Предисловие.

Самым сложным в любом деле является первый шаг, а в любой книге – первые предложения. Хотя мои лекции и не являются книгой, как таковой, однако я постараюсь связно и последовательно рассказать Вам, дорогие читатели, про основные (местами банальные, местами откровенно скучные, но иногда очень интересные и необходимые) моменты современного геймдевелопмента. Уделять большую часть времени я буду, естественно, программированию, но достаточно подробно постараюсь описать и другие связанные с разработкой моменты, типа общей организации работы, положения в области в данный момент, некоторых экономических вопросов. Ранее данные документы писались для личного пользования, а для преподавания использовались презентации, но сейчас это можно назвать новой версией – расширенной и исправленной. Ну что же, это были общие слова, а теперь начнем☺

Занятие 1. Инструменты разработки игр. Необходимые приготовления. Простейший пример на языке C#.

Введение.

Итак, Вы поиграли во все современные шутеры, стратегии, квесты, аркады, RPG и еще несколько десятков всевозможных жанров игр. Кому-то нравятся высокобюджетные новинки, а некоторые, возможно, играли или до сих пор играют во что-то старое и культовое, так сказать олдскул. В любом случае, Вы загорелись идеей создать нечто новое, нечто СВОЕ. И это правильно! Места под солнцем всем хватит – игры выходят постоянно, а их создатели обзаводятся толпой фанатов и чемоданами с деньгами. И когда за дело возьметесь Вы, то Ваши игры непременно будут еще круче, фанаты одержимее, а чемоданы – толще. Если, конечно, все сделать правильно и по уму☺
И вот, в Вашей голове уже зреет мега-геймплей и Вы готовы создать свою «супер ММОРПГ» с «графоном круче чем в Кризисе и сюжетом круче чем в Скайриме!»©? Похвальное желание. И, главное - вполне реализуемое! Ведь сделать игру с помощью всех инструментов, что доступны современным разработчикам – это не сложно, а даже весело!
Самое главное, что нужно понять для себя еще до начала разработки – не стоит бояться ошибок и «падений». Не стоит бояться выглядеть глупо в глазах других разработчиков – ведь вы только начинаете, все успехи у Вас еще впереди! Из любого провала Вы должны извлекать опыт, зерно истины и анализировать свои ошибки с целью не допустить их в будущем.

На самом деле даже имеет место существовать мнение, что полезность одной ошибки для развития человека, как личности и специалиста, равна десяти успехам. Хотя в утверждении и есть доля лукавства, однако по большей части оно абсолютно верно! Не зря самыми успешными, богатыми и знаменитыми людьми являются индивидуумы, всего добившиеся сами и начинавшие с нуля, на энтузиазме и вере в будущее – Билл Гейтс, Стив Джобс, Марк Цукерберг (примеры из IT, но есть и другие). Сколько этапов они прошли, сколько неудач перетерпели – известно только им, но именно благодаря этому сейчас они те, кем они и являются.

Выбор инструмента.

Успех игры определен множеством составляющих. Это и проработанный, интересный геймплей, и оригинальный сюжет, и качественный контент – картинки, модели, звуки и музыка.

Возможный возглас из зала: “А как же атмосфЭра!?”
*Ответ: Атмосфера – это не составляющая игры, а показатель удачного сочетания составляющих. Она естественным образом появляется, когда сценарист, художник, моделлер, композитор делают вместе один мир, а не каждый свой собственный. Ваша задача как программиста (читай – самого психологически адекватного) – не дать им выжить при этом из ума. И написать безбажный код, чтобы не проср*ть всю эту атмосферу, что будет крайне обидно ☺*

email: imalexsmith@ya.ru
skype: imalexsmith

Но что по факту представляет собой “Игра”? Игра – это программа. Точно так же как кино – это бобина с пленкой или видеофайл, а музыка – аудиосигнал.
Есть масса игр, которым вообще не понадобились ни художники, ни сценаристы. Но без программистов разработка точно никуда не сдвинется. Потому что именно МЫ делаем игру игрой.
Создать игру - значит написать программу. Всю ту красоту, что сделали ваши работники (нашли в интернете, украли из других игр, вы сами накрапали в Пэинте – нужное подчеркнуть) надо привести к общему знаменателю, заставить взаимодействовать друг с другом и дать реальную возможность погрузить игрока в этот новый, волшебный мир Вашей игры.
Для реализации вышеописанного нам необходим мощный и быстрый, но в тоже время простой и гибкий инструмент. Как раз таким и является Microsoft XNA Framework 4.0, на примере которого мы будем учиться и про который подробнее я расскажу ниже. В будущем Вы свободны выбирать инструмент в зависимости от своих предпочтений и навыков, конкретных задач и сроков разработки – везде есть свои плюсы и минусы, но очень сложно рассказать сразу про все. Да и новинки в данной области появляются достаточно часто, поэтому любые письменные советы по данному вопросу будут заведомо являться устаревшими. Однако краткую спецификацию, которая устоялась уже достаточно давно, я позволю себе обозначить.

Краткая классификация инструментов для создания (программирования) игр.

Название	**Описание**	**Примеры**
Игровые конструкторы	Это наименее интересный для нас тип программного обеспечения для создания игр. Весь процесс разработки – перетаскивание графических элементов. Игру можно создать, не написав ни строчки программного кода. Многие советую начинать обучаться с таких конструкторов, однакоя лично я бы Вам этого не советовал – польза нулевая.	Самый известный пример, пожалуй, это ***Game Maker***. Ходят целые легенды про его баганутось и корявость, однако надо признать, что функционал в нем действительно неплохой. Можно даже писать свои скрипты, компилировать игру под несколько платформ – одако в силу глючности подходит только для прототипов и обучения.
Игровые движки	Тут, чаще всего, тоже присутствует некий графический интерфейс и перетаскивание объектов. Но самое интересное начинается после того, как все окружение расставлено, все элементы перетащены. Начинается программирование. В игровых движках программирование построено на так называемых скриптах, из которых мы можем получить доступ к самому движку и расширять его функцинальность. Скрипт – простенькая программа(если выражаться на профессинальном языке – функция, метод), запускаемая по некоему условию. Открыть дверь, добавить вещь в инвентарь – как примеры. Но как таковая, программная часть игры скрыта от глаз программиста – т.е. сам движок, «основная часть айсберга».	Приведу в качестве примера один из самых популярных на данный момент движков – ***Unity3D***. Это целая платформа для разработки мультимедия приложений – и презентаций, и 3Д миров, и игр, да даже для бизнес-приложений он может пригодится! Большое коммьюнити, много примеров, компиляция под несколько платформ, в том числе и мобильных, полная свободна в написании кода, т.е. расширении функционала. Однако, все равно остаеться проблема закрытости основного ядра, скудный набор бесплатных «печенюшек». Иногда попадаются баги в самом движке, что уже не исправишь.

Набор графических библиотек + язык программирования (+дополнительные фреймворки)	Самый интересный, на мой взгляд, подход. При таком подходе создание игры – это написание настоящей, полноценной программы, с несколькими незначительными условиями. К примеру должен присутствовать игровой цикл, для вывода графики должен быть использован специализированный набор библиотек. С другой стороны, вам доступны (без особых ухищрений) все возможности того ЯП, на котором вы привыкли писать. Такой подход является самым мощным и функционально полноценным, но и самым сложным!	Ну что сказать. Это настоящее, 100% программирование. Из наиболее распространенного и известного – ***C++ & DirectX***, которое в ходу уже 15 лет. Полная свобода для фантазии, полный контроль за работой приложения. А еще несколько лет на обучение и непонятно сколько, чтобы написать что-то более менее красивое☺ Для начала пути в геймдеве не самый лучший выбор.

Microsoft XNA Framework – определения и примеры.

Выше было упомянуто такое непонятное словосочетание, как Microsoft XNA Framework 4.0. Это, как видно из названия, фреймворк, который очень сильно упрощает жизнь разработчикам игр, а в особенности начинающим. Чаще, правда, используется просто термин XNA *(русскоязычный сленг – хна, как краску для волос☺)*, который в свою очередь может обозначать целый ряд понятий. Итак, перечислим их:

- ***XNA Game Studio.*** Интегрированая среда разработки (IDE), в которой мы пишем программную часть игры. Является всего лишь надстройкой, расширением, специальной версией над Microsoft Visual Studio C# Express с некоторыми дополнительными возможностями, необходимыми специально при создании игр. К примеру выбор целевого устройства, на котором будет запущена программа – подключенный XBox, эмулятор Win Phone, операционная система Windows и т.д.
- ***XNA Framework.*** Наиболее часто подразумеваемое понятие. Набор библиотек с кодом, необходимых для создания игры. В ней уже есть:
  1. упрощенный доступ к DirectX;
  2. набор вспомогательных классов/структур для упрощения математических расчетов;
  3. специфические контент-импортеры и контент-процессоры системы XNA;
  4. каркас игрового приложения (классы Game, GameComponent, GameWindow и т.п.) и т.д.
- ***XNA Framework Redistributable.*** Набор библиотек, необходимых для запуска игры на машине конечного пользователя. Совсем не тоже самое, что XNA Framework.
- ***Код программы.*** Код, который мы пишем в окне Visual Studio(Game Studio) с использованием .Net Framework, языка C#, XNA Framework.

Платформа XNA, по моему мнению, является крайне интересной и перспективной, подходящей именно для студентов и школьников, начинающих команд разработчиков. Ведь благодаря особым принципам *(которые нам совсем не обязательно сейчас затрагивать, да и есть неплохой шанс, что не все поймут☺)* с помощью XNA можно создавать приложения сразу для нескольких платформ: Windows, Xbox, Windows Phone. А упрощенный доступ к DirectX значительно увеличивает скорость разработки. К примеру, вывод 3Д модели на экран в XNA можно реализовать одной строкой! А можно же и в цикле, применяя множество эффектов и шейдеров, со сложной обработкой физики – полёт фантазии ограничен, пожалуй, только мощностью вашего железа.

Как было сказано выше, мир IT – стремительно и неумолимо развивается. Быть в курсе последних событий иногда очень и очень сложно. Так и тут – когда я только начал изучать хну, она поддерживала только обозначенные три платформы. Однако за этот год я узнал о планах Mono по портированию XNA на Linux (проект называется MonoGame – можно поискать в интернете), а мелкомягкие выпустили Windows 8. С разработкой на XNA под последний я лично не сталкивался, но говорят, что все не так радостно и два продукта Microsoft не очень хорошо уживаются друг с другом. Однако можно быть точно уверенным, что для Xbox и почти наверняка под Windows Phone разрабатывать можно будет спокойно.

Вывод: XNA является хорошим, сбалансированным решением между созданием игры с помощью игрового движка, где может быть недостаточный функционал или ошибки в логике самого движка, и созданием игры с помощью более классической спайки DirectX/OpenGL & C++/C#/Java, что более требовательно к

профессионализму разработчика и времени разработки. Перед последними может наблюдаться небольшой проигрыш в скорости, но он достаточно незначителен. Намного проще потерять на контроле за памятью, сборке мусора, отсечении невидимых частей – очень много моментов, от слежения за которыми XNA нас освобождает.

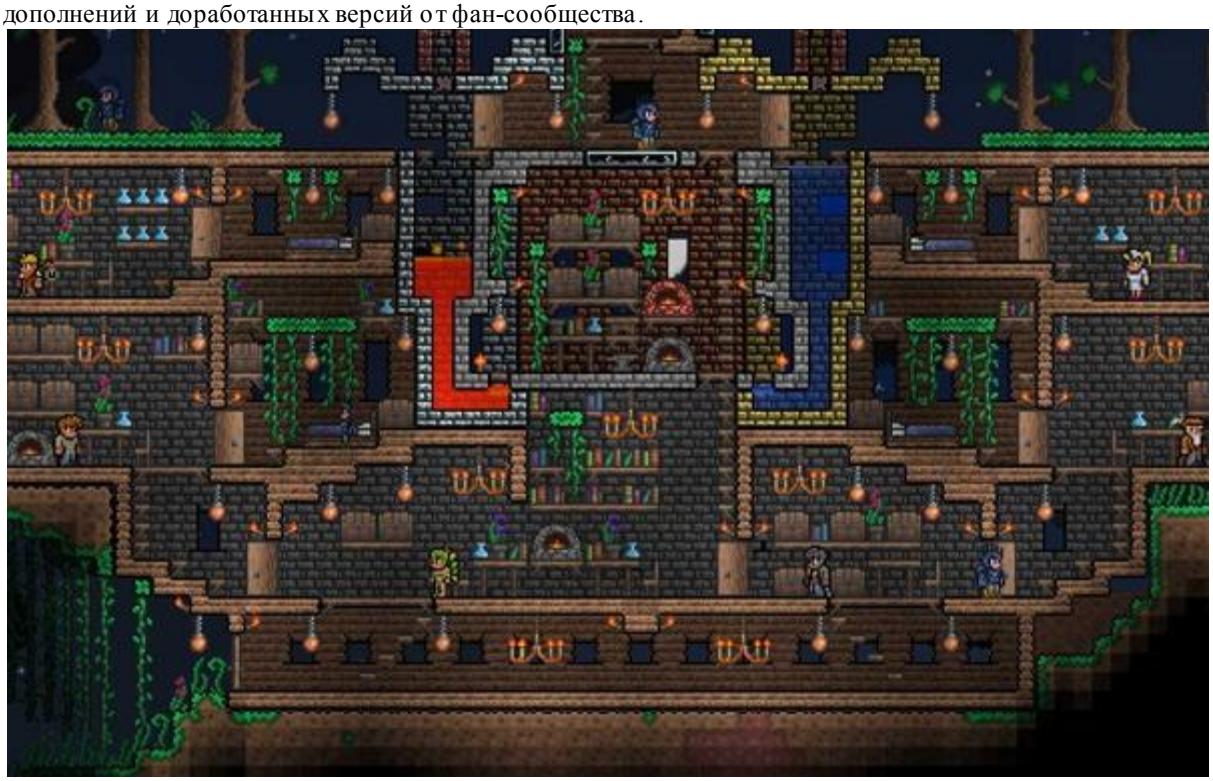
Примеры игр, разработанных с применением XNA Framework.

Lost Empire:Immortals. Глобальная экономическая стратегия в космическом антураже.

Magicka. «Ну очень эпическая игра». Командная РПГ с яркой, сочной картинкой, хорошим юмором и оригинальным геймплеем. Так сказать маст хев.

Bastion. Диабло—подобная РПГ с красивой графикой и оригинальными геймплейными решениями. Простенько и со вкусом.

Terraria. Про нее, думаю, многие слышали. Называют «майнкрафт в 2д». Интересная песочница с кучей дополнений и доработанных версий от фан-сообщества.



Инструментарий для разработки игр с помощью XNA Framework 4.0.

Для программирования с использованием XNA Framework 4-ой версии используются Visual Studio версии 2010. Тут сразу же надо определиться, интересно ли Вам разрабатывать под платформу WinPhone, потому что от этого зависит конкретная разновидность студии, которую вам нужно будет установить. От себя посоветую установить обе. Первая позволяет писать под WinPhone – во-первых, платформа интересная и развивающаяся, во-вторых, такая версия позволит писать для ПК и Xbox, но не наоборот, и наконец, в-третьих – её проще установить, все в один этап☺ Однако такая студия не позволит писать «обычные»

приложения для винды, только игры. Вторая позволяет писать приложения со стандартными интерфейсами для настольных ПК, но никак не получиться писать на телефоны. Подробности установки по обоим вариантам:

1) ***Visual Studio 2010 Express для Windows Phone.*** Вариант номер один, позволяющий разрабатывать под Windows, Xbox и Windows Phone. Установка простейшая – скачать маленький файл размером в несколько килобайт и запустить. Все, дальше процесс полностью автоматизированный, после проверки необходимые компоненты скачаются самостоятельно, без Вашего участия. Так же есть возможность скачать образом.
2) ***Visual C# 2010 Express.*** Вариант номер два, только две платформы и несколько этапов установок. Первый этап – скачать и поставить студию из ссылки. Установка онлайн или путем скачивания образа, на Ваш выбор. Второй этап – скачиваем и ставим ***Microsoft XNA Game Studio 4.0***. После этого при запуске Visual C# 2010 Express у нас появится возможность создавать проекты нового типа – игры для Windows и игры для Xbox.

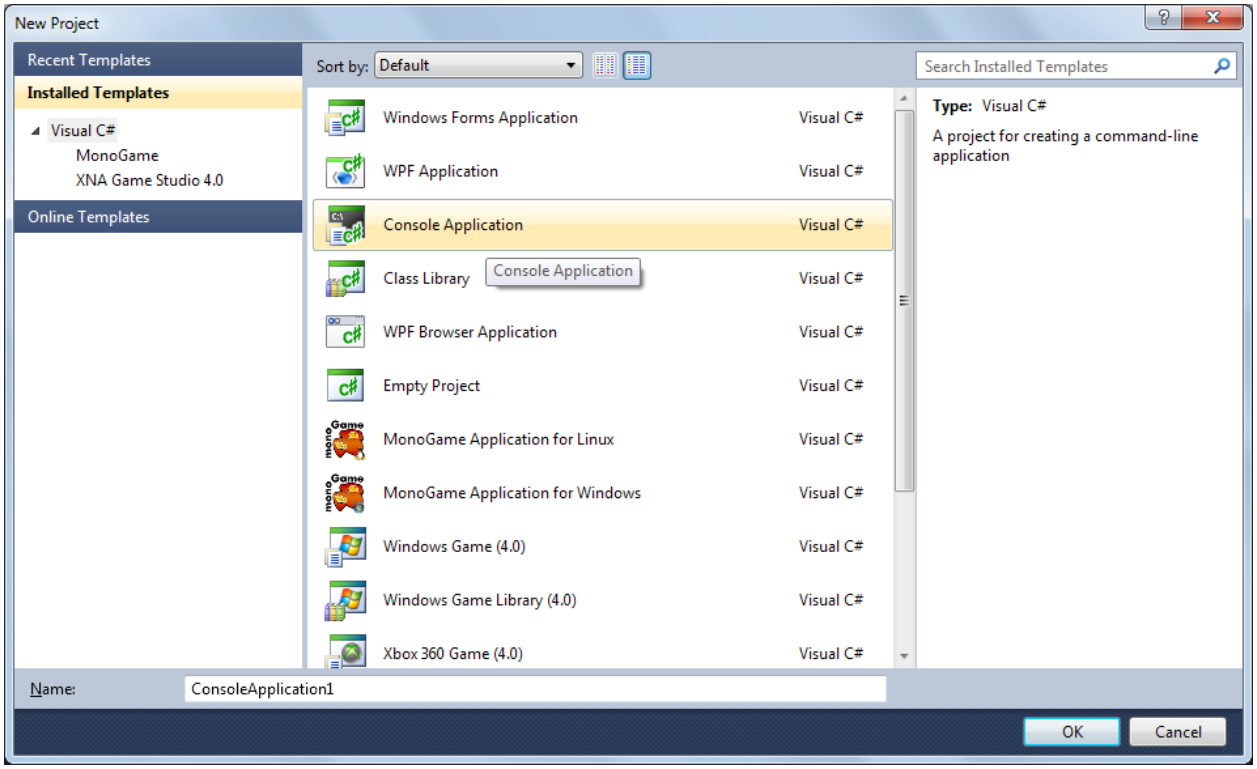
Кстати, рекомендую после установки все обновить по возможности, так как иногда студия может ругаться на несоответствие версий. Обновления проходят через Windows Update (Центр обновления Windows).

Далее будет немного лести в честь доблестной компании Microsoft, а конкретно их такого замечательного продукта, как Visual Studio☺ В своей не такой, чтобы долгой, но и не короткой, профессиональной деятельности на стезе программирования я познакомился со многими интегрированными средами для разработки. По научному IDE (от англ. Integrated development environment). Так вот, мне кажется, что именно в студии умудрились все сбалансировать – она очень мощная и функциональная, однако, с другой стороны, её интерфейс не перегружен и интуитивно понятен. Про подсветку синтаксиса и подсказки по коду вообще молчу – как будто читает мысли. Особенно её начинаешь любить, когда необходимо писать программы в специальных средах для математической разработки ☺

Простейшее консольное приложение на C#.

Для нормальной работы над игрой (т.е. для уверенного использования дополнительного фреймворка XNA) нужно иметь хотя бы минимальные навык работы с языком, в котором происходит этот самый процесс разработки. Ну, думаю все уже догадались, что речь идет про язык C# ☺ Ниже мы напишем первое, простейшее, но полноценное консольное приложение – т.е. приложение без графического интерфейса, взаимодействие с которым (ввод-вывод данных) может осуществляться только с помощью текста. Консольные приложения – хороший способ проверять, тестировать какие-то отдельные алгоритмы и так же хорошо в приложениях, где интерфейс вовсе не нужен – некоторые приложения, работающие с серверами, базами данных, возможно, большими объемами текста.
Приведу последовательный порядок, что делать и куда нажимать:

1. Чтобы запустить Visual C# Express, в меню «Пуск» укажите «Все программы» и затем выберите «Microsoft Visual C# Express Edition». Если на рабочем столе уже есть нужный вам значок (ярлык), просто щелкните по нему.
2. Для создания нового проекта откройте меню File («Файл»), щелкните New project («Новый проект») и выберите тип проекта. Сегодня мы попрактикуемся на простейшем консольном приложении, т.е. ***«Console Application».***



3. Попробуем создать новое консольное приложение и нажмем «ОК». Откроется студия разработки с множеством окон. Центральное окно предназначено для программирования (написания программного кода), здесь будет показан автоматически созданный код класса **Program.cs**. Удалите его полностью и наберите текст программы, приведенной ниже. Как бы это не звучало дико, но именно наберите руками – когда Вы только учитесь языку и среде, нужно как можно больше писать с нуля.

```
using System;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
            Console.Read();
        }
    }
}
```

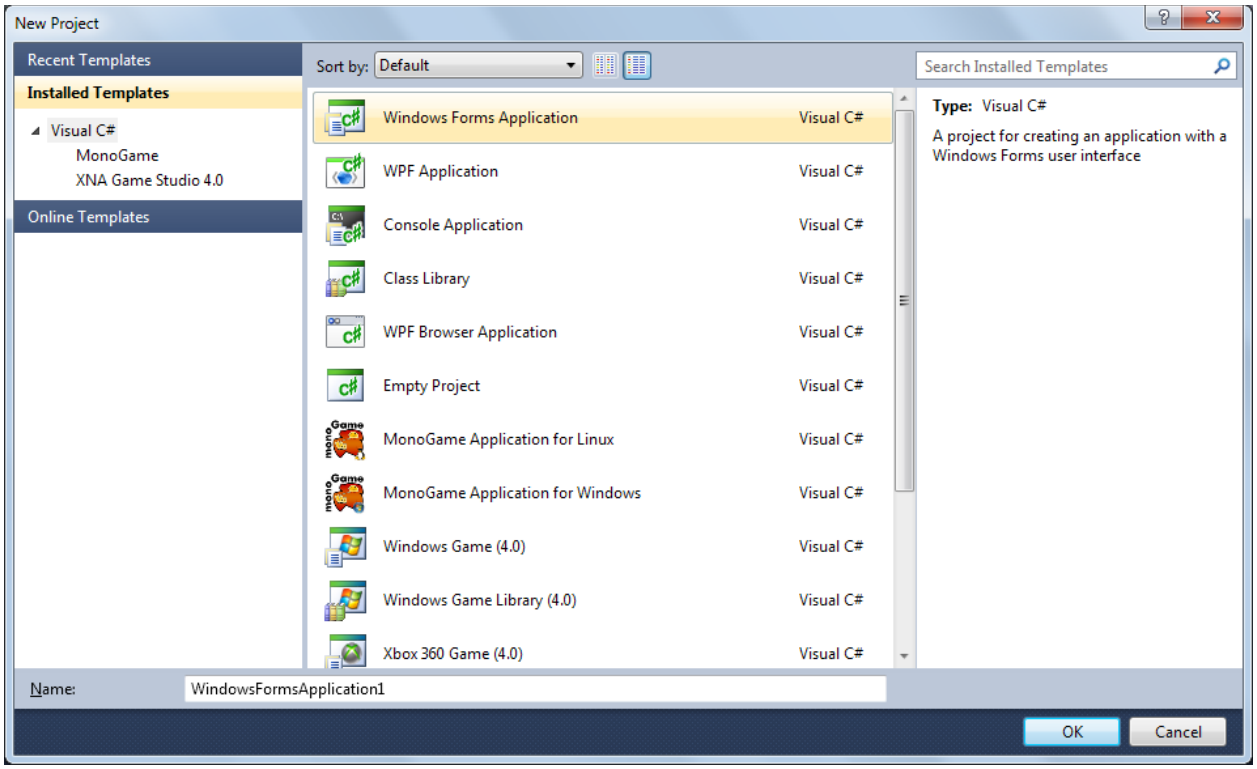
4. Убедитесь, что набранный вами текст в точности соответствует данному. Затем нажмите кнопку «Выполнить/Debug» (или клавишу F5).
5. Если в коде программы обнаружится ошибка, Вы получите предупреждение. Обратите внимание на то, что в языке программирования C# в конце каждого предложения или выражения должна стоять точка с запятой. Попробуем удалить последнюю точку с запятой («;») и затем выполнить программу – на экране появится сообщение об ошибке. Внимательно на него посмотрите и запомните его – еще много-много раз Вы с ним увидитесь☺ Можете нажать «нет/no».

Простейшее приложение с графическим интерфейсом на C#.

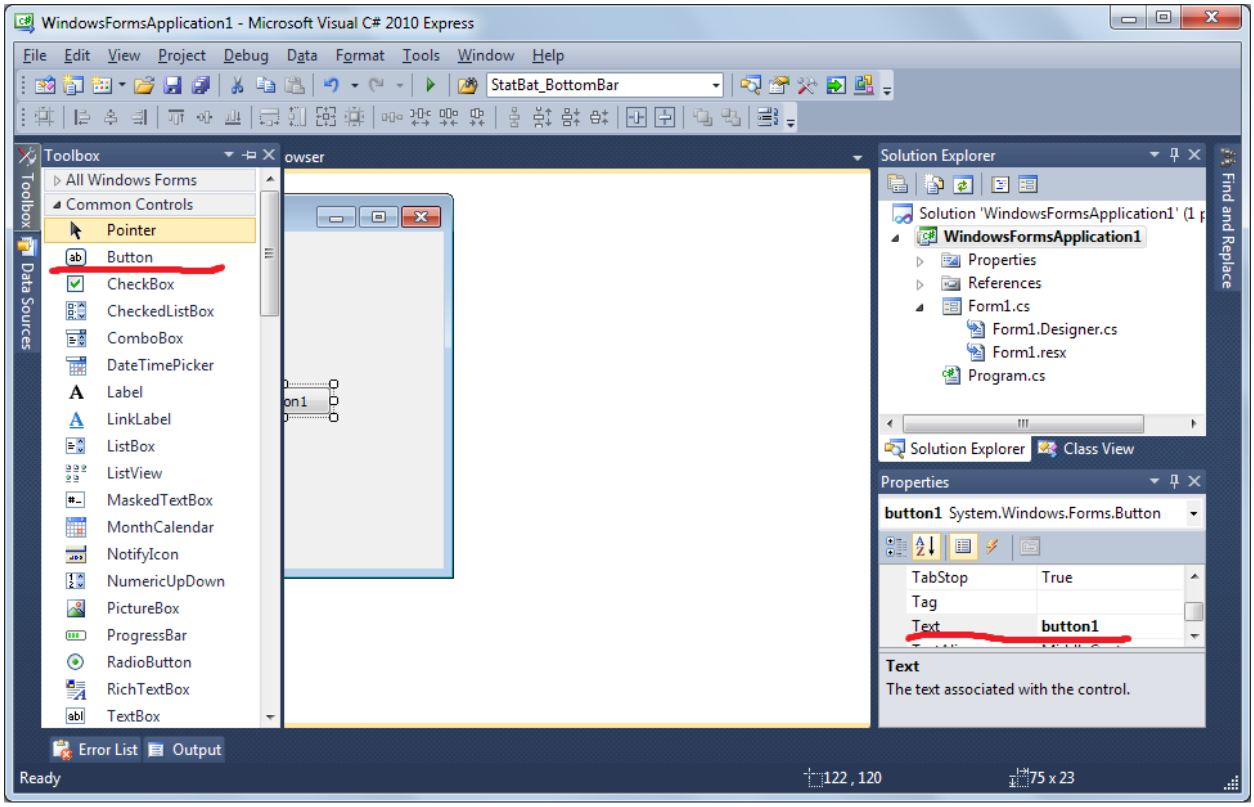
Если Вы захотите писать программы, похожие на привычные приложения Windows, то Вам понадобиться что-то пофункциональнее обычных консольных приложений, а именно приложения с ***GUI(от англ.***

Graphical User Interface) – приложения с графическим интерфейсом пользователя. Они позволяют задействовать кнопки, списки, текстовые поля, меню, окна сообщений и множество других «элементов управления» (на слеге «контролов»). Элементы управления - это то, что Вы помещаете в форму, т.е. в окно Вашего приложения. Так же, как и в случае консольного приложения, приведу порядок создания:

1. Чтобы запустить Visual C# Express, в меню «Пуск» укажите «Все программы» и затем выберите «Microsoft Visual C# Express Edition». Если на рабочем столе уже есть нужный вам значок (ярлык), просто щелкните по нему.
2. Для создания нового проекта откройте меню File («Файл»), щелкните New project («Новый проект») и выберите тип проекта. Теперь нам нужно приложение типа ***Windows Forms Application.***



3. После нажатия кнопки «Ок» откроется окно редактора, в центре которого будет пустая форма. Далее добавим контрол типа Button. На рисунке он виден слева во вкладке ToolBox, добавление происходит путем перетаскивания элемента на форму или просто двойного клика. После этого изменим свойство Text на «Привет», т.е. справа внизу увидим все свойства объекта, и нажмем на поле, где написанно button1, изменив этот текст на нужный



4. Теперь два раза нажмем на добавленную кнопку, после чего должен открыться экран редактора кода с курсором внутри нужного нам метода – а именно метода, который выполнится при нажатии на кнопку. Хотя я советую удалить все, что там уже есть и написать приведенный код сначала.

```
using System;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            MessageBox.Show("И тебе здравствуй!");
        }
    }
}
```

5. Убедитесь, что набранный вами текст в точности соответствует данному. Затем нажмите кнопку «Выполнить/Debug» (или клавишу F5).
6. Если в коде программы обнаружится ошибка, Вы получите предупреждение. Обратите внимание на то, что в языке программирования C# в конце каждого предложения или выражения должна стоять точка с запятой. Попробуем удалить последнюю точку с запятой («;») и затем выполнить программу – на экране появится сообщение об ошибке. Внимательно на него посмотрите и запомните его – еще много-много раз Вы с ним увидитесь☺ Можете нажать «нет/no».

Ура, можете похвалить себя – первые шаги сделан и первые приложения успешно написаны. Если Вы совсем не знакомы с языком C#, то я рекомендую прочитать книгу ***Мартина Дрейера, «C# для***

школьников» (из которой я взял некоторые определения), а точнее ***часть 2, Учимся общаться с компьютером.*** Это около 50 страниц очень хорошо объясненного материала, с примерами и пояснениями.

Заключение.

Вот и завершился первый урок. Подведем итоги, с чем же мы ознакомились сегодня:

1. С возможными инструментами для создания игр и примерами таких инструментов.
2. С общей характеристкой платформы XNA и примерами игр на ней.
3. С необходимым инструментарием для разработки приложений, узнали где его брать.
4. Было написано простейшее приложение на языке C# и приложение с графическим интерфейсом.

Домашнее задание.

Установить обозначенный инстументарий для разработки (Visual Studio). Прочитать, по возможности, книгу ***Мартина Дрейера, «C# для школьников»*** или аналогичную, расчитанную на начальный уровень знаний.

email: imalexsmith@ya.ru
skype: imalexsmith