AI.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace TicTacToeQLearning
{
    class AI
    {
        static double highest = 0;
        static double lowest = 1;
        public static double EPSILON = 1;
        public static double GAMMA = 0.5;
        public static double ALPHA = 0.2;

        public static double stepPrice = 0;
        public static double[][] Q;
        int team;
        Field field;
        public AI(Field field, int team)
        {
            this.team = team;
            this.field = field;

            Q = new double[Field.states.Count][];
            for (int a = 0; a < Field.states.Count; a++)
            {
                Q[a] = new double[9];
            }
            ResetQ();
        }
        public void FillQ(StreamReader sr)
        {
            int c = 0;
            for(int a = 0; a < Field.states.Count; a++)
            {
                for(int b = 0; b < 9; b++)
                {
                    c++;
                    string v = sr.ReadLine();
                    double val = double.Parse(v);
                    Q[a][b] = val;
                }
            }
        }
        public bool MakeCertainMove(int action)
        {
            State currentState = field.GetCurrentState(team);

            if (currentState.Aim != 0)
                throw new Exception("MakeCertainMove: current State is Final:\n" +
currentState.getContent(0));

            int currentIndex = currentState.getIndex();
            if (currentState.nextState[0][action] == null)
            {
                Console.WriteLine("Not possible: " + currentState.Aim);
                return false;
            }
            Console.WriteLine(Field.Printable(currentState.getContent(0)));
            field.SetPosition(action, team, false);
            State enemyState = field.GetCurrentState(team % 2 + 1);
            int enemyStateIndex = enemyState.getIndex();
            double reward = field.getReward();
            if (enemyState.Aim == 2 || currentState.Aim == 2)
            {
                reward = 40;
            }

            double q = Q[currentIndex][action];
            double newQ = 0;
            if (reward != 0)
            {
                newQ = reward - q;
            }
            else
            {

                double qualityNextState = HighestValue(Q[enemyStateIndex]) - stepPrice;
                if (enemyState.Aim == 1)
                {
                    qualityNextState = 100;
                }
                newQ = (reward + GAMMA * (100 - qualityNextState)) - q;
            }

            Q[currentIndex][action] = q + ALPHA * newQ;

            if (q + ALPHA * newQ > highest)
                highest = q + ALPHA * newQ;
            else if (q + ALPHA * newQ < lowest)
                lowest = q + ALPHA * newQ;

            return true;
        }
        public bool MakeMove(bool rand, int start)
        {
            State currentState = field.GetCurrentState(team);
            if (currentState.Aim != 0)
                throw new Exception("MakeMove: current State is Final:\n" +
currentState.getContent(0));

            int currentIndex = currentState.getIndex();

            int action = 0;

            if(rand)
            {
                List<int> possibleActions = new List<int>();
                for(int a = 0; a < 9; a++)
                {
                    if(currentState.getContent(0)[a] == '0')
                    {
                        possibleActions.Add(a);
                    }

                }
                action = possibleActions[Program.random.Next(0, possibleActions.Count)];
            }
            else
            {
                action = HighestIndex(Q[currentIndex]);
            }
            field.SetPosition(action, team, false);
            State enemyState = field.GetCurrentState(team % 2 + 1);
            int enemyStateIndex = enemyState.getIndex();
            double reward = field.getReward();
            if(enemyState.Aim == 2 || currentState.Aim == 2)
            {
                reward = 40;
            }

            double q = Q[currentIndex][action];
            double newQ = 0;

            if (reward != 0)
            {
                newQ = reward - q;
            }
            else
            {

                double qualityNextState = HighestValue(Q[enemyStateIndex]) - stepPrice;
                if(enemyState.Aim == 1)
                {
                    qualityNextState = 100;
                }
                newQ = (reward + GAMMA * ( 100 - qualityNextState)) - q;
            }


            Q[currentIndex][action] = q + ALPHA * newQ;

            if (reward != 0 && EPSILON > 0)
            {
                EPSILON *= 1 - (1.0000 / 10000000);
                //Console.WriteLine(EPSILON);
                return false;
            }
            else if(reward != 0)
            {
                return false;
            }
                return true;
            //TODO
            /*
             * Q Array füllen
             * vorausschauendes Verhalten einfügen
             *
             * */
        }
        void GoBack(string s, double highest)
        {
            int currentIndex = field.FindStateIndex(s);
            double highestValue = HighestValue(Q[currentIndex]);
            highestValue = highest > highestValue ? highest : highestValue;
            int C1 = Field.CharCount(s, '1');
            int C2 = Field.CharCount(s, '2');
            if (C1 == 0 && C2 == 0)
                return;
            else if (C1 > C2)
            {
                for (int a = 0; a < s.Length; a++)
                {
                    if (s[a] == '1')
                    {
                        string newString = "";
                        for (int b = 0; b < s.Length; b++)
                        {
                            if (b == a)
                            {
                                newString += 0;
                                continue;
                            }
                            newString += s[b];
                        }
                        int newIndex = field.FindStateIndex(newString);
                    if (Q[newIndex][a] == highestValue * GAMMA && Q[newIndex][a] !=
0)
                    {
                        Q[newIndex][a] = 0;
                        GoBack(newString, highestValue);
                    }
                }
            }
        }
        else if (C1 < C2)
        {
            for (int a = 0; a < s.Length; a++)
            {
                if (s[a] == '2')
                {
                    string newString = "";
                    for (int b = 0; b < s.Length; b++)
                    {
                        if (b == a)
                        {
                            newString += 0;
                            continue;
                        }
                        newString += s[b];
                    }
                    int newIndex = field.FindStateIndex(newString);
                    if (Q[newIndex][a] == highestValue * GAMMA && Q[newIndex][a] !=
0)
                    {
                        Q[newIndex][a] = 0;
                        GoBack(newString, highestValue);
                    }
                }
            }
        }
        else if (C1 == C2)
        {
            for (int a = 0; a < s.Length; a++)
            {
                if (s[a] != '0')
                {
                    string newString = "";
                    for (int b = 0; b < s.Length; b++)
                    {
                        if (b == a)
                        {
                            newString += 0;
                            continue;
                        }
                        newString += s[b];
                    }
                    int newIndex = field.FindStateIndex(newString);
                    if (Q[newIndex][a] == highestValue * GAMMA && Q[newIndex][a] !=
0)
                    {
                        Q[newIndex][a] = 0;
                        GoBack(newString, highestValue);
                    }
                }
            }
        }
    }
    public bool MakeMove(int start, StreamWriter sw)
    {
        State current = field.GetCurrentState(team);
        int index = current.getIndex();
        int action = HighestIndex(Q[index]);
        sw.Write("\t"+ team + " : " + action);
        sw.Flush();
        field.SetPosition(action, team, false);
        double reward = field.getReward();
        if (reward != 0)
        {
            if (reward == 100)
            {
                sw.Write("\tWin\a");
            }
            else
            {
                sw.Write("\tDraw\a");
            }
            return false;
        }
        else
            return true;
    }
    #region Utility

    int HighestIndex(double[] vals)
    {
        double highest = double.MinValue;
        int index = 0;
        for (int a = 0; a < vals.Length; a++)
        {
            if (vals[a] > highest)
            {
                highest = vals[a];
                index = a;
            }
        }
        return index;
    }
    double HighestValue(double[] vals)
    {
        double highest = double.MinValue;
        for (int a = 0; a < vals.Length; a++)
        {
            if (vals[a] > highest)
            {
                highest = vals[a];
            }
        }
        return highest == -200 ? 0 : highest;
    }

    #endregion

    public static void ResetQ()
    {
        for (int a = 0; a < Field.states.Count; a++)
        {
            for (int b = 0; b < 9; b++)
            {
                Q[a][b] = Field.states[a].nextState[0][b] == null ? -200 : 0;
            }
        }
    }
  }
}
```

Field.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace TicTacToeQLearning
{

    class Field
    {
        public static List<State> states = new List<State>();
        public List<State> finishedStates = new List<State>();
        public State[] currentState;
        bool gotCombinations = false;

        double WinReward = 100;
        double DrawReward = 20;

        int[][] field;

        public Field()
        {
            currentState = new State[2];
            field = new int[3][];
            for(int a = 0; a < 3; a++)
            {
                field[a] = new int[3];
                for(int b = 0; b < 3; b++)
                {
                    field[a][b] = 0;
                }
            }
        }
        public void Reset()
        {
            for(int a = 0; a < 3; a++)
            {
                for(int b = 0; b < 3; b++)
                {
                    field[a][b] = 0;
                }
            }
            currentState[0] = states[0];
            currentState[1] = states[0];
        }
        public void setFirstState(int index)
        {
            State startState = states[index];
            State startState2 = FindState(startState.getContent(2));

            if(CharCount(startState.getContent(0), '1') <= CharCount(startState.getContent(0),
'2'))
            {
                currentState[0] = startState;
                currentState[1] = startState2;
            }
            else
            {
                currentState[1] = startState;
                currentState[0] = startState2;
            }

        }
        public void SetRandomState(int start)
        {
            State startState = states[0];
            do
            {
                startState = states[Program.random.Next(0, states.Count)];
                if (startState.Aim == 0 && CharCount(startState.getContent(0), '1') ==
CharCount(startState.getContent(0), '2'))
                {
                    //Console.WriteLine(CharCount(startState.getContent(0), '1') + ":" +
CharCount(startState.getContent(0), '2'));
                    break;
                }

            } while (true);

            /*Console.WriteLine(startState.getContent(1));
            Console.WriteLine(startState.getContent(2));
            */

            currentState[0] = FindState(startState.getContent(2));
            currentState[1] = startState;
            if (currentState[0] == null || currentState[1] == null)
                throw new Exception("No state found");
        }
        public double getReward()
        {
            if (currentState[0].Aim == 1)
                return WinReward;
            else if (currentState[0].Aim == 2)
                return DrawReward;
            else
                return 0;
        }
        public State GetCurrentState(int team)
        {
            return currentState[team - 1];
        }
        public string getCombination(int view)
        {

            if(currentState[0] == null)
            {
                string r = "";
                for(int a = 0; a < 3; a++)
                {
                    for(int b = 0; b < 3; b++)
                    {
                        r += field[a][b];
                    }
                }
                if (view == 2)
                {
                    r.Replace('1', '3');
                    r.Replace('2', '1');
                    r.Replace('3', '2');
                }
                return r;
            }
            return currentState[0].getContent(view);
        }
        public void SetPosition(int position, int value, bool ignoreNotZero)
        {
            if(field[position / 3][position % 3] == 0 || ignoreNotZero)
            {
                field[position / 3][position % 3] = value;
            }
            if (ignoreNotZero)
                return;

            State lastState1 = currentState[0];
            State lastState2 = currentState[1];
            currentState[0] = lastState1.nextState[value - 1][position];
            currentState[1] = lastState2.nextState[value % 2][position];

            if (currentState[0] == null || currentState[1] == null)
                throw new Exception("No state found");
        }

#region GetCombinations
        List<string> combinations = new List<string>();
        public void GetCombinations(string file, bool print)
        {
            if (gotCombinations)
                return;

            gotCombinations = true;
            if (file.Contains(":/"))
            {
                FileStream f = new FileStream(file, FileMode.Open);
                StreamReader sr = new StreamReader(f);//ermöglicht das Lesen aus einem
Input-Stream wie dem FileStream

                string s = sr.ReadLine();
                while (s != null)
                {
                    if (print) Console.WriteLine("|" + s + "|");
                    this.combinations.Add(s);
                    s = sr.ReadLine();
                }
                goto ConvertStringToState;
            }
            //find every possible state, the field can be in
            Layer(0, print);
            //remove every combination, that cant be reached in a normal game (one player
makes more than 1 move more than the other player
            List<string> remove = new List<string>();
            for (int a = 0; a < combinations.Count; a++)
            {
                int ones = CharCount(combinations[a], '1');
                int twos = CharCount(combinations[a], '2');
                if (Math.Abs(twos - ones) > 1)
                {
                    remove.Add(combinations[a]);
                }
            }
            for (int a = 0; a < remove.Count; a++)
            {
                combinations.Remove(remove[a]);
            }
            if (print)
            {


                for (int a = 0; a < combinations.Count; a++)
                {
                    Console.WriteLine(Printable(combinations[a])/*siehe "Print"*/);
                }
            }
            Console.WriteLine("Where do you want to save the data?");
            file = Console.ReadLine();
            FileStream fs = new FileStream(file, FileMode.Create);
            StreamWriter sw = new StreamWriter(fs);
            for (int a = 0; a < combinations.Count; a++)
            {
                sw.WriteLine(combinations[a]);
                sw.Flush();
            }
            ConvertStringToState:;
            for (int a = 0; a < combinations.Count; a++)
            {
                State s = new State(combinations[a], a);
                states.Add(s);
            }
            for (int a = 0; a < states.Count; a++)
            {
                string currentState = states[a].getContent(0);

                states[a].Aim = CheckAim(currentState);
                if(states[a].Aim != 0)
                {
                    finishedStates.Add(states[a]);
                }

                Console.WriteLine(Printable(currentState));
                for (int b = 0; b < 2; b++)
                {
                    for (int c = 0; c < 9; c++)
                    {
                        State temp = FindState(StateAfterAction(currentState, c, b + 1));

                        if (currentState[c] != '0') // wird ein benutztes Feld überschrieben, wird ein
Zustand ausgewählt, der nicht möglich ist
                            temp = null;

                        states[a].nextState[b][c] = temp;
                        /*if (temp == null)
                        {
                            Console.Write("--\t");
                        }
                        else
                        {
                            Console.Write(temp.getContent(0)+ "\t");
                        }*/
                    }
                }
            }
            finishedStates.Sort(delegate (State a, State b)
            { return a.Aim.CompareTo(b.Aim);
            }
            );
        }
        int CheckAim(string s)
        {

            if (s[0] != '0' && s[0] == s[1] && s[0] == s[2])
            {              //X|X|X
                return 1;   // | |
            }              // | |
            else if (s[3] != '0' && s[3] == s[4] && s[3] == s[5])
            {                  // |  |
                return 1;   //X|X|X
            }              // | |
            else if (s[6] != '0' && s[6] == s[7] && s[6] == s[8])
            {                 // | |
                return 1;   // | |
            }              //X|X|X
            else if (s[0] != '0' && s[0] == s[3] && s[0] == s[6])
            {                 //X| |
                return 1;   //X| |
            }                 //X| |
            else if (s[1] != '0' && s[1] == s[4] && s[1] == s[7])
            {                 // |X|
                return 1;   // |X|
            }                 // |X|
            else if (s[2] != '0' && s[2] == s[5] && s[2] == s[8])
            {              // | |X
                return 1;   // | |X
            }               // | |X
            else if (s[0] != '0' && s[0] == s[4] && s[0] == s[8])
            {               //X| |
                return 1;   // |X|
            }               // | |X
            else if (s[2] != '0' && s[2] == s[4] && s[2] == s[6])
            {               // | |X
                return 1;   // |X|
            }               //X| |
            else if(CharCount(s, '0') == 0)
            {
                return 2;
            }
            return 0;
        }
        string StateAfterAction(string s, int action, int value)
        {
            string r = "";
            for(int a = 0; a < s.Length; a++)
            {
                if( a == action)
                {
                    r += value;
                    continue;
                }
                r += s[a];
            }
            return r;
        }
        public State StateAfterAction(int action, int team)
        {
            State current = GetCurrentState(team);
            return current.nextState[1][action];
        }
        void Layer(int layer, bool print)
        {
            if (layer == 9)
            {
                string s = getCombination(0);
                combinations.Add(s);
                if (print) Console.WriteLine(s);
                return;
            }
            for (int a = 0; a < 3; a++)
            {
                SetPosition(layer, a, true);
                Layer(layer + 1, print);
            }

        }

        public static int CharCount(string s, char c)
        {
            int count = 0;
            for (int a = 0; a < s.Length; a++)
            {
                if (s[a] == c)
                    count++;
            }
            return count;
        }

        public State FindState(string s)
        {
            for (int a = 0; a < states.Count; a++)
            {
                if (states[a].getContent(0) == s)
                    return states[a];
            }

            return null;
        }
        public int FindStateIndex(string s)
        {
            for (int a = 0; a < combinations.Count; a++)
            {
                if (combinations[a] == s)
                    return a;
            }
            return -1;
        }
        //------------------------------End Get Combinations-------------------------------------
        public static string Printable(string s)
        {
            string r = "";
            for (int a = 0; a < s.Length; a++)
            {
                if (a % 3 == 0)
                {
                    r += "\n";
                }
                r += s[a] + "|";
            }
            return r;
        }
#endregion

    }
}
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.IO;
using System.Threading.Tasks;

namespace TicTacToeQLearning
{
    class Program
    {
        static FileStream QStream;

        static FileStream fs;
        public static Random random = new Random();
        static Field field = new Field();
        static string file;
        static bool gotCombinations = false;
        static AI[] players;
        static bool playersInitiated = false;

        static StreamWriter ActionLog;

        public static void Main(string[] args)
        {
            Console.WriteLine("Enter best Action Log");
            string f = Console.ReadLine();

            ActionLog = new StreamWriter(new FileStream(f, FileMode.OpenOrCreate));
            while (true)
            {
                Console.WriteLine("0: Get Combinations\n1: Train\n2: Print Sequence\n3:
Print Q\n4: Save Q\n5: Exit\n6: Assign Values");
                int input = int.Parse(Console.ReadLine());
                switch (input)
                {
                    case (0):
                        if (gotCombinations)
                            break;
                        gotCombinations = true;
                        Console.WriteLine("Where are the combinations?");
                        string file = Console.ReadLine();
                        if (file == "__")
                            file = "D:/TicTacToeFile.txt";

                        field.GetCombinations(file, true);
                        break;
                    case (1):
                        if (!gotCombinations)
                        {
                            Console.WriteLine("Where are the combinations?");
                            file = Console.ReadLine();
                            if (file == "__")
                                file = "D:/TicTacToeFile.txt";

                            field.GetCombinations(file, false);
                            gotCombinations = true;
                        }
                        if (!playersInitiated)
                        {
                            players = new AI[2];
                            for (int a = 0; a < 2; a++)
                            {
                                players[a] = new AI(field, a + 1);
                            }
                            playersInitiated = true;
                            Console.WriteLine("Q-Data?[Y/N]");
                            string inS = Console.ReadLine();
                            if (inS == "N" || inS == "n")
                            {
                                Console.WriteLine("Where?");
                                string path = Console.ReadLine();
                                if (path == "__")
                                    path = "D:/QData.txt";
                                QStream = new FileStream(path, FileMode.Create);

                                //ConfigureLastActions();
                            }
                            else
                            {
                                Console.WriteLine("Where?");
                                string path = Console.ReadLine();
                                if (path == "__")
                                    path = "D:/QData.txt";

                                QStream = new FileStream(path, FileMode.Open);

                                for(int a = 0; a < 2; a++)
                                {
                                    QStream.Seek(0, SeekOrigin.Begin);
                                    players[a].FillQ(new StreamReader(QStream));
                                }
                            }

                        }
                        Console.WriteLine("Use EPSILON-GREEDY EXPLOITATION?
[Y/N]");
                        string s = Console.ReadLine();
                        if (s == "N" || s == "n")
                        {
```

```
                    AI.EPSILON = 0;
                }
                else if (AI.EPSILON == 0)
                {
                    AI.EPSILON = 1;
                }
                Train();
                break;
            case (2):
                PrintSolution();
                break;
            case (3):
                PrintQ();
                break;
            case (4):
                SaveQ();
                break;
            case (5):
                return;
            case (6):
                AssignValues();
                break;
            case (7):
                PlayManually();
                break;
            case (8):
                AutomateLearnAndCapture();
                break;

        }
    }
}
static void AssignValues()
{
    double oldA = AI.ALPHA;
    double oldG = AI.GAMMA;
    double oldS = AI.stepPrice;
    Console.WriteLine("New Alpha?");
    AI.ALPHA = double.Parse(Console.ReadLine().Replace('.', ','));
    Console.WriteLine(AI.ALPHA);
    Console.WriteLine("New Gamma?");
    AI.GAMMA =
double.Parse(Console.ReadLine().Replace('.',','));//(Console.ReadLine());
    Console.WriteLine(AI.GAMMA);
    Console.WriteLine("New Step Prize?");
    AI.stepPrice = double.Parse(Console.ReadLine().Replace('.', ','));
    Console.WriteLine(AI.stepPrice);
    if (!playersInitiated)
        return;

    DoActionLog(oldA, oldG, oldS);


    AI.ResetQ();
}
static void DoActionLog(double oldA, double oldG, double oldS)
{
    ActionLog.WriteLine("A:" + oldA + "  G:" + oldG + "  S:" + oldS);
    ActionLog.WriteLine();
    ActionLog.Flush();
    field.Reset();
    int start = random.Next(0, 2);
    int player = start;
    while (players[player].MakeMove(start, ActionLog))
    {
        player = player == 0 ? 1 : 0;
    }
    ActionLog.WriteLine();
    ActionLog.Flush();
    field.Reset();
    player = start;
    while (players[player].MakeMove(false, start))
    {
        player = player == 0 ? 1 : 0;
        ActionLog.WriteLine(field.getCombination(0));
        ActionLog.Flush();
    }

    ActionLog.WriteLine(field.getCombination(0));
    ActionLog.WriteLine();
    ActionLog.WriteLine();
    ActionLog.Flush();
}
static void AutomateLearnAndCapture()
{
    if (!gotCombinations)
    {
        Console.WriteLine("Where are the combinations?");
        file = Console.ReadLine();
        if (file == "__")
            file = "D:/TicTacToeFile.txt";

        field.GetCombinations(file, false);
        gotCombinations = true;
    }
    if (!playersInitiated)
    {
        players = new AI[2];
        for (int a = 0; a < 2; a++)
        {
            players[a] = new AI(field, a + 1);
            }
            playersInitiated = true;
        }

        for (double alpha = 0.1; alpha <= 0.9; alpha += 0.2)
        {

            for(double gamma = 0.1; gamma <= 0.9; gamma += 0.2)
            {
                for(int step = 0; step <= 10; step += 5)
                {
                    AI.ALPHA = alpha;
                    AI.GAMMA = gamma;
                    AI.stepPrice = step;
                    AI.ResetQ();
                    Train(new StreamWriter(new FileStream("D:/TTTData/" + alpha * 100 +
"-" + gamma * 10 + "-" + step + ".txt", FileMode.Create)));
                    DoActionLog(alpha, gamma, step);

                }
            }

        }
    }
    static void Train(StreamWriter sw)
    {
        if (fs == null)
        {
            Console.WriteLine("Where do you want to save the log?");
            string s = Console.ReadLine();
            if (s == "__")
                s = "D:/TTTLog.txt";

            fs = new FileStream(s, FileMode.Create);

        }
        StreamWriter writer = new StreamWriter(fs);
        for (int q = 0; q < 40; q++)
        {
            for (int p = 0; p < Field.states.Count; p++)
            {
                writer.WriteLine("Run: " + q + ":" + p);
                Console.WriteLine("__________________________________________\n"
+
                            "________________________________________\n" +
                            "Run: " + q + ":" + p);
                for (int a = 0; a < 9; a++)
                {
                    //Game();
                    Game(p, a);
                }
                Console.WriteLine("--------------------------------");
                field.Reset();
                int start = random.Next(0, 2);
                int player = start;
                while (players[player].MakeMove(start, writer))
                {
                    player = player == 0 ? 1 : 0;
                }
                writer.WriteLine();
                writer.Flush();
                field.Reset();
                player = start;
                while (players[player].MakeMove(false, start))
                {
                    player = player == 0 ? 1 : 0;
                    writer.WriteLine(field.getCombination(0));
                    writer.Flush();
                }

                writer.WriteLine(field.getCombination(0));
                writer.WriteLine();
                writer.WriteLine();
                writer.Flush();
                field.Reset();
                player = start;
                while (players[player].MakeMove(false, start))
                {
                    player = player == 0 ? 1 : 0;
                }
                if(field.currentState[0].Aim == 1)
                {
                    sw.WriteLine("1");
                }
                else
                {
                    sw.WriteLine("0");
                }
                sw.Flush();
            }
        }
    }
    static void Train()
    {


        if (fs == null)
        {
            Console.WriteLine("Where do you want to save the log?");
            string s = Console.ReadLine();
            if (s == "__")
```

```
                s = "D:/TTTLog.txt";

            fs = new FileStream(s, FileMode.Create);

        }
        StreamWriter writer = new StreamWriter(fs);
        for (int q = 0; q < 40; q++)
        {
            for (int p = 0; p < Field.states.Count; p++)
            {
                writer.WriteLine("Run: " + q + ":"+ p);
                Console.WriteLine("____________________________________________\n"
+
                                  "____________________________________________\n" +
                                  "Run: " + q + ":" + p);
                for(int a = 0; a < 9; a++)
                {
                    //Game();
                    Game(p,a);
                }
                Console.WriteLine("--------------------------------");
                field.Reset();
                int start = random.Next(0, 2);
                int player = start;
                while (players[player].MakeMove(start, writer))
                {
                    player = player == 0 ? 1 : 0;
                }
                writer.WriteLine();
                writer.Flush();
                field.Reset();
                player = start;
                while (players[player].MakeMove(false, start))
                {
                    player = player == 0 ? 1 : 0;
                    writer.WriteLine(field.getCombination(0));
                    writer.Flush();
                }

                writer.WriteLine(field.getCombination(0));
                writer.WriteLine();
                writer.WriteLine();
                writer.Flush();
            }
        }
    }
    static void Game()
    {
        field.Reset();
        int start = random.Next(0,2);
        int p = start;

        Console.WriteLine();
        Console.WriteLine("Player " + (p + 1));
        while (players[p].MakeMove(true, start))
        {
            p = p == 0 ? 1 : 0;
            Console.WriteLine("Player " + (p + 1));
            Console.WriteLine(Field.Printable(field.getCombination(0)));
            //Console.ReadLine();
        }
    }
    static void Game(int g, int action)
    {
        Console.WriteLine(Field.states[g].getContent(0) + "\t" + action);
        field.Reset();
        field.setFirstState(g);
        if (field.currentState[0].Aim != 0)
        {
            return;
        }
        int start = Field.CharCount(field.getCombination(0), '1') >=
Field.CharCount(field.getCombination(0), '2') ? 1 : 0;
        int p = start;
        Console.WriteLine();
        Console.WriteLine("First Player " + (p + 1));
        if (!players[p].MakeCertainMove(action))
        {
            Console.WriteLine("Returning");
            return;
        }
        if (field.currentState[0].Aim != 0)
        {
            return;
        }
        p = p == 0 ? 1 : 0;

        Console.WriteLine();
        Console.WriteLine("Player " + (p + 1));
        while (players[p].MakeMove(true, start))
        {
            p = p == 0 ? 1 : 0;
            Console.WriteLine("Player " + (p + 1));
            Console.WriteLine(Field.Printable(field.getCombination(0)));
            //Console.ReadLine();
        }
        //Console.WriteLine(Field.Printable(field.getCombination(0)) + "\n------------------
--------------");
    }

    static void PrintSolution()
    {
        field.Reset();
        int start = random.Next(0, 2);
        int p = start;
        while (players[p].MakeMove(false, start))
        {
            p = p == 0 ? 1 : 0;
            Console.WriteLine(Field.Printable(field.getCombination(0)));
            for (int a = 0; a < 9; a++)
            {
                Console.WriteLine(AI.Q[field.GetCurrentState(p + 1).getIndex()][a]);
            }
        }
        Console.WriteLine(Field.Printable(field.getCombination(0)) + "\n-------------------
------------");
    }
    static void PlayManually()
    {
        field.Reset();
        int team = 1;
        while(field.getReward() == 0)
        {
            Console.WriteLine(Field.Printable(field.getCombination(0)));
            int action = int.Parse(Console.ReadLine());
            field.SetPosition(action, team, false);
            /*Console.WriteLine(field.currentState[0].getContent(0));
            Console.WriteLine("--------------------------------------------------------------");
            Console.WriteLine(field.currentState[1].getContent(0));
            */
            for(int a = 0; a < 9; a++)
            {
                Console.WriteLine(AI.Q[field.currentState[0].getIndex()][a]);
            }
            team = team % 2;
            team++;
        }
    }
    static void PrintQ()
    {
        for(int index = 0; index < AI.Q.Length; index++)
        {
            Console.WriteLine(Field.Printable(Field.states[index].getContent(0)));
            for(int action = 0; action < 9; action++)
            {
                string v1 = "" + AI.Q[index][action];
                if (v1 == "-200")
                    v1 = "-";

                Console.Write(v1 +"\t");
            }
            Console.WriteLine();
        }
    }
    static void ConfigureLastActions()
    {
        for(int a = 0; a < field.finishedStates.Count; a++)
        {
            State firstState = field.finishedStates[a];
            field.setFirstState(firstState.getIndex());

            for (int p = 0; p < 9; p++)
            {
                string s = "";
                string n = firstState.getContent(0);
                for(int x = 0; x < n.Length; x++)
                {
                    if( x == p)
                    {
                        s += "0";
                        continue;
                    }
                    s += n[x];
                }
                State state = field.FindState(s);
                if (state == null||state.Aim != 0)
                    continue;
                string content = state.getContent(0);
                field.setFirstState(state.getIndex());
                if(Field.CharCount(content, '1') > Field.CharCount(content, '2'))
                {
                    int player = 1;
                    for(int action = 0; action < 9; action++)
                    {
                        players[player].MakeCertainMove(action);
                        field.setFirstState(state.getIndex());
                    }
                }
                else if (Field.CharCount(content, '1') < Field.CharCount(content, '2'))
                {
                    int player = 0;
                    for (int action = 0; action < 9; action++)
                    {
                        players[player].MakeCertainMove(action);
                        field.setFirstState(state.getIndex());
                    }
                }
                else
                {

                    for (int player = 0; player < 2; player++)
                    {
                        for (int action = 0; action < 9; action++)
                        {
                            players[player].MakeCertainMove(action);
                            field.setFirstState(state.getIndex());
                        }
                    }
                }

                for (int q = 0; q < 9; q++)
                {
                    string t = "";
                    for (int x = 0; x < n.Length; x++)
                    {
                        if (x == p)
                        {
                            t += "0";
                            continue;
                        }
                        t += s[x];
                        State state2 = field.FindState(s);
                        if (state2 == null||state2.Aim != 0)
                            continue;
                        string content2 = state2.getContent(0);
                        field.setFirstState(state2.getIndex());
                        if (Field.CharCount(content2, '1') > Field.CharCount(content2, '2'))
                        {
                            int player = 1;
                            for (int action = 0; action < 9; action++)
                            {
                                players[player].MakeCertainMove(action);
                                field.setFirstState(state2.getIndex());
                            }
                        }
                        else if (Field.CharCount(content2, '1') < Field.CharCount(content2, '2'))
                        {
                            int player = 0;
                            for (int action = 0; action < 9; action++)
                            {
                                players[player].MakeCertainMove(action);
                                field.setFirstState(state2.getIndex());
                            }
                        }
                        else
                        {

                            for (int player = 0; player < 2; player++)
                            {
                                for (int action = 0; action < 9; action++)
                                {
                                    players[player].MakeCertainMove(action);
                                    field.setFirstState(state2.getIndex());
                                }
                            }
                        }

                    }
                }
                Console.WriteLine(a + ": " + p );
            }
        }
        StreamWriter sw = new StreamWriter(QStream);
        for(int a = 0; a < Field.states.Count; a++)
        {
            for(int b = 0; b < 9; b++)
            {
                sw.WriteLine(AI.Q[a][b]);
                sw.Flush();
            }
        }
    }
    static void SaveQ()
    {
        if(QStream != null)
        {
            QStream.Close();
        }
        Console.WriteLine("Where?");
        string path = Console.ReadLine();
        if (path == "__")
            path = "D:/QData.txt";

        QStream = new FileStream(path, FileMode.Create);
        StreamWriter sw = new StreamWriter(QStream);

        for(int state = 0; state < Field.states.Count; state++)
        {
            for(int a = 0; a < 9; a++)
            {
                sw.WriteLine(AI.Q[state][a]);
                sw.Flush();
            }
        }
    }
  }
}
```

State.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TicTacToeQLearning
{
    class State
    {
        protected string content;
        protected int index;
        public State[][] nextState;
        public int Aim;

        public State(string c, int i)
        {
            Aim = 0;
            nextState = new State[2][];
            for(int a = 0; a < 2; a++)
            {
                nextState[a] = new State[9];
            }
            content = c;
            index = i;
        }

        public void setContent(string newContent)
        {
            content = newContent;
        }
        public string getContent(int view)
        {
            string result = "";
            if(view == 2)
            {
                for(int a = 0; a < content.Length; a++)
                {
                    char c = content[a];
                    if (c == '1')
                        c = '2';
                    else if (c == '2')
                        c = '1';

                    result += c;
                }
            }
            else
            {
                result = content;
            }
            return result;
        }

        public void setIndex(int newIndex)
        {
            index = newIndex;
        }
        public int getIndex()
        {
            return index;
        }
    }
}
```