

Assessment Feedback Sheet

|  |  |
| --- | --- |
| **Overall Assessment Grade Achieved** |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Learner Name** | |  | | | |
| **Assessor Name** | | C. Livesey | | | |
| **Qualification Title** | | Pearson BTEC Level 3 National Extended Diploma in Computing | | | |
| **Unit/Module No./Title** | | Unit 23: System Methodology | | | |
| **Assignment No./Title** | | **23.2 Apply and Review the use of Systems Methodology** | | | |
| **Learning Aim(s)** | | Learning aim B: Apply systems methodology tools and techniques to identify and solve a computing problem  Learning aim C: Review a solution to a computing problem | | | |
| **Issue Date** |  | **Planned Submission Date** |  | **Actual Submission Date** |  |

|  |  |  |
| --- | --- | --- |
| First/Second Submission | | |
| Criteria | Criteria Achieved | Assessor’s Feedback *Your feedback should include:*   * *What the learner has done well. (Knowledge, skills, etc.)* * *What the learner has not achieved and what was missing.* * *Information or guidance available to the learner they could have drawn on (e.g. class notes; handouts; resources in assignment brief etc.)* |
| P3  Define a problem statement for an identified scenario. |  |  |
| P4  Develop a solution to a computing problem, using appropriate systems methodologies. |  |  |
| P5  Review the solution to a computing problem, considering feedback from others and identifying possible improvements. |  |  |
| M2  Justify the choice of systems methodology tools and techniques used to provide a solution to a computing problem. |  |  |
| M3  Assess potential improvements to the solution in relation to the problem statement. |  |  |
| D2  Evaluate the process of applying a systems methodology to a computing problem in order to develop a proposed solution. |  |  |
| D3  Demonstrate individual responsibility and effective self-management in the application and review of systems methodologies to identify and develop a solution to a problem. |  |  |
| **BTEC Rules**  All resubmissions must be authorised by the **Lead Internal Verifier**. Only **one** resubmission is possible per assignment, providing:   * The learner has met initial deadlines set in the assignment, or has met an agreed deadline extension. * The tutor considers that the learner will be able to provide improved evidence without further guidance. * Evidence submitted for assessment has been authenticated and accompanied by a signed and dated declaration of authenticity by the learner.   Any resubmission evidence **must** be submitted within 10 working days of receipt of results of assessment (BTEC only) | | |
| **Wider Skills (Linked to Positive Futures)** *Comment on the quality of the learner work, the learner’s process and practice during assessment, research skills, presentation, general behaviour and conduct, meeting deadlines, etc.* | | |
|  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Assessor Declaration | *I certify that, to the best of my knowledge, the evidence submitted for this assignment/assessment is the learner’s own. I understand that false declaration is a form of malpractice.* | | |
| Assessor Signature: |  | Date: |  |
| Learner Declaration | *I certify that the evidence submitted for this assignment/assessment is my own. I have clearly referenced any sources used in the work. I understand that false declaration is a form of malpractice.* | | |
| Learner Signature: |  | Date: |  |

**Learner Actions**

|  |  |
| --- | --- |
| Learner’s Targets/Actions  *What needs to be developed to improve future work?* | *Actions linked to the following Positive Futures outcomes* |
|  | Confidence  Commitment  Collaboration  Resilience |

**Apply and Review the use of Systems Methodology**

# Task1: Definition

## Analysis and Design

Introduction:

The purpose of this project is to recreate a retro game in Visual studio with a working GUI. The game I will choose to represent is going to be noughts and crosses.

Initial Problem Statement:

The game is analogue and needs to be virtual, this will be changed in this project by recreating a game in windows forms and C#. The problem was observed by playing the game and wanting it to be virtual. The system also needs to be electronic.

I need more detail and information on the task to produce an appropriate problem statement.

Chosen Methodology, tools and techniques:

My chosen methodology is RAD, it is suitable for this purpose as it allows for pauses mid project and allows for more functionality to be added on the subsequent iterations that it will go through in its development.

It is suitable because it allows for user feedback on the program and builds off the feedback to improve itself. It is also meant for quick development which is useful since it can be completed in a short amount of time.

Task List:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Task | Tools and techniques | Justification |
| 1 | Plan the project | Create a list of tasks in Microsoft Planner. | Planner is used to make a clear checklist of thing to do and to have a timeframe for them. It helps you to know what to do and not miss any deadlines. |
| 2 | Analysis: Research | Secondary Research on the internet and a feasibility study | Research on the internet. So you have information to implement into your code. Secondary research will allow the brief to be more specific. |
| 3 | Analysis: Requirements | Gather requirements that are needed | You need to know the rules and things needed to make the program, this is the step to create guidelines to go by to know what the client wants the program to do. |
| 4 | Design the programme | Flowchart, Pseudocode, appearance (GUI),  Structure charts, Screen designs, visio | You need guidelines to go by. For this design stage a flowchart, made in visio, can be used to determine the steps to be taken towards creating the program and as a basis for the code to be formed. You casn also create a pseudocode. You can make some sketches of what you would like the GUI to look like before finalizing it in visio. |
| 5  6  7 | Develop prototype 1  Develop prototype 2  Develop final version | Visual Studio  C#  Windows forms | This language allows for event driven programming, for the code and GUI. This development step can be done in visual studio, by using windows forms to create the GUI and using C# to code the game. |
| 8 | Test | People (user testing)  System testing  Blackbox testing  Whitebox testing  System testing  User testing | You need to do this to make sure there are no errors or code breaks.  This makes sure there are no errors left in the program before it is released. This makes sure the game is fully functioning and will work when it gets released to the public using various testing methods, it is also used to find any areas that can be improved upon in the future. |

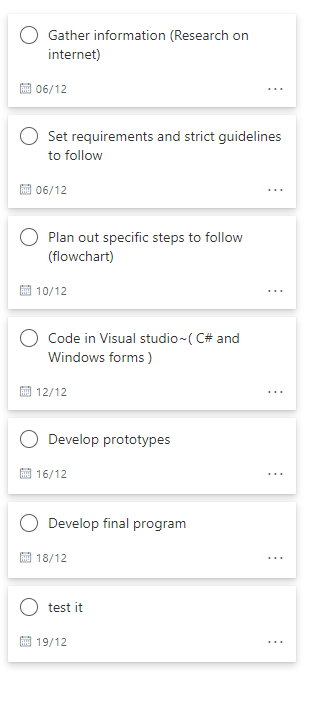
Final project definition:

The game is analogue and needs to be virtual, this will be changed in this project by recreating a game in windows forms and C#. The problem was observed by playing the game and wanting it to be virtual. The system also needs to be electronic.

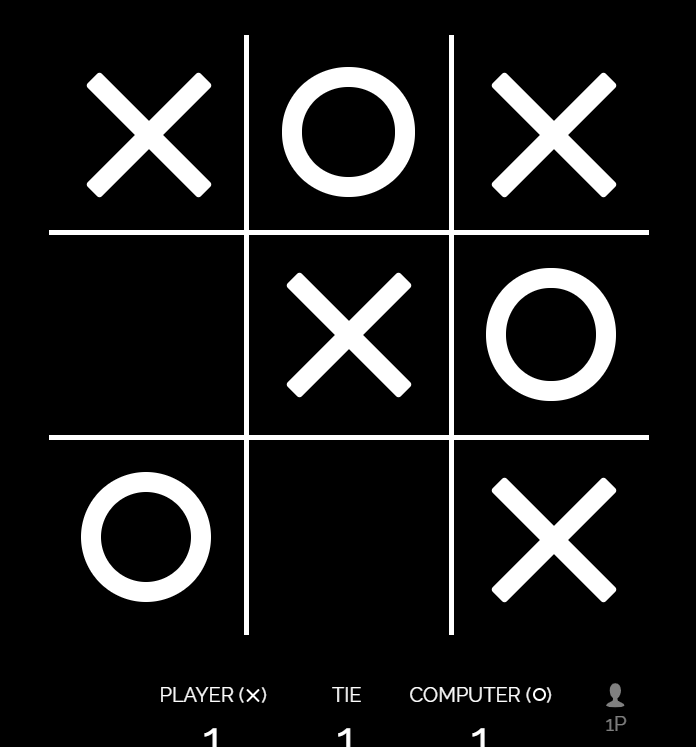
The game must not crash or have errors. The game must be player versus player. The game must have 2 modes, a single round mode and a best of 3 mode. Make it so a line is placed in the middle of the winning sequence. The best of 3 mode must have a score counter for win, lose and tie. Use windows forms for the game to have buttons that can be pressed to input the ‘X’ or ‘O’. A win or lose screen after the 3 matches. Able to press buttons and they have a response on the grid.

# Task2: Apply Systems Methodology to develop the project

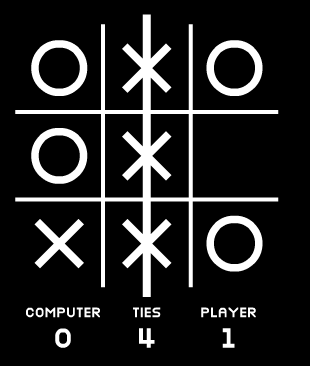
Planning: I will use Microsoft planner to keep track of the task.



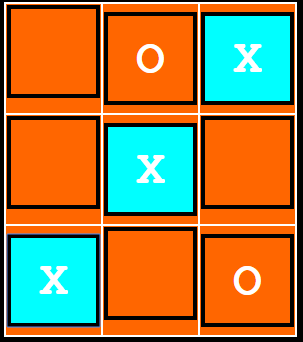
Research – I have tried a few different web games for tic tac toe or noughts and crosses.



* This is a tic tac toe game that I found on the internet.
* This goes straight into the game against a computer and freezes once you win the game, if you lose or tie it plays on.
* There is no indicator to show that you won besides the score counter.
* It is black and white.
* The player and computer take turns to place their respective points (X & O).
* Fairly simple game.



* This is the 2nd tic tac toe game I found.
* This gives you an option of 3 modes easy, medium and hard. Easy will always let you win, medium will be a bit harder to win as you will draw sometimes and hard is impossible to win as you will keep getting blocked and have no chance to get 3 in a row.
* There is a line through the winning sequence.

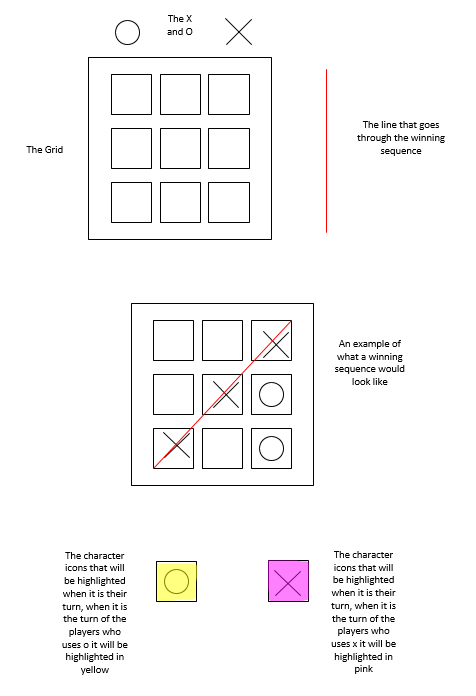


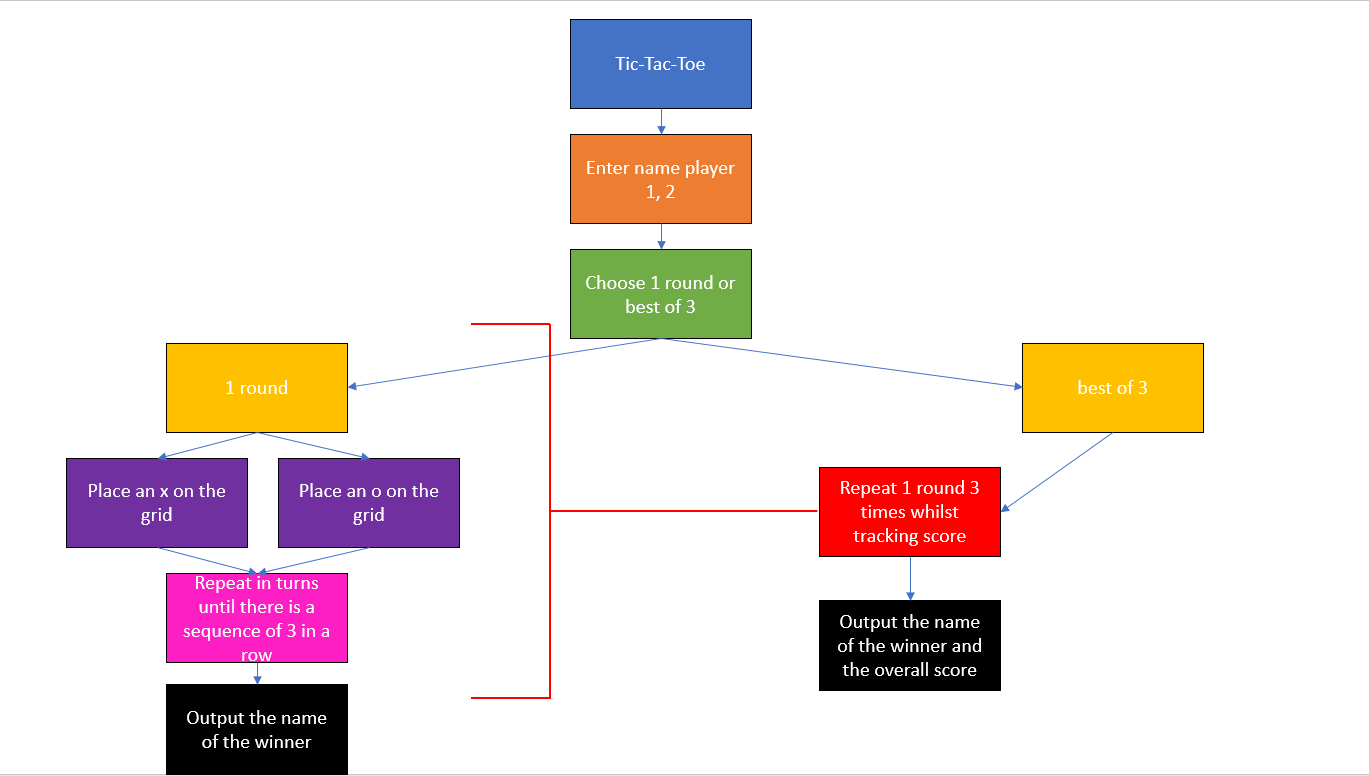
* This is the 3rd tic tac toe game.
* The winning sequence is highlighted.
* It is in colour.
* There is also a choice of player versus player games and not just player versus computer.

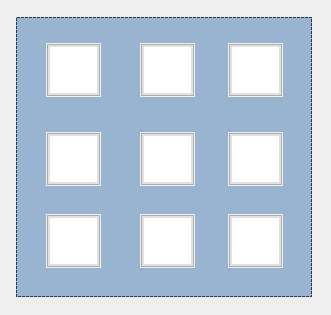
# Analysis & Requirements:

* The game must not crash or have errors.
  + so that it runs properly.
* The game must be player versus player.
  + So that there is a competitive aspect of it and has an appeal to the players so that they will enjoy the game.
* The game must have 2 modes, a single round mode and a best of 3 mode.
  + So there is some variation and the game isn’t too simple.
* Make it so a line is placed in the middle of the winning sequence.
  + So that it is aesthetically pleasing (at least in my eyes) and it shows which sequence is the winning one.
* The best of 3 mode must have a score counter for win, lose and tie.
  + To keep track of the scores visually so you can know who is in the lead and the final score.
* Use windows forms for the game to have buttons that can be pressed to input the ‘X’ or ‘O’.
  + So that the game has a GUI as that is one of the requirements, and so that it is more interactive and you don’t have to manually input the coordinates on a grid to enter ‘X’ or ‘O’.
* A win or lose screen after the 3 matches.
  + So that the player will know the results of the match.
* Able to press buttons and they have a response on the grid.
  + So the game will work and not be useless, this is also a way to make the game interactive and more straightforward as clicking a button is faster and easier than entering coordinates on a grid.

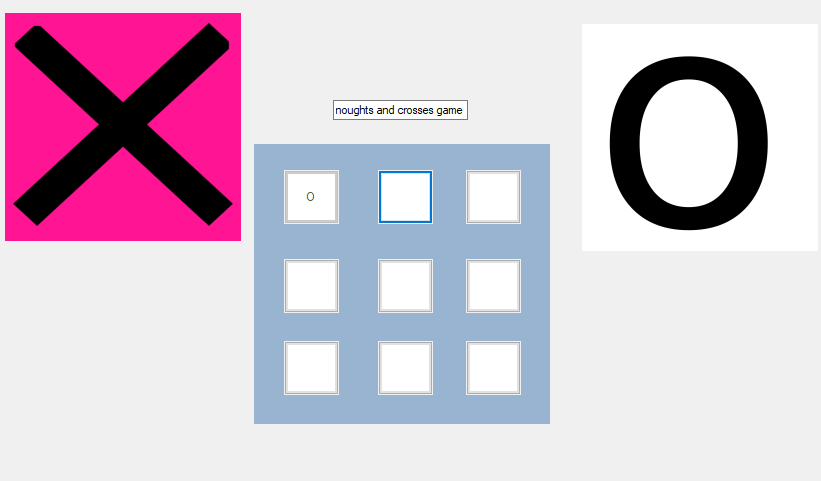
Design:



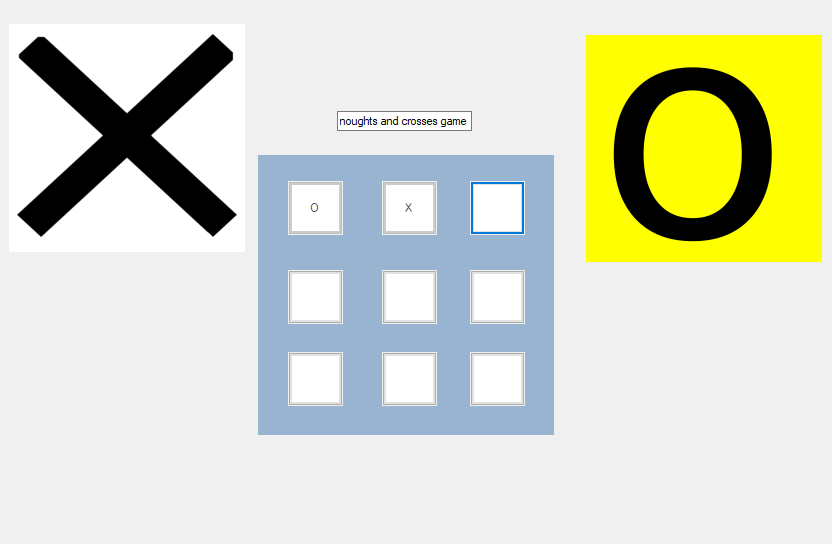
­­­­­



windows forms. 9 buttons inside of an image box



This is when it is the turn of X.



This is when it is the turn of O

## Cycle 1

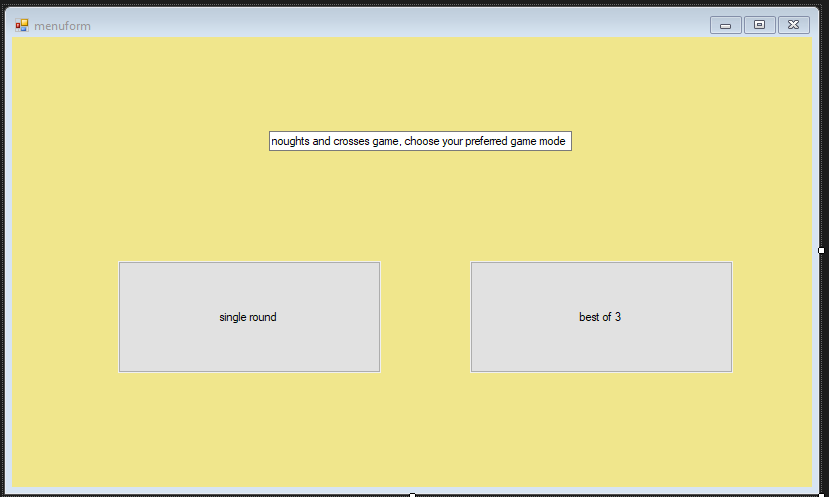
|  |  |
| --- | --- |
| **Interface and code:** Main Menu | **Code** |
| **Interface and code:** | **Code** |
| **Testing and Feedback from this cycle**  What do you think of the interface?  I think that the game runs smoothly and looks good.  Is the game doing its intended purpose?  It is playing but there should be a menu that gives the option of a 3 round or single round game.  Does the game have any bugs?  No. | |
| **Objectives/Requirements for next Cycle**   1. **Add a menu that allows you to choose between a single round and best of 3 matches.** 2. **A score counter for the best of 3 matches.** | |

## Cycle 2

|  |  |
| --- | --- |
| **Interface and code:** Main Menu | **Code** |
| **Interface and code:** | **Code** |
| **Testing and Feedback from this cycle**  **1. The score counter works but it is not visually displayed**  **2. menu allows you to choose between 2 game modes**  **3. first to achieve 2 points is the winner, draws can break the game – if x = 1 and o = 1 and then you draw you cannot proceed with the game.** | |
| **Objectives/Requirements for next Cycle**   1. **Make it so that a draw does not break the game** | |

Code:

Menuform code –



namespace OandX

{

public partial class menuform : Form

{

public menuform()

{

InitializeComponent();

}

private void Button1\_Click(object sender, EventArgs e)

{

Form1 open = new Form1();

open.Show();

}

private void Button2\_Click(object sender, EventArgs e)

{

Form11 open = new Form11();

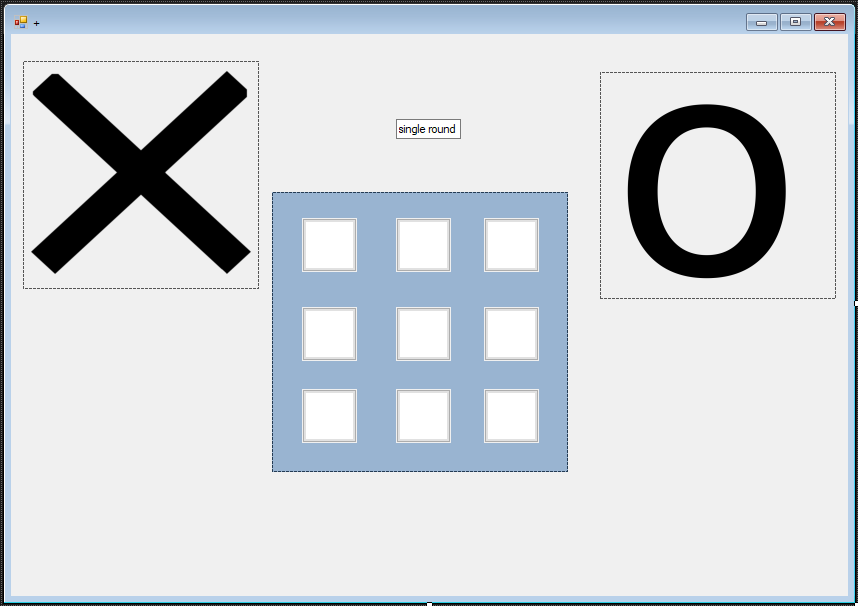
open.Show();

}

}

}

Form1 code –



namespace OandX

{

public partial class Form1 : Form

{

bool turn = true;

int turnCount = 0;

public Form1()

{

InitializeComponent();

}

private void Button1\_Click(object sender, EventArgs e)

{

try

{

Button b = (Button)sender;

if (turn == true)

{

b.Text = "O";

pictureBox2.BackColor = Color.DeepPink;

pictureBox3.BackColor = Color.White;

}

else

{

b.Text = "X";

pictureBox3.BackColor = Color.Yellow;

pictureBox2.BackColor = Color.White;

}

turn = !turn;

turnCount++;

b.Enabled = false;

check();

}

catch { }

}

public void check()

{

bool ThereIsAWinner = false;

if (a1.Text == a2.Text && a2.Text == a3.Text && a1.Text != "")

ThereIsAWinner = true;

else if (b1.Text == b2.Text && b2.Text == b3.Text && b1.Text != "")

ThereIsAWinner = true;

else if (c1.Text == c2.Text && c2.Text == c3.Text && c1.Text != "")

ThereIsAWinner = true;

else if (a1.Text == b1.Text && b1.Text == c1.Text && a1.Text != "")

ThereIsAWinner = true;

else if (a2.Text == b2.Text && b2.Text == c2.Text && a2.Text != "")

ThereIsAWinner = true;

else if (a3.Text == b3.Text && b3.Text == c3.Text && a3.Text != "")

ThereIsAWinner = true;

else if (a1.Text == b2.Text && b2.Text == c3.Text && a1.Text != "")

ThereIsAWinner = true;

else if (a3.Text == b2.Text && b2.Text == c1.Text && a3.Text != "")

ThereIsAWinner = true;

if (ThereIsAWinner)

{

string w = "";

if (turn)

w = "X";

else w = "O";

MessageBox.Show("The winner is " + w + "!");

this.Close();

}

else if (turnCount == 9)

{

MessageBox.Show("Its a draw!");

}

}

private void A2\_Click(object sender, EventArgs e)

{

try

{

Button b = (Button)sender;

if (turn == true)

{

b.Text = "O";

pictureBox2.BackColor = Color.DeepPink;

pictureBox3.BackColor = Color.White;

}

else

{

b.Text = "X";

pictureBox3.BackColor = Color.Yellow;

pictureBox2.BackColor = Color.White;

}

turn = !turn;

turnCount++;

b.Enabled = false;

check();

}

catch { }

}

private void A3\_Click(object sender, EventArgs e)

{

try

{

Button b = (Button)sender;

if (turn == true)

{

b.Text = "O";

pictureBox2.BackColor = Color.DeepPink;

pictureBox3.BackColor = Color.White;

}

else

{

b.Text = "X";

pictureBox3.BackColor = Color.Yellow;

pictureBox2.BackColor = Color.White;

}

turn = !turn;

turnCount++;

b.Enabled = false;

check();

}

catch { }

}

private void B1\_Click(object sender, EventArgs e)

{

try

{

Button b = (Button)sender;

if (turn == true)

{

b.Text = "O";

pictureBox2.BackColor = Color.DeepPink;

pictureBox3.BackColor = Color.White;

}

else

{

b.Text = "X";

pictureBox3.BackColor = Color.Yellow;

pictureBox2.BackColor = Color.White;

}

turn = !turn;

turnCount++;

b.Enabled = false;

check();

}

catch { }

}

private void B2\_Click(object sender, EventArgs e)

{

try

{

Button b = (Button)sender;

if (turn == true)

{

b.Text = "O";

pictureBox2.BackColor = Color.DeepPink;

pictureBox3.BackColor = Color.White;

}

else

{

b.Text = "X";

pictureBox3.BackColor = Color.Yellow;

pictureBox2.BackColor = Color.White;

}

turn = !turn;

turnCount++;

b.Enabled = false;

check();

}

catch { }

}

private void B3\_Click(object sender, EventArgs e)

{

try

{

Button b = (Button)sender;

if (turn == true)

{

b.Text = "O";

pictureBox2.BackColor = Color.DeepPink;

pictureBox3.BackColor = Color.White;

}

else

{

b.Text = "X";

pictureBox3.BackColor = Color.Yellow;

pictureBox2.BackColor = Color.White;

}

turn = !turn;

turnCount++;

b.Enabled = false;

check();

}

catch { }

}

private void C1\_Click(object sender, EventArgs e)

{

try

{

Button b = (Button)sender;

if (turn == true)

{

b.Text = "O";

pictureBox2.BackColor = Color.DeepPink;

pictureBox3.BackColor = Color.White;

}

else

{

b.Text = "X";

pictureBox3.BackColor = Color.Yellow;

pictureBox2.BackColor = Color.White;

}

turn = !turn;

turnCount++;

b.Enabled = false;

check();

}

catch { }

}

private void C2\_Click(object sender, EventArgs e)

{

try

{

Button b = (Button)sender;

if (turn == true)

{

b.Text = "O";

pictureBox2.BackColor = Color.DeepPink;

pictureBox3.BackColor = Color.White;

}

else

{

b.Text = "X";

pictureBox3.BackColor = Color.Yellow;

pictureBox2.BackColor = Color.White;

}

turn = !turn;

turnCount++;

b.Enabled = false;

check();

}

catch { }

}

private void C3\_Click(object sender, EventArgs e)

{

try

{

Button b = (Button)sender;

if (turn == true)

{

b.Text = "O";

pictureBox2.BackColor = Color.DeepPink;

pictureBox3.BackColor = Color.White;

}

else

{

b.Text = "X";

pictureBox3.BackColor = Color.Yellow;

pictureBox2.BackColor = Color.White;

}

turn = !turn;

turnCount++;

b.Enabled = false;

check();

}

catch { }

}

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

{

turnCount = 0;

a1.Text = "";

a1.Enabled = true;

a2.Text = "";

a2.Enabled = true;

a3.Text = "";

a3.Enabled = true;

b1.Text = "";

b1.Enabled = true;

b2.Text = "";

b2.Enabled = true;

b3.Text = "";

b3.Enabled = true;

c1.Text = "";

c1.Enabled = true;

c2.Text = "";

c2.Enabled = true;

c3.Text = "";

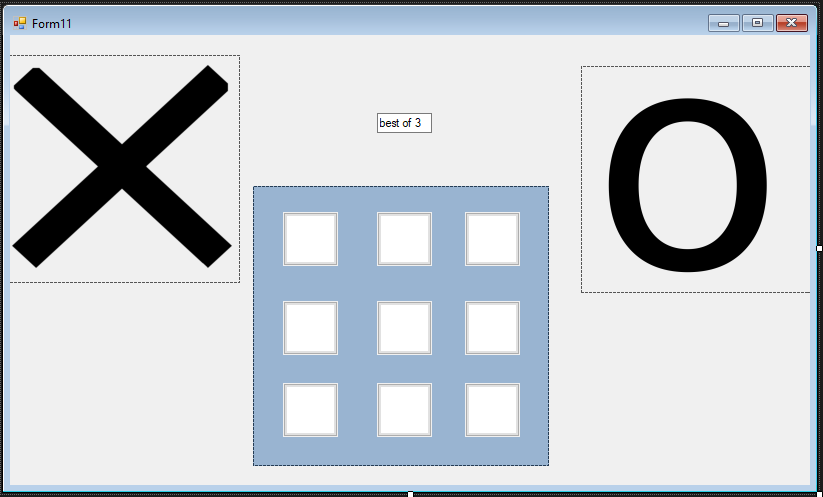
c3.Enabled = true;

}

}

}

Form11 code –



namespace OandX

{

public partial class Form11 : Form

{

bool turn = true;

int turnCount = 0;

int scoreX = 0;

int scoreO = 0;

int scoreD = 0;

public Form11()

{

InitializeComponent();

}

private void A1\_Click(object sender, EventArgs e)

{

try

{

Button b = (Button)sender;

if (turn == true)

{

b.Text = "O";

pictureBox2.BackColor = Color.DeepPink;

pictureBox3.BackColor = Color.White;

}

else

{

b.Text = "X";

pictureBox3.BackColor = Color.Yellow;

pictureBox2.BackColor = Color.White;

}

turn = !turn;

turnCount++;

b.Enabled = false;

check();

}

catch { }

}

public void check()

{

bool ThereIsAWinner = false;

if (a1.Text == a2.Text && a2.Text == a3.Text && a1.Text != "")

{ ThereIsAWinner = true; }

else if (b1.Text == b2.Text && b2.Text == b3.Text && b1.Text != "")

{ ThereIsAWinner = true; }

else if (c1.Text == c2.Text && c2.Text == c3.Text && c1.Text != "")

{ ThereIsAWinner = true; }

else if (a1.Text == b1.Text && b1.Text == c1.Text && a1.Text != "")

{ ThereIsAWinner = true; }

else if (a2.Text == b2.Text && b2.Text == c2.Text && a2.Text != "")

{ ThereIsAWinner = true; }

else if (a3.Text == b3.Text && b3.Text == c3.Text && a3.Text != "")

{ ThereIsAWinner = true; }

else if (a1.Text == b2.Text && b2.Text == c3.Text && a1.Text != "")

{ ThereIsAWinner = true; }

else if (a3.Text == b2.Text && b2.Text == c1.Text && a3.Text != "")

{ ThereIsAWinner = true; }

if (ThereIsAWinner)

{

string w = "";

if (turn)

{

w = "X";

scoreX = scoreX + 1;

MessageBox.Show("The winner is " + w + "! - X Score = " + scoreX);

}

else

{

w = "O";

scoreO = scoreO + 1;

MessageBox.Show("The winner is " + w + "! - O Score = " + scoreO);

}

if (scoreX == 2)

{

MessageBox.Show("The winner is " + w + "!");

}

else if (scoreO == 2)

{

MessageBox.Show("The winner is " + w + "!");

}

else if (scoreX == 1 && scoreO == 1 && scoreD == 1)

{

MessageBox.Show("Draw, there is no winner");

}

else

{

turnCount = 0;

a1.Text = "";

a1.Enabled = true;

a2.Text = "";

a2.Enabled = true;

a3.Text = "";

a3.Enabled = true;

b1.Text = "";

b1.Enabled = true;

b2.Text = "";

b2.Enabled = true;

b3.Text = "";

b3.Enabled = true;

c1.Text = "";

c1.Enabled = true;

c2.Text = "";

c2.Enabled = true;

c3.Text = "";

c3.Enabled = true;

}

}

else if (turnCount == 9)

{

scoreD = scoreD + 1;

MessageBox.Show("Its a draw!");

}

}

private void A2\_Click\_1(object sender, EventArgs e)

{

try

{

Button b = (Button)sender;

if (turn == true)

{

b.Text = "O";

pictureBox2.BackColor = Color.DeepPink;

pictureBox3.BackColor = Color.White;

}

else

{

b.Text = "X";

pictureBox3.BackColor = Color.Yellow;

pictureBox2.BackColor = Color.White;

}

turn = !turn;

turnCount++;

b.Enabled = false;

check();

}

catch { }

}

private void A3\_Click(object sender, EventArgs e)

{

try

{

Button b = (Button)sender;

if (turn == true)

{

b.Text = "O";

pictureBox2.BackColor = Color.DeepPink;

pictureBox3.BackColor = Color.White;

}

else

{

b.Text = "X";

pictureBox3.BackColor = Color.Yellow;

pictureBox2.BackColor = Color.White;

}

turn = !turn;

turnCount++;

b.Enabled = false;

check();

}

catch { }

}

private void B1\_Click(object sender, EventArgs e)

{

try

{

Button b = (Button)sender;

if (turn == true)

{

b.Text = "O";

pictureBox2.BackColor = Color.DeepPink;

pictureBox3.BackColor = Color.White;

}

else

{

b.Text = "X";

pictureBox3.BackColor = Color.Yellow;

pictureBox2.BackColor = Color.White;

}

turn = !turn;

turnCount++;

b.Enabled = false;

check();

}

catch { }

}

private void B2\_Click(object sender, EventArgs e)

{

try

{

Button b = (Button)sender;

if (turn == true)

{

b.Text = "O";

pictureBox2.BackColor = Color.DeepPink;

pictureBox3.BackColor = Color.White;

}

else

{

b.Text = "X";

pictureBox3.BackColor = Color.Yellow;

pictureBox2.BackColor = Color.White;

}

turn = !turn;

turnCount++;

b.Enabled = false;

check();

}

catch { }

}

private void B3\_Click(object sender, EventArgs e)

{

try

{

Button b = (Button)sender;

if (turn == true)

{

b.Text = "O";

pictureBox2.BackColor = Color.DeepPink;

pictureBox3.BackColor = Color.White;

}

else

{

b.Text = "X";

pictureBox3.BackColor = Color.Yellow;

pictureBox2.BackColor = Color.White;

}

turn = !turn;

turnCount++;

b.Enabled = false;

check();

}

catch { }

}

private void C1\_Click(object sender, EventArgs e)

{

try

{

Button b = (Button)sender;

if (turn == true)

{

b.Text = "O";

pictureBox2.BackColor = Color.DeepPink;

pictureBox3.BackColor = Color.White;

}

else

{

b.Text = "X";

pictureBox3.BackColor = Color.Yellow;

pictureBox2.BackColor = Color.White;

}

turn = !turn;

turnCount++;

b.Enabled = false;

check();

}

catch { }

}

private void C2\_Click(object sender, EventArgs e)

{

try

{

Button b = (Button)sender;

if (turn == true)

{

b.Text = "O";

pictureBox2.BackColor = Color.DeepPink;

pictureBox3.BackColor = Color.White;

}

else

{

b.Text = "X";

pictureBox3.BackColor = Color.Yellow;

pictureBox2.BackColor = Color.White;

}

turn = !turn;

turnCount++;

b.Enabled = false;

check();

}

catch { }

}

private void C3\_Click(object sender, EventArgs e)

{

try

{

Button b = (Button)sender;

if (turn == true)

{

b.Text = "O";

pictureBox2.BackColor = Color.DeepPink;

pictureBox3.BackColor = Color.White;

}

else

{

b.Text = "X";

pictureBox3.BackColor = Color.Yellow;

pictureBox2.BackColor = Color.White;

}

turn = !turn;

turnCount++;

b.Enabled = false;

check();

}

catch { }

}

private void Form11\_Load(object sender, EventArgs e)

{

}

}

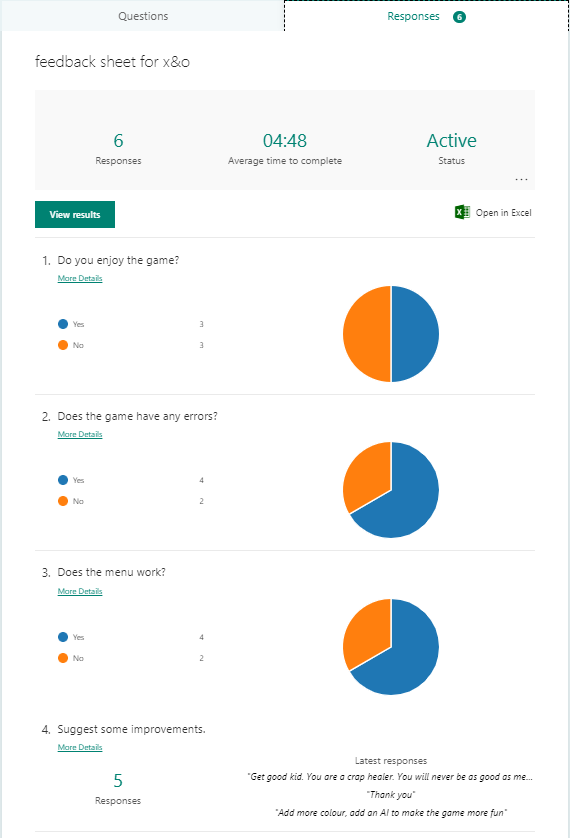
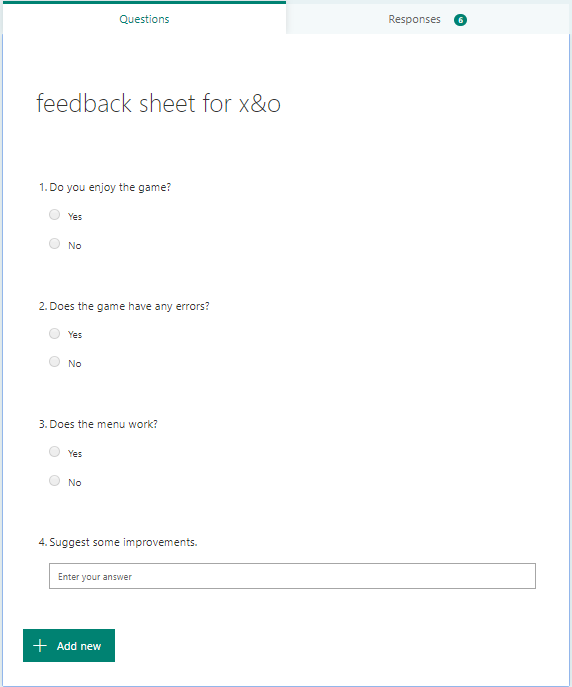
}

Program.cs –



# Task 3: Review of the Final Solution

The purpose of testing this is so that I can make sure that there are no errors or discrepancies in the code and that the game works perfectly. The testing structure I follow is called black box testing which is where I will have another person test my program to find errors without any prior knowledge of it, or having previously seen the code.



This is my response form I sent to my classmates that attempted my game. The feedback has been mostly positive and the most common response for improvement suggestions was to include more vibrant colours to make the game aesthetically pleasing.

Strengths:

* The game runs properly without crashing – there were no errors as of right now and the game ran properly without crashing.
* The GUI is well designed and is aesthetically pleasing whilst also being functionable and providing a satisfactory user experience – the testers complimented this aspect a lot and they enjoyed the game in its competitive aspects, they also did not have much problem with the appearance of the game.
* The best of 3 game mode has a competitive atmosphere – the 2 people who were playing had quite a lot of matches as they were not willing to be beaten, so I think this served its purpose.
* The Menu is simple and easy to understand – there is 2 buttons and a textbox that says “pick one of the 2 buttons”, it can’t really get any simpler than that.
* All buttons work as they are designed to – there were no responses that identified any errors as this was fixed in the testing phase.

Improvements:

* Include an AI or Computer Player – to increase the difficulty to make it more fun, so you don’t always instantly win.
* Ensure win conditions – so you don’t win or lose when you haven’t even placed a winning sequence yet, I just fixed the problem where X would win instead of O in a 3 in a row O on the top row ‘A’.
* Add more colour – to make the game more aesthetically pleasing and in turn make more people want to play it over a bland and boring game as the vibrant colours will attract their attention.
* Add different grid sizes like a 4x4 grid – to make the game harder to win and thus increase the competitive aspect.

Order of improvements (from easiest to implement to the hardest to implement):

1. Ensure win conditions – I have done this already, the problem that was occurring was that any wins that were in a diagonal row would not count and the game would not accept it as a win so I went back to the code and changed this so that the game works properly. This also increases the users overall experience as this is a bug that can ruin the game.
2. Add more colour – This will be the fastest and easiest to implement as all I will have to do is recolour the background and some buttons in vibrant contrasting colours like neon green and dark blue.
3. Add different grid sizes – This will only require 20 to 30 lines of code, this is so that the game will have to be 4 in a row instead of 3 in a row and will be a lot harder as you will need a longer sequence to win and there is no middle button to have a higher starting point from.
4. Add an AI to play against – This will be the hardest and most time consuming as I don’t even know where to start, but I can make an easy mode where the AI places everything randomly and a hard mode where the ai places them in strategically correct places.

# Task 4: Evaluation of Systems Methodology

For this project I used the RAD methodology.

RAD or Rapid Application Development is based off of the waterfall methodology. Its aim is completing a program or objective in a short amount of time by splitting it up into stages.

It focuses of input and output sources and delivering updates to the projects in smaller chunks, large projects are split into several smaller projects for different groups to do.

When I used this, I analysed the project and found out my requirements using the internet to research existing versions of this game and taking into account what they do well and what I think they should include, designed the program using flowcharts and Microsoft Visio to make prototypes for the GUI, then I developed it using C# on visual studio and windows forms on visual studio, and tested it myself and got other people to test it using Whitebox and Blackbox testing, Then I improved on certain aspects that they included in their feedback such as the bugs and errors that occur while playing and some aesthetic features.

Planning stage:

* Planning a timeframe for what to do – some of the timeframes were finished before the specified dates, they could have been shortened and I could have lengthened other ones. This was very good as it helped me keep track of time and not miss my deadline, it helped to know what to do and when to finish a certain stage.
* However, I could have broken the task down further and put more detail into it.

Analysis stage:

* Research my game and existing games – looked at existing O&X games and took inspiration from them. Setting requirements and strict guidelines to follow – so that I can know what is needed in my game and what I have to do. This was very good as it helped me know what I could use as a basis and how to shape my game, it also gave me ideas of how to proceed with the project and what I should have done by the end of the deadline.
* However, I could have researched more games, and I only searched for digital games whilst I should have also analysed the analogue variants of the game such as on paper or blackboards.

Design stage:

* Design the GUI in Microsoft Visio – To have a rough idea of what it is going to look like. Make a flowchart / hierarchy graph – to have a basis to work from and know what steps to code and in what order. This was very good as I further cementer the steps which I would take to complete the project, I also had a flowchart specifically for the game which I used as a reference for my coding.
* However, I should have done more graphs for different aspects and I could have done more than 2 different types of graphs.

Development stage:

* Code in visual studio using C# and windows forms – to make the game work and have it linked to the GUI. Develop prototypes – To have something to evaluate and correct and then work forwards from it. Develop final program – This should meet all of the requirements that I have set and not have any errors in the code. This was good as I completed the game and made it work within the time frame.
* However, this took too long as I wasn’t familiar with many of the keywords and had to consult the internet multiple times.

Testing stage:

* Test it – To make sure that there are no errors using both Blackbox and Whitebox testing. Have people test it and receive feedback from them – To know what I can improve on using feedback. This was good as it helped to remove any lingering errors or bugs that I found and that my testers found.
* However, I could have introduced more than one form of testing.

Conclusion:

* This (RAD) was the best methodology for the problem – because I had to complete it in a relatively short timeframe.
* Waterfall would have been a more suitable method because – it is simpler, has less steps and evaluation.
* I would use the Waterfall method if I had to try again because – it is more straightforward and less planning and analysis work would have been involved so it would be easier for me and cause me less stress.

# Task 5: Demonstrate Individual Responsibility

How I demonstrated individual responsibility:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Task | Tools | Justification |
| 1 | Plan the project | Microsoft Planner | I set deadlines and achieved them. |
| 2 | Analysis: Research | Internet | I researched individually and did not ask for outside help. |
| 3 | Analysis: Requirements | Setting guidelines | I set strict guidelines that I wanted to achieve in this project and did achieve them. |
| 4 | Design the programme | Flowchart  Hierarchy chart | I made a hierarchy chart and used it to guide the steps I coded the game in. |
| 5 | Develop GUI | Microsoft Visio  Visual Studio | I roughly designed a GUI in Visio. |
| 6 | Develop prototypes | Visual Studio  C#  Windows forms | I coded in C# and fixed my errors repeatedly myself. |
| 7 | Develop final version | Visual Studio  C#  Windows forms | I finished the game and made sure there were no errors in both the code and GUI. |
| 8 | Test | People (user testing)  System testing | I reached out to others and had others test my code for any errors and had them give me feedback the aspects of it that they enjoyed and what they think could be improved. |