1. **Latar Belakang**

Dalam era globalisasi dan teknologi sekarang ini, penggunaan komputer sebagai salah satu alat teknologi informasi sangat dibutuhkan keberadaannya hampir di setiap aspek kehidupan. Komputer saat ini tidak hanya digunakan dalam masalah bisnis, manajemen maupun proyek, tetapi juga digunakan untuk hiburan seperti contohnya bermain game.

Game adalah sebuah permainan video yang berbasis komputer yang digunakan sebagai alat hiburan sehari-hari yang bisa digunakan secara individu atau dalam sebuah kelompok kecil yang biasanya 2 hingga 4 orang. Game tentunya tidak asing lagi di mata masyarakat, bahkan perkembangan smart phone yang sangat berkembang akhir-akhir ini menyebabkan pemilik smart phone tergila-gila bahkan menghabiskan sebagian waktunya hanya untuk bermain game mobile baik dari segi **device**, **console**, **grafis**, **storyline**, **popularity**, **after effect** yang terdapat dalam game.

Fungsi dari game ini sendiri sangat banyak, diantaranya sebagai media hiburan, ilmu pembelajaran, melenyapkan rasa suntuk atau bosan bahkan membangun hubungan sosialisasi dalam bermain game. Bila kita mencoba melihat game komputer yang ada sekarang ini, maka kita dapatkan banyak sekali game dengan tipe dan variasi yang berbeda-beda. Jenis dan tipe game ini disebut dengan genre. Banyak cara untuk melihat bagaimana suatu game dapat dikelompokkan, karenanya tidak ada kesepakatan tentang apa dan bagaimana suatu game dapat dikelompokkan.

Game merupakan salah satu produk teknologi informasi yang cukup digemari saat ini, game juga merupakan bentuk aplikasi yang edukatif, artinya bisa dijadikan sebagai media pembelajaran dimana prosesnya bisa dilakukan dengan konsep belajar sambil bermain. Game telah mengalami kemajuan yang sangat pesat. Ini bisa kita lihat di kota-kota besar, tidak terkecuali juga kota-kota kecil, banyak sekali game center yang muncul. Game center itu sendiri tidak seperti halnya warnet, mereka memiliki pelanggan tetap yang lebih banyak daripada warnet. Inilah yang membuat game center hampir selalu ramai dikunjungi.

Game saat ini tidak seperti game terdahulu, jika dahulu game hanya bisa maksimal dimainkan dua orang, sekarang dengan kemajuan teknologi terutama jaringan internet, game bisa dimainkan 100 orang lebih sekaligus dalam waktu yang bersamaan.

Walaupun game ditujukan untuk anak-anak, tidak sedikit pula orang dewasa kerap memainkannya bahkan tidak sedikit yang menjadikannya sebagai pekerjaan dan mendapat penghasilan dari bermain game. Bahkan untuk bulan Juli mendatang akan terjadi **The International 4** untuk para tim game **DotA2** (Defense of the Ancient) berlaga dalam event setiap tahunnya yang diselenggarakan oleh **Valve** yang mempunyai prize pool hingga tercatat pada 23 Juni 2014 sudah mencapai **$ 9.869.222 USD**. Tetapi masih ada orang yang beranggapan membuat game itu susah, serta masih kurangnya minat para pecinta game untuk membuat game sendiri dengan cara yang mudah. Sehingga banyaknya game luar negeri yang di jual di Indonesia, dari pada game buatan Indonesia itu sendiri. Padahal pada kenyataannya membuat game itu tidak sesulit yang mereka bayangkan. Misalnya kita bisa membuat game kecil atau game flash hanya dengan waktu yang sangat singkat.

Permainan snake merupakan permainan populer dalam telepon selular beberapa tahun yang lalu. Inti dari permainan ini adalah agar snake yang kita kontrol mendapatkan sebanyak banyaknya makanan tanpa membentur dinding atau bagian tubuhnya sendiri. Semakin banyak makanan yang snake dapatkan, tubuhnya akan tumbuh sehingga semakin panjang. Sejatinya permainan ini merupakan single player game atau permainan yang dimainkan sendiri. Dalam permainan ini komputer hanya memunculkan makanan secara random /acak di layar untuk dimakan. Kesuksesan permainan ini bergantung kepada kecepatan dan perhitungan sang pemain agar ular yang dikontrolnya tidak terjebak dinding atau bagian tubuhnya sendiri.

Tetapi permainan snake pada telepon selular telah mengalami banyak perubahan dan mungkin tidak ada yang tidak tahu akan game ini karena kepopuleran dan simple yang membuat permainan snake ini merupakan nenek moyang nya para game yang telah ada.

Beberapa faktor mendorong penulis untuk merancang suatu game untuk dapat menghibur orang-orang yang bermain game ini, dengan membuat mereka tertawa karena game hiburan yang penulis buat, dan juga dapat membuat seorang anak kecil yang asalnya menangis dapat menjadi tertawa karena bermain game ini, dan juga orang-orang yang stress atau suntuk dan bahkan galau karena sesuatu pekerjaan atau masalah pribadinya dapat tertawa dan senyum kembali setelah bermain game ini, dan dapat melupakan masalah mereka. Serta untuk meningkatkan kecerdasan dan daya tangkap seorang anak dalam masa pertumbuhan. Dimana penulis menambahkan fitur baru yaitu dapat dimainkan oleh 2 player sekaligus dan membuktikan siapa yang memegang juara dalam permainan simple ini. Target penulis membuat game ini adalah untuk kalangan semua umur, mulai dari anak-anak sampai dewasa. Hal ini pula yang menjadi latar belakang penulis membuat permainan snake yang penulis sajikan dalam tugas ini yang berjudul **”Snake Multiplayer”**.

1. **Landasan Teori**

**Object Oriented Programming**

*Object oriented programming* adalah sebuah pendekatan pemrograman untuk menyelesaikan permasalahan secara fungsional. Dalam *object oriented programmin*g, sebuah struktur data yang merupakan objek tidak hanya mempunyai variable yang merupakan *state* tetapi terdapat fungsi(method) yang merupakan *behavior* yang dilakukan objek tersebut. Suatu objek dapat saling berkomunikasi satu sama lain dengan menggunakan fungsi yang ada di dalamnya tanpa perlu mengetahui *internal state* masing-masing objek (*data encapsulation*).

Keuntungan dalam menggunakan *object oriented programming* dibandingkan *procedural programming* adalah memungkinkan developer menggunakan fungsi tersebut pada lain waktu dengan hanya memanggil dan dalam perbaikan/*debugging* akan jauh lebih mudah karena telah terstruktur dan perubahan hanya terletak pada satu lokasi saja. Dalam *object oriented programming* ini sering digunakan penurunan yang dapat mempersingkat dan merapikan aplikasi tersebut. Hal ini membuat *object oriented programming* mudah dalam pengembangannya.

Penggunakan static variable juga sangat membantu karena penggunaan variable global dapat menyebabkan program dapat diubah oleh pihak lain dengan gampang. Juga terdapat pemilihan pendefinisian pada variable, method, assessor maupun mutator yaitu public, private dan protected. Perbedaan antara ketiga ini adalah:

* **Private,** hanya dapat diakses oleh kelasnya sendiri. Sehingga penggunaan ini menjamin keamanan class tersebut.
* **Protected,** dapat diakses oleh kelasnya sendiri dan kelas turunannya. Ini kebanyakan digunakan untuk kelas parent.
* **Public,** dapat diakses oleh kelasnya maupun kelas lain baik program. Ini merupakan cara tergampang dalam pemilihan tetapi security dan keamanan nya tidak terjamin.

Umumnya terdiri dari 3 teknik, yaitu :

1. **Enkapsulasi,** suatu cara untuk menyembunyikan informasi detail dari suatu kelas. Biasanya digunakan pendefinisian private dan protected agar informasi dari sifat nya terlindungi dan tidak dapat diakses dari luar dengan bebas.
2. **Inheritance,** suatu cara untuk membuat suatu kelas dapat menggunakan dan mempunyai sifat dari kelas lainnya, baik method, variable dan sifat-sifat lainnya. Kelas yang mewarisi sifat(inherited) dapat mempunyai perubahan dari kelas yang diwarisi sifatnya oleh kelas tersebut.
3. **Polymorphism,** suatu cara untuk membuat suatu kelas mempunyai sifat yang sama dengan kelas lain, sehingga kelas tersebut dapat menjadi kelas lain secara implisit. Biasanya sifat tersebut hampir menyerupai satu sama lain.

**Threading**

Thread yang biasanya kebanyakan orang menyamakan nya dengan multitasking atau multithreading yang artinya proses yang ada dapat dijalankan sekaligus tanpa harus saling menunggu untuk mempersingkat waktu pengerjaan dan proses yang diperlukan.

Kita menggunakan threading dalam snake multiplayer ini untuk membagi proses dari snake yang berbeda dapat dijalankan secara bersamaan. Sehingga kedua snake ini tidak perlu menunggu satu sama lain dalam permainan ini, tentu saja kita tidak mau hal itu terjadi, oleh karena itu, penggunaan threading ini sangat membantu dalam project ini.

Dalam Visual Studio C Sharp(#) telah terdapat library thread sehingga teknik ini dapat diterapkan langsung pada project ini. Walaupun penulis hanya menggunakan maksimal 2 threading yang digunakan saat kita memainkan dual player dalam project ini, tetapi threading ini dapat dengan beberapa kelebihan, yaitu:

1. **Tanggap,** Threading mengizinkan program untuk berjalan terus walaupun pada bagian program tersebut di-*block* atau sedang dalam keadaan menjalankan operasi yang lama atau panjang.
2. **Pembagian sumber daya,** Secara *default*, thread membagi memori dan sumber daya dari proses. Keuntungan dari pembagian kode adalah aplikasi mempunyai perbedaan aktifitas *thread* dengan alokasi memori yang sama.
3. **Ekonomis,** Mengalokasikan memori dan sumber daya sangat membuang waktu proses. Alternatifnya, karena *thread* membagi sumber daya dari proses, ini lebih ekonomis.
4. **Pembahasan**

Snake Multiplayer dibuat menggunakan console application dan digabung dengan menu yang dibuat menggunakan Windows Form application.

Console application menggunakan library:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading; //added

using System.IO; //added

Windows Form application menggunakan library:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

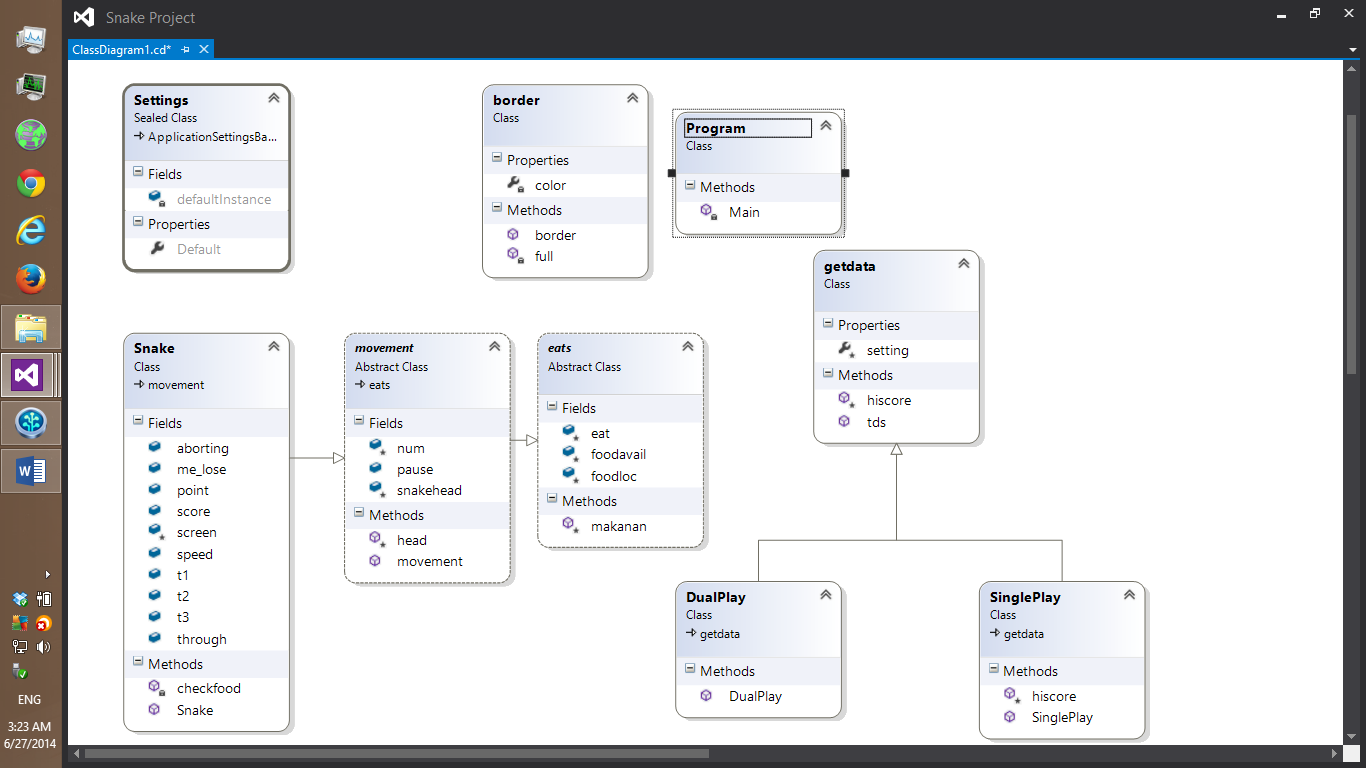
using System.IO; //added

using System.Diagnostics; //added

using System.Media; //added

1. System.Threading kami gunakan untuk melakukan Thread.Sleep(milisecond) pada snake.
2. System.IO kami gunakan untuk menulis data pada satu file seperti setting dan lainnya. Kami menggunakan File.ReadAllText(“dir/data”) untuk membaca text, File.WriteAllText(“dir/data”) untuk menulis text ke file, dan new File Info(“dir/data”).Exists untuk mengecek keadaan data sudah ada atau belum.
3. System.Diagnostics kami gunakan untuk menjalankan program external karena form application dan console application kami buat secara terpisah. Kami menggunakan Process.Start(“execfile”) untuk menjalankan file exec yang dituju.
4. System.Media kami gunakan untuk melakukan perputaran music. Kami menggunakan SoundPlayer sound = new SoundPlayer(“wavfile”) dan sound.Play() untuk memainkan music tersebut

Pada console snake, kami menambahkan 7 class:



**Class eats** menentukan lokasi makanan

**Class movement:eats** menentukan arah snake

**Class Snake:movement** menentukan seluruh bagian dari snake

**Class getdata** mengambil data yang ada+memberi pause sejenak

**Class DualPlay:getdata** membuat dualplayer

**Class SinglePlay:getdata** membuat singleplayer

**Class Border** membuat border

Karena applikasi kami dibuat secara terpisah console snake kami menggunakan args dengan kata lain harus ada argument tertentu untuk mengakses **Console Snake.exe** kami jadi untuk menjalankannya harus dijalankan melalui applikasi windows form kami yaitu **Snake MultiPlayer.exe**. Dan applikasi ini dibuat dengan Library **.NET Framework 4.5 (Visual Studio 2012)** dengan kata lain untuk menjalankan applikasi hendaknya terlebih dahulu menginstall **.NET Framework 4.5** karena kami dari awal sudah terlanjur menggunakan .NET Framework 4.5 dan kami takut jika mengubah .NET Framework secara paksa akan mengakibatkan error pada applikasi kami.

1. **Kesimpulan**

Dibuatnya Snake Multiplayer berdasarkan atas kecintaan para penulis akan game dan memiliki tujuan untuk menghibur para gamers yang bermain permainan ini. Karena selain menghibur, game ini juga menguji kecerdasan dan daya tangkap bagi para pemain baik dari kalangan anak-anak, remaja, maupun dewasa karena game ini tidak memiliki batasan umur untuk memainkannya.

Penciptaan snake multiplayer juga menggunakan beberapa fitur dari permainan Snake lama dan di modifikasi sehingga dapat dimainkan lebih dari 1 orang. Pengerjaan snake menggunakan menggunakan *Object oriented programming* yang merupakan pendekatan pemrograman untuk menyelesaikan permasalahan secara fungsional dengan memanfaatkan keuntungan OOP yang memungkinkan developer menggunakan fungsi tersebut pada lain waktu dengan hanya memanggil dan dalam perbaikan/*debugging* akan jauh lebih mudah karena telah terstruktur dan perubahan hanya terletak pada satu lokasi saja dimana kita menggunakannya pada saat penurunan. Pada OOP juga terdapat 3 teknik yang dapat membantu pengerjaan Snake Multiplayer tersebut, 3 teknik tersebut meliputi enkapsulasi, inheritance, dan polymorphism.

Hal yang tak kalah penting yaitu Threading dimana kita dapat membagi proses dari snake yang berbeda dapat dijalankan secara bersamaan. Sehingga kedua snake ini tidak perlu menunggu satu sama lain dalam permainan yang dijalankan agar tidak terjadi kecurangan.

Untuk pengerjaan design kami menggunakan fitur Adobe Photoshop dan memasukkannya pada program Form Application. Tentu saja, sebelum memasuki permainan kita akan diberikan tampilan gambar. Untuk penentuan Kemenangan pada Snake Multiplayer didasarkan pada Player yang menjalankan Console dimana jika Player menabrak snake Player lainnya, maka Player yang ditabrak akan menang, tetapi bila kepala snake mereka bertemu akan terjadi Draw!.

Untuk penentuan Score didasarkan atas panjang pendeknya Snake yang ada, dimana panjang dari snake dapat bertambah jika memakan makanan pada permainan Snake Multiplayer. Score diproses pada Solo Player, untuk Multiplayer Score tidak akan diproses karena adanya 2 pemain yang bermain sekaligus.

1. **Source Code**

Console Application

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading;

using System.IO;

using System.Security;

namespace Console\_Snake

{

class Program

{

static void Main(string[] args)

{

switch(args[0])

{

case "1":

new SinglePlay();

break;

case "2":

new DualPlay();

break;

default:

break;

}

}

}

}

border.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Console\_Snake

{

class border

{

private ConsoleColor color

{

get

{

if (Snake.through)

return ConsoleColor.Green;

else

return ConsoleColor.Magenta;

}

}

private void full()

{

for (int i = 0; i < 80; i++)

{

Console.ForegroundColor = color;

Console.Write("#");

}

}

public border()

{

full();

for (int i = 0; i < 24; i++)

{

Console.Write("#");

Console.SetCursorPosition(79, i + 1);

Console.Write("#");

}

full();

Console.SetCursorPosition(0, 26);

Console.WriteLine("Score:");

}

}

}

eats.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Console\_Snake

{

abstract class eats

{

static protected bool eat = false;

static protected bool foodavail = false;

static protected int[] foodloc = new int[2];

static protected void makanan(bool[,] loc)

{

Random x = new Random();

Random y = new Random();

bool locavailable = false;

do

{

int rx = x.Next(1,78);

int ry = y.Next(1,23);

bool c = false;

if (loc[rx,ry]==true)

{

c = true;

}

if (!c)

{

locavailable = true;

Console.SetCursorPosition(rx, ry);

Console.ForegroundColor = ConsoleColor.Yellow;

Console.Write((char)15);

foodavail = true;

foodloc[0] = rx;

foodloc[1] = ry;

}

} while (!locavailable);

}

}

}

movement.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading;

namespace Console\_Snake

{

abstract class movement:eats

{

static public bool pause = false;

protected byte num=1;

static protected byte[] snakehead = new byte[2];//1=up 2=down 3=left 4= right

public movement()

{

snakehead[0] = 1;

snakehead[1] = 1;

}

protected void head(ConsoleKeyInfo i)

{

{

if (i.Key == ConsoleKey.UpArrow && snakehead[0] != 2&&!pause)

snakehead[0]=1;

else if (i.Key == ConsoleKey.DownArrow && snakehead[0] != 1 && !pause)

snakehead[0]=2;

else if (i.Key == ConsoleKey.LeftArrow && snakehead[0] != 4 && !pause)

snakehead[0]=3;

else if (i.Key == ConsoleKey.RightArrow && snakehead[0] != 3 && !pause)

snakehead[0]=4;

else if (i.Key == ConsoleKey.W && snakehead[1] != 2 && !pause)

snakehead[1] = 1;

else if (i.Key == ConsoleKey.S && snakehead[1] != 1 && !pause)

snakehead[1] = 2;

else if (i.Key == ConsoleKey.A && snakehead[1] != 4 && !pause)

snakehead[1] = 3;

else if (i.Key == ConsoleKey.D && snakehead[1] != 3 && !pause)

snakehead[1] = 4;

else if(i.Key==ConsoleKey.P)

pause=!pause;

}

}

}

}

snake.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading;

using System.IO;

namespace Console\_Snake

{

class Snake:movement

{

public bool me\_lose=false;

static public bool through = true;

public int score = 0;

static public int speed=200;

static private void checkfood(int x, int y)

{

if (x == foodloc[0] && y == foodloc[1])

{

eat = true;

foodavail = false;

}

}

static public bool aborting = false;

static public Thread t1, t3;

public Thread t2;

public List<int[]> point;

static protected bool[,] screen=new bool[80,26];

public Snake(int x=2,int y=2,byte nomor=1,ConsoleColor color=ConsoleColor.Cyan)

{

num = nomor;

int tailx =x;

int taily =y+2;

point = new List<int[]>();

for (int i = taily; i > y; i--)

{

int[] xy1 = { tailx, i };

point.Add(xy1);

screen[tailx, i] = true;

}

Console.ForegroundColor = color;

Console.SetCursorPosition(x, y);

Console.Write((char)2);

for (int i = 0; i < point.LongCount(); i++)

{

int tx, ty;

tx = point[i][0];

ty = point[i][1];

Console.SetCursorPosition(tx, ty);

Console.ForegroundColor = color;

Console.Write((char)2);

}

Console.SetCursorPosition(point[0][0], point[0][1]);

screen[point[0][0], point[0][1]] = false;

Console.Write(' ');

Console.ForegroundColor = ConsoleColor.Yellow;

Console.SetCursorPosition(0, 26 + num);

score=point.Count-2;

Console.WriteLine("Player{0} : {1}", num, score);

t2 = new Thread(new ThreadStart(delegate

{

while (true)

{

Console.SetCursorPosition(x,y);

Console.ForegroundColor = color;

Console.Write((char)2);

int[] tmp1 = { x, y };

if (screen[x, y] == true)

{

point.Add(tmp1);

aborting = true;

t1.Abort();

}

point.Add(tmp1);

screen[x, y] = true;

checkfood(x, y);

if (!foodavail)

{

makanan(screen);

foodavail = true;

}

if (!eat)

{

Console.SetCursorPosition(point[0][0], point[0][1]);

screen[point[0][0], point[0][1]] = false;

Console.Write(' ');

point.RemoveAt(0);

}

else

{

Console.SetCursorPosition(0, 26 + num);

score=point.Count-2;

Console.WriteLine("Player{0} : {1}", num, score);

if (point[point.Count - 1][0] == x && point[point.Count - 1][1] == y)

eat = false;

else

{

Console.SetCursorPosition(point[0][0], point[0][1]);

screen[point[0][0], point[0][1]] = false;

Console.Write(' ');

point.RemoveAt(0);

}

}

Thread.Sleep(speed);

if (snakehead[nomor-1] == 1)

{

if (y > 1)

{

y--;

}

else

{

if (through)

y += 23;

else

{

me\_lose=true;

Thread.Sleep(100);

aborting = true;

Thread.Sleep(100);

}

}

}

else if (snakehead[nomor-1] == 2)

{

if (y<24)

{

y++;

}

else

{

if (through)

y = 1;

else

{

me\_lose = true;

Thread.Sleep(100);

aborting = true;

Thread.Sleep(100);

}

}

}

else if (snakehead[nomor-1] == 3)

{

if (x > 1)

{

x--;

}

else

{

if (through)

x += 77;

else

{

me\_lose = true;

Thread.Sleep(100);

aborting = true;

Thread.Sleep(100);

}

}

}

else if (snakehead[nomor-1] == 4)

{

if (x < 78)

{

x++;

}

else

{

if (through)

x = 1;

else

{

me\_lose = true;

Thread.Sleep(100);

aborting = true;

Thread.Sleep(100);

}

}

}

}

}));

t1 = new Thread(new ThreadStart(delegate

{

while (t2.IsAlive)

{

head(Console.ReadKey(true));

Thread.Sleep(speed/2+speed/4);

}

}));

t3 = new Thread(new ThreadStart(delegate

{

bool cek1 = false;

while (true)

{

if (pause)

{

cek1 = true;

t2.Suspend();

Console.SetCursorPosition(60, 26);

Console.ForegroundColor = ConsoleColor.Yellow;

Console.Write("Pause!!!");

}

else

{

if (cek1)

{

cek1 = false;

Console.SetCursorPosition(60, 26);

Console.ForegroundColor = ConsoleColor.Yellow;

Console.Write(" ");

new getdata().tds();

t2.Resume();

}

}

}

}));

}

}

}

getdata.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading;

using System.IO;

namespace Console\_Snake

{

class getdata

{

protected string[] setting

{

get

{

if (new FileInfo("setting.dat").Exists)

{

return File.ReadAllText("setting.dat").Split(' ');

}

else

return null;

}

}

public void tds()

{

for (int i = 3; i > 0; i--)

{

Console.SetCursorPosition(60, 26);

Console.ForegroundColor = ConsoleColor.Yellow;

Console.Write(i);

Thread.Sleep(900);

}

Console.SetCursorPosition(60, 26);

Console.ForegroundColor = ConsoleColor.Yellow;

Console.Write("GO!!!");

Thread.Sleep(500);

Console.SetCursorPosition(60, 26);

Console.ForegroundColor = ConsoleColor.Yellow;

Console.Write(" ");

}

protected virtual int hiscore()

{

return 0;

}

}

}

SinglePlay.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading;

using System.IO;

namespace Console\_Snake

{

class SinglePlay:getdata

{

protected override int hiscore()

{

if (new FileInfo("score.dat").Exists)

return int.Parse(File.ReadAllText("score.dat"));

else

return base.hiscore();

}

public SinglePlay()

{

Snake.speed = 200;

Snake.through = true;

if (setting != null)

{

if (setting[1] == "0")

Snake.through = false;

else

Snake.through = true;

int ss = int.Parse(setting[0]) \* 10;

Snake.speed = 250 - ss;

}

border a = new border();

Console.SetWindowSize(Console.WindowWidth, 30);

Snake snake1 = new Snake(40,10);

Console.SetCursorPosition(0, 0);

Console.SetCursorPosition(40, 26);

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("Hi-Score = " + hiscore());

tds();

snake1.t2.Start();

Snake.t1.Start();

Snake.t3.Start();

while (!Snake.aborting)

{

}

Console.SetCursorPosition(20, 26);

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("GAME OVER!!!");

snake1.t2.Abort();

Snake.t3.Abort();

Snake.t1.Abort();

int hi\_score = hiscore();

if (snake1.score > hi\_score)

{

Console.SetCursorPosition(20, 27);

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("Hi-Score");

File.WriteAllText("score.dat", (snake1.score).ToString());

Console.SetCursorPosition(40, 26);

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("Hi-Score = " + hiscore());

}

Console.ReadKey();

}

}

}

DualPlay.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading;

using System.IO;

namespace Console\_Snake

{

class DualPlay:getdata

{

public DualPlay()

{

Snake.speed = 200;

Snake.through = true;

if (setting != null)

{

if (setting[1] == "0")

Snake.through = false;

else

Snake.through = true;

int ss = int.Parse(setting[0]) \* 10;

Snake.speed = 250 - ss;

}

border a = new border();

Console.SetWindowSize(Console.WindowWidth, 30);

Snake snake1 = new Snake(50,10);

Snake snake2 = new Snake(2, 10, 2, ConsoleColor.Red);

Console.SetCursorPosition(0, 0);

tds();

snake1.t2.Start();

Thread.Sleep(75);

snake2.t2.Start();

Snake.t1.Start();

Thread pausing = new Thread(new ThreadStart(delegate

{

bool cek1 = false;

while (true)

{

if (Snake.pause)

{

cek1 = true;

snake1.t2.Suspend();

snake2.t2.Suspend();

Console.SetCursorPosition(60, 26);

Console.ForegroundColor = ConsoleColor.Yellow;

Console.Write("Pause!!!");

}

else

{

if (cek1)

{

cek1 = false;

Console.SetCursorPosition(60, 26);

Console.Write(" ");

tds();

snake1.t2.Resume();

Thread.Sleep(75);

snake2.t2.Resume();

}

}

}

}));

pausing.Start();

while (!Snake.aborting)

{

}

snake1.t2.Abort();

snake2.t2.Abort();

pausing.Abort();

Snake.t1.Abort();

bool snakea=false;

bool snakeb=false;

snakea = snake1.me\_lose;

snakeb = snake2.me\_lose;

foreach (var x in snake2.point)

{

if (snake1.point.Last()[0] == x[0] && snake1.point.Last()[1] == x[1])

{

if (snake2.point.Last()[0] != x[0] && snake2.point.Last()[1] != x[1])

{

snakeb = true;

break;

}

snakea = true;

break;

}

}

foreach (var x in snake1.point)

{

if (snake2.point.Last()[0] == x[0] && snake2.point.Last()[1] == x[1])

{

if (snake1.point.Last()[0] != x[0] && snake1.point.Last()[1] != x[1])

{

snakeb = true;

break;

}

}

}

if (snake2.point.Last()[0] == snake1.point.Last()[0] && snake2.point.Last()[1] == snake1.point.Last()[1])

{

snakea = true;

snakeb = true;

}

if (snakea && snakeb)

{

Console.SetCursorPosition(20, 26);

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("GAME OVER!!!");

if (snake1.score > snake2.score)

{

Console.SetCursorPosition(20, 27);

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("Player1 Wins!!!");

}

else if (snake1.score < snake2.score)

{

Console.SetCursorPosition(20, 27);

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("Player2 Wins!!!");

}

else

{

Console.SetCursorPosition(20, 27);

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("Draw!!!");

}

}

else if (snakea && !snakeb)

{

Console.SetCursorPosition(20, 26);

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("GAME OVER!!!");

Console.SetCursorPosition(20, 27);

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("Player2 Wins!!!");

}

else if (!snakea && snakeb)

{

Console.SetCursorPosition(20, 26);

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("GAME OVER!!!");

Console.SetCursorPosition(20, 27);

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("Player1 Wins!!!");

}

else if (!snakea && !snakeb)

{

int counta = 0;

foreach (var x in snake1.point)

{

if (snake1.point.Last()[0] == x[0] && snake1.point.Last()[1] == x[1])

{

counta++;

}

}

if (counta > 1)

{

Console.SetCursorPosition(20, 26);

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("GAME OVER!!!");

Console.SetCursorPosition(20, 27);

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("Player2 Wins!!!");

}

else

{

Console.SetCursorPosition(20, 26);

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("GAME OVER!!!");

Console.SetCursorPosition(20, 27);

Console.ForegroundColor = ConsoleColor.Magenta;

Console.WriteLine("Player1 Wins!!!");

}

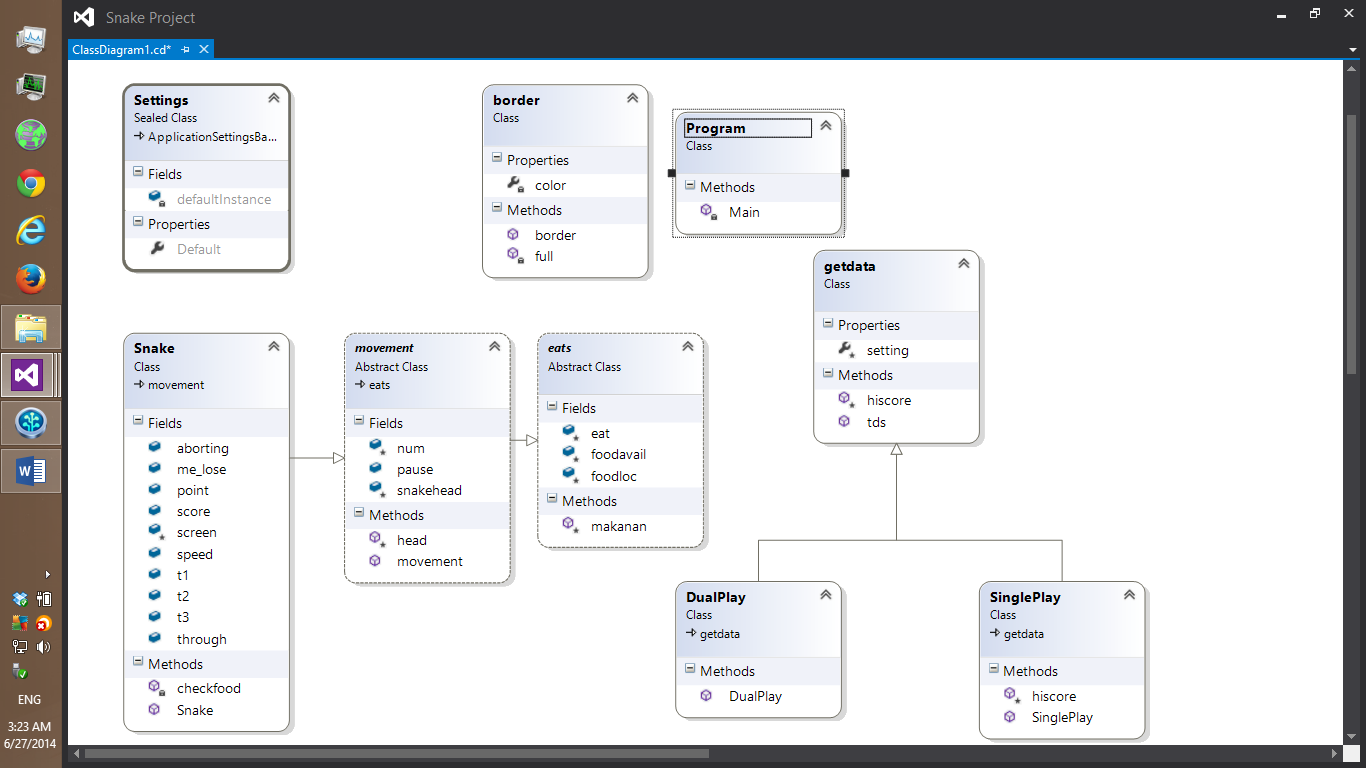
}

Console.ReadKey();

}

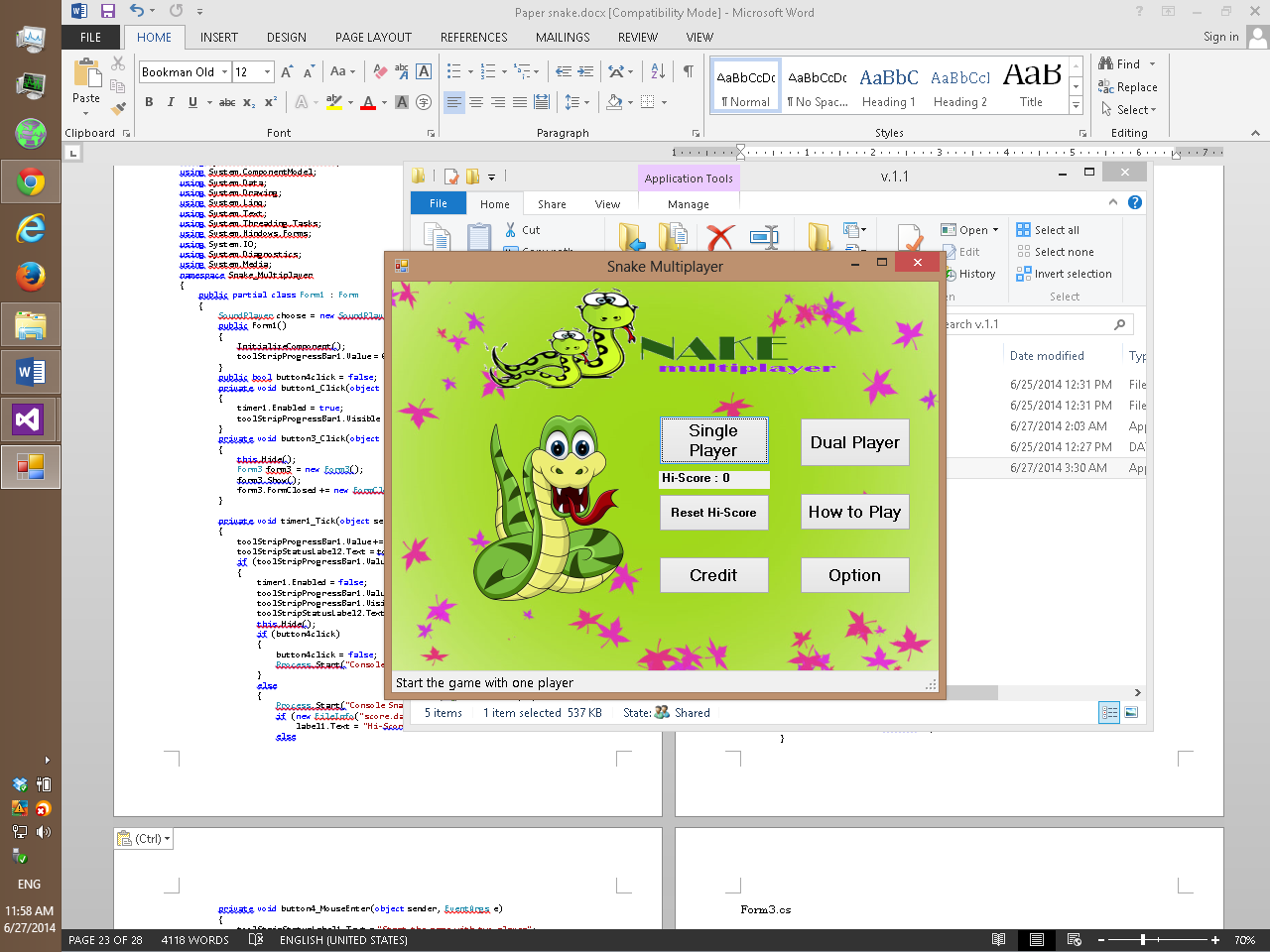
}

}



Snake MultiPlayer Form Application source code

Form1.cs



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

using System.Diagnostics;

using System.Media;

namespace Snake\_Multiplayer

{

public partial class Form1 : Form

{

SoundPlayer choose = new SoundPlayer("Music/up-down.wav");

public Form1()

{

InitializeComponent();

toolStripProgressBar1.Value = 0;

}

public bool button4click = false;

private void button1\_Click(object sender, EventArgs e)

{

timer1.Enabled = true;

toolStripProgressBar1.Visible = true;

}

private void button3\_Click(object sender, EventArgs e)

{

this.Hide();

Form3 form3 = new Form3();

form3.Show();

form3.FormClosed += new FormClosedEventHandler(muncul);

}

private void timer1\_Tick(object sender, EventArgs e)

{

toolStripProgressBar1.Value += 1;

toolStripStatusLabel2.Text = toolStripProgressBar1.Value.ToString("D2") + "%";

if (toolStripProgressBar1.Value == toolStripProgressBar1.Maximum)

{

timer1.Enabled = false;

toolStripProgressBar1.Value = 0;

toolStripProgressBar1.Visible = false;

toolStripStatusLabel2.Text = "Complete";

this.Hide();

if (button4click)

{

button4click = false;

Process.Start("Console Snake.exe", "2").WaitForExit();

}

else

{

Process.Start("Console Snake.exe", "1").WaitForExit();

if (new FileInfo("score.dat").Exists)

label1.Text = "Hi-Score : " + File.ReadAllText("score.dat");

else

label1.Text = "Hi-Score : 0";

}

this.Show();

}

}

private void Form1\_MouseEnter(object sender, EventArgs e)

{

toolStripStatusLabel1.Text = "";

}

private void button1\_MouseEnter(object sender, EventArgs e)

{

toolStripStatusLabel1.Text = "Start the game with one player";

choose.Play();

}

private void button2\_MouseEnter(object sender, EventArgs e)

{

toolStripStatusLabel1.Text = "See How to Play snake";

choose.Play();

}

private void button3\_MouseEnter(object sender, EventArgs e)

{

toolStripStatusLabel1.Text = "See the credit of the snake multiplayer";

choose.Play();

}

private void button2\_Click(object sender, EventArgs e)

{

this.Hide();

Form2 form2 = new Form2();

form2.Show();

form2.FormClosed+=new FormClosedEventHandler(muncul);

}

private void muncul(object sender, EventArgs e)

{

this.Show();

}

private void button4\_Click(object sender, EventArgs e)

{

button4click = true;

timer1.Enabled = true;

toolStripProgressBar1.Visible = true;

}

private void Form1\_Load(object sender, EventArgs e)

{

if (new FileInfo("score.dat").Exists)

label1.Text ="Hi-Score : "+ File.ReadAllText("score.dat");

else

label1.Text = "Hi-Score : 0";

}

private void button5\_Click(object sender, EventArgs e)

{

new FileInfo("score.dat").Delete();

label1.Text ="Hi-Score : 0";

}

private void button4\_MouseEnter(object sender, EventArgs e)

{

toolStripStatusLabel1.Text = "Start the game with two player";

choose.Play();

}

private void button5\_MouseEnter(object sender, EventArgs e)

{

toolStripStatusLabel1.Text = "Reset current Hi-Score";

choose.Play();

}

private void button6\_MouseEnter(object sender, EventArgs e)

{

toolStripStatusLabel1.Text = "Configure the snake game";

choose.Play();

}

private void button6\_Click(object sender, EventArgs e)

{

this.Hide();

Form4 form4 = new Form4();

form4.Show();

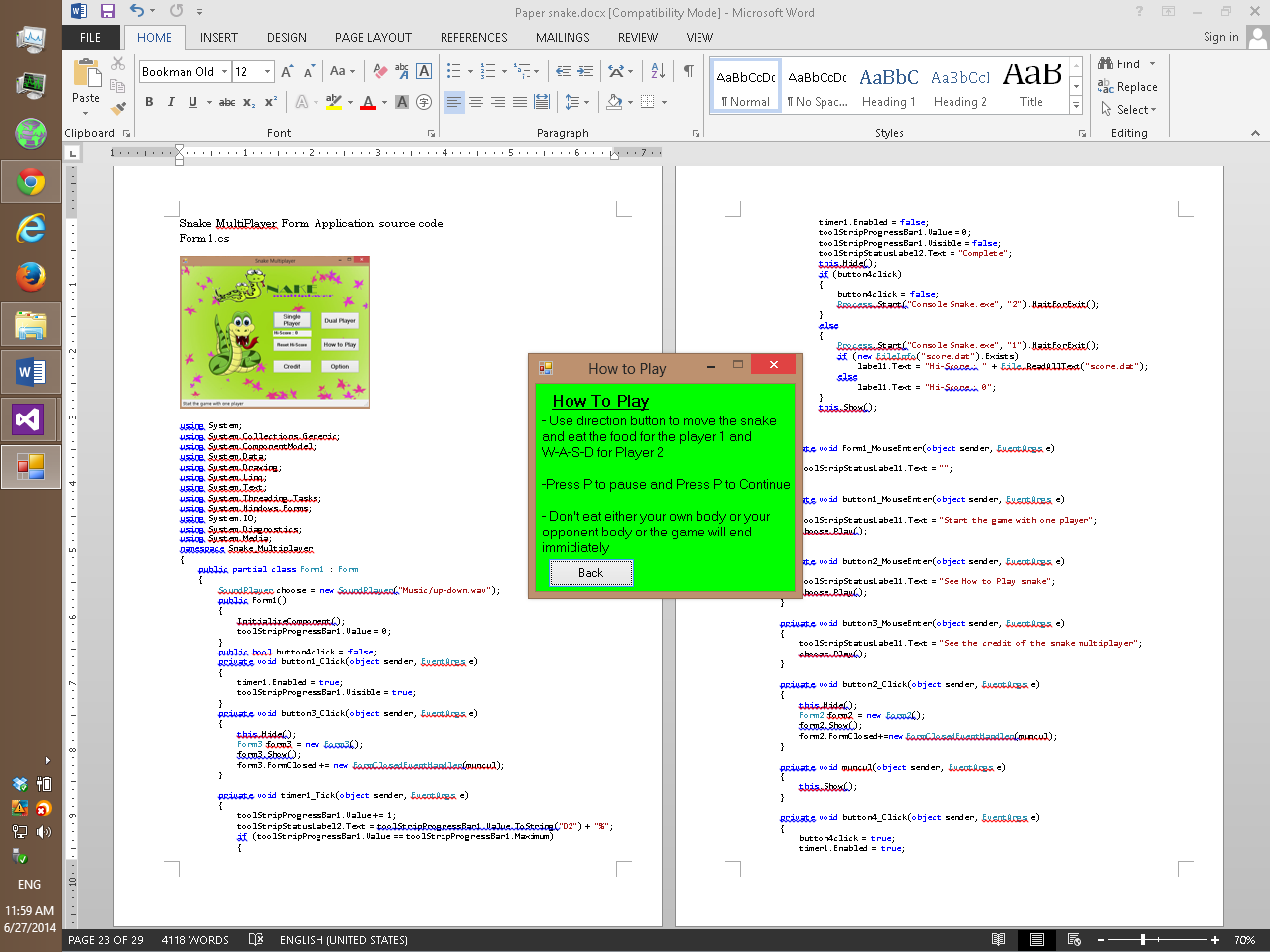
form4.FormClosed += new FormClosedEventHandler(muncul);

}

}

}

Form2.cs



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Media;

namespace Snake\_Multiplayer

{

public partial class Form2 : Form

{

SoundPlayer choose = new SoundPlayer("Music/up-down.wav");

public Form2()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button1\_MouseEnter(object sender, EventArgs e)

{

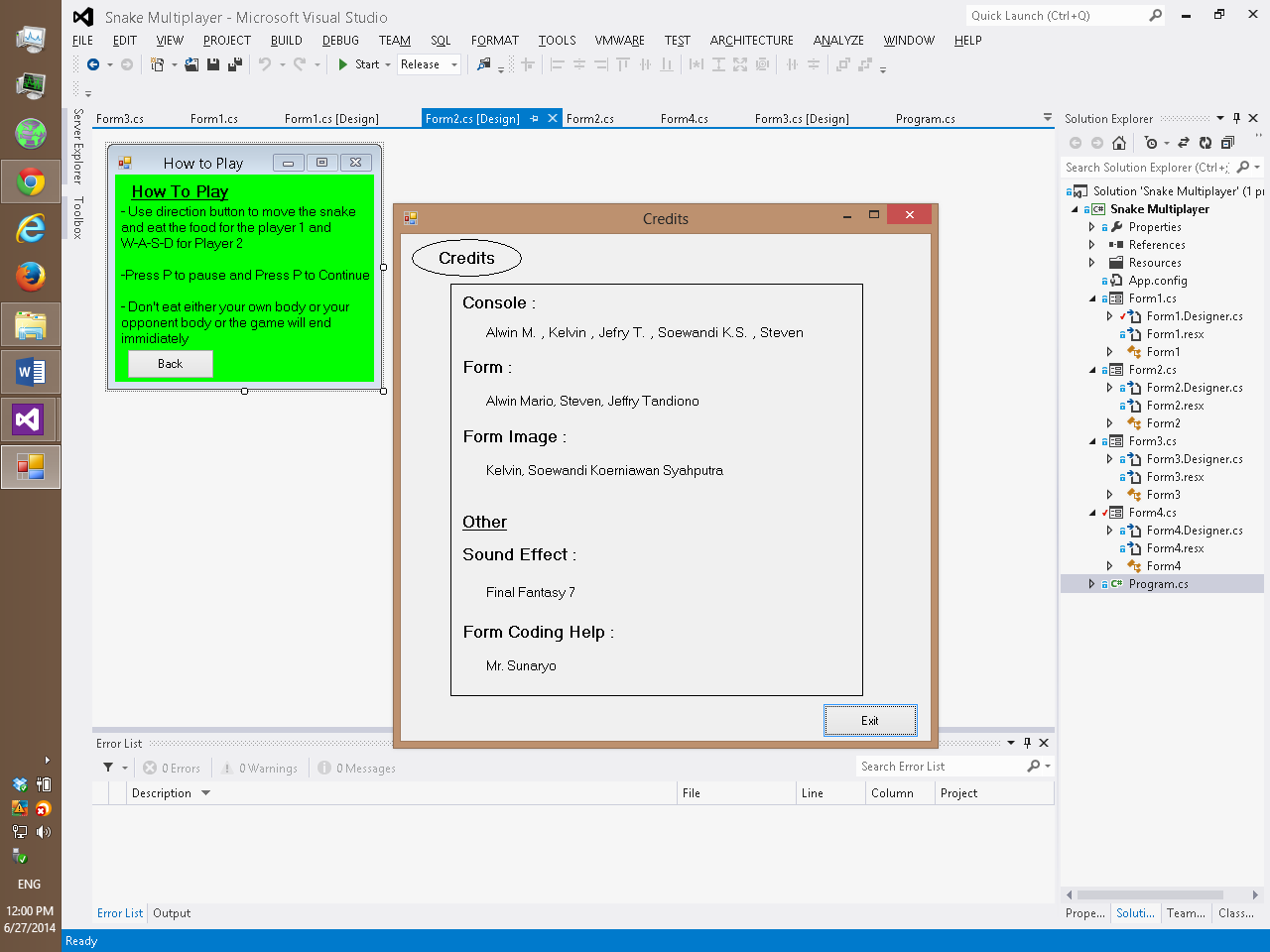
choose.Play();

}

}

}

Form3.cs



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Media;

namespace Snake\_Multiplayer

{

public partial class Form3 : Form

{

SoundPlayer choose = new SoundPlayer("Music/up-down.wav");

public Form3()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button1\_MouseEnter\_1(object sender, EventArgs e)

{

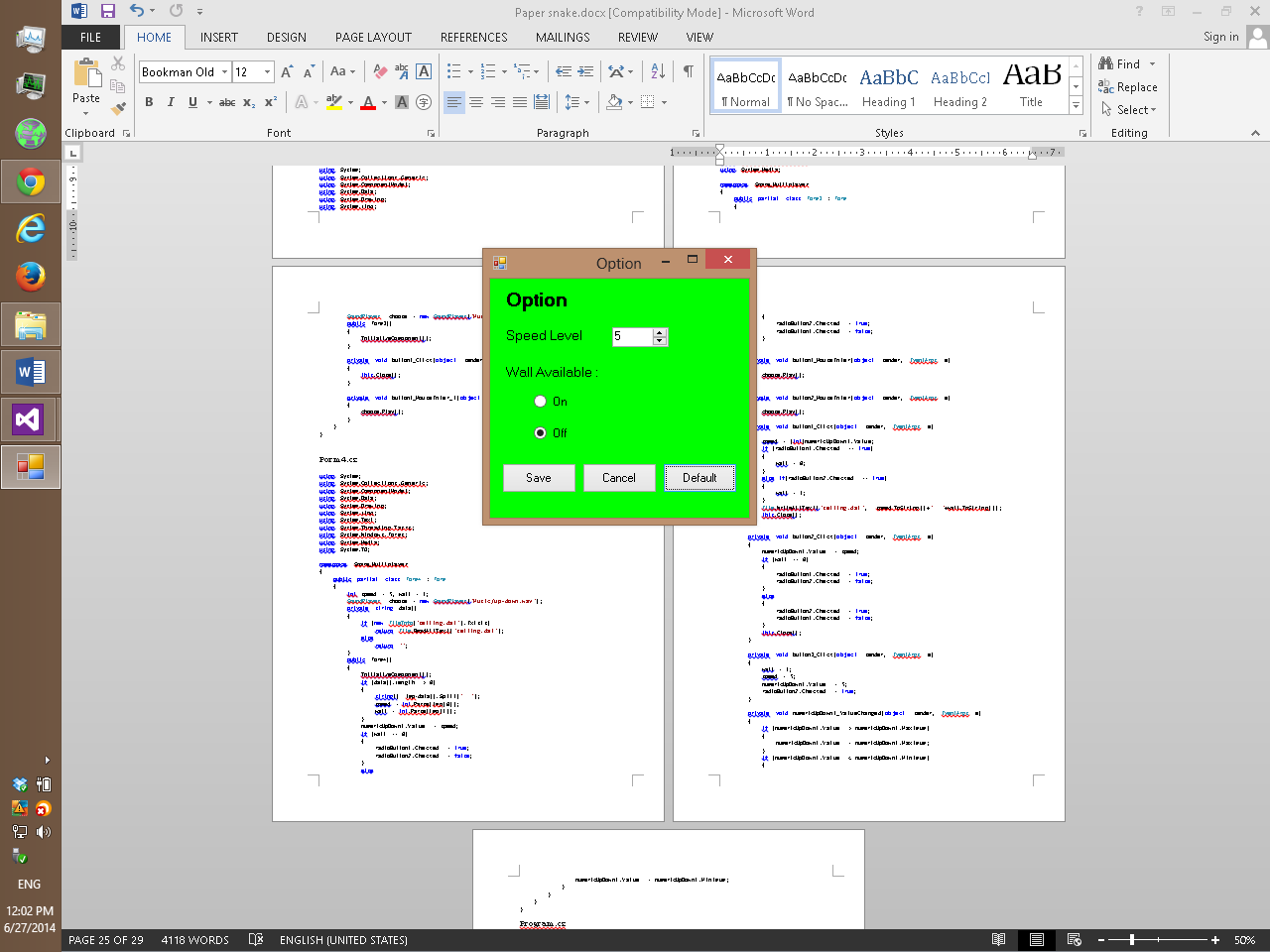
choose.Play();

}

}

}

Form4.cs



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Media;

using System.IO;

namespace Snake\_Multiplayer

{

public partial class Form4 : Form

{

int speed = 5, wall = 1;

SoundPlayer choose = new SoundPlayer("Music/up-down.wav");

private string data()

{

if (new FileInfo("setting.dat").Exists)

return File.ReadAllText("setting.dat");

else

return "";

}

public Form4()

{

InitializeComponent();

if (data().Length > 0)

{

string[] tmp=data().Split(' ');

speed = int.Parse(tmp[0]);

wall = int.Parse(tmp[1]);

}

numericUpDown1.Value = speed;

if (wall == 0)

{

radioButton1.Checked = true;

radioButton2.Checked = false;

}

else

{

radioButton2.Checked = true;

radioButton1.Checked = false;

}

}

private void button1\_MouseEnter(object sender, EventArgs e)

{

choose.Play();

}

private void button2\_MouseEnter(object sender, EventArgs e)

{

choose.Play();

}

private void button1\_Click(object sender, EventArgs e)

{

speed = (int)numericUpDown1.Value;

if (radioButton1.Checked == true)

{

wall = 0;

}

else if(radioButton2.Checked == true)

{

wall = 1;

}

File.WriteAllText("setting.dat", speed.ToString()+" "+wall.ToString());

this.Close();

}

private void button2\_Click(object sender, EventArgs e)

{

numericUpDown1.Value = speed;

if (wall == 0)

{

radioButton1.Checked = true;

radioButton2.Checked = false;

}

else

{

radioButton2.Checked = true;

radioButton1.Checked = false;

}

this.Close();

}

private void button3\_Click(object sender, EventArgs e)

{

wall = 1;

speed = 5;

numericUpDown1.Value = 5;

radioButton2.Checked = true;

}

private void numericUpDown1\_ValueChanged(object sender, EventArgs e)

{

if (numericUpDown1.Value > numericUpDown1.Maximum)

{

numericUpDown1.Value = numericUpDown1.Maximum;

}

if (numericUpDown1.Value < numericUpDown1.Minimum)

{

numericUpDown1.Value = numericUpDown1.Minimum;

}

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Snake\_Multiplayer

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

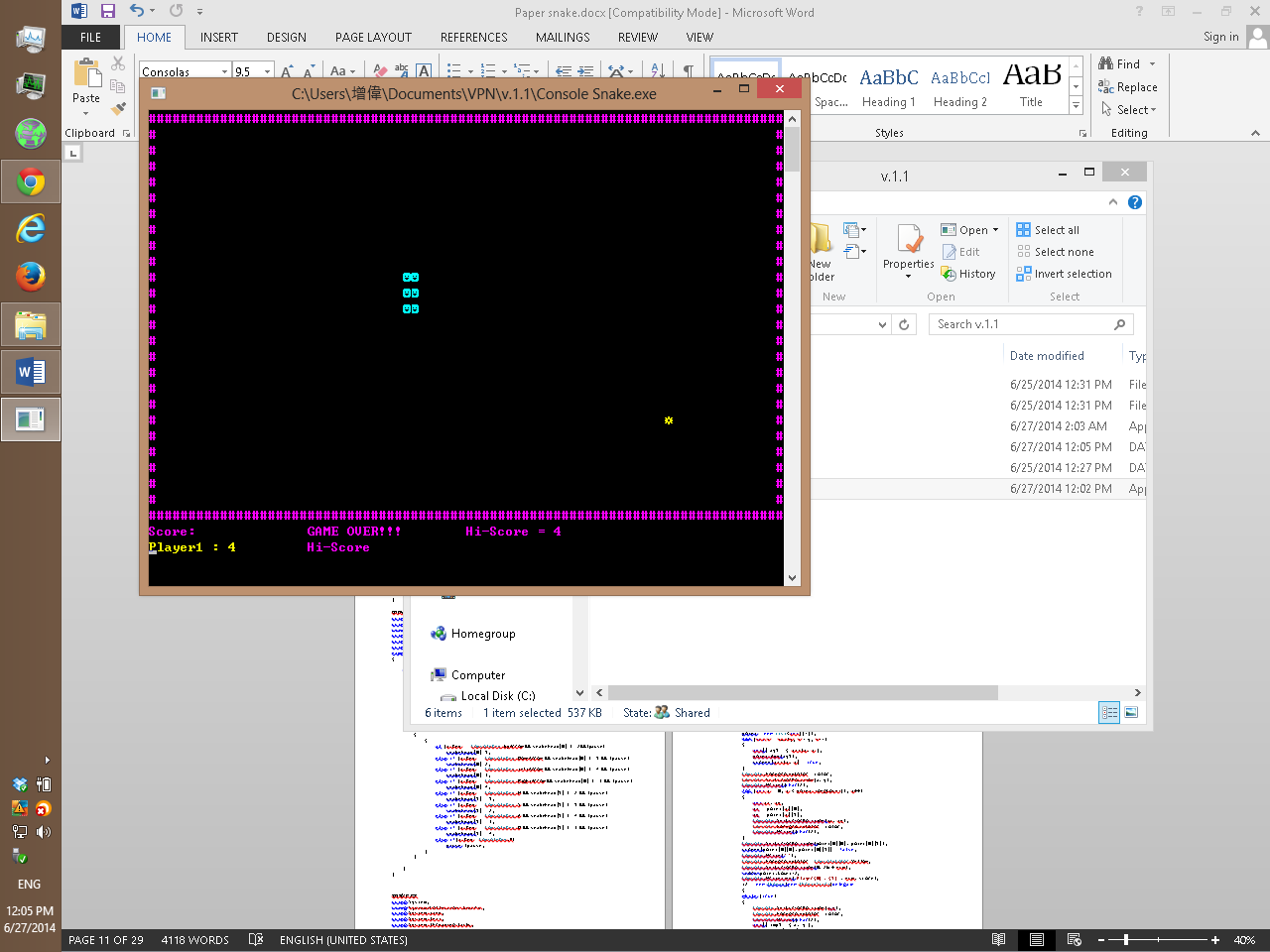
Application.Run(new Form1());

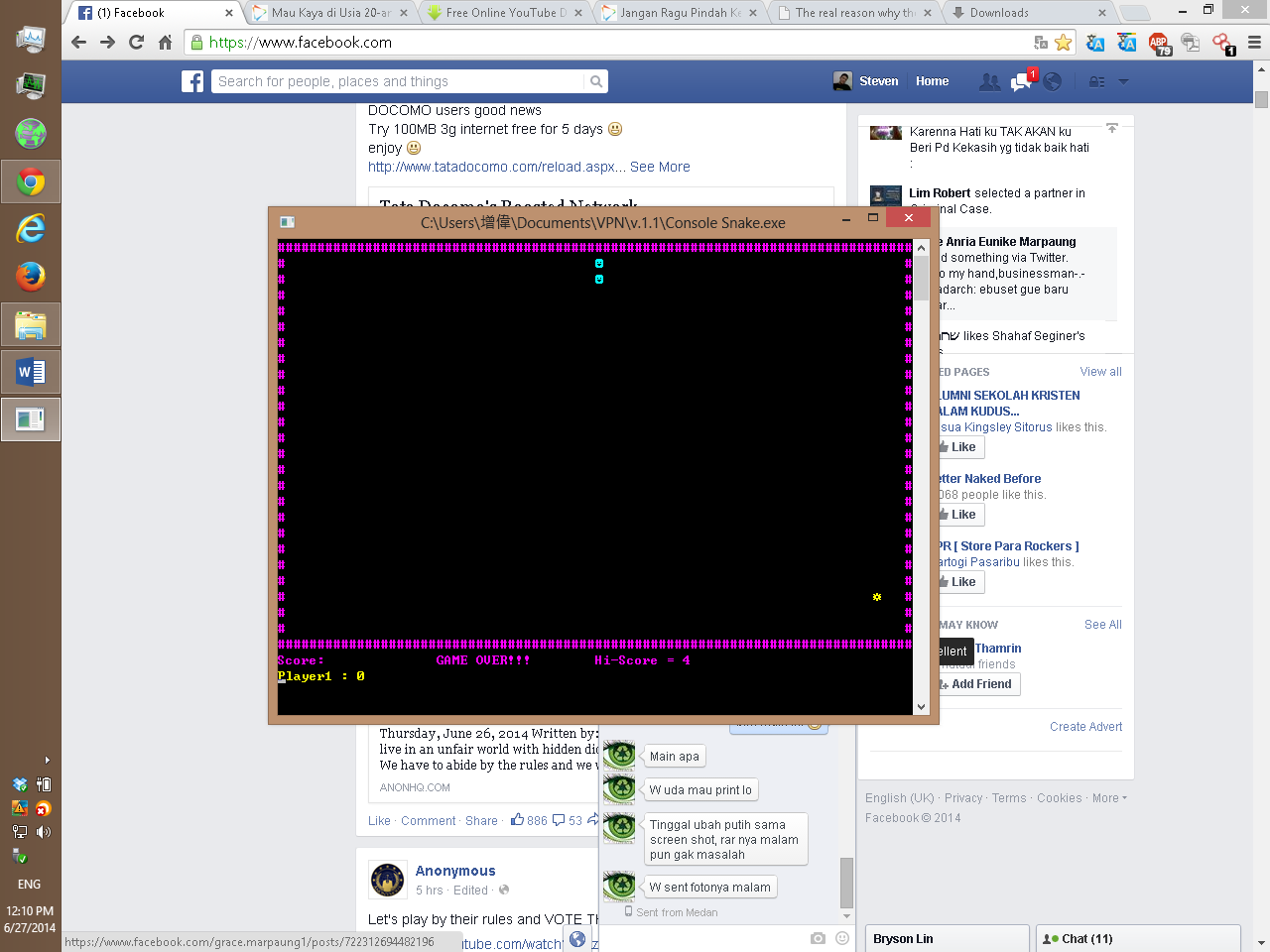
}

}

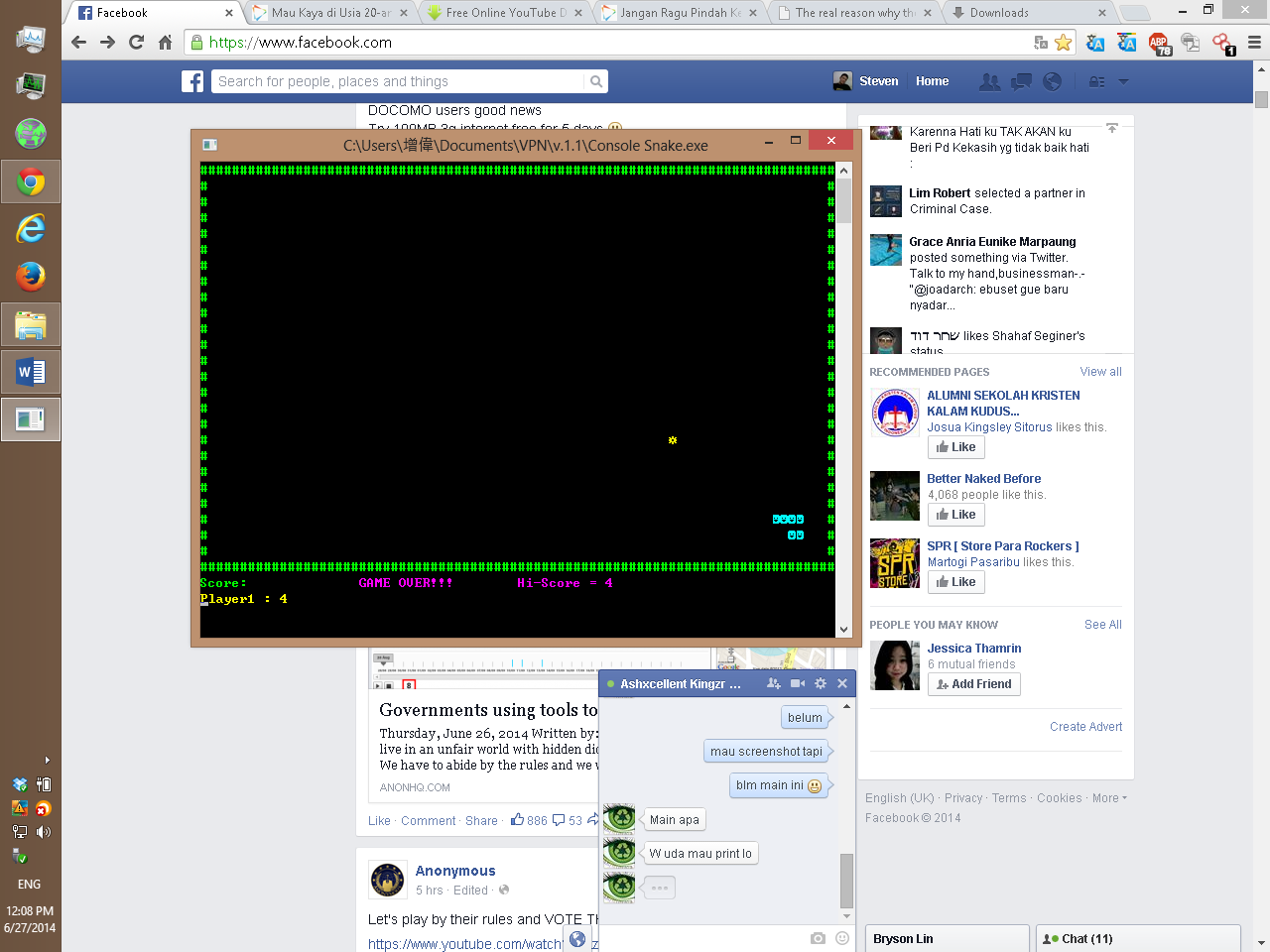
}

Snake Single Player with wall:

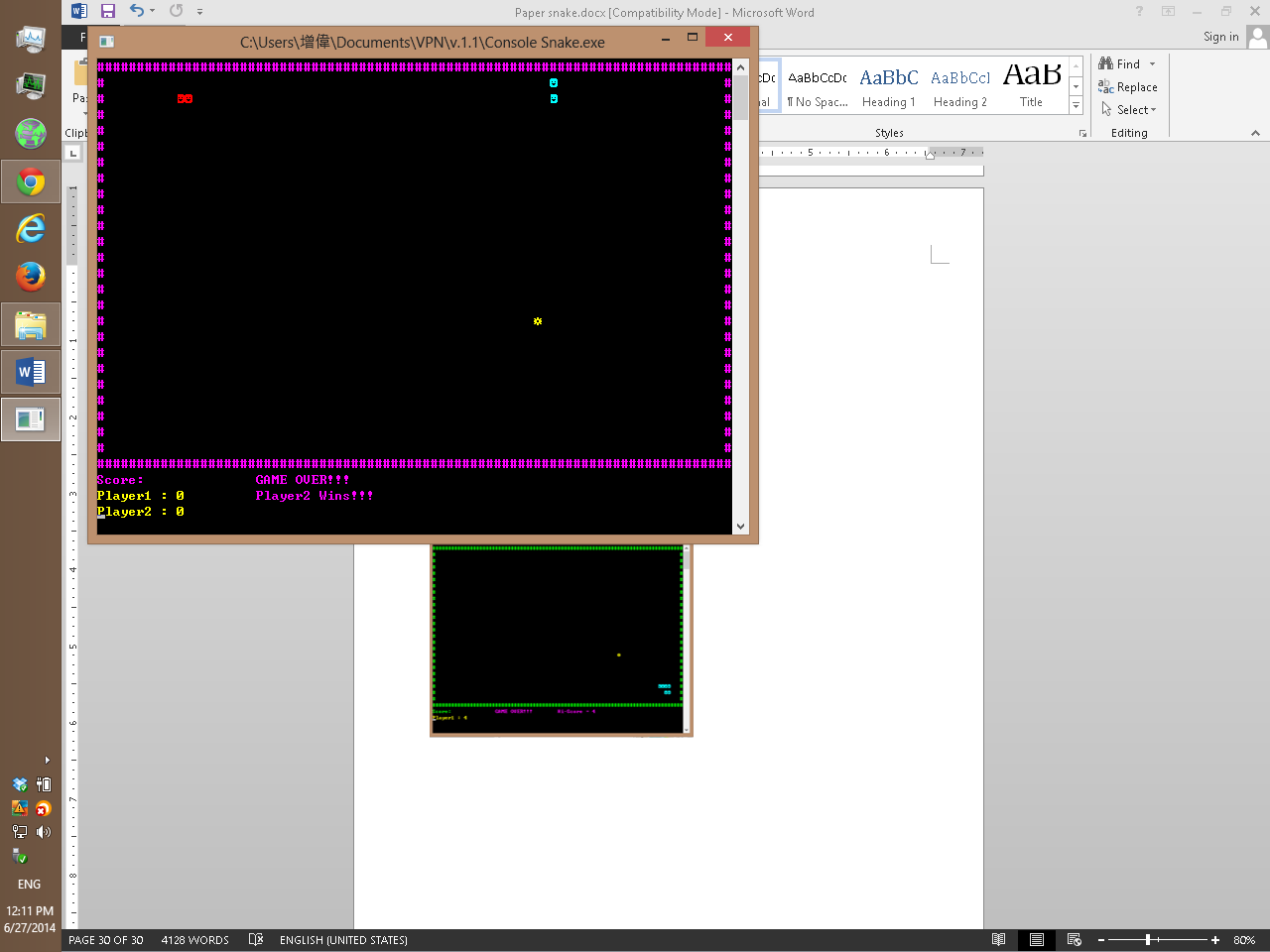




Snake Single Player without wall:



Snake DualPlayer with wall:

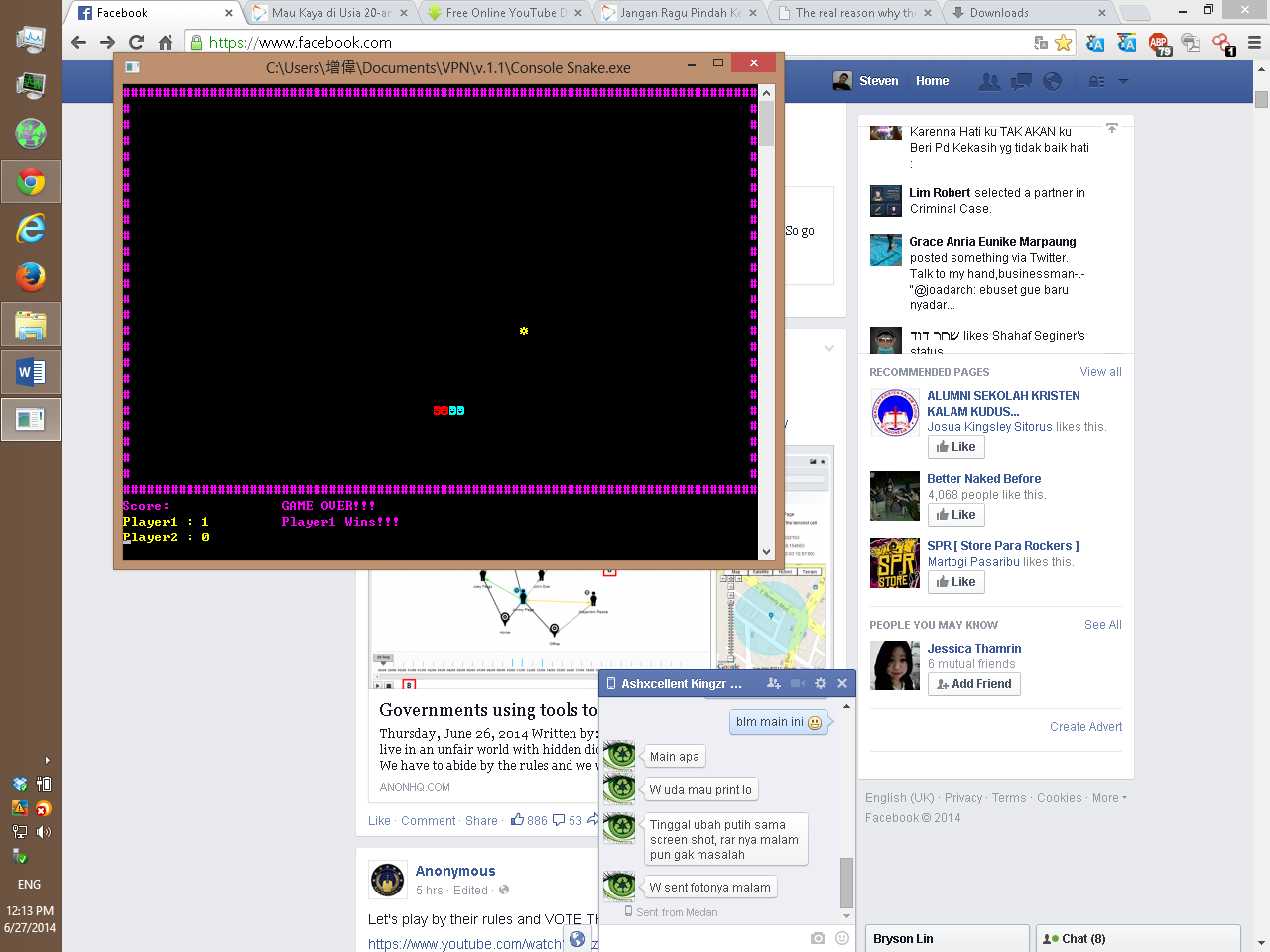


Snake DualPlayer head vs head:

If snake 1>snake2 snake1 win

If snake1<snake2 snake2 win

If snake1==snake2 draw



Snake DualPlayer without wall:

If(snake1 kills himself) snake2 win

If(snake2 kills himself) snake1 win

If(snake1 eats snake2 body or tail) snake2 win

If(snake2 eats snake1 body or tail) snake1 win

