Programming - CS1S446

Minecfield Game in C#.NET:

Developing a simple minefield game in C#.NET

Allan Vermeulen – <https://github.com/AllancVe>

Contents

[Flow Chart 2](#_Toc43644315)

[Description of development tools and Process 3](#_Toc43644316)

[**Code listing and description** 4](#_Toc43644317)

[Sources 9](#_Toc43644318)

# Flow Chart

Start-up

Play Music

Game loads

Loads Sprite

Mines placed

Exit Game

Play

Check Input

Exit

Down

Up

Right

Left

Restart

Move

Show mines

Sprite Collides with mine

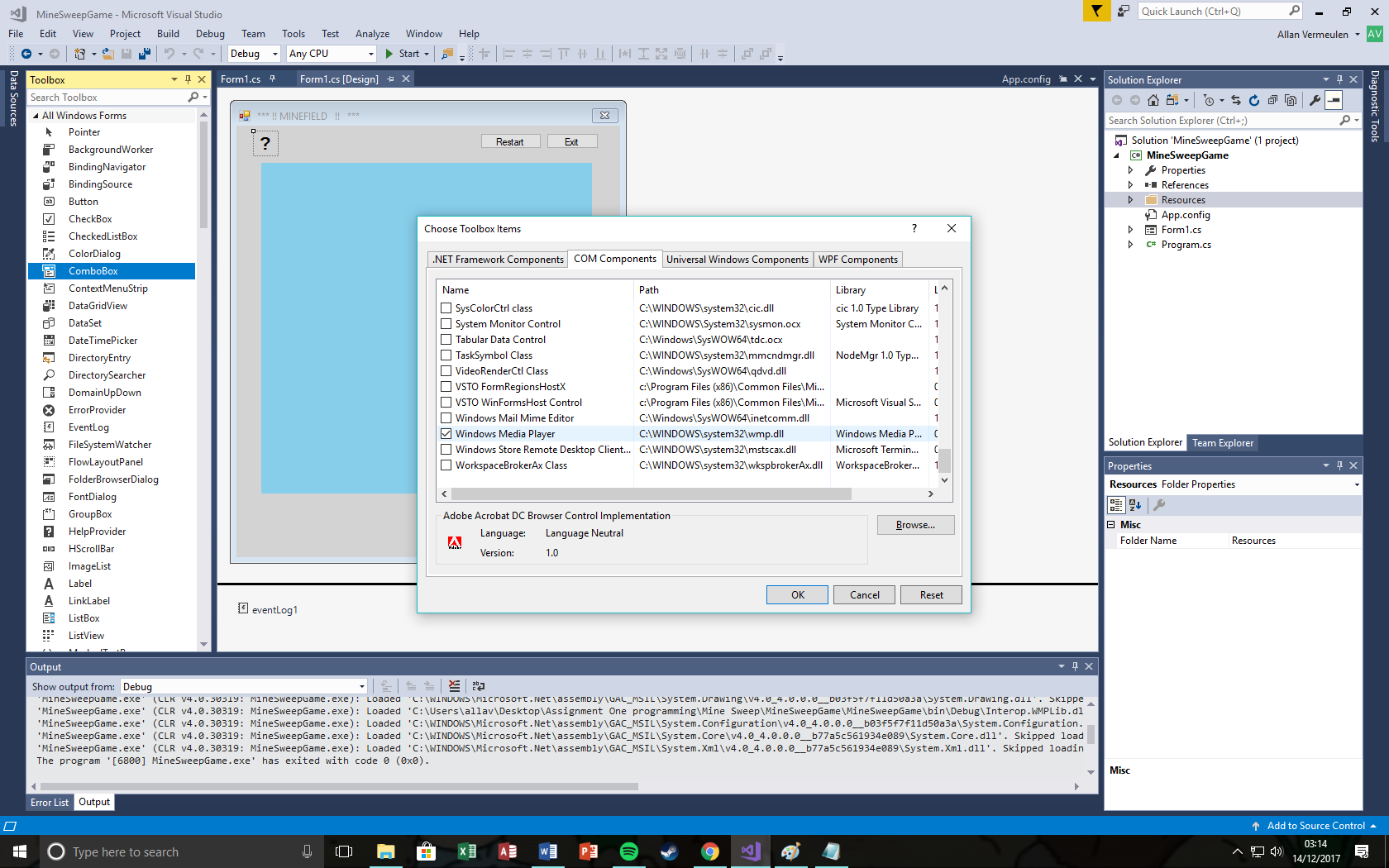
Collision?

Draw

Count Bombs

# Description of development tools and Process

**Requirements**: The requirements for this application was to follow a step by step guide that will allow me to create a minefield game. This game that I was to produce would have buttons for controlling the sprite, a bomb counter & when the sprite hits a mine the game will show the rest of the hidden mines the player couldn’t see before. Also adding colours to represent the player failing in their mission. The other requirement for extra marks was to add additional aspects to the game. I will talk about that for the designing stage.

**Designing**: For the designing stage I used visual studios - windows application form: Which is object orientated programming that is used to create games. I added the features that programme needed such as a label for counting bombs around the sprite (player), buttons that would be used for moving the sprite & also adding two buttons for exiting & restarting the game. For additional purposes I thought that music would be suitable for a game of this kind. So, I went online & found some free music for the background & placed it in the Debug file in the games Bin file. This will be useful for me, once I start programming.

**Implementation**: This part is where I started coding; adding code to the buttons & making then fully functional, most of this was already done due to the mini tasks that I had to do, however for the Exit & also the Refresh button I had to do my own coding which took some time as I had to look at all the different ways I could go about creating the right code for the programme. I also had to add “Windows Media Player” in order for music to play in the background of the game & once that was done. I left a screenshot to show how I added it.

**Verification**: For my verification I had to do some white-box testing during the implementation of my game. This allowed me to see if a piece of code worked properly. I also did Black-box testing which allowed me to see if there were any errors in the programme that I would have to fix without me having to look at the overall code. Fortunately, there were not many mistakes in this process other than some minor spelling errors.

Show Error the error -  
“No value in textbox!”

# **Code listing and description**

namespace MineSweepGame

{

public partial class Form1 : Form

{

int atX = 10;

int atY = 20;

//Allowing sound to be played

WindowsMediaPlayer player = new WindowsMediaPlayer();

public Form1()

{

InitializeComponent();

//place the sprite at its start-up location

drawsprite(atX, atY);

//place the sprite at its start-up location

drawsprite(atX, atY);

//place the mines (bombs)

placeBombs(80);

//Telling which music file should be played

player.URL = "Background\_music.wav";

}

//a Boolen Array that indicates where the mines are

bool[,] bombs = new bool[21, 21];

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

{

//for the music to be played when the form loads

player.controls.play();

}

//function to draw the sprite at location (x,y)

private void drawsprite(int x, int y)

{

Label lbl = getLabel(atX, atY);

lbl.BackColor = Color.White;

lbl.Image = Properties.Resources.w;

}

private Label getLabel(int x, int y)

{

int k = (y - 1) \* 20 + x;

string s = "label" + k.ToString();

foreach (Control c in panel1.Controls)

{

if (c.GetType() == typeof(System.Windows.Forms.Label))

{

if (c.Name == s)

{

return (Label)c;

}

}

}

return null;

}

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

{

{

//if allowed, update location

if (atX < 20)

{

//delete sprite at current location

wipesprite(atX, atY);

//move up by one row

atX++;

//draw sprite at current location

drawsprite(atX, atY);

}

}

}

private void Form1\_Load\_1(object sender, EventArgs e)

{

}

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

{

{

//if allowed, update location

if (atY > 1)

{

//delete sprite at current location

wipesprite(atX, atY);

//move up by one row

atY--;

//draw sprite at current location

drawsprite(atX, atY);

//check for bombs

chkbomb(atX, atY);

}

}

}

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

{

{

//if allowed, update location

if (atY < 20)

{

//delete sprite at current location

wipesprite(atX, atY);

//move up by one row

atY++;

//draw sprite at current location

drawsprite(atX, atY);

//check for bombs

chkbomb(atX, atY);

}

}

}

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

{

{

//if allowed, update location

if (atX > 1)

{

//delete sprite at current location

wipesprite(atX, atY);

//move up by one row

atX--;

//draw sprite at current location

drawsprite(atX, atY);

//check for bombs

chkbomb(atX, atY);

}

}

}

private void wipesprite(int x, int y)

{

Label lbl = getLabel(atX, atY);

lbl.Image = null;

}

//place those mines (bombs)

private void placeBombs(int target)

{

//create a random number generator

Random r = new Random();

//set up variables

int x;

int y;

int k = target;

//clear the current mines list

Array.Clear(bombs, 0, bombs.Length);

//loop to fill with desired number of mines

do

{

x = r.Next(1, 20);

y = r.Next(1, 20);

if (!bombs[x, y])

{

bombs[x, y] = true;

k--;

}

} while (k > 0);

}

// counting bombs

private void countBombs(int X, int Y)

{

int count = 0;

int newx;

int newy;

newx = X - 1;

if (newx > -1)

{

if (bombs[newx, Y])

count++;

}

newx = X + 1;

if (newx < 21)

{

if (bombs[newx, Y])

count++;

}

newy = Y - 1;

if (newy > -1)

{

if (bombs[X, newy])

count++;

}

newy = Y + 1;

if (newy < 21)

{

if (bombs[X, newy])

count++;

}

label404.Text = count.ToString();

}

//show the bombs

private void showBombs()

{

Label lbl;

for (int y = 1; y < 21; y++)

{

for (int x = 1; x < 21; x++)

{

lbl = getLabel(x, y);

if (bombs[x, y])

{

lbl.BackColor = Color.Yellow;

}

else

{

lbl.BackColor = Color.Black;

}

}

}

if (MessageBox.Show("Do you want to restart?", "Restart", MessageBoxButtons.YesNo) ==

DialogResult.Yes) Application.Restart();

}

//check for bomb at current location

private void chkbomb(int X, int Y)

{

if (bombs[X, Y])

{

this.BackColor = Color.Red;

//end of game

btnDown.Enabled = false;

btnUp.Enabled = false;

btnRight.Enabled = false;

btnLeft.Enabled = false;

showBombs();

}

else

{

//count bombs around current location

countBombs(X, Y);

}

}

// Code to allow the user to exit the application. If selected yes then application closes, else the forms stays open.

private void btn\_Restart\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Exit application?", "Exit?", MessageBoxButtons.YesNo) ==

DialogResult.Yes) Application.Exit();

}

// This code allows the user to decide if they would like to restart the game

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

{

if (MessageBox.Show("Do you want to restart?", "Restart", MessageBoxButtons.YesNo) ==

DialogResult.Yes) Application.Restart();

}

}

}

# Sources

Purple-planet (2017). *Dark Backgrounds – Purple Plant Royalty Free Music.* Available at: http://www.purple-planet.com/dark-backgrounds/4584537439 (Accessed 14 December 2017).

YouTube (2017). *How to add background music in Visual Studio*. Available at: http://www.purple-planet.com/dark-backgrounds/4584537439 (Accessed: 14 December 2017).