 1].x < (snake.snakeBody[0].x + settings.startSizeSnake))

&& (walls[walls.size() - 1].y >(snake.snakeBody[0].y - settings.startSizeSnake)) && (walls[walls.size() - 1].y < (snake.snakeBody[0].y + settings.startSizeSnake)))

{

walls[walls.size() - 1].x = rand() % settings.width;

walls[walls.size() - 1].y = rand() % settings.height;

}

else

{

switch (rand() % LINEPOS){

case HOR:

{

walls.push\_back({ walls[walls.size() - 1].x + 1, walls[walls.size() - 1].y });

walls.push\_back({ walls[walls.size() - 1].x + 1, walls[walls.size() - 1].y });

}

break;

case VERT:

{

walls.push\_back({ walls[walls.size() - 1].x, walls[walls.size() - 1].y + 1 });

walls.push\_back({ walls[walls.size() - 1].x, walls[walls.size() - 1].y + 1 });

}

break;

}

isPlaced = true;

}

}

return isPlaced;

}

#define CORNERPOS 4

#define UPLEFT 0

#define DOWNLEFT 1

#define UPRIGTH 2

#define DOWNRIGTH 3

bool TWall::placeCorner()

{

walls.push\_back({ rand() % settings.width, rand() % settings.height });

bool isPlaced = false;

while (!isPlaced)

{

if ((walls[walls.size() - 1].x > (snake.snakeBody[0].x - settings.startSizeSnake)) && (walls[walls.size() - 1].x < (snake.snakeBody[0].x + settings.startSizeSnake))

&& (walls[walls.size() - 1].y >(snake.snakeBody[0].y - settings.startSizeSnake)) && (walls[walls.size() - 1].y < (snake.snakeBody[0].y + settings.startSizeSnake)))

{

walls[walls.size() - 1].x = rand() % settings.width;

walls[walls.size() - 1].y = rand() % settings.height;

}

else

{

switch (rand() % CORNERPOS){

case UPLEFT:

{

walls.push\_back({ walls[walls.size() - 1].x, walls[walls.size() - 1].y - 1 });

walls.push\_back({ walls[walls.size() - 1].x - 1, walls[walls.size() - 1].y });

}

break;

case DOWNLEFT:

{

walls.push\_back({ walls[walls.size() - 1].x, walls[walls.size() - 1].y + 1 });

walls.push\_back({ walls[walls.size() - 1].x - 1, walls[walls.size() - 1].y });

}

break;

case UPRIGTH:

{

walls.push\_back({ walls[walls.size() - 1].x, walls[walls.size() - 1].y - 1 });

walls.push\_back({ walls[walls.size() - 1].x + 1, walls[walls.size() - 1].y });

}

break;

case DOWNRIGTH:

{

walls.push\_back({ walls[walls.size() - 1].x, walls[walls.size() - 1].y + 1 });

walls.push\_back({ walls[walls.size() - 1].x + 1, walls[walls.size() - 1].y });

}

break;

}

isPlaced = true;

}

}

return isPlaced;

}

bool TWall::placeQuad()

{

walls.push\_back({ rand() % settings.width, rand() % settings.height });

bool isPlaced = false;

while (!isPlaced)

{

if ((walls[walls.size() - 1].x > (snake.snakeBody[0].x - settings.startSizeSnake)) && (walls[walls.size() - 1].x < (snake.snakeBody[0].x + settings.startSizeSnake))

&& (walls[walls.size() - 1].y >(snake.snakeBody[0].y - settings.startSizeSnake)) && (walls[walls.size() - 1].y < (snake.snakeBody[0].y + settings.startSizeSnake)))

{

walls[walls.size() - 1].x = rand() % settings.width;

walls[walls.size() - 1].y = rand() % settings.height;

}

else

{

walls.push\_back({ walls[walls.size() - 1].x, walls[walls.size() - 1].y + 1 });

walls.push\_back({ walls[walls.size() - 1].x + 1, walls[walls.size() - 1].y });

walls.push\_back({ walls[walls.size() - 1].x, walls[walls.size() - 1].y - 1 });

isPlaced = true;

}

}

return isPlaced;

}

void TWall::spawnWalls(int count)

{

for (int i = 0; i < count; i++){

switch (rand() % COUNTOFTYPEOFWALL)

{

case line3:

placeLine();

break;

case corner:

placeCorner();

break;

case quad:

placeQuad();

break;

}

}

}

void TWall::drawWalls()

{

for (int i = 0; i < walls.size(); i++)

settings.drawBlock(walls[i].x, walls[i].y, RED);

}

void TSnake::drawBody()

{

settings.drawBlock(snake.snakeBody[0].x, snakeBody[0].y, BLUE);

for (int i = 1; i < snakeBody.size(); i++)

{

settings.drawBlock(snake.snakeBody[i].x, snakeBody[i].y, WHITE);

}

}

void TSnake::makeMove()

{

for (int i = snakeBody.size() - 1; i > 0; i--){

snakeBody[i].x = snakeBody[i - 1].x;

snakeBody[i].y = snakeBody[i - 1].y;

}

switch (direction)

{

case GLUT\_KEY\_UP:

snakeBody[0].y++;

break;

case GLUT\_KEY\_DOWN:

snakeBody[0].y--;

break;

case GLUT\_KEY\_LEFT:

snakeBody[0].x--;

break;

case GLUT\_KEY\_RIGHT:

snakeBody[0].x++;

break;

}

}

bool TSnake::eat()

{

if ((food.x == snakeBody[0].x) && (food.y == snakeBody[0].y))

{

snakeBody.push\_back({ -1, -1 });

sizeSnake++;

settings.field[food.y][food.x] = NONE;

food.drop();

settings.score = settings.score + 10 \* settings.difficult.scoreMultiplier;

char temp[50];

temp[0] = '\0';

strcat(temp,settings.getTitle());

glutSetWindowTitle(temp);

sound.playSound(SOUNDEATINDEX);

return true;

}

return false;

}

void TSnake::makeAction()

{

snake.isSnakeTurn = true;

snake.makeMove();

snake.eat();

}

void mainEvent(int value)

{

if (!settings.isGameStopped)

{

snake.makeAction();

glutPostRedisplay();

if (settings.timeToShowRecords)

{

settings.showRecords();

}

else

if (settings.isGameOver())

{

settings.leaveGame();

}

}

glutTimerFunc(settings.delay, mainEvent, 0);

}

void drawCurrState()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_QUADS);

settings.drawBlock(food.x, food.y, GREEN);

wall.drawWalls();

snake.drawBody();

glEnd();

glutSwapBuffers();

}

void keyboardInterrControl(int key, int x, int y){

if (!settings.isGameStopped)

{

switch (key)

{

case GLUT\_KEY\_UP:

if ((snake.direction != GLUT\_KEY\_DOWN) && (snake.isSnakeTurn))

{

if (snake.direction != GLUT\_KEY\_UP)

sound.playSound(SOUNDMOVEINDEX);

snake.direction = GLUT\_KEY\_UP;

snake.isSnakeTurn = false;

}

break;

case GLUT\_KEY\_DOWN:

if ((snake.direction != GLUT\_KEY\_UP) && (snake.isSnakeTurn))

{

if (snake.direction != GLUT\_KEY\_DOWN)

sound.playSound(SOUNDMOVEINDEX);

snake.direction = GLUT\_KEY\_DOWN;

snake.isSnakeTurn = false;

}

break;

case GLUT\_KEY\_LEFT:

if ((snake.direction != GLUT\_KEY\_RIGHT) && (snake.isSnakeTurn))

{

if (snake.direction != GLUT\_KEY\_LEFT)

sound.playSound(SOUNDMOVEINDEX);

snake.direction = GLUT\_KEY\_LEFT;

snake.isSnakeTurn = false;

}

break;

case GLUT\_KEY\_RIGHT:

if ((snake.direction != GLUT\_KEY\_LEFT) && (snake.isSnakeTurn))

{

if (snake.direction != GLUT\_KEY\_RIGHT)

sound.playSound(SOUNDMOVEINDEX);

snake.direction = GLUT\_KEY\_RIGHT;

snake.isSnakeTurn = false;

}

break;

}

}

switch (key)

{

case GLUT\_KEY\_F1:

settings.stopGame();

settings.showMessage(FindWindowA(NULL, settings.getTitle()), CONTROLMSG, NOTFIRSTTIME);

break;

case GLUT\_KEY\_F2:

settings.stopGame();

settings.showMessage(FindWindowA(NULL, settings.getTitle()), HINTMSG, NOTFIRSTTIME);

break;

default:

break;

}

}

#define SPACE 32

#define KEY\_INCVOLMUSIC '1'

#define KEY\_DECVOLMUSIC '2'

#define KEY\_INCVOLSOUND '3'

#define KEY\_DECVOLSOUND '4'

void keyboardInterr(unsigned char key, int x, int y)

{

switch (key)

{

case SPACE: settings.stop\_continueGame();

break;

case KEY\_INCVOLMUSIC: sound.changeMusicVol(3);

break;

case KEY\_DECVOLMUSIC: sound.changeMusicVol(-3);

break;

case KEY\_INCVOLSOUND: sound.changeSoundVol(5);

break;

case KEY\_DECVOLSOUND: sound.changeSoundVol(-5);

default:

break;

}

}

void initGame()

{

LPCSTR title = "SNAKE";

SetConsoleTitleA(title);

char answ;

cout << "Please, input your name:";

cin >> settings.playerName;

if (settings.playerName == "")

strcat\_s(settings.playerName, "player");

cout << endl << "please, chose defficult:" << endl;

for (int i = 1; i <= COUNTOFDIFFICULTS; i++)

cout << i << '(' << DIFFICULTS[i - 1].info << ')' << endl;

bool isCorrectAnsw = false;

while (!isCorrectAnsw)

{

cin >> answ;

if ((answ > '7') || (answ < '1'))

{

cout << "Incorrect answer. Please, input data:" << endl;

isCorrectAnsw = false;

}

else

isCorrectAnsw = true;

}

switch (answ)

{

case '1': settings.difficult = { VERYEASY };

break;

case '2': settings.difficult = { EASY };

break;

case '3': settings.difficult = { NORMAL };

break;

case '4': settings.difficult = { ADVANCED };

break;

case '5': settings.difficult = { HARD };

break;

case '6': settings.difficult = { VERYHARD };

break;

case '7': settings.difficult = { HARDCORE };

break;

default:

break;

}

system("cls");

settings.descrCon = FindWindowA(NULL, title);

ShowWindow(settings.descrCon, SW\_HIDE);

}

void exitFromGame(int value)

{

glutLeaveMainLoop();

}

void initSound()

{

sound.addToPlaylist("1.mp3");

sound.addToPlaylist("2.mp3");

sound.addToPlaylist("3.mp3");

sound.addToPlaylist("4.mp3");

sound.addToPlaylist("5.mp3");

sound.addToPlaylist("6.mp3");

//sound.addToPlaylist("7.mp3");

}

void startGame(int argc, \_TCHAR argv[], TDifficult difficult)

{

srand(time(NULL));

initSound();

wall.spawnWalls(difficult.countOfWalls);

settings.delay = difficult.speed;

settings.fillField();

settings.fillSpace();

food.drop();

glutInit(&argc, (char\*\*)argv);

glutInitDisplayMode(GLUT\_RGB);

glutInitWindowSize(settings.width\*settings.scaleWin \* 2, settings.height\*settings.scaleWin \* 2);

glutInitWindowPosition(settings.startPosWinX, settings.startPosWinY);

char temp[50];

temp[0] = '\0';

strcat(temp, settings.getTitle());

settings.showMessage(NULL, HINTMSG, FIRSTTIME);

settings.showMessage(NULL, CONTROLMSG, FIRSTTIME);

settings.windowIndex = glutCreateWindow(temp);

HWND handle = FindWindowA(NULL, settings.getTitle());

glutDisplayFunc(drawCurrState);

glutTimerFunc(settings.delay, mainEvent, 0);

glutSpecialFunc(keyboardInterrControl);

glutKeyboardFunc(keyboardInterr);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0, settings.width, 0, settings.height, -1, 1);

sound.changeMusicVol(-80);

sound.startPlayMusic();

glutMainLoop();

SetFocus(handle);

}

int \_tmain(int argc, \_TCHAR argv[])

{

initGame();

startGame(argc, argv, settings.difficult);

return(0);

}

typedef struct \_TDifficult{

char name[10];

int index;

int speed;

int countOfWalls;

char info[400];

float scoreMultiplier;

}TDifficult;

#define VERYEASY "Very easy" , 0, 250, 0, "slow speed, no walls", 1

#define EASY "Easy", 1, 100, 0, "normal speed, no walls", 1.5

#define NORMAL "Normal", 2, 250, 50, "slow speed, few walls", 2.5

#define ADVANCED "Advanced", 3, 150, 50, "normal speed, few walls", 3.5

#define HARD "Hard", 4, 250, 100, "slow speed, many walls", 4

#define VERYHARD "Very hard", 5, 150, 100, "normal speed, many walls", 5

#define HARDCORE "Hardcode", 6, 100, 100, "HARDCORE!!!!", 7

#define COUNTOFDIFFICULTS 7

const TDifficult DIFFICULTS[COUNTOFDIFFICULTS] = { { VERYEASY }, { EASY }, { NORMAL }, { ADVANCED }, { HARD }, { VERYHARD }, { HARDCORE } };

using namespace std;

typedef struct \_TPlayerInfo{

char playerName[20];

long int countOfScore;

char difficult[10];

int diffIndex;

} TPlayerInfo;

#define ENDOFVECTORINFO {"", -1, "", -1}

typedef vector <TPlayerInfo> TArrInfo;

typedef vector<TPlayerInfo>::iterator TRange;

typedef vector<TRange> TRanges;

typedef TRange TPlayerInfoIter;

class TInfo{

public:

void printData();

void addDataToList(TPlayerInfo data);

TInfo();

private:

char \*fileName;

TPlayerInfo player;

TArrInfo arrInfo;

TRanges ranges;

int prevCountScore;

TArrInfo getDataFromFile();

void writeDataToFile();

TRanges getRanges();

void sortData();

void partSort();

void printPlayerResults();

};

#include "stdafx.h"

#include "commonTypes.h"

#include "scoreBoard.h"

using namespace std;

#define NEWRECORFIRSTTIME -1

#define NORECORD -2

void TInfo::printPlayerResults()

{

switch (prevCountScore)

{

case NEWRECORFIRSTTIME:

printf("Congratulations, %s. Your result is %d score(difficult - %s).", player.playerName, player.countOfScore, player.difficult);

break;

case NORECORD:

printf("Congratulations, %s. Your result is %d score(difficult - %s). But you haven't beaten your previous record on this difficult.", player.playerName, player.countOfScore, player.difficult);

break;

default:

printf("Congratulations, %s. You have beaten your previous record on this difficult. Your result is %d score(difficult - %s). Your previous result is %d score.", player.playerName, player.countOfScore, player.difficult, prevCountScore);

break;

}

}

void TInfo::printData()

{

printPlayerResults();

cout << endl << endl <<"-------------------ScoreBoard------------------";

printf("\n%-20s%-10s%-8s", "Player name", "Difficult", "Score");

printf("\n-----------------------------------------------");

for (int i = 0; i < (arrInfo.size()) - 1; i++)

{

printf("\n%-20s%-10s%-8d", arrInfo[i].playerName, arrInfo[i].difficult, arrInfo[i].countOfScore);

}

printf("\n\nTo quit the game print q\n");

}

class TFindFunc{

public:

TFindFunc(TPlayerInfo currPlayer) : currPlayer(currPlayer) {};

bool operator()(TPlayerInfo player) const {

if ((strcmp(player.playerName, currPlayer.playerName) == 0) && (player.diffIndex == currPlayer.diffIndex))

return true;

else

return false;

};

private:

TPlayerInfo currPlayer;

};

void TInfo::addDataToList(TPlayerInfo data)

{

player = data;

if (data.countOfScore != 0)

{

TPlayerInfoIter founded = find\_if(arrInfo.begin(), arrInfo.end(), TFindFunc(player));

if (founded != arrInfo.end())

{

if (founded->countOfScore < player.countOfScore)

{

prevCountScore = founded->countOfScore;

founded->countOfScore = player.countOfScore;

}

else

prevCountScore = NORECORD;

}

else

{

arrInfo.push\_back(player);

prevCountScore = NEWRECORFIRSTTIME;

}

sortData();

writeDataToFile();

}

}

bool compareFuncDiff(TPlayerInfo first, TPlayerInfo second)

{

return (first.diffIndex > second.diffIndex);

}

bool compareFuncScore(TPlayerInfo first, TPlayerInfo second)

{

return (first.countOfScore > second.countOfScore);

}

TRanges TInfo::getRanges()

{

TRanges res;

int currIndex = COUNTOFDIFFICULTS - 1;

TRange currIter = arrInfo.begin();

res.push\_back(arrInfo.begin());

for (int i = 0; i < arrInfo.size(); i++)

{

if (arrInfo[i].diffIndex != currIndex)

{

res.push\_back(currIter);

currIndex--;

}

currIter++;

}

res.push\_back(arrInfo.end());

return (res);

}

void TInfo::partSort()

{

for (int i = 1; i < ranges.size(); i++)

{

sort(ranges[i - 1], ranges[i], &compareFuncScore);

}

}

void TInfo::sortData()

{

stable\_sort(arrInfo.begin(), arrInfo.end(), &compareFuncDiff);

ranges = getRanges();

partSort();

}

long long filesize(FILE \*stream)

{

long long pos, length;

pos = ftell(stream);

fseek(stream, 0L, SEEK\_END);

length = ftell(stream);

fseek(stream, pos, SEEK\_SET);

return length;

}

void TInfo::writeDataToFile()

{

FILE \*file;

file = fopen(fileName, "wb");

TPlayerInfo \*buff = new TPlayerInfo;

for (int i = 0; i < arrInfo.size(); i++)

{

\*buff = arrInfo[i];

fwrite(buff, sizeof(TPlayerInfo), 1, file);

}

fclose(file);

}

TArrInfo TInfo::getDataFromFile()

{

FILE \*file;

vector <TPlayerInfo> res;

TPlayerInfo \*buff = new TPlayerInfo;

file = fopen(fileName, "rb");

if (file == NULL)

{

file = fopen(fileName, "wb");

res.push\_back(ENDOFVECTORINFO);

}

else

{

int count = filesize(file) / sizeof(TPlayerInfo);

for (int i = 0; i < count; i++)

{

fread(buff, sizeof(TPlayerInfo), 1, file);

res.push\_back(\*buff);

}

}

fclose(file);

return res;

}

TInfo::TInfo()

{

fileName = "scoreBoard.dat";

arrInfo = getDataFromFile();

prevCountScore = NORECORD;

}

using namespace std;

typedef char\* TFileName;

typedef HSTREAM TFileDescr;

typedef HSAMPLE TFileDescrSample;

typedef struct \_TPlaylistItem

{

QWORD songLength;

TFileDescr fileDescr;

} TPlaylistItem;

typedef vector <TPlaylistItem> TPlayList;

typedef vector <TFileDescrSample> TSoundList;

#define SOUNDEATINDEX 0

#define SOUNDMOVEINDEX 1

#define SOUNDENDOFTHEGAMEINDEX 2

class TSound{

public:

void startPlayMusic();

void stopPlayMusic();

void changeSoundVol(int add);

void changeMusicVol(int add);

void playSound(int soundIndex);

void addToPlaylist(TFileName musicName);

TSound::TSound(TFileName soundOfEat, TFileName soundOfMove, TFileName soundOfEnd, TFileName endSongName);

void setNextMusicIndex();

TPlayList playList;

int currMusicIndex;

HCHANNEL soundChannel;

void playEndSong();

private:

float currSoundVol;

TSoundList soundList;

HSTREAM endSong;

HSTREAM getDescrFromFileStr(TFileName fileName);

HSAMPLE getDesrcFromFileSmp(TFileName fileName);

};

#include "stdafx.h"

#include "TSound.h"

HSTREAM TSound::getDescrFromFileStr(TFileName fileName)

{

HSTREAM res;

res = BASS\_StreamCreateFile(false, fileName, 0, 0, 0);

return res;

}

HSAMPLE TSound::getDesrcFromFileSmp(TFileName fileName)

{

HSAMPLE res;

res = BASS\_SampleLoad(false, fileName, 0, 0, 3, 0);

return res;

}

TSound::TSound(TFileName soundOfEat, TFileName soundOfMove, TFileName soundOfEnd, TFileName endSongName)

{

BASS\_Init(-1, 44100, BASS\_DEVICE\_STEREO, 0, NULL);

soundList.push\_back(getDesrcFromFileSmp(soundOfEat));

soundList.push\_back(getDesrcFromFileSmp(soundOfMove));

endSong = getDescrFromFileStr(endSongName);

currSoundVol = 1.0f;

}

void CALLBACK playMusic(HSYNC handle, DWORD channel, DWORD data, void \*user)

{

((TSound\*)user)->setNextMusicIndex();

BASS\_ChannelPlay(((TSound\*)user)->playList[((TSound\*)user)->currMusicIndex].fileDescr, true);

}

void TSound::startPlayMusic()

{

setNextMusicIndex();

BASS\_ChannelPlay(playList[currMusicIndex].fileDescr, false);

}

void TSound::addToPlaylist(TFileName musicName)

{

TPlaylistItem currSong;

currSong.fileDescr = getDescrFromFileStr(musicName);

currSong.songLength = BASS\_ChannelGetLength(currSong.fileDescr, BASS\_POS\_BYTE);

BASS\_ChannelSetSync(currSong.fileDescr, BASS\_SYNC\_END, 0, playMusic, this);

playList.push\_back(currSong);

}

void TSound::setNextMusicIndex()

{

int prevIndex = currMusicIndex;

currMusicIndex = rand() % playList.size();

if (playList.size() != 1)

while (currMusicIndex == prevIndex)

{

currMusicIndex = rand() % playList.size();

}

}

void TSound::playSound(int soundIndex)

{

soundChannel = BASS\_SampleGetChannel(soundList[soundIndex], false);

BASS\_ChannelSetAttribute(soundChannel, BASS\_ATTRIB\_VOL, currSoundVol);

BASS\_ChannelPlay(soundChannel, true);

}

void TSound::stopPlayMusic()

{

BASS\_ChannelPause(playList[currMusicIndex].fileDescr);

}

void TSound::changeMusicVol(int add)

{

float currVol;

BASS\_ChannelGetAttribute(playList[currMusicIndex].fileDescr, BASS\_ATTRIB\_VOL, &currVol);

currVol += (add / 100.0f);

if (currVol < 0.0f)

currVol = 0.0f;

if (currVol > 1.0f)

currVol = 1.0f;

for (int i = 0; i < playList.size(); i++)

{

BASS\_ChannelSetAttribute(playList[i].fileDescr, BASS\_ATTRIB\_VOL, currVol);

}

BASS\_ChannelSetAttribute(endSong, BASS\_ATTRIB\_VOL, currVol);

}

void TSound::changeSoundVol(int add)

{

float addedVol = (add / 100.0f);

currSoundVol += addedVol;

if (currSoundVol < 0.0f)

currSoundVol = 0.0f;

if (currSoundVol > 1.0f)

currSoundVol = 1.0f;

}

void TSound::playEndSong()

{

BASS\_ChannelPlay(endSong, false);

}