МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ КОМП’ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра

Економічної кібернетики

Комплексне практичне індивідуальне завдання

з дисципліни

«Конструювання ПЗ»

Виконав:

студент групи ПЗАС-32

Цвид Б.В.

Перевірив:

Сирник О.Й.

Тернопіль

2015

Зміст

Вступ…………………………………………….……………….…………….3

1. Загальна інформація про git……………………………………………….4
2. Программа Catch the Ball…………………………………….……………5
3. Лістинг коду…………………………………………………………….…..8

Висновки………………………………………………...………………….…15

Вступ

Аркадні комп'ютерні ігри почали з'являтись ще з самого початку функціонування комп’ютерів як таких. “Тетріс”, “Арканоїд” ,”Змійка” та багато інших ігор стали буквально культовими і не тільки для представників старшого покоління. Завдяки різним модифікаціям та нововведенням, ці ігри не втратили актуальності і по наш час. У даному КПІЗі представлена аркадна гра, яку я спробував створити опираючись на свої знання фреймворку .NET та мови С#.

1.Загальна інформація про git

В першу чергу потрібно вивести таке поняття як систему управління версіями.

Система управління версіями — програмний інструмент для керування версіями одиниці інформації: вихідного коду програми, скрипту, веб-сторінки, веб-сайту, 3D моделі, текстового документу тощо.Система керування версіями — це потужний інструмент, який дозволяє одночасно, без завад один одному, проводити роботу над груповими проектами.Системи керування версіями зазвичай використовуються при розробці програмного забезпечення для відстеження, документування та контролю над поступовими змінами в електронних документах: у ихідному коді, кресленнях, електронних моделях та інших документах, над змінами яких одночасно працюють декілька людей.Кожна версія позначається унікальною цифрою чи літерою, зміни документу занотовуються. Зазвичай також зберігається автор зробленої зміни та її час.Інструменти для контролю версій входять до складу багатьох інтегрованих середовищ розробки.

Система git спроектована як набір програм, спеціально розроблених з урахуванням їх використання в скриптах. Це дозволяє зручно створювати спеціалізовані системи контролю версій на базі Git або користувацькі інтерфейси. Наприклад, Cogito є саме таким прикладом оболонки до репозиторіїв Git, а StGit використовує Git для управління колекцією виправлень (патчів).

Git підтримує швидке розділення і злиття версій, включає інструменти для візуалізації і навігації по нелінійній історії розробки. Як і Darcs, BitKeeper, Mercurial, Bazaar і Monotone, Git надає кожному розробнику локальну копію всієї історії розробки, зміни копіюються з одного репозиторію в іншій.

Віддалений доступ до репозиторіїв Git забезпечується git-daemon, SSH- або HTTP-сервером. TCP-сервіс git-daemon входить в дистрибутив Git і є разом з SSH найбільш поширеним і надійним методом доступу. Метод доступу по HTTP, незважаючи на ряд обмежень, дуже популярний в контрольованих мережах, тому що дозволяє використовувати існуючі конфігурації мережевих фільтрів

2.Програма BallGame

Дана программа являє собою аркадну гру, суть якої заключається у наборі очок при кліканні на м'яч, який постійно рухається.

Так виглядає головне вікно програми:



Рис 1.1 – Головне вікно програми

Головне вікно містить у собі три навігаційні кнопки. Кнопка “Exit”, як зрозуміло з її назви закриває программу, а кнопка “Settings” дозволяє перейти у меню вибору теми для гри :



Рис 1.2 – Меню вибору теми

За замовчуванням у грі використовується тема Purple, але якщо її можна змінити:



Рис 1.3 – Оформлення програми при виборі теми Lime

Для того щоби почати гру, потрібно повернутись до головного меню та натиснути “New Game”. Після чого користувач побачить наступне вікно

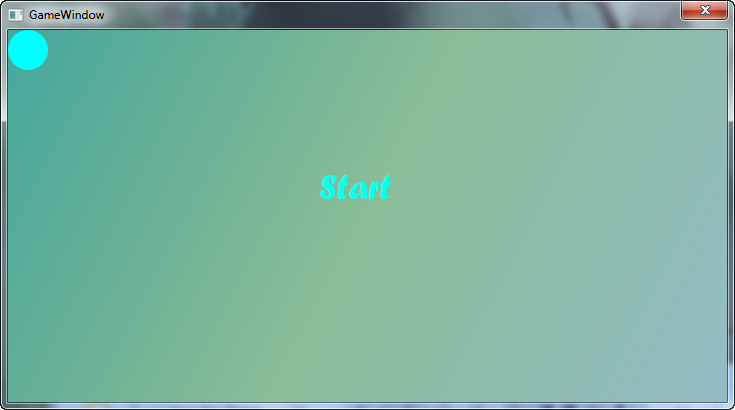


Рис 1.4 – Вікно початку гри

Після натискання на Start, гра починається, з'являються параметри гравця(життя та кількість набраних очок) і м'ячик починає рухатись, відбиваючись від стінок

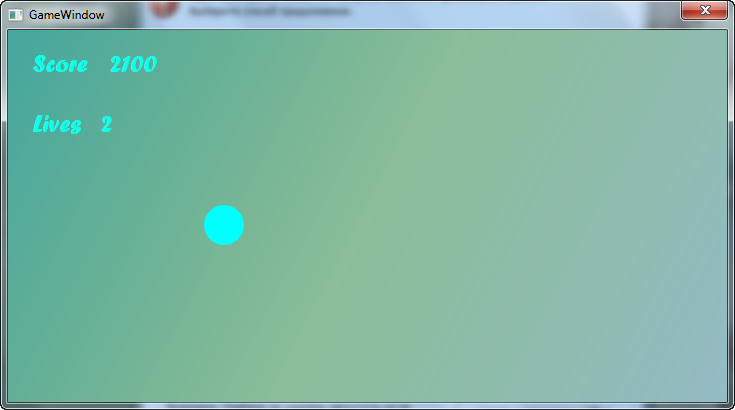


Рис 1.5 – Вікно гри після старту

Коли гравець вичерпує всі свої життя, гра закінчується, программа виводить набраний результат і повідомляє про закінчення гри:

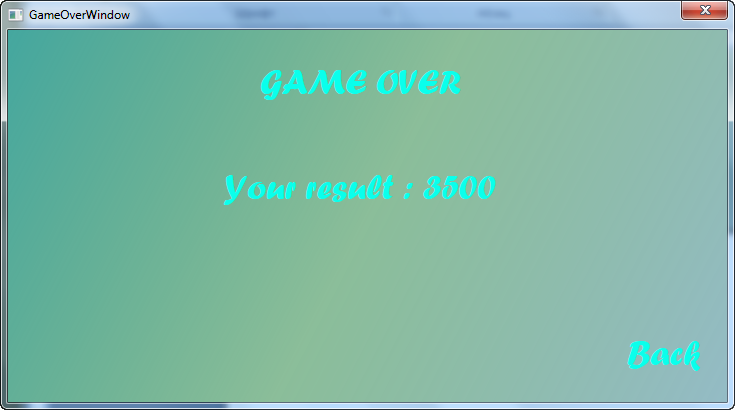


Рис 1.6 – Вікно після закінчення гри

3.Лістинг коду

MainWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.IO;

namespace ball\_game

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

// зафарбування вікна градієнтом при завантаженні

public MainWindow()

{

InitializeComponent();

// перевірка обраної користувачем теми

if (SettingsWindow.themeID == 1)

{

SettingsWindow.painting(255, 172, 150, 183, 181, 46, 31, 128, 245, 189, 104, 203, grid);

}

// перевірка обраної користувачем теми

if (SettingsWindow.themeID == 2)

{

SettingsWindow.painting(255, 68, 167, 159, 164, 74, 153, 97, 196, 116, 167, 180, grid);

}

}

// перехід до вікна з грою

private void label2\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

GameWindow gw = new GameWindow();

gw.Show();

this.Hide();

}

// закриття програми при закритті вікна

private void Window\_Closed(object sender, EventArgs e)

{

Environment.Exit(0);

}

// перехід у меню налаштувань

private void label4\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

SettingsWindow sw = new SettingsWindow();

sw.Show();

this.Hide(); } } }

GameWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using System.Timers;

using System.Windows.Media.Animation;

using System.IO;

namespace ball\_game

{

/// <summary>

/// Логика взаимодействия для GameWindow.xaml

/// </summary>

public partial class GameWindow : Window

{

public static int coeficient = 1; // ініціалізація змінної, яка буде відповідати за прискорення м'ячика

public static int lives = 4; // ініціалізація змінної ,яка характеризує "життя" гравця

public static int result = 0; // ініціалізація змінної, яка характеризує результат гравця

// блок змінних, необхідних для функціонування алгоритму пересування м'ячика

double X = 0;

double Y = 0;

double x = 0;

double y = 0;

System.Windows.Threading.DispatcherTimer timer = new System.Windows.Threading.DispatcherTimer();

// зафарбування вікна градієнтом при завантаженні

public GameWindow()

{

InitializeComponent();

// перевірка обраної користувачем теми

if (SettingsWindow.themeID == 1)

{

SettingsWindow.painting(255, 172, 150, 183, 181, 46, 31, 128, 245, 189, 104, 203, grid);

}

// перевірка обраної користувачем теми

if (SettingsWindow.themeID == 2)

{

SettingsWindow.painting(255, 68, 167, 159, 164, 74, 153, 97, 196, 116, 167, 180, grid);

}

}

// функція, яка пересуває м'ячик з кожним тіком таймера

private void timerTick(object sender, EventArgs e)

{

// рух м'ячика за траекторыэю X

if (x > 660)

{

X -= coeficient;

if (X < 0) x = 0;

};

if (x == 0)

{

X += coeficient;

if (X > 670) x = 670;

};

// рух м'ячика за траекторією Y

if (y >310)

{

Y -= coeficient;

if (Y < 0) y = 0;

};

if (y == 0)

{

Y += coeficient;

if (Y > 320) y = 320;

};

// перемальовування м'ячика згідно зі зміненими координатами

ell.Margin = new Thickness(X,Y,0,0);

}

// початок гри при натисканні на кнопку Start

private void label1\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

label1.Visibility = System.Windows.Visibility.Hidden;

label2.Visibility = System.Windows.Visibility.Visible;

label3.Visibility = System.Windows.Visibility.Visible;

ell.Visibility = System.Windows.Visibility.Visible;

timer.Tick += new EventHandler(timerTick); // ініціалізація таймера

timer.Interval = new TimeSpan(0, 0, 0, 0, 10); // проміжок, через який буде спрацьовувати таймер

timer.Start(); // початок роботи таймера

ell.MouseLeftButtonDown += ell\_MouseLeftButtonDown; // створення функції, яка буде виконуватись після кліку на об'єкт ell

}

// попадання гравця по м'ячику

private void ell\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

// пришвидшення руху м'ячика зі збільшенням очок

result += 100 ;

if (result > 1000)

coeficient = 2;

if (result > 2000)

coeficient = 3;

if (result > 3000)

coeficient = 4;

if (result > 4000)

coeficient = 5;

if (result > 5000)

coeficient = 6;

if (result > 6000)

coeficient = 7;

if (result > 7000)

coeficient = 8;

lives += 1;

label4.Content = result.ToString();

}

// непопадання гравця по м'ячику

private void Grid\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

lives -= 1;

label6.Content = lives.ToString();

GameOverWindow.result\_import = result;

// кінець гри, коли гравець витратив усі свої життя

if (lives == 0)

{

label1.Visibility = System.Windows.Visibility.Hidden;

label2.Visibility = System.Windows.Visibility.Hidden;

label3.Visibility = System.Windows.Visibility.Hidden;

label6.Visibility = System.Windows.Visibility.Hidden;

label4.Visibility = System.Windows.Visibility.Hidden;

ell.Visibility = System.Windows.Visibility.Hidden;

label9.Visibility = System.Windows.Visibility.Visible;

result = 0;

lives = 4;

coeficient = 1;

GameOverWindow gow = new GameOverWindow();

gow.Show();

this.Hide();

}

}

// вихід із програми при закритті вікна

private void Window\_Closed(object sender, EventArgs e)

{

//Environment.Exit(0);

}

// перехід до головного меню

private void label5\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

MainWindow mw = new MainWindow();

mw.Show();

this.Hide();

}

// перехід до головного меню

private void label9\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

MainWindow mw = new MainWindow();

mw.Show();

this.Hide();

}

}

}

GameOverWIndow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace ball\_game

{

/// <summary>

/// Логика взаимодействия для GameOverWindow.xaml

/// </summary>

public partial class GameOverWindow : Window

{

public static int result\_import;

// зафарбування вікна градієнтом при завантаженні

public GameOverWindow()

{

InitializeComponent();

// перевірка обраної користувачем теми

if (SettingsWindow.themeID == 1)

{

SettingsWindow.painting(255, 172, 150, 183, 181, 46, 31, 128, 245, 189, 104, 203, grid);

}

// перевірка обраної користувачем теми

if (SettingsWindow.themeID == 2)

{

SettingsWindow.painting(255, 68, 167, 159, 164, 74, 153, 97, 196, 116, 167, 180, grid);

}

}

// перехід до головног оменю

private void label9\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

MainWindow mw = new MainWindow();

mw.Show();

this.Hide();

}

// закриття програми при закритті вікна

private void Window\_Closed(object sender, EventArgs e)

{

Environment.Exit(0);

}

}

}

SettingsWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace ball\_game

{

/// <summary>

/// Логика взаимодействия для SettingsWindow.xaml

/// </summary>

public partial class SettingsWindow : Window

{

public static int themeID = 0;

// зафарбування вікна градієнтом при завантаженні

public SettingsWindow()

{

InitializeComponent();

// перевірка обраної користувачем теми

if (SettingsWindow.themeID == 1)

{

SettingsWindow.painting(255, 172, 150, 183, 181, 46, 31, 128, 245, 189, 104, 203, grid);

}

// перевірка обраної користувачем теми

if (SettingsWindow.themeID == 2)

{

SettingsWindow.painting(255, 68, 167, 159, 164, 74, 153, 97, 196, 116, 167, 180, grid);

}

}

// перехід до головного меню

public void label9\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

MainWindow mw = new MainWindow();

mw.Show();

this.Hide();

}

// функція, яка реалізовує зафарбування градієнтом вікна

public static void painting(byte o1, byte r1, byte g1, byte b1, byte o2, byte r2, byte g2, byte b2, byte o3, byte r3, byte g3, byte b3, Grid grid)

{

Color color1 = Color.FromArgb(o1, r1, g1, b1);

Color color2 = Color.FromArgb(o2, r2, g2, b2);

Color color3 = Color.FromArgb(o3, r3, g3, b3);

LinearGradientBrush gradient = new LinearGradientBrush();

gradient.StartPoint = new Point(0, 0);

gradient.EndPoint = new Point(1, 1);

gradient.GradientStops.Add(

new GradientStop(color1, 0.0));

gradient.GradientStops.Add(

new GradientStop(color2, 0.5));

gradient.GradientStops.Add(

new GradientStop(color3, 1.0));

grid.Background = gradient;

}

// вихід із програми при закритті вікна

private void Window\_Closed(object sender, EventArgs e)

{

Environment.Exit(0);

}

// вибір теми при натисканні на картинку

private void image2\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

themeID = 2;

image2.Opacity = 1;

image1.Opacity = 0.7;

}

// вибір теми при натисканні на картинку

private void image1\_MouseLeftButtonDown\_1(object sender, MouseButtonEventArgs e)

{

themeID = 1;

image1.Opacity = 1;

image2.Opacity = 0.7;

}

}

}

Висновки

При побудові даної програми я ознайомився із новими прийомами у програмуванні на .NET, покращив свої знання технології WPF та мови С#. Також я зрозумів певні принципи ігрової логіки та функціонування алгоритмів, потрібних для данного типу ігор.

Більше того, я ознайомився із технологією git, на основі сайту github.com та програми TortoiseGit.