Київський національний торговельно-економічний університет

Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка програмного забезпечення «Happy Smile»»

Студента 4 курсу, 6 групи,
спеціальності 121
«Інженерія програмного
забезпечення» _____ Перетокіна
підпис студента Володимира Ігоревича

Науковий керівник
кандидат технічних наук,
професор _____ Пашорін Валерій
підпис керівника Іванович

Гарант освітньої програми
кандидат технічних наук,
доцент _____ Цензура Микола
підпис керівника Олександрович

КИЇВ – 2020

Київський національний торговельно-економічний університет

Факультет *інформаційних технологій*
Кафедра *інженерії програмного забезпечення та кібербезпеки*
Освітній ступінь *бакалавр*
Спеціальність *121 «Інженерія програмного забезпечення»*
Спеціалізація / освітня програма *121 «Інженерія програмного забезпечення»*

Затверджую
Зав. кафедри інженерії
програмного забезпечення
та кібербезпеки
Криворучко О. В.
"7" листопада 2019 р.

Завдання
на випускну кваліфікаційну роботу студентові

Перетокіна Володимира Ігоревича

1. Тема випускної кваліфікаційної: *«Розробка програмного продукту «Happy Smile»».*
Затверджена наказом ректора від *«02» грудня 2019 р. № 4112*
2. Строк здачі студентом закінченої роботи _____
3. Цільова установка та вихідні дані до роботи

Мета роботи *аналіз засобів, методів, побудови, розроблення і застосування довідково-інформаційної системи, яка реалізує довідки та виводить результати на екран.*

Об'єкт дослідження *дослідження програмних додатків, що сприяють розвитку реакції*

Предмет дослідження *методи та алгоритми розробки Windows – додатку.*

4. Консультанти роботи із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

4. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. Аналіз задачі та програмних засобів

1. Загальна частина

1.1. Постановка задачі

1.2. Дослідження і аналіз об'єкту програмування

1.3. Використані прогамні засоби

1.4. Вимоги до апаратного та програмного забезечення

Висновок до розділу 1

РОЗДІЛ2.Створення додатку

2.1. Створення програми

2.2. Опис програми та її алгоритмів

2.3. Технічні засоби

2.4. Інструкція користувача

Висновки до розділу 2

РОЗДІЛ 3. Економічна частина

3.1. Оцінка вартості продукту

3.2. Розрахунок собівартості програмного продукту

3.3. Розрахунок ціни програмного продукту

6. Календарний план виконання роботи

№ пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		за планом	фактично
1	2	3	4
1.	*Вибір теми випускної кваліфікаційної роботи*		
2.	*Вступ та перелік літературних джерел*		
3.	*Розділ 1. «Аналіз задачі та програмних засобів»*		
4.	*Розділ 2. «Створення додатку»*		
5.	*Розділ 3. «Економічна частина»*		
6.	*Висновки*		
7.	*Здача випускної кваліфікаційної роботи на кафедру (перша перевірка)*		
8.	*Підготовка автореферату та презентації доповіді*		
9.	*Попередній захист випускної кваліфікаційної роботи*		
10.	*Зовнішнє рецензування випускної кваліфікаційної роботи*		
11.	*Здача прошитої випускної кваліфікаційної роботи на кафедру*		
12.	*Публічний захист випускної кваліфікаційної роботи*		

7. Дата видачі завдання «___»_____20__ р.

8. Науковий керівник випускної кваліфікаційної роботи Котенко Н.О.

9. Гарант освітньої програми Цензура М. О.

10. Завдання прийняв до виконання студент Перетокін В.І.

11. Відгук керівника випускної кваліфікаційної роботи

____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____

Науковий керівник випускної кваліфікаційної роботи ____________________

(підпис, дата)

Відмітка про попередній захист Котенко Н.О _________________

(ПІБ, підпис, дата)

12. Висновок про випускну кваліфікаційну робота

Випускна кваліфікаційна робота студента _______________________________

(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми ____

(прізвище, ініціали, підпис)

Завідувач кафедри ____

(підпис, прізвище, ініціали)

«_____»_________________20 ___ р.

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (50с., 18 рис., 6 таб., 1 додаток).

Об’єкт розробки – програмний додаток під назвою «Happy Smile» –гра-додаток для стаціонарних комп’ютерів та телефонів?, що тренує швидкість реакції, яку можна випускати у повноцінне використання для людей будь-якого віку.

Завданням дослідження є: розробка гри-додатку, можливість гри на будь-яких пристроях, можливість монетизації.

В процесі розробки було використано мову програмування C#, а також C#

– модулі. Visual Studio – для розробки графічного інтерфейсу. UWP – для створення додатку.

Ключові слова: гра-додаток, об'єктно-орієнтоване програмування, розробка графічного інтерфейсу, C#, Visual Studio, UWP.

ANNOTATION

Qualification work contains explanatory note (50 pages, 18 pictures, 6 tables, 1 addion).

An object of development – «Happy Smile» is a developmental game application for desktop computers and phones that can be released for full use for people of all ages. The task of research is: development of visual addition of advertising agency of "Argo", creation of individual included in addition, determination of catalogue of commodities of enterprise.

In the process of development, a programming of C#, and also C# language – modules was used. Visual Studio – for development of graphic interface. UWP – to create an application.

Keywords: game, application , object-oriented programming, development of graphic interface, C#, Visual Studio, UWP.

ЗМІСТ

					КНТЕУ 121 06-20.БР		
Зм.	*Аркуш*	*№ докум*	*Підпис*	*Дата*			
Зав. кафедри		*Криворучко О.В.*			*Розробка програмного забезпечення «Happy Smile»*	*Стадія*	*Аркуш*	*Аркушів*
Керівник		*Пашорін В.І..*				*З*	*8*	*43*
Гарант		*Цензура М.О.*				Факультет інформаційних технологій, 4 курс, 6 група		
Розроб.		*Перетокін В.І.*			*Зміст*			

ВСТУП

Кожен день людині необхідно розвиватися, задля того щоб ставати краще. У цьому їй допомагае безліч гаджетів, задач, та багато іншого. Також, людині необхідна гарна реакція, задля того, щоб швидко реагувати на події, які можуть несподівано статися, зловити телефон, що випадає з рук, або швидкий автомобіль, що рухається порушуючи правила дорожнього руху. Одним з додатків, що зможуть допомогти програмний продукт під назвою «Happy Smile», який буде розробений у даному дипломному проекті.

Об’єкт дослідження – процес розвитку.

Предмет дослідження – розроблені користувацькі додатки для розвитку швидкості мишлення та реакції.

Метою дослідження кваліфікаційної роботи є покращення навичок у мові програмування C# та UWP, та з їх допомогою створення гри на реакцію.

Завдання дослідження:

• Розробка гри;

• реалізація логіки на перехід на нові рівні;

• підготовка гри до релізу.

Методи дослідження будуть використанні спостереження, дослідження та аналіз отриманих даних.

РОЗДІЛ 1.
АНАЛІЗ ЗАДАЧІ ТА ПРОГРАМНИХ ЗАСОБІВ

1 Загальна частина

1.1 Постановка задачі

Завдання дипломної роботи – розробити гру для розвитку навичок «Happy Smile». В програмі повинні бути реалізовані наступні функції:.

1) Зчитування інформації з екрану.
2) Перехід між рівнями.
3) Таблиця рекордів користувача.
4) Необмежена кількість рівнів.

Користувач повинен серед багатьох облич вибрати лише веселе, та з кожним рівнем задача буде лишь складнішою, через зростання рівню складності. Програма повинна мати зручний інтерфейс і забезпечувати ряд операцій для допомоги користувачу під час використання програми.

Коли гравець програє він матиме змогу подивитися таблицю рекордів та почати гру з самого початку.

Гра буде вважатися закінченою, коли гравець за вівдведенний срок не встигне вибрати необхідне обличчя.

Після запуску, для спрощення дій користувача, програма надає за замовчуванням рандомне ім’я з можливістю його зміни, для подальшого ведення рахунку очок.

Гра розрахована на широку аудиторію користувачів, без вікових обмежень.

1.2 Дослідження і аналіз аналогів додатку

Навчальна або освітня гра — це вид гри, розроблений з метою кращого засвоєння навчального матеріалу за допомогою ігрових елементів. Навчальні ігри покликані допомогти людині у вивченні конкретних об'єктів, розробці концепцій, осмисленні історичних подій, культурного розвитку, а також освоєнні нових вмінь в ході ігрового процесу.

Для гри «Happy Smile» є багато ігор аналогів, деякі приклади приведенні нижче.

Знайти кота

Ця гра має спільну рису з грою з мого дипломного проекту. Ця риса необхідність уважно дивитися на картинку з ціллю пошуку певного предмету.

Рис.1.1. Гра «Асоціації»

Where is Waldo?

Ця книга є дуже популярною у країнах Европи серед дітей. Її головною задачею є знайти персонажа Waldo, який захований серед людей, предметів тощо.

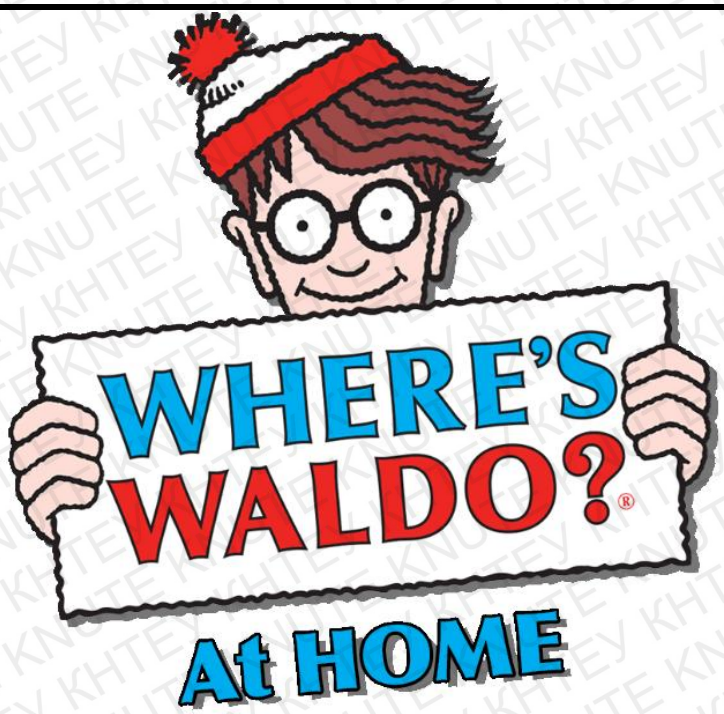

Рис. 1.2. Книга «Where is Waldo?»

1.3 Використані програмні засоби

Виходячи із задач, що були поставлені у моему диплоному проекті та аналізу схожих додатків для розробки свого проекту я обрав мову програмування C# та UWP.

C # - це програма програмування багатопарадигми загального призначення, яка охоплює сильний набір тексту, лексично обмежений, імперативний, декларативний, функціональний, родовий, об'єктний -орієнтована (на основі класу) та компонентно-орієнтована дисципліна програмування.

Він був розроблений близько 2000 року Microsoft в рамках своєї ініціативи .NET, а згодом затверджений як міжнародний стандарт Ecma (ECMA-334) у 2002 році та ISO (ISO / IEC 23270) у 2003 році. Моно - назва вільного та відкритого - ресурсний проект з розробки компілятора та часу виконання мови. C # - одна з мов програмування, розроблена для загальної мовної інфраструктури (CLI)

C # був розроблений Андерсом Хейлсбергом, а його команду з розробки в даний час очолює Мадс Торгерсен. Найновіша версія - 8.0, яка вийшла у 2019 році разом із Visual Studio 2019 версії 16.3.

Універсальна платформа Windows (UWP) - це обчислювальна платформа, створена Microsoft та вперше представлена в Windows 10. Мета цієї платформи - допомогти розробити універсальні додатки, які працюють у Windows 10, Windows 10 Mobile, Xbox One та HoloLens без необхідності. переписаний для кожного. Він підтримує розробку додатків для Windows за допомогою C ++, C #, VB.NET та XAML. API реалізований у C ++ та підтримується в C ++, VB.NET, C #, F # та JavaScript. Розроблений як розширення до платформи Windows Runtime, вперше представленої в Windows Server 2012 та Windows 8, UWP дозволяє розробникам створювати додатки, які потенційно можуть працювати на кількох типах пристроїв.

UWP не націлений на системи, що не належать Microsoft. Цією задачею займається Xamarin.Forms, API з відкритим кодом, створений Xamarin, дочірньою компанією Microsoft з 2016 року.

Як підказує назва платформи, вона є універсальною - універсальної для всіх пристроїв екосистеми Windows 10. А це звичайні дестопи, планшети, мобільні пристрої, пристрої IoT (інтернет речей), Xbox, пристрої Surface Hub. І додаток UWP може однаково працювати на всіх цих платформах, якщо на них встановлена Windows 10.

Основні переваги програми UWP:

- Безпека. Додатки UWP оголошують, до яких ресурсів пристрою та даними вони здійснюють доступ. Користувач повинен дозволити такий доступ.
- Можливість використовувати загальний API на всіх пристроях під управлінням Windows 10.

- Можливість використання можливостей окремих пристроїв і адаптації користувальницького інтерфейсу до різних розмірами екранів, дозволами і плотностям точок.
- Доступність в Microsoft Store, на всіх пристроях (або тільки тих, яких ви вкажете), що працюють під управлінням Windows 10. У Microsoft Store передбачено кілька способів монетизувати ваш додаток.
- Можливість встановлюватися і віддалятися без ризику для комп'ютера або "деградації" ПО.
- Захопливість: можливість використовувати живі плитки, push-повідомлення і призначені для користувача дії, які взаємодіють з тимчасовою шкалою Windows і функцією "Продовжити з місця зупинки" Кортан, для підтримки інтересу користувачів до додатка.
- Можливість програмування на C #, C ++, Visual Basic і JavaScript. Для призначеного для користувача інтерфейсу можна використовувати XAML, HTML або DirectX.

Безпека

У маніфестах додатків UWP оголошуються можливості пристрою, необхідні додатком, – наприклад, доступ до мікрофона, годинних, веб-камері, USB-пристроїв, файлам і т. Д. Перш ніж додаток отримає доступ до можливості, користувач повинен підтвердити і дозволити такий доступ.

Загальна поверхня API для всіх пристроїв

У Windows 10 вперше з'явилася універсальна платформа Windows (UWP), яка надає загальну платформу додатків на будь-якому пристрої, що працює під управлінням Windows 10. Основні API UWP однакові на всіх пристроях Windows. Якщо додаток використовує тільки основні API, воно буде працювати на будь-якому пристрої під управлінням Windows 10 незалежно від того, під який пристрій воно розроблялося – ПК, Xbox, гарнітуру змішаної реальності і т.п.

Додаток UWP, написане на C ++ / WinRT або C ++ / CX, має доступ до API-інтерфейсів Win32, які входять до складу UWP. Ці API Win32 реалізуються усіма пристроями Windows 10.

Пакети SDK розширень надають унікальні можливості для конкретних типів пристроїв.

Якщо ви розробляєте додаток для універсальних API, воно зможе працювати на всіх пристроях під управлінням Windows 10. Але якщо ви хочете, щоб ваше додаток UWP могло користуватися перевагами API конкретних пристроїв, це також можливо.

Пакети SDK розширень дозволяють викликати спеціалізовані API для різних пристроїв. Наприклад, якщо ваше додаток UWP призначене для пристрою Інтернету речей, ви можете додати в свій проект пакет SDK розширення для Інтернету речей, щоб реалізувати функції, характерні для пристроїв Інтернету речей. Детальніше про додавання пакетів SDK розширень див. Розділ Extension SDKs (Пакети SDK розширень) статті Device families overview (Загальні відомості про сімейства пристроїв).

Ви можете написати додаток так, щоб воно було призначене для запуску тільки на пристроях певного типу, а потім обмежити його поширення в Microsoft Store тільки цим типом пристроїв. Або ж ви можете реалізувати умовну перевірку на наявність того чи іншого API під час виконання і відповідним чином адаптувати поведінку свого застосування. Детальніше див. У розділі Writing Code (Написання коду) статті Device families overview (Загальні відомості про сімейства пристроїв).

Адаптивні елементи управління і введення

Елементи призначеного для користувача інтерфейсу реагують на розмір і щільність точок екрану, на якому запущено програму, яка і вибирають відповідний масштаб і макет. Крім того, додатки UWP відмінно працюють з різними засобами введення, такими як клавіатура, миша, сенсорні пристрої, перо

і пристрої управління Xbox One. Якщо потрібно додатково налаштувати користувальницький інтерфейс у відповідності з певним розміром екрану або типом пристрою, нові панелі макета і інструменти допоможуть вам розробити інтерфейс, здатний адаптуватися до різних пристроїв і форм-факторів, на яких може працювати ваш додаток.

Windows дозволяє орієнтувати призначений для користувача інтерфейс на безліч пристроїв за допомогою наступних функцій:

Універсальні елементи управління і панелі макета допомагають оптимізувати призначений для користувача інтерфейс під будь-який дозвіл екрана на конкретному пристрої. Наприклад, такі елементи управління, як кнопки і повзунки, автоматично адаптуються до розміру і щільності точок на екрані пристрою. Панелі макета допомагають коригувати компонування вмісту в залежності від розміру екрана. Адаптивне масштабування підлаштовується під відмінності в дозволі і DPI на всіх пристроях.

Єдина обробка введення дозволяє отримувати вхідні дані за допомогою торкань, пера, миші, клавіатури або контролера, наприклад Microsoft Xbox.

Інструменти допоможуть розробити вам призначений для користувача інтерфейс, здатний адаптуватися під різні дозволи екрану.

Деякі характеристики призначеного для користувача інтерфейсу додатка автоматично коригуються під різні пристрої. Однак при проектуванні користувальницького інтерфейсу додатку можуть знадобитися деякі зміни в поведінці додатки в залежності від пристрою, на якому воно працює. Наприклад, з додатком для фотографування при роботі на маленькому наладонних пристрої слід адаптувати свій інтерфейс так, щоб з ним зручно було працювати однією рукою. Коли це ж додаток для фотографування запускається на настільному комп'ютері, призначений для користувача інтерфейс повинен адаптуватися так, щоб використовувати додатковий простір екрану.

Актуальна інформація в реальному часі, яка спонукає користувачів знову і знову звертатися до додатка

- Є безліч способів підтримувати інтерес користувачів до додатка UWP.
- Живі плитки і плитки екрану блокування, на які виводиться короткий огляд актуальної і значущою в певному контексті інформації з програми.
- Push-повідомлення, які пропонують до уваги користувачів важливі оповіщення в потрібний момент.
- Дії користувачів, які дозволяють їм продовжити роботу в додатку з того місця, де вони зупинилися - навіть на іншому пристрої.
- Центр повідомлень забезпечує організацію повідомлень, що надходять з вашого застосування.
- Фонове виконання і тригери дозволяють вашому додатку відновлювати роботу саме тоді, коли це потрібно користувачеві.
- Додаток може використовувати голосове управління і пристрою Bluetooth LE, щоб користувачі могли взаємодіяти з навколишнім світом.
- Інтеграція з Кортаной дозволяє додати в ваше додаток можливості голосового управління.

Додавання служб

- Використовуйте хмарні служби, щоб виконувати синхронізацію між пристроями.
- Дізнайтеся, як підключатися до веб-службам для поліпшення взаємодії з додатком.
- Включіть у свій план push-повідомлення і покупки з додатків. Ці функції повинні працювати на всіх пристроях.

Зіставлення універсальної платформи Windows (UWP) і API часу виконання Windows

Якщо ви розробляєте додаток для універсальної платформи Windows (UWP), вам буде набагато простіше і зручніше вважати терміни "Універсальна платформа Windows" і "Середовище виконання Windows (WinRT)" майже синонімами. Але у вас є можливість зазирнути "під капот" цих технологій і визначити відмінності між концепціями. Якщо вам це цікаво, то цей останній розділ написаний саме для вас.

Середовище виконання Windows і API-інтерфейси WinRT розвиваються на основі API-інтерфейсів Windows. Спочатку Windows програмувались через плоскі API Win32 в стилі C. Поверх них були додані API-інтерфейси COM (з яких найяскравішим прикладом можна вважати DirectX). Windows Forms, WPF, .NET і керовані мови привнесли власні способи створення програмного забезпечення Windows і специфічні технології API. Середовище виконання Windows фактично є наступним етапом розвитку моделі COM. На рівні двійкового інтерфейсу (ABI) родинні зв'язки з COM стають очевидними. Однак Виконавча Windows була розроблена так, щоб її можна було викликати з великого числа різних мов програмування. Ці виклики здійснюються самим природним способом для кожного з цих мов. Тому доступ до середовища виконання Windows реалізований через особливий механізм мовних проекцій. Існують мовні проекції середовища виконання Windows для C #, Visual Basic, стандартного C ++, JavaScript і т. Д. Крім того, після одноразової упаковки (докладніше див. В описі моста для класичних додатків) ви можете викликати API-інтерфейси WinRT з програми, створеного на основі будь-якої з багатьох моделей додатків: Win32, .NET, WinForms і WPF.

І, само собою, API-інтерфейси WinRT можна викликати з програми UWP. Модель додатки UWP створена на основі середовища виконання Windows. З технічної точки зору модель додатки UWP заснована на CoreApplication, але

деякі подробиці можуть бути недоступні залежно від вибору мови програмування. Як описано в цьому розділі, з точки зору цінності платформа UWP призначена для створення єдиного виконуваного файлу, який ви зможете при бажанні опублікувати в Microsoft Store і запустити на будь-яких пристроях різних форм-факторів. Охоплення пристроїв для додатків UWP залежить від API середовища виконання Windows, які може викликати додаток або які ви можете викликати умовно.

Використання вже знайомого мови

Додатки UWP можуть використовувати середу виконання Windows, тобто власний API, вбудований в операційну систему. Цей API реалізований на мові C ++ і підтримується в C #, Visual Basic, C ++ і JavaScript. Деякі з мов і технологій, придатних для створення програмного забезпечення UWP:

XAML (для призначеного для користувача інтерфейсу) і C #, VB або C ++;

DirectX (для призначеного для користувача інтерфейсу) і C ++;

JavaScript і HTML.

Сумісність

Універсальна платформа – частина Windows 10 і 10 Mobile. Універсальні програми Windows не запускаються на версіях Windows до 8.Додатки, які здатні реалізувати дану платформу, створюються з використанням Visual Studio 2015 року, Visual Studio 2017 і Visual Studio 2019. Старі Metro-додатки для Windows 8.1 або Windows Phone 8.1 потребують зміни коду, щоб підтримувати UWP.

Під час Build 2015 Microsoft представила набір так званих «мостів» UWP для портування додатків для Android і iOS в середу Windows 10 Mobile . Міст Windows для Android (з кодовою назвою «Astoria») дозволяє перенести додатки Android, написані на Java або C ++, в середу Windows 10 Mobile і опублікувати їх в Windows Store. Кевін Галло (англ. Kevin Gallo), керівник Windows Developer Platform, пояснив, що дана реалізація має деякі обмеження: сервіси Google і

основне API недоступно, тому додатки, які мають «фонову діяльність», наприклад, додатки для швидких повідомлень, не працюватимуть коректно . Міст Windows для iOS (з кодовою назвою «Islandwood») - відкрита єднальна-утиліта, що дозволяє перенести додатки iOS, написані на Objective-C, в середу Windows 10 Mobile, використовуючи Visual Studio 2015 конвертіровав код з Xcode. Ранні збірки моста для iOS почали поширюватися як відкрите програмне забезпечення під ліцензією MIT з 6 серпня 2015; міст для Android поки знаходиться в закритому тестуванні.

У лютому 2016 Microsoft оголосила про придбання компанії Xamarin. Незабаром після покупки Microsoft оголосила про закриття розробки моста Android і підтримки даних додатків в Windows 10. Головним напрямком компанії залишився міст iOS.

Становлення платформи

UWP була лише доповненням до Windows Runtime. Універсальні програми Windows, створені з використанням технології UWP, не потребують позначенні, для якої ОС вони призначені; крім того, вони підтримують як ПК, так і смартфони, планшети або Xbox One, використовуючи мости UWP. Дане розширення дозволяє автоматично підтримувати всі можливі платформи. Універсальне додаток може бути запущено на будь-якому мобільному телефоні або планшеті. Воно ж, запущене на смартфоні, може вести себе так, як ніби запущено на ПК, якщо підключено до останнього за допомогою док-станції.

1.4 Вимоги до апаратного забезпечення:

- Процесор – Intel Pentium I3
- Обсяг ОЗУ – не нижче 128 Мб
- Відеокарта
- Вільний простір на диску - 100 Мб

- Клавіатура\
- Наявність комп’ютерної миші або touch screen
- ОС Windows10

Висновок до Розділу 1.

У першому розділі своєї курсової роботи я провів аналіз ігор, які можуть бути аналогами до гри, яка буде розроблена у данному курсовому проекті. Також були вибрані програмні засоби для розробки програми, проведений їх аналіз, та наведені доводи у їхню користь.

РОЗДІЛ 2.
СТВОРЕННЯ ДОДАТКУ

2.1 Створення програми

Розробка інтерфейсу та дизайну веб- додатку

До розробки інтерфейсу та дизайну додатку також можна переходити відразу після написання основної частини проекту.

Розробка інтерфейсу та дизайну− це творчий процес. Під дизайном додатку розуміється не просто шаблон, а повне оформлення сторінок додатку в єдиному стилі. Найважливіше значення тут має не нагромадження робочого простору щоб не відволікати користувача непотрібними речами.

Також додатки мають властивість становитися менш актуальними. Це означає, що для ефективної роботи необхідно проводити оновлення додатку як мінімум один раз на рік. Переважно особливості додатку повинні бути такими:

— функціональна навігація;

— правильно скомпонованим;

— інтуїтивний дизайн

— гармонійним, щодо підбору кольорів щоб не втомлюватися при довгій роботі;

Написання программного коду

Під написанням коду мається на увазі програмування додатку та його функцій.В даний час жоден додаток не обходиться без програмних розробок. Використання програмування дозволяє зробити додаток більш функціональним, а також, що важливо, спростити його подальше використання, адже саме правильно запрограмовані додатки дають можливість власнику або особі яка за нього відповідає вносити зміни якщо вони потребуються.

Саме розробка інтерфейсу надає додатку остаточний вигляд, який і побачать користувачі на сайті. Це накладає на людину яка це розробляє відповідальність так як він повинен дати користувачу зрозумілий інтерфейс в якому зможе розібратися кожна людина, що захоче придбати данну гру, що розроблена..

Тестування – фінальний етап створення проекту

Цей етап являє собою контроль якості роботи, що виконується. Перевіряється все: зручність використання, працездатність і наявність всіх необхідних функцій правильність написання.

На тестування додаток, як правило, віддається людям, які не брали участь в його створенні, так як в процесі роботи помилки стають непомітні, свіжий погляд тут просто необхідний.

Мова програмування C#

C# (вимовляється Сі-шарп) – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників – мов C++, Delphi, Модула і Smalltalk - C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

C# підтримує строго типізовані неявні оголошення змінних з ключовим словом var і неявно типізовані масиви з ключовим словом new [], за яким слідує ініціалізатор колекції.

C# підтримує суворий тип даних Boolean, bool. Вирази, які приймають умови, такі як while та if, вимагають висловлювання, що реалізує оператор true або false. Хоча C++ також має тип Boolean, він може бути вільно перетворений в цілі числа та з них, а вирази, такі як if(a), вимагають тільки того, щоб a був конвертований в bool, що дозволяє бути a int-типу або вказівником. C# забороняє «ціле значення означає справжній або помилковий підхід» на тій підставі, що примус програмістів використовувати вирази, які повертають точно bool, можуть створювати деякі типи помилок програмування, наприклад if (a = b) (використання присвоювання = замість рівності ==, які, хоча і не є помилкою на C або C ++, все одно будуть спіймані компілятором).

C# безпечніший в порівнянні з C++. Єдиними неявними перетвореннями за умовчуванням є ті, які вважаються безпечними, наприклад, розширення цілих чисел. Це застосовується під час компіляції, під час JIT і, в деяких випадках, під час виконання. Не відбувається неявних перетворень між булевими і цілими числами, а також між членами перерахування і цілими числами (крім літерала 0, який може бути неявно перетворений будь-який нумерований тип). Будь-яке призначене для користувача перетворення повинно бути явно позначене як явне або неявне, на відміну від конструкторів копіювання C++ і операторів перетворення, які за умовчанням є неявними.

C# має явну підтримку коваріації та контраваріантності в родових типах, на відміну від C++, яка має певний рівень підтримки контраваріантності просто через семантику типів,що повертаються, на віртуальні методи.

Члени перерахування розміщуються в своєму власному обсязі.

Мова C # не допускає глобальних змінних або функцій. Всі методи і члени повинні бути оголошені всередині класів. Статичні члени відкритих класів можуть замінювати глобальні змінні та функції.

Фреймворк програмний каркас

Фреймворк (англ. Framework, каркас, платформа, структура, інфраструктура) – інфраструктура програмних рішень, що полегшує розробку складних систем. Спрощено дану інфраструктуру можна вважати своєрідною комплексною бібліотекою, але при цьому вона має ряд обмежень, що задають правила створення структури проекту та написання коду.

Програмний фреймворк (англ. software framework) - це готовий до використання комплекс програмних рішень, включаючи дизайн, логіку та базову функціональність системи або підсистеми. Відповідно програмний фреймворк може містити в собі також допоміжні програми, деякі бібліотеки коду, скрипти та загалом все, що полегшує створення та поєднання різних компонентів великого програмного забезпечення чи швидке створення готового і не обов'язково об'ємного програмного продукту. Побудова кінцевого продукту відбувається, зазвичай, на базі єдиного API.

Одна з головних переваг, при використанні каркасних застосунків, полягає в тому, що такі програми мають стандартну структуру. Каркаси за стосунків стали популярними з появою елементів інтерфейсу, які мали тенденцію до реалізації стандартної структури для додатків. З їх використанням стало набагато простіше створювати засоби для автоматичного створення графічних інтерфейсів, оскільки структура внутрішньої реалізації коду програми стала відома заздалегідь.

Для забезпечення каркасу, зазвичай, використовують підходи об'єктно-орієнтованого програмування, наприклад, частини програми можуть успадковуватися від базових класів фреймворка.

2.2 Опис програми та її алгоритмів

Готова програма дозволяє грати в гру «Happy Smile». Гравцю необхідно знайти за визначений термін часу яке з облич щастливе серез похмурих, які він бачить на екрані. Якщо гравець знаходить вірне обличчя, то при натиску ЛКМ

на нього з’являється наступний рівень, де знайти щасливе обличчя буде більш складною задачею. Гра завершується коли гравець не встигаж знайти необхідну посмішку на обличчі.

Опис функцій системи

Функції системи зображені у вигляді UML діаграми прецедентів з одним актором – Гравець на рисунку 2.1.

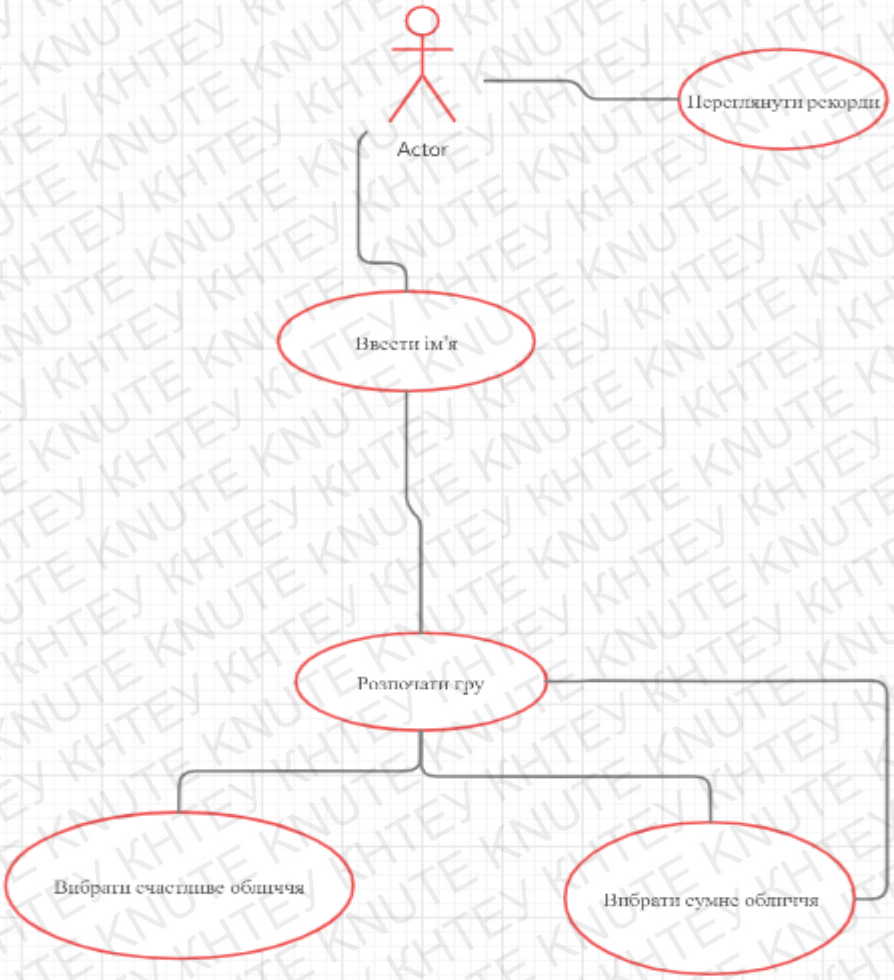

Рис. 2.1. UML діаграма прецедентів

Ця діаграма зображує, що може зробити гравець.

Гравець може:

1) Ввести ім’я, для запису у таблицю рекордів.
2) Вибрати счастливе обличчя.
3) Вибрати сумне обличчя.
4) Закінчити гру за бажанням.
5) Програти.
6) Переглянути таблицю рекордів.

Опис ієрархії власних класів

В процесі написання гри було розроблено ієрархію класів представлену на рисунку 2.2.

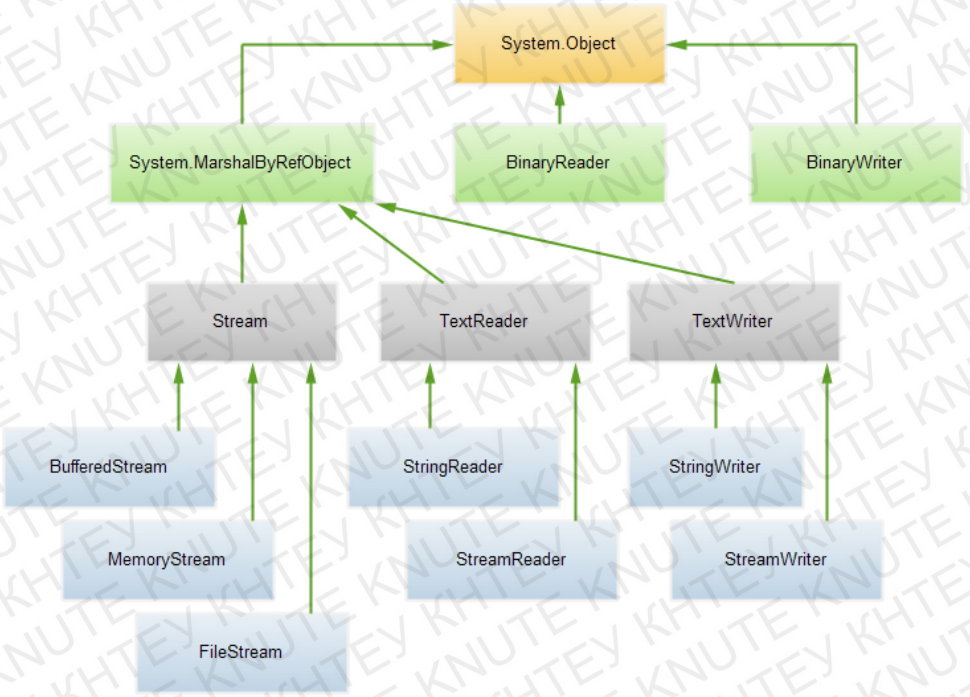

Рис. 2.2. Ієрархія класів

Нижче подано опис представлених на діаграмі класів.

Клас «MainPage» - ініціалізація об’єктів і властивостей, а також виклик функції завантаження першого рівня.

У таблиці 2.1 представлено опис методів класу.

Опис компонентів

В результаті написання програми було створено компоненти, які є

окремимими файлами. А саме:

1) Папка Assets- сховище для фотографій, що використовуються у проекті.

2) Файл App.xaml- сховище для програмних засобів.

3) Файл App1_TemporaryKey.pfx- сертифікат для запуску додатку.

4) Файл MainPage.xaml і пов'язаний з ним MainPage.xaml.cs / vb- основна сторінка додатку, на якій розробляється інтерфейс

5) Package.appxmanifest.- редактор коду.

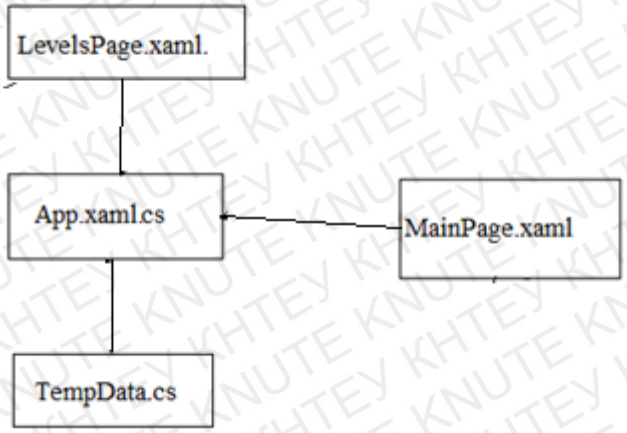

Рис. 2.3. Діаграма компонентів.

Опис алгоритму

Користувач вводзнаходить щасливе обличчя. Якщо відповідь була вибрана правильна, то він переходить на наступний рівень, якщо ні, то виводиться повідомлення про поразку і пропонується почати гру спочатку.

На рисунку 2.4 зображено циклічний, розгалужений алгоритм роботи програми.

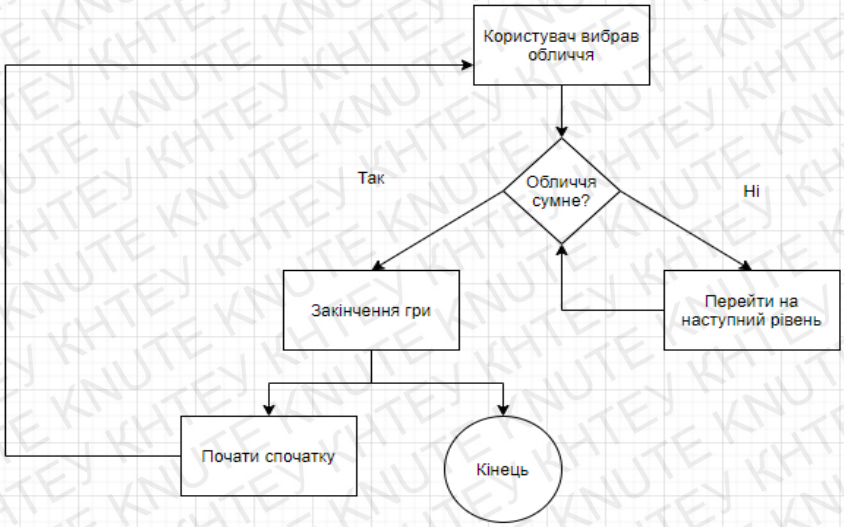

Рис. 2.4 . Алгоритм роботи програми

2.3. Техічні засоби

Проект являє собою гру для будь-якого пристрою під керуванням ОС Windows.

— Автор: Перетокін В.І.

— Мова програмування: C#

— Середовище програмування: Microsoft Visual Studio 2017

Вимоги до апаратного забезпечення:

— Процесор – Intel Pentium I3

— Обсяг ОЗУ – не нижче 128 Мб

— Відеокарта

— Вільний простір на диску - 100 Мб

— Клавіатура

— Наявність комп’ютерної миші або touch screen

— ОС Windows10

2.4. Інструкція користувача

Для запуску гри потрібно перейти до папки з її проектом і натиснути ЛКМ x2 на HappySmile.exe.

Гра запуститься і буде відображено вікно авторизації, у якому запропоноване випадкове ім’я для спрощення авторизації, проте у користувача залишається можливість ввести своє ім’я, далі необхідно натиснути Start game, щоб вибрати рівень, як зображено на рисунку 2.5.

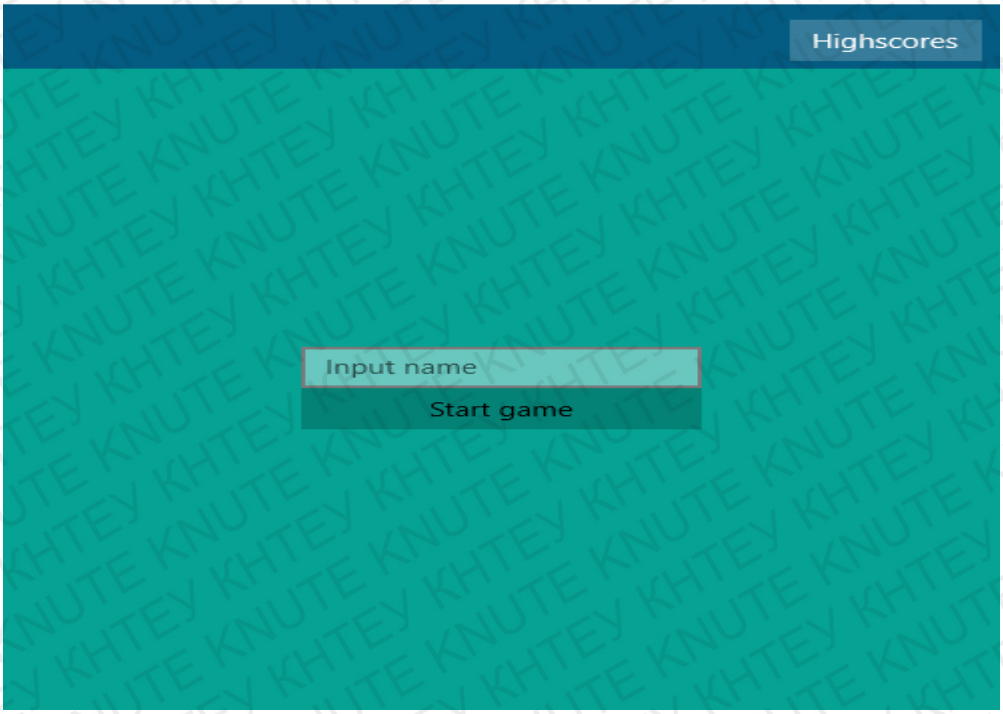

Рис. 2.5. Вікно авторизації після запуску програми

Якщо гравець програє, то відрау може почату гру наново як показано на рисунку 2.6.

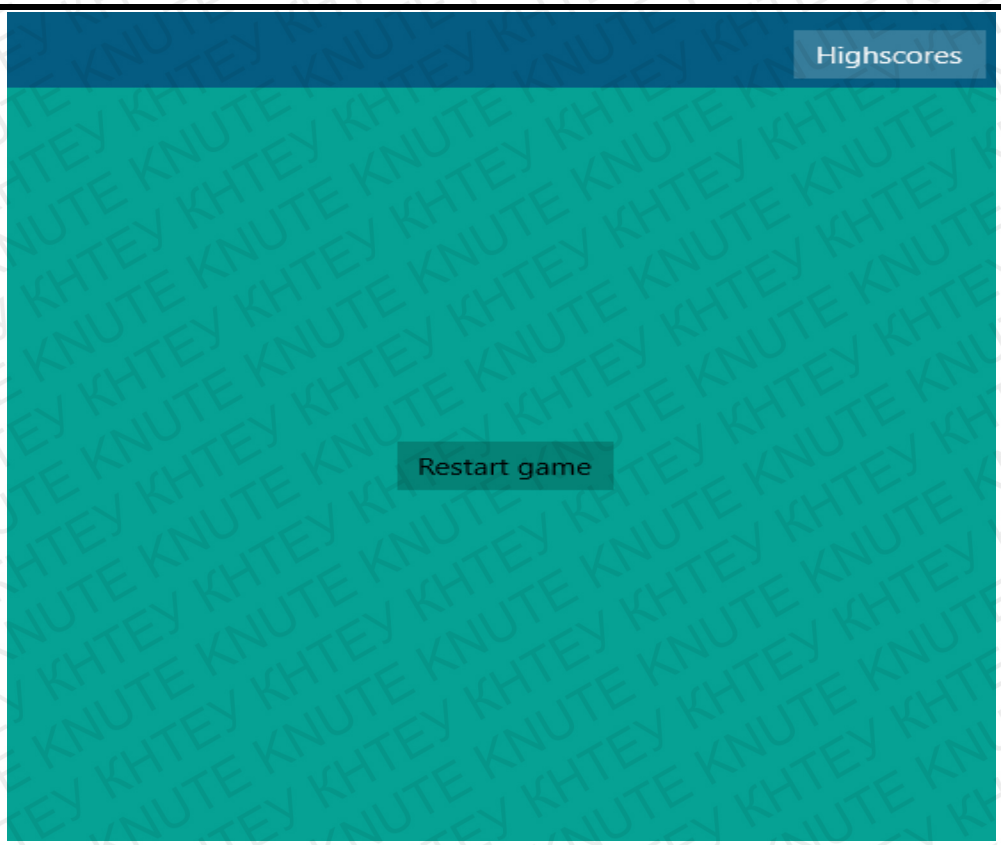

Рис. 2.6. Можливість почати спочатку.

Після того, як гравець натисне на рівень, з якого хоче розпочати, гра почнеться.

Кількість набраних очок, за проходження певної кількості рівнів. Приклад на рисунку 2.7

Рис. 2.7. Кількість зібраних очок за гру.

На рисунку 2.8 показано, як зростає складність відповідно до обраного рівня.

Рис. 2.8. Складність зростає з кожним рівнем.

Також в грі реалізований таймер і на проходження кожного рівня дається лише 20 секунд, як зображено на рис. 2.9

Рис. 2.9. Таймер.

Якщо гравець натискає на не правильний елемент, тобто сумний смайл, або не встигає знайти веселий смайл і час виходить, то гра закінчується і з'являється повідомлення про програш, як показано на рисунку 2.10.

Рис. 2.10. Програш.

Коли гравець закінчив гру- його ім'я і рахунок заноситься до списку рекордів, який можна переглянути, натиснувши на кнопку «Highscores» у верхньому правому кутку вікна. Приклад зображено на рисунку 2.11 і 2.12.

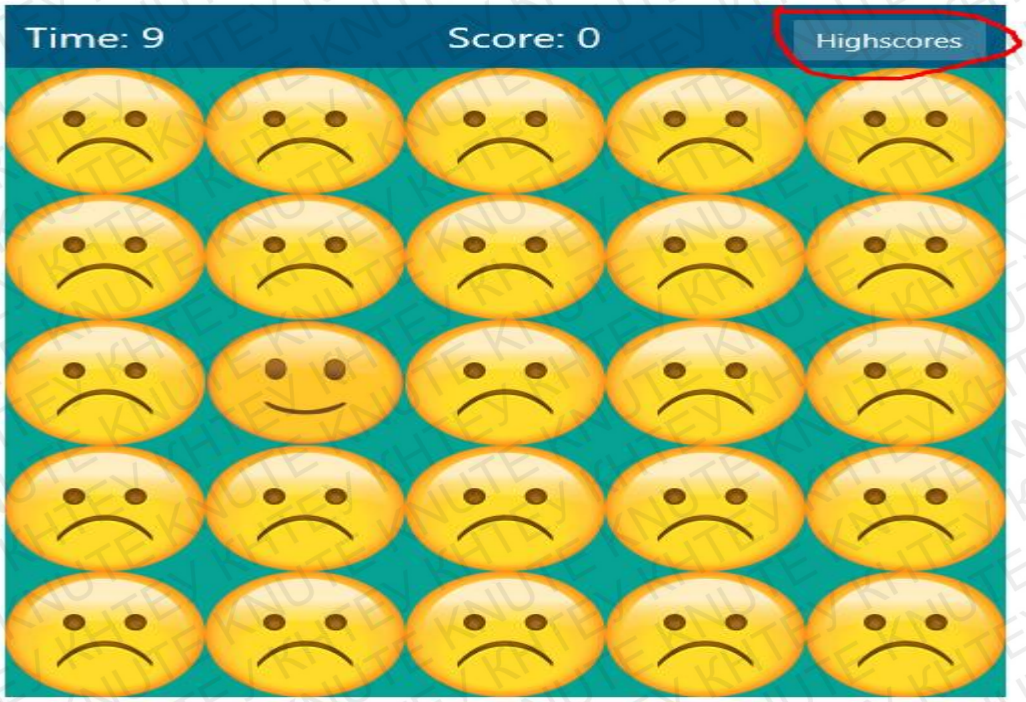

Рис. 2.11. Кнопка Highscores.

Рис. 2.12 . Список рекордів.

Під час створення гри були використані наступні класи та функції реалізовані в них(у вигляді UML – діаграмми):

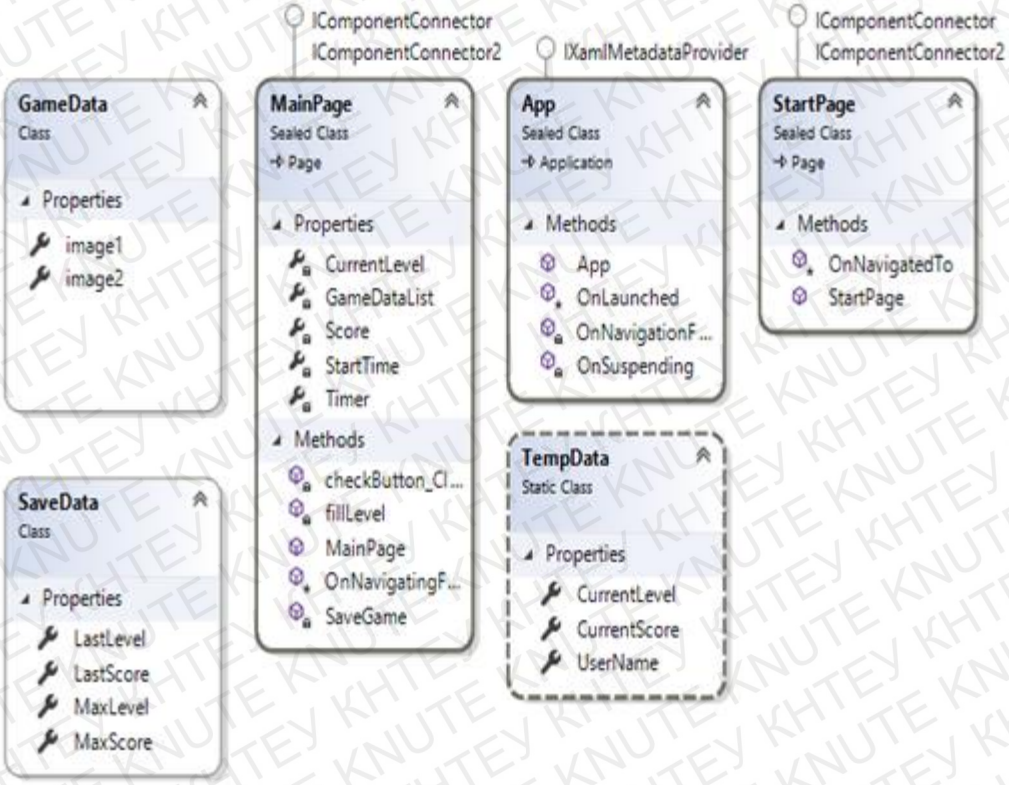

Рис. 2.13. UML-діаграма

Висновок до розділу 2

У другому розділі була проведена робота по створенню програмного додатку,роззробки його інтерфейсу та функцій, які стануть фішкою данного проекту, також наведена інструкція користувача, у якій описані усі функції додатку.

РОЗДІЛ 3
ЕКОНОМІЧНА ЧАСТИНА

3.1 Оцінка вартості програмного продукту

У цьому розділі свого дипломного проекту буде проводитися аналіз та розрахунок затрат на розробку даного додатку. Розробка гри вимагає деяких трудових та інтелектуальних затрат, що допомагають у процесі розробки, також важливою частиною є використання комп'ютерної техніки, що додає певні витрати у процес розробки. Оцінка вартості продукту складається з певних факторів, таких як затрати коштів, часу, стаж розробника, та сил необхідних для використання.

Загальний час розробки може складатися з різних факторів. Структура часу, що необхідний для розробки буде представлений у таблиці 3.1.

За допомогою розрахунку часу я матиму змогу у подальшому вирахувати свої інтелектуальні та трудові витрати, кошти, які на них необіхдно буде використати, для того щоб внести їх в фінальну версію вартості додатку, що я розробляв у даному дипломному проекті.

Взявши данні з таблиці, я матиму змогу у подальшому їх використовувати у різних формулах, для підготовки фінальної вартості продукту, що був розроблений.

У подальшому це допоможе у розробці інших програм, та дасть змогу зрозуміти, що можна покращити у данному увипадку, як зєкономити, або перерозподілити час витрачений на розробку програмного додатку.

Таблиця 3.1 – Таблиця часу, який затрачений на розробку додатку

№ етапу	Позначення часу певного процесу	Зміст процесу
1	$T_{ПО}$	Підготовка до проекту
2	$T_{О}$	Опис самого завдання
3	$T_{А}$	Розробка алгоритмів для розробки
4	$T_{БС}$	Розробка блок-схеми для алгоритмів додатку
5	$T_{ПІ}$	Проектування інтерфейсу користувача
6	$T_{Н}$	Написання коду програми
7	$T_{ВТ}$	Тестування програми
8	$T_{Д}$	Оформлення документації

Час вираховується в людино-годинах, коли $T_{ПО}$ та $T_{Д}$ беруться по фактично відпрацьованому часу, а розрахунок останніх етапів визначається за умовним числу команд *Q*.

Число команд *Q* визначається за формулою

$$Q = q \cdot з, \qquad (3.1)$$

де *q* – коефіцієнт, який враховує умовне число команд відповідно від типів завдання. Значення коефіцієнта *q* вибирається з таблиці 4.2.

Значення коефіцієнта *з* вибирається з таблиці 4.3.

Таблиця 3.2 – Данні для коефіцієнта *q*.

Тип завдання	Значення коефіцієнта q
Завдання для обліку даних	1400…16500
Завдання управління	1600…1800
Завдання щодо планування	3100…3600
Багатоваріантне завдання	4600…5100
Комплексне завдання	5100…5600

Для даного завдання коефіцієнт *q* приймається = 1650

Програмні продукти розподіляються до однієї з нижче наведених груп, у відповідності з їх новизною:

— група А: розробка абсолютно нових завдань;

— група Б: розробка програм, що відрізняються своєю оригінальністю;

— група В: розробка програм при використанні стандартних рішень;

— група Г: разове завдання.

Для даного завдання міра новизни: В

За складністю програмні продукти розподіляються до однієї з трьох груп:

— алгоритми моделювання і оптимізації систем;

— завдання обліку, звітності і статистики;

— алгоритми стандартнох розробки.

Дане завдання може бути віднесено до другої групи складності низького рівня.

Коефіцієнт *з* визначається з таблиці 3.3 в залежності від складності і міри новизни.

Таблиця 3.3 – Дані для коефіцієнта *з*

Мова програмування	Група складності	Міра новизни			
		А	Б	В	Г
Високого рівня	1	1,38	1,16	1,17	0,7
	2	1,31	1,2	1,07	0,64
	3	1,21	1,11	1,01	0,61
Низького рівня	1	1,59	1,46	1,33	0,78
	2	1,50	1,38	1,28	0,75
	3	1,39	1,27	1,16	0,67

Для даного завдання коефіцієнт *з* = 1,00.

За формулою 4.1 визначається умовне число команд *Q*.

$$Q = 1400 \cdot 1{,}00 = 1400{,}00 \text{ чоловіко / годин.}$$

За формулою $T_{ПО}$ визначається час, що був затрачений на кожен з етапів розробки програмного забезпечення:

— $T_{ПО}$ (час, що витрачено на підготовку опису завдання), береться по факту і для даного завдання

$$T_{ПО} = 25 \text{ чоловіко / годин.}$$

T_O (час, що витрачажться на опис завдання) визначається за формулою

$$T_O = Q \cdot B / (50 \cdot K), \qquad (3.2)$$

де *B* — коефіцієнт обліку змін завдання. Для даного завдання *B* = 1,4;

K — коефіцієнт, що показує кваліфікацію програміста, який був вибраний для розробки завдання. Значення коефіцієнта вибирається з таблиці 4.4.

Таблиця 3.4 – Дані для коефіцієнта К.

Стаж програміста	Значення коефіцієнта К
до 3-х років	0,81
від 3 до 4 років	1,1
від 5 до 6 років	1,2…1,3
від 6 до 11 років	1,3…1,4
понад 11 років	1,4…1,6

В даному випадку коефіцієнт *K* = 1,0.

За формулою 3.2 підраховується час на опис завдання.

$$To = 1650{,}00 * 1{,}4/(50*1{,}0) = 46{,}3 \text{ [чоловіко / годин].}$$

— T_A (час, що витрачений на розробку алгоритмів) розраховується за формулою

$$T_A = Q / (50 * K), \qquad (3.3)$$

За формулою 4.3 проводяться розрахунки на розробку алгоритмів.

$$T_A = 1650{,}00 / (50 \cdot 1{,}0) = 33 \text{[чоловіко / годин].}$$

— $T_{БС}$ (час, що був витрачений, на розробку блок-схеми) визначається аналогічно *ТА* за формулою 4.3 і складає

$$T_{БС} = 1400{,}00 / (50 \cdot 1{,}0) = 34 \text{[чоловіко / годин]}.$$

— T_H (час написання програми на мові програмування) визначається за формулою

$$T_H = Q \cdot 1{,}5 / (50 \cdot K), \qquad (3.4)$$

За формулою 4.4 підраховується час написання програми на мові програмування.

$$Тн = 1650{,}00 * 1{,}5/(50*1{,}0)=49{,}5 \text{ [чол. / годин]}.$$

- $T_{ПІ}$ (час на проектування інтерфейсу) визначається за формулою

$$T_{ПІ} = Q / 50, \qquad (3.5)$$

За формулою 4.5 підраховується час на проектування інтерфейсу програми.

$$T_{ПІ} = 1400 / 50 = 43 \text{ [чол. / годин]}.$$

— $T_{ВТ}$ (час відладки і тестування програми) визначається за формулою

$$T_{ВТ} = Q \cdot 4{,}2 / (50 \cdot К), \qquad (3.6)$$

За формулою 4.6 підраховується час відладки і тестування програми.

$$Твт = 1650{,}00 \cdot 4{,}2 / (50 \cdot 1{,}0) = 138{,}6 \text{ [чол. / годин]}.$$

— $T_Д$ (час на оформлення документації), береться по факту і для даного завдання

$$T_Д = 25 \text{ чол / годин}.$$

Підраховується загальний час, що був використаний, на створення програмного продукту. Він рахується за формулою:

$$T = T_{ПО} + T_О + T_А + T_{БС} + T_Н + T_{ПІ} + T_{ВТ} + T_Д$$

$$T = 25 + 46{,}2 + 33 + 33 + 48{,}8 + 49{,}5 + 138{,}6 + 25 = 399{,}1 \text{ [чоловіко / годин]}.$$

Основна заробітня платня визначається за формулою

$$ЗПосн.=(ЗП\ 1р \cdot Кт \cdot Т)/(Чр \cdot тр.д.)\cdot(1 + П/100), \quad (3.8)$$

де *ЗП 1р* — місячна зарплата 1-го розряду, грн.;

Кт — коефіцієнт, що вибирається із тарифної сітки, за яким працює виконавець;

Т — загальний час, що використаний на розробку додатку, чол. / годин;

Чр — кількість робочих днів розробника в місяць;

тр.д. — тривалість робочого дня в годинах;

П — відсоток премії для розробника.

Для даного завдання:

— *ЗП 1р* = 450,00 грн.;

— виконавець 14 розряду;

— відсоток премії з проету *П*=15%;

— загальний час, що був використанний для створення продукту *Т*=399,1 чоловіко / годин;

— кількість робочих днів у місяць *Чр* = 21;

— для 14 розряду тарифний коефіцієнт складає *Кт* = 3,36;

— Тривалість одного робочого дня *тр.д.* = 8 годин;

Таким чином, визначається основну заробітну плату розробника під час створення програмного продутку:

$$ЗП\ осн. = (450{,}00 \cdot 3{,}36 \cdot 399{,}1)/(21 \cdot 8)\cdot(1+15/100)=4130{,}68\ [грн].$$

Премія розраховується у розмірі 15% від основних затрат на проект, та визначається за формулою:

$$ЗП\ дод. = ЗПосн. \cdot 15 / 100, \quad (3.9)$$

$$ЗП\ дод. = 4130{,}68 \cdot 15 / 100=619{,}6\ [грн].$$

Загальна заробітна плата розраховується за формулою 4.10:

$$ЗПзагальна = ЗПосн + ЗПдод., \quad (3.10)$$

ЗПзагальна = 4130,68 + 619,6 =4750,28 [грн].

ЄСВ нараховується на заробітну плату за формулою 3.11 і нараховує 22 % від загальної заробітної плати

$$ЄСВ = ЗПзагальна \cdot 22 / 100, \quad (3.11)$$

$$ЄСВ = 4750,28 \cdot 22 / 100 = 1045,06 \text{ [грн]}.$$

Розрахунок витрат на збереження та експлуатацію ПК, на якому виконується розробка розраховується за формулою

$$В\ зб.\ експл. = ЗП\ осн. \cdot 130 / 100, \quad (3.12)$$

$$В\ зб.\ експл. = 4130,68 \cdot 130 / 100 = 5369,88 \text{ [грн]}.$$

3.2 Розрахунок собівартості програмного додатку

Собівартість програмного продукту визначають:

— основна ЗП виконавця;

— додаткова ЗП виконавця;

— ЄСВ;

— витрати на ПК;

— Інші витрати.

Інші витрати визначаються у формулі 3.13:

$$І.вит. = (ЗП\ осн. + ЗП\ дод. + ЄСВ + В\ зб.\ експл.) \cdot 10 / 100, \quad (3.13)$$

$$І.вит. = (4130,68 + 619,6 + 1045,06 + 5369,88) \cdot 10 / 100 = 1116,52 \text{ [грн]}.$$

Собівартість програмного додатку визначається за формулою

$$Сп.п. = ЗП\ осн. + ЗП\ дод. + ЄСВ + В\ зб.експл. + І.вит, \quad (3.14)$$

$$Сп.п. = 4130,68 + 619,6 + 1045,06 + 5369,88 + 1116,52 = 13281,74 \text{[грн]}.$$

Структура собівартості програмного продукту відображена в таблиці 3.5.

Таблиця 3.5 – Дані з проекту з яких складається сума проекту

Елементи структури	Сума грн.	Співвідношення %
1 Плата за проект для розробника	4130,68	33.7
2 Додаткова плата розробника за проект	619,6	5
3 ЄСВ	1045,06	8.5
4 Витрати на ПК	5369,88	43.7
5 Інші види витрати	1116,52	9.1
Собівартість програмного додатку	13281,74	100

3.3 Розрахунок ціни програмного додатку

Ціна програмного додатку складається з декількох певних компонентів і розраховуються за допомогою формули 3.15:

$$Ц = Сп.п. + П + ПДВ, \quad (3.15)$$

де *Сп.п.* — собівартість програмного додатку, грн.;

П — прибуток на який розраховують, грн.;

ПДВ — за законодавством податок на додану вартість, грн.

П — прибуток може складати 40% від повної собівартості програмного додатку і розраховуються за допомогою формулі 3.16:

$$П = Сп.п. \cdot 40 / 100, \quad (3.16)$$

$$П = 1381,79 \cdot 40 / 100 = 4812,69 \text{ [грн]}.$$

ПДВ — податок на додану вартість, розраховується у розмірі 20% від прибутку та суми собівартості, та розраховуються за допомогою формули 4.17:

$$ПДВ = (Сп.п. + П) \cdot 20 / 100, \quad (3.17)$$

$$ПДВ = (1281,74+4812,69) \cdot 20 /100 = 3538,88 \text{ [грн]}.$$

За формулою 4.15 визначається ціна програмного продукту.

$Ц$ = 13281,74+ 4812,69 + 3538,88 = 21633,3 [грн].

3.4 Зведена таблиця показників

Результати розрахунків відображаються в підсумкової таблиці 2.6

Таблиця 3.6 – Результати розрахунку витрат

Найменування показника	Сума, грн.
Собівартість додатку	13281,74
Розрахований прибуток	4812,69
Податок на додану вартість	3538,88
Ціна програмного додатку	21633,3

Висновок до розділу 3

У третьому розділі були наведені економічні розрахунки, які були використані за стандартними формулами та дають змогу розрахувати усі витрати та прибуток з проекту. За допомогою даного розділу, я зміг визначити вартість своєї роботи, сміг проаналізувати усі витрати, які можуть трапитись у процесі розробки веб-додатку. З вище наведених формул та розрахунків, можна зробити висновок про доцільність створення даного веб-додатку, виходячи з цього вирішувати робити його на продаж чи ні.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

В ході написання дипломної роботи був розроблений додаткок «HappySmile» для Windows мовою C#.У ході роботи я ознайомився та покращив навички у мові програмування C# та UWP. Сформував ТЗ та план робот, необхідних для виконання. Також розробив проект, що готовий до використання та реалізаціїЯк середовище для розробки я видрав Visual Studio 2017, через його функціонал та легкість у використанні, та можливість розробки додатків із підключенням певних функцій.

У роботі були наведні приклади схожих додатків, проведений їх аналіз, що дало змогу зробити інтерфейс користувача приємним, та наповнити функціями, що стануть зручними та цікавими для усіх користувачів.

У практичній частині були розроблені усі функції, які були задумані на початку реалізації проекту, та наведені алгоритми їх створення та використання

Цілі, що поставленні на початку дипломнох роботи я вважаю виконаними за допомогою мову програмування C# та UWP.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Законодавчі та нормативні документи

1. Інструкція зі статистики заробітної плати: наказ Держ. комітету статистики України від 13.01.2004 № 5.
2. Про деякі питання практики застосування конкурентного законодавства : Інформ. лист Вищого господарського суду України від 13.04.2007 № 01-8/229.
3. Закон України Про оплату праці.
4. Закон України Про підприємництво.
5. Закон України Про оподаткування прибутку підприємств.
6. ГОСТ 19.001-77 ЄСПД. Загальні положення.
7. ГОСТ 19.002-80 ЄСПД. Схеми алгоритмів і програм. Правила виконання.
8. ГОСТ 19.003-80 ЄСПД. Схеми алгоритмів і програм. Позначення умовні графічні.
9. ГОСТ 19.105-78 ЄСПД. Загальні вимоги до програмних документів.
10. ГОСТ 19.401-78 ЄСПД. Текст програми. Вимоги до змісту і оформлення.
11. ГОСТ 19.402-78 ЄСПД. Опис програми.
12. ГОСТ 19.504-79 ЄСПД. Інструкція програміста.
13. ГОСТ 19.505-79 ЄСПД. Інструкція оператора.
14. ГОСТ 19.506-79 ЄСПД. Опис мови. Вимоги до змісту і оформлення
15. ДСТУ – 2293 – 93. Система стандартів безпеки праці. Охорона праці.

Книги колективу авторів

16. Р. Шелдон, Д. Мойе, MySQL: базовый курс - Beginning MySQL-533с.
17. М. Кузнецов, И. Симдянов, MySQL на примерах. – Спб.: БХВ Петербург, 2008.-211 с.

					КНТЕУ 121 06-20.БР		
Зм.	Аркуш	№ докум	Підпис	Дата			
Зав. кафедри		Криворучко О.В.			Розробка програмного забезпечення «Happy Smile»	Стадія	Аркуш	Аркушів
Керівник		Пашорін В.І.				СП	46	43
Гарант		Цензура М.О.				Факультет інформаційних технологій, 4 курс, 6 група		
Розроб.		Перетокін В.І.			Список використаних джерел			

Електронні ресурси

18.Сайт мови MySql - https://www.mysql.com/

19. Сайт допомоги із створенням класів - https://professorweb.ru/my/programs/visual-studio/level3/3_6.php

20. Сайт Microsoft з документацією. C#- https://docs.microsoft.com/en-us/dotnet/csharp/

21.Сайт із допомогою UWP- https://metanit.com/sharp/uwp/1.1.php

ДОДАТКИ

Додаток А

Код програми

MainPage.xaml.cs

```
<Page
    x:Class="HappySmile.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:HappySmile"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" Width="500" Height="550">

    <Grid Background="{ThemeResource
ApplicationPageBackgroundThemeBrush}">
        <RelativePanel HorizontalAlignment="Left" Height="50"
VerticalAlignment="Top" Width="500" RequestedTheme="Dark"
Background="#FF055C82">
            <TextBlock x:Name ="scoreBlock"  HorizontalAlignment="Left" Height="34"
TextWrapping="Wrap" VerticalAlignment="Top" Width="168" Margin="221,10,-
327.333,-24" FontSize="22"/>
            <TextBlock  x:Name ="timeBlock" HorizontalAlignment="Left" Height="34"
TextWrapping="Wrap" VerticalAlignment="Top" Width="184" Margin="10,10,-
132.333,-24" FontSize="22"/>
            <Button x:Name="recordBotton"  Content="Highscores"
HorizontalAlignment="Left" Height="32" VerticalAlignment="Top" Width="96"
Margin="394,12,-425.667,-12"/>
        </RelativePanel>
        <Grid x:Name="grid"  Background="#FF06A294" Width="500" Height="500"
HorizontalAlignment="Left" VerticalAlignment="Bottom">
            <StackPanel HorizontalAlignment="Center" Height="68"
VerticalAlignment="Center" Width="200">
                <TextBox x:Name="nameBox"  TextWrapping="Wrap" Text=""
VerticalAlignment="Center" HorizontalAlignment="Center"
HorizontalContentAlignment="Left" VerticalContentAlignment="Top" Width="200"
PlaceholderText="Input name"/>
                <Button x:Name="startButton" Content="Start game"
HorizontalAlignment="Stretch" VerticalAlignment="Stretch"/>
            </StackPanel>

        </Grid>
```

```
    </Grid>
</Page>
```

MainPage.xaml.cpp

```
//
// MainPage.xaml.cpp
// Реализация класса MainPage.
//

#include "pch.h"
#include "MainPage.xaml.h"
#include "iostream";
#include <fstream>
#include <ppltasks.h>


using namespace HappySmile;
using namespace Platform;
using namespace Windows::Storage;
using namespace Windows::Foundation;
using namespace Windows::Foundation::Collections;
using namespace Windows::UI::Xaml;
using namespace Windows::UI::Xaml::Controls;
using namespace Windows::UI::Xaml::Controls::Primitives;
using namespace Windows::UI::Xaml::Data;
using namespace Windows::UI::Xaml::Input;
using namespace Windows::UI::Xaml::Media;
using namespace Windows::UI::Xaml::Navigation;
using namespace Windows::UI::Xaml::Media::Imaging;
using namespace Windows::UI::Popups;
using namespace std;

// Документацию по шаблону элемента "Пустая страница" см. по адресу
https://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x419

DispatcherTimer^ timer;
time_t startTime;
std::wofstream m_OutStream;
bool isGame = true;

MainPage::MainPage()
{
        InitializeComponent();
        size = 5;
```

```
	startButton->Click += ref new Windows::UI::Xaml::RoutedEventHandler(this,
&HappySmile::MainPage::OnClick);

	timer = ref new DispatcherTimer();
	TimeSpan timeSpan;
	timeSpan.Duration = 1000L;
	timer->Interval = timeSpan;
	timer->Tick += ref new Windows::Foundation::EventHandler<Platform::Object
^>(this, &HappySmile::MainPage::OnTick);

	recordBotton->Click += ref new
Windows::UI::Xaml::RoutedEventHandler(this,
&HappySmile::MainPage::RecordOnClick);
}

void MainPage::loadLevel() {
	grid->Children->Clear();
	double x = 500 / size;
	int randI = rand() % size;
	int randJ = rand() % size;
	for (int i = 0; i < size; i++)
		for (int j = 0; j < size; j++)
		{
			auto img = ref new Image();
			img->Height = x;
			img->Width = x;
			img->HorizontalAlignment =
Windows::UI::Xaml::HorizontalAlignment::Left;
			img->VerticalAlignment =
Windows::UI::Xaml::VerticalAlignment::Top;
			img->Margin = Thickness(x * i, x * j, 0, 0);
			img->PointerReleased += ref new
Windows::UI::Xaml::Input::PointerEventHandler(this,
&HappySmile::MainPage::OnPointerReleased);
			grid->Children->Append(img);
			if (i == randI && j == randJ) {
				img->Source = ref new BitmapImage(ref new
Windows::Foundation::Uri("ms-appx:///Assets/Emoji_Smile.png"));
				for (int k = 0; k < grid->Children->Size; k++) {
					if (grid->Children->GetAt(k) == img)
						smilePos = k;
				}
			}
			else
```

```
                img->Source = ref new BitmapImage(ref new
Windows::Foundation::Uri(img->BaseUri->AbsoluteUri, "Assets/Emoji_Sad.png"));

            }
        startTime = time(NULL);
        timer->Start();


}


void HappySmile::MainPage::OnPointerReleased(Platform::Object ^sender,
Windows::UI::Xaml::Input::PointerRoutedEventArgs ^e)
{
        if (dynamic_cast<Image^>(grid->Children->GetAt(smilePos)) ==
dynamic_cast<Image^>(sender)) {
            size += 1;
            score += 100;
            scoreBlock->Text = "Score: " + score;
            loadLevel();
        }
        else {
            auto msg = ref new MessageDialog("Sorry", "You lose.");
            size = 5;

            saveToFile(score);

            score = 0;
            scoreBlock->Text = "Score: " + score;
            msg->ShowAsync();
            restartGame();
        }
}

void HappySmile::MainPage::OnClick(Platform::Object ^sender,
Windows::UI::Xaml::RoutedEventArgs ^e)
{
        loadLevel();
        scoreBlock->Text = "Score: 0";
}

void HappySmile::MainPage::OnTick(Platform::Object ^sender, Platform::Object
^args)
{
```

```
	auto remainTime = 10 -  (int)(time(NULL) - startTime);
	timeBlock->Text = "Time: " + remainTime;
	if (remainTime == 0) {
		timer->Stop();
		auto msg = ref new MessageDialog("Sorry", "You lose.");
		size = 5;

		score = 0;
		scoreBlock->Text = "Score: " + score;
		msg->ShowAsync();
		restartGame();
	}
}

void MainPage::saveToFile(int newScore) {


	StorageFolder^ storageFolder = ApplicationData::Current->LocalFolder;
	concurrency::create_task(storageFolder->CreateFileAsync("record.txt",
CreationCollisionOption::OpenIfExists));




	concurrency::create_task(storageFolder-
>GetFileAsync("record.txt")).then([=](StorageFile^ sampleFile)
	{
		concurrency::create_task(FileIO::AppendTextAsync(sampleFile,
nameBox->Text + ": " + newScore + "\n"));
	});

}

void MainPage::restartGame() {

	grid->Children->Clear();
	timer->Stop();
	scoreBlock->Text = "";
	timeBlock->Text = "";
	auto button = ref new Button();
	button->HorizontalAlignment =
Windows::UI::Xaml::HorizontalAlignment::Center;
	button->VerticalAlignment = Windows::UI::Xaml::VerticalAlignment::Center;
	button->Content = "Restart game";
```

```
	button->Click += ref new Windows::UI::Xaml::RoutedEventHandler(this,
&HappySmile::MainPage::restartOnClick);
	grid->Children->Append(button);
}

void HappySmile::MainPage::RecordOnClick(Platform::Object ^sender,
Windows::UI::Xaml::RoutedEventArgs ^e)
{
	if (isGame) {
		recordBotton->Content = "Continue";
		isGame = false;
		timer->Stop();
		scoreBlock->Text = "";
		timeBlock->Text = "";
		StorageFolder^ storageFolder = ApplicationData::Current->LocalFolder;
		concurrency::create_task(storageFolder->CreateFileAsync("record.txt",
CreationCollisionOption::OpenIfExists));
		concurrency::create_task(storageFolder-
>GetFileAsync("record.txt")).then([=](StorageFile^ sampleFile)
		{
			return FileIO::ReadBufferAsync(sampleFile);

		}).then([=](Streams::IBuffer^ buffer)
		{
			auto dataReader = Streams::DataReader::FromBuffer(buffer);
			auto bufferText = dataReader->ReadString(buffer->Length);

			grid->Children->Clear();
			auto textBlock = ref new TextBlock();
			textBlock->Height = 500;
			textBlock->Width = 500;
			textBlock->HorizontalAlignment =
Windows::UI::Xaml::HorizontalAlignment::Left;
			textBlock->VerticalAlignment =
Windows::UI::Xaml::VerticalAlignment::Top;
			textBlock->Text = bufferText;
			grid->Children->Append(textBlock);

		});
	}
	else {
		recordBotton->Content = "Highscores";
		isGame = true;
		scoreBlock->Text = "Score: " + 0;
```

```
            loadLevel();
        }

}


void HappySmile::MainPage::restartOnClick(Platform::Object ^sender,
Windows::UI::Xaml::RoutedEventArgs ^e)
{
        scoreBlock->Text = "Score: " + 0;
        loadLevel();
}
                                                MainPahe.xaml.h
//
// MainPage.xaml.h
// Объявление класса MainPage.
//

#pragma once

#include "MainPage.g.h"

namespace HappySmile
{
        /// <summary>
        /// Пустая страница, которую можно использовать саму по себе или для
перехода внутри фрейма.
        /// </summary>
        public ref class MainPage sealed
        {
        public:
                MainPage();

                void OnPointerReleased(Platform::Object ^sender,
Windows::UI::Xaml::Input::PointerRoutedEventArgs ^e);
        private:
                int smilePos;
                int size;
                void loadLevel();
                int score = 0;
                void OnClick(Platform::Object ^sender,
Windows::UI::Xaml::RoutedEventArgs ^e);
                int sec = 0;
                void OnTick(Platform::Object ^sender, Platform::Object ^args);
```

```
            void saveToFile(int newScore);
            void RecordOnClick(Platform::Object ^sender,
Windows::UI::Xaml::RoutedEventArgs ^e);
            void restartGame();
            void restartOnClick(Platform::Object ^sender,
Windows::UI::Xaml::RoutedEventArgs ^e);
        };
}
```