**Yu Feng**
**CS 4523**
**Orlando Karam**
**05/02/2011**

# AI Final Project Explanation

## Game Explanation:

In this game, basically it is a mini strategy game. Each turn player select one fruit and get it(means gets the point), then the PC will automatically choose one. During the selecting, player can see his or her path of moving to the destination using the search method. Player also able to see how the PC go to the destination. Therefore, my AI system is the searching. What's more, after player choose the fruit, he or she can get the point. At last, score determines who will win in the game.

## Core of AI search code:

It is very easy search code, which just determine which places player or PC can go. Therefore, at first, it will search up grid to see whether it can move. If it is 0, that means can go. If it is 1, that means cannot. After that, see down, left and right. As result, repeat these steps and move the character to the position which player's mouse determined.

```
// Core AI code (this Search is based on the Depth First Search):
private void Search(int row, int colum)
        {
            //has a check point to determine whether or not go that way.
            //check up,down,left,right to determine which is the still
not choose.

            if ((Form[row - 1, colum, 1] == 1) && (Form[row + 1, colum,
1] == 1) && (Form[row, colum - 1, 1] == 1) && (Form[row, colum + 1, 1]
== 1))
            {
                return;
            }
            //up
            if (Form[row - 1, colum, 1] == 0)
            {
                //if search to the form[1,1,1], then put it to 2
                if ((row - 1 == 1) && (colum == 1))
                {
                    Form[row - 1, colum, 1] = 2;
                }
                else
                {
                    Form[row - 1, colum, 1] = 1;
                }
                stack.Push(Form[row - 1, colum, 0]);

                //form[1,1,1] becomes to 2 then put a 20 as mark, it
                //means in stack after the 20 is the goal.
                if (Form[row - 1, colum, 1] == 2)
                {
                    stack.Push(20);
```

```
                }

                TempRowDepth = row - 1;
                TempColumDepth = colum;
            }
            else
            {
                //down
                if (Form[row + 1, colum, 1] == 0)
                {
                    if ((row + 1 == 1) && (colum == 1))
                    {
                        Form[row + 1, colum, 1] = 2;
                    }
                    else
                    {
                        Form[row + 1, colum, 1] = 1;
                    }
                    stack.Push(Form[row + 1, colum, 0]);

                    //form[1,1,1] becomes to 2 then put a 20 as mark,
it
                    //means in stack after the 20 is the goal.
                    if (Form[row + 1, colum, 1] == 2)
                    {
                        stack.Push(20);
                    }
                    TempRowDepth = row + 1;
                    TempColumDepth = colum;
                }
                else
                {
                    //left
                    if (Form[row, colum - 1, 1] == 0)
                    {
                        if ((row == 1) && (colum - 1 == 1))
                        {
                            Form[row, colum - 1, 1] = 2;
                        }
                        else
                        {
                            Form[row, colum - 1, 1] = 1;
                        }
                        stack.Push(Form[row, colum - 1, 0]);

                        //form[1,1,1] becomes to 2 then put a 20 as
mark, it
                        //means in stack after the 20 is the goal.
                        if (Form[row, colum - 1, 1] == 2)
                        {
                            stack.Push(20);
                        }
                        TempRowDepth = row;
                        TempColumDepth = colum - 1;
                    }
                    else
                    {
```

```
                        //right
                        if (Form[row, colum + 1, 1] == 0)
                        {
                            if ((row == 1) && (colum + 1 == 1))
                            {
                                Form[row, colum + 1, 1] = 2;
                            }
                            else
                            {
                                Form[row, colum + 1, 1] = 1;
                            }
                            stack.Push(Form[row, colum + 1, 0]);

                            //form[1,1,1] becomes to 2 then put a 20 as
mark, it
                            //means in stack after the 20 is the goal.
                            if (Form[row, colum + 1, 1] == 2)
                            {
                                stack.Push(20);
                            }
                            TempRowDepth = row;
                            TempColumDepth = colum + 1;
                        }
                    }
                }
            }
            Search(TempRowDepth, TempColumDepth);
        }
```

**The Whole Game Code: (Using XNA C#):**

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using Microsoft.Xna.Framework.Net;
using Microsoft.Xna.Framework.Storage;

namespace MiniStrategyGame
{
    public class Game1 : Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;
        MouseState newMouseState, oldMouseState;
        SpriteFont PointofPCandPlayer;
        Texture2D ItemApple, ItemOrange, ItemGrape, ItemBanana;
        Texture2D Background;
        Texture2D WinScene, LoseScene;
        Texture2D CharactorPC, CharactorPlayer;
```

```
Texture2D Path;
Vector2 PCVec, PlayerVec;
Random number;

string[,] JudgeDraw;

int[,,] Form;
int[] StoreStack;
int[] StoreStackPC;
int CountofStore;
int CountofStorePC;
double TimeSinceLastUpdate;

bool isPressButton;
bool isFinishPC;
bool isFinishInitial;
bool isPlayerFinish;
bool isPathDrawPlayer, isPathDrawPC;
bool isGameFinish;
bool isPlayerWin;
bool isPCWin;

static int PlayerScroe, PCScore;
static int[] tempRowPC;
static int[] tempColumPC;
static int RowPC;
static int ColumPC;
static int row, colum;
static int TempRowDepth, TempColumDepth;

static Stack<int> stack;

public Game1()
{
    isPCWin = false;
    isPlayerWin = false;
    isPlayerFinish = false;
    isFinishInitial = false;
    isPathDrawPC = false;
    isPathDrawPlayer = false;
    isPressButton = false;
    isGameFinish = false;

    Form = new int[5, 5, 2];
    tempRowPC = new int[3];
    tempColumPC = new int[3];
    StoreStack = new int[12];
    StoreStackPC = new int[12];

    JudgeDraw = new string[5, 5];
    stack = new Stack<int>();
    number = new Random();
    PCVec = new Vector2(201, 150);
    PlayerVec = new Vector2(201, 111);

    PlayerScroe = 0;
    CountofStore = 0;
```

```
            CountofStorePC = 0;

            graphics = new GraphicsDeviceManager(this);
            Content.RootDirectory = "Content";
        }

        protected override void Initialize()
        {
            // TODO: Add your initialization logic here
            this.IsMouseVisible = true;

            base.Initialize();
        }

        protected override void LoadContent()
        {
            // Create a new SpriteBatch, which can be used to draw
textures.
            spriteBatch = new SpriteBatch(GraphicsDevice);

            //sprite font
            PointofPCandPlayer = Content.Load<SpriteFont>("Point");

            //Initialize two charactors
            CharactorPlayer =
Content.Load<Texture2D>("Materials\\PlayerArrow");
            CharactorPC  =  Content.Load<Texture2D>("Materials\\PCArrow");

            //Initialize a background and friut pictures
            Background =
Content.Load<Texture2D>("Materials\\MainScene");
            ItemApple = Content.Load<Texture2D>("Materials\\Apple");
            ItemBanana = Content.Load<Texture2D>("Materials\\Banana");
            ItemGrape = Content.Load<Texture2D>("Materials\\Grape");
            ItemOrange = Content.Load<Texture2D>("Materials\\Orange");
            //initialize the win or lose scene
            WinScene = Content.Load<Texture2D>("Materials\\Win");
            LoseScene = Content.Load<Texture2D>("Materials\\Lose");
            Path = Content.Load<Texture2D>("Materials\\Path");

            //bulid a wall, sign a 1,when judge the 1 that means cannot
move
            //to there.
            for (int i = 0; i < 5; i++)
                for (int j = 0; j < 5; j++)
                {
                    if (i < 1 || i > 3)
                    {
                        Form[i, j, 1] = 1;
                    }
                    else
                    {
                        Form[i, 0, 1] = 1;
                        Form[i, 4, 1] = 1;
                    }
                }
```

```
            //excpet the wall all the place is 0, unsign
            //that means PC can move to there
            for (int i = 1; i < 4; i++)
            {
                for (int j = 1; j < 4; j++)
                {
                    Form[i, j,0] = number.Next(-1, 3);
                    Form[i, j, 1] = 0;

                    //juge which one is which one, and draw in correct
place
                    if (Form[i, j,0] == -1)
                    {
                        JudgeDraw[i, j] = "Orange";
                    }
                    if (Form[i, j,0] == 1)
                    {
                        JudgeDraw[i, j] = "Apple";
                    }
                    if (Form[i, j,0] == 2)
                    {
                        JudgeDraw[i, j] = "Banana";
                    }
                    if (Form[i, j, 0] == 0)
                    {
                        Form[i, j, 0] = 3;
                        JudgeDraw[i, j] = "Grape";
                    }

                }
            }
            isFinishInitial = true;
            // TODO: use this.Content to load your game content here
        }

        protected override void UnloadContent()
        {
            // TODO: Unload any non ContentManager content here
        }

        protected override void Update(GameTime gameTime)
        {
            TimeSinceLastUpdate +=
gameTime.ElapsedGameTime.TotalSeconds;

            // Allows the game to exit
            if (Mouse.GetState().LeftButton == ButtonState.Pressed)
            {
                if((Mouse.GetState().X > 747)&&(Mouse.GetState().X <
788))
                    if ((Mouse.GetState().Y > 6) && (Mouse.GetState().Y
< 46))
                    {
                        this.Exit();
                    }
            }
```

```
            newMouseState = Mouse.GetState();
            Console.WriteLine(oldMouseState.X);
            Console.WriteLine(oldMouseState.Y);

            //*****************************************************
            // Player first go.
            //*****************************************************
            if (!isGameFinish)
            {
                if (Mouse.GetState().LeftButton == ButtonState.Pressed
&& newMouseState != oldMouseState)
                {
                    isPathDrawPC = false;
                    isPathDrawPlayer = true;

                    //reset to the 0
                    int RemoveStoreStack = 0;
                    CountofStore = 0;

                    while (StoreStack[RemoveStoreStack] != 0)
                    {
                        StoreStack[RemoveStoreStack] = 0;
                        RemoveStoreStack++;
                    }

                    //using for to decide where is row and column.
                    for (int j = 1; j < 4; j++)
                    {
                        for (int i = 1; i < 4; i++)
                        {
                            if ((Mouse.GetState().X > 341 + 130 * (i -
1)) && (Mouse.GetState().X < 341 + 130 * (i - 1) + 60))
                                if ((Mouse.GetState().Y > 130 + 130 *
(j - 1)) && (Mouse.GetState().Y < 130 + 130 * (j - 1) + 60))
                                {

                                    //push the number you select
                                    stack.Push(Form[j, i, 0]);
                                    Form[j, i, 1] = 1;
                                    JudgeDraw[j, i] = " 1 ";
                                    PlayerScroe += Form[j, i, 0];


//*****************************************************************
//using depth first search to search the way which charactor need to go
//AI - using search to decide the path of Player moving.
//*****************************************************************
                                    DepthFirstSearch(j, i);

                                    //judge whether push it, if push it
then move the charactor
                                    isPressButton = true;

                                    //all the sign switch to 0 except
player selected one.
                                    for (int k = 1; k < 4; k++)
                                        for (int h = 1; h < 4; h++)
```

```
                                    {
                                        Form[k, h, 1] = 0;
                                    }

                                //tell the character specific
position
                                row = j;
                                colum = i;

                                //store the stack into the string
                                int temp = 0;

                                while (temp != 20)
                                {
                                    temp = stack.Pop();
                                    if ((stack.Count == 0) &&
(temp != 20))
                                    {
                                        //move to the first colum
                                        PlayerVec.X = 340;
                                        PlayerVec.Y = 111;
                                        TimeSinceLastUpdate = 0.0;
                                        JudgeDraw[1, 1] = " 1 ";

                                        isPressButton = false;
                                        isPlayerFinish = true;
                                        isPathDrawPlayer = false;

                                        break;
                                    }
                                }

                                while (stack.Count != 0)
                                {
                                    StoreStack[CountofStore] =
stack.Pop();
                                    CountofStore++;
                                }

                            }
                        }
                    }


                }
                //move the character, except to first colum
                if (isPressButton)
                {
                    PlayerVec.X = 340;
                    PlayerVec.Y = 130;

                    PlayerVec.X += 130 * (colum - 1);
                    PlayerVec.Y += 130 * (row - 1);


                    isPressButton = false;
                    isPlayerFinish = true;
```

```
                    TimeSinceLastUpdate = 0.0;
                }


//*****************************************************
                // PC then go.

//*****************************************************
                if (isPlayerFinish)
                {
                    if (TimeSinceLastUpdate > 1)
                    {
                        isPathDrawPC = true;
                        //set to 0
                        int removeCountofStorePC = 0;
                        CountofStorePC = 0;
                        while (StoreStackPC[removeCountofStorePC] != 0)
                        {
                            StoreStackPC[removeCountofStorePC] = 0;
                            removeCountofStorePC++;
                        }

                        Random rowPC = new Random();
                        Random columPC = new Random();

                        RowPC = rowPC.Next(1, 4);
                        ColumPC = columPC.Next(1, 4);

                        for (int i = 1; i < 4; i++)
                        {
                            for (int j = 1; j < 4; j++)
                            {
                                if ((JudgeDraw[i, j] == " 1 ") &&
(JudgeDraw[i, j] == JudgeDraw[RowPC, ColumPC]))
                                {
                                    rowPC = new Random();
                                    columPC = new Random();
                                    RowPC = rowPC.Next(1, 4);
                                    ColumPC = columPC.Next(1, 4);

                                    i = 1;
                                    j = 0;
                                }
                            }
                        }

                        stack.Push(Form[RowPC, ColumPC, 0]);
                        Form[RowPC, ColumPC, 1] = 1;
                        JudgeDraw[RowPC, ColumPC] = " 1 ";
                        PCScore += Form[RowPC, ColumPC, 0];


//*****************************************************
//AI-Using search to determing PC where to go

//*****************************************************
```

```
                        DepthFirstSearch(RowPC, ColumPC);

                        //all the sign switch to 0 except player
selected one.
                        for (int k = 1; k < 4; k++)
                            for (int h = 1; h < 4; h++)
                            {
                                Form[k, h, 1] = 0;
                            }

                        //store the stack into the string
                        int temp1 = 0;

                        while (temp1 != 20)
                        {
                            temp1 = stack.Pop();
                            if ((stack.Count == 0) && (temp1 != 20))
                            {
                                PCVec.X = 340;
                                PCVec.Y = 150;

                                isPathDrawPlayer = false;
                                isPathDrawPC = false;
                                isFinishPC = false;
                                TimeSinceLastUpdate = 0.0;
                                break;
                            }
                        }

                        while (stack.Count != 0)
                        {
                            StoreStackPC[CountofStorePC] = stack.Pop();
                            CountofStorePC++;
                            isFinishPC = true;
                        }

                        isPlayerFinish = false;

                    }
                }


                //move the PC to the position
                if (isFinishPC)
                {
                    PCVec.X = 340;
                    PCVec.Y = 130;

                    PCVec.X += 130 * (ColumPC - 1);
                    PCVec.Y += 130 * (RowPC - 1);
                    isPathDrawPC = true;
                    isPathDrawPlayer = false;
                    isFinishPC = false;

                    TimeSinceLastUpdate = 0.0;
                }
```

```
                //Juge Whether game end
                int count = 0;

                for (int a = 1; a < 4; a++)
                    for (int b = 1; b < 4; b++)
                    {
                        if (JudgeDraw[a, b] == " 1 ")
                        {
                            count++;
                        }
                    }
                if (count == 9)
                {
                    isGameFinish = true;
                }
            }
            //if game finish is true, then decide which one win
             if (isGameFinish)
             {
                 if (PlayerScroe > PCScore)
                 {
                     isPlayerWin = true;
                 }
                 else
                 {
                     isPCWin = true;

                 }
             }
                     // TODO: Add your update logic here
                     oldMouseState = newMouseState;

            base.Update(gameTime);
        }

        //******************************************************
        //AI - Search
        //******************************************************
        private void DepthFirstSearch(int row, int colum)
        {
            //has a check point to determine whether or not go that way.
            //check up,down,left,right to determine which is the still
not choose.

            if ((Form[row - 1, colum, 1] == 1) && (Form[row + 1, colum,
1] == 1) && (Form[row, colum - 1, 1] == 1) && (Form[row, colum + 1, 1]
== 1))
            {
                return;
            }
            //up
            if (Form[row - 1, colum, 1] == 0)
            {
                //if search to the form[1,1,1], then put it to 2
                if ((row - 1 == 1) && (colum == 1))
                {
                    Form[row - 1, colum, 1] = 2;
```

```
                }
                else
                {
                    Form[row - 1, colum, 1] = 1;
                }
                stack.Push(Form[row - 1, colum, 0]);

                //form[1,1,1] becomes to 2 then put a 20 as mark, it
                //means in stack after the 20 is the goal.
                if (Form[row - 1, colum, 1] == 2)
                {
                    stack.Push(20);
                }

                TempRowDepth = row - 1;
                TempColumDepth = colum;
            }
            else
            {
                //down
                if (Form[row + 1, colum, 1] == 0)
                {
                    if ((row + 1 == 1) && (colum == 1))
                    {
                        Form[row + 1, colum, 1] = 2;
                    }
                    else
                    {
                        Form[row + 1, colum, 1] = 1;
                    }
                    stack.Push(Form[row + 1, colum, 0]);

                    //form[1,1,1] becomes to 2 then put a 20 as mark,
it
                    //means in stack after the 20 is the goal.
                    if (Form[row + 1, colum, 1] == 2)
                    {
                        stack.Push(20);
                    }
                    TempRowDepth = row + 1;
                    TempColumDepth = colum;
                }
                else
                {
                    //left
                    if (Form[row, colum - 1, 1] == 0)
                    {
                        if ((row == 1) && (colum - 1 == 1))
                        {
                            Form[row, colum - 1, 1] = 2;
                        }
                        else
                        {
                            Form[row, colum - 1, 1] = 1;
                        }
                        stack.Push(Form[row, colum - 1, 0]);
```

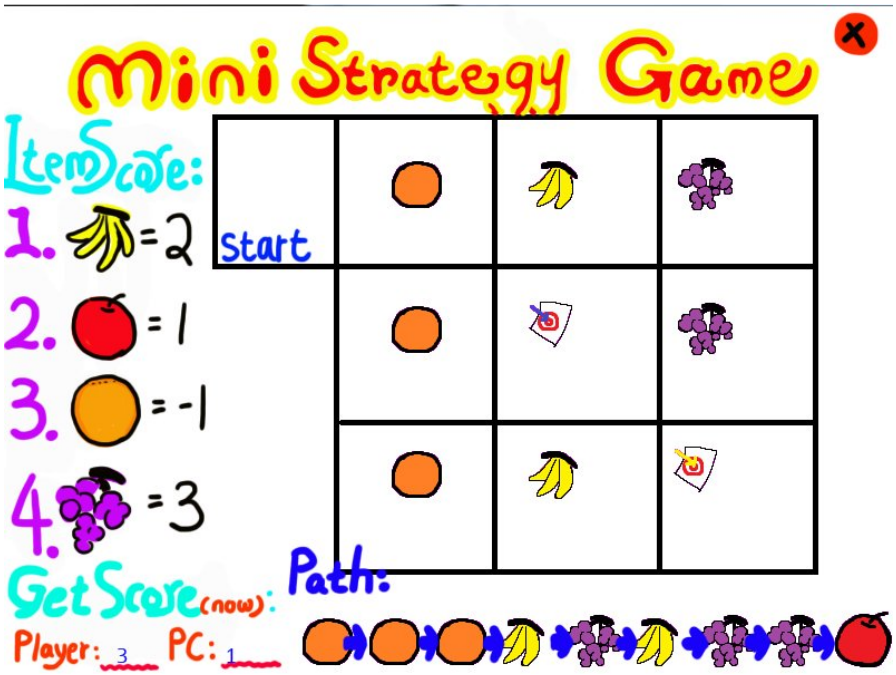
```
                        //form[1,1,1] becomes to 2 then put a 20 as
mark, it
                        //means in stack after the 20 is the goal.
                        if (Form[row, colum - 1, 1] == 2)
                        {
                            stack.Push(20);
                        }
                        TempRowDepth = row;
                        TempColumDepth = colum - 1;
                    }
                    else
                    {
                        //right
                        if (Form[row, colum + 1, 1] == 0)
                        {
                            if ((row == 1) && (colum + 1 == 1))
                            {
                                Form[row, colum + 1, 1] = 2;
                            }
                            else
                            {
                                Form[row, colum + 1, 1] = 1;
                            }
                            stack.Push(Form[row, colum + 1, 0]);

                            //form[1,1,1] becomes to 2 then put a 20 as
mark, it
                            //means in stack after the 20 is the goal.
                            if (Form[row, colum + 1, 1] == 2)
                            {
                                stack.Push(20);
                            }
                            TempRowDepth = row;
                            TempColumDepth = colum + 1;
                        }
                    }
                }
            }
            DepthFirstSearch(TempRowDepth, TempColumDepth);
        }


        protected override void Draw(GameTime gameTime)
        {
            GraphicsDevice.Clear(Color.CornflowerBlue);

            // TODO: Add your drawing code here
            spriteBatch.Begin();
            spriteBatch.Draw(Background,new Vector2(0,0),Color.White);

            //draw the two charactors
            spriteBatch.Draw(CharactorPlayer,PlayerVec,Color.White);
            spriteBatch.Draw(CharactorPC,PCVec, Color.White);

            //begin to draw the fruit pictures
            if (isFinishInitial)
            {
```

```
                for (int i = 1; i < 4; i++)
                    for (int j = 1; j < 4; j++)
                    {
                        if (JudgeDraw[j, i] == "Apple")
                            spriteBatch.Draw(ItemApple, new Vector2(341
+ 130 * (i - 1), 130 + 130 * (j - 1)), Color.White);
                        if (JudgeDraw[j, i] == "Banana")
                            spriteBatch.Draw(ItemBanana, new
Vector2(341 + 130 * (i - 1), 130 + 130 * (j - 1)), Color.White);
                        if (JudgeDraw[j, i] == "Orange")
                            spriteBatch.Draw(ItemOrange, new
Vector2(341 + 130 * (i - 1), 130 + 130 * (j - 1)), Color.White);
                        if (JudgeDraw[j, i] == "Grape")
                            spriteBatch.Draw(ItemGrape, new Vector2(341
+ 130 * (i - 1), 130 + 130 * (j - 1)), Color.White);
                    }
            }

            //draw the path of the player
            if (isPathDrawPlayer)
            {
                int l = 0;

                for (int a = 0; a < CountofStore; a++)
                {
                    if (StoreStack[a] == -1)
                    {
                        spriteBatch.Draw(ItemOrange, new Vector2(261 +
l, 540), Color.White);
                        l += 60;
                    }
                    if (StoreStack[a] == 3)
                    {
                        spriteBatch.Draw(ItemGrape, new Vector2(261 + l,
540), Color.White);
                        l += 60;
                    }
                    if (StoreStack[a] == 1)
                    {
                        spriteBatch.Draw(ItemApple, new Vector2(261 + l,
540), Color.White);
                        l += 60;
                    }
                    if (StoreStack[a] == 2)
                    {
                        spriteBatch.Draw(ItemBanana, new Vector2(261 +
l, 540), Color.White);
                        l += 60;
                    }


                }
                //draw the path
                spriteBatch.Draw(Path, new Vector2(0, 0), Color.White);

            }
            //draw the path of the PC
```

```
            if (isPathDrawPC)
            {
                int l = 0;
                //StoreStack[CountofStore]
                for (int a = 0; a < CountofStorePC; a++)
                {
                    if (StoreStackPC[a] == -1)
                    {
                        spriteBatch.Draw(ItemOrange, new Vector2(261 +
l, 540), Color.White);
                        l += 60;
                    }
                    if (StoreStackPC[a] == 3)
                    {
                        spriteBatch.Draw(ItemGrape, new Vector2(261 + l,
540), Color.White);
                        l += 60;
                    }
                    if (StoreStackPC[a] == 1)
                    {
                        spriteBatch.Draw(ItemApple, new Vector2(261 + l,
540), Color.White);
                        l += 60;
                    }
                    if (StoreStackPC[a] == 2)
                    {
                        spriteBatch.Draw(ItemBanana, new Vector2(261 +
l, 540), Color.White);
                        l += 60;
                    }
                }
                //draw the path
                spriteBatch.Draw(Path, new Vector2(0, 0), Color.White);

            }

            //draw player and pc's point
            spriteBatch.DrawString(PointofPCandPlayer,
PlayerScroe.ToString(), new Vector2(102, 570), Color.Blue);
            spriteBatch.DrawString(PointofPCandPlayer,
PCScore.ToString(), new Vector2(201, 570), Color.Blue);

            //if player win draw it, otherwise pc
            if (isPlayerWin)
            {
                spriteBatch.Draw(WinScene,new  Vector2(0,0),Color.White);
            }

            if (isPCWin)
            {
                spriteBatch.Draw(LoseScene, new Vector2(0, 0),
Color.White);
            }

            spriteBatch.End();

            base.Draw(gameTime);
```

```
        }
    }
}
```

**Screen Shot:**




This is the whole screen, and in the bottom is the path which using search way to achieve it.

**Conclusion:**

All in all, I think in this project, I learned how to use search in a game, and I also feel that AI in game is not a simple thing. Only this easy programming cost me many times. Therefore, I think in this semester, I learned a lot. I know the AI world is huge, and there are still lots of things that I did not know, and need to continue learning. I think this is a great class that gives me so many information about the AI. I will keep learning it in the future, and find more interesting things in the AI world.