Konzolové piškvorky

Jako můj /projekt jsem si tedy vybral vytvořit piškvorky v programovacím jazyce C#. Chtěl jsem se naučit vytvářet jednoduché hry a pracovat s objektově orientovaným programováním v C#.
Projekt jsem začal tím, že jsem si vytvořil nový projekt v programovacím prostředí Visual Studio a začal psát kód. Nejprve jsem vytvořil třídu Board, která reprezentuje hrací pole piškvorek. Tato třída má v sobě uloženou informaci o stavu hrací desky, tj. které pozice jsou volné a které jsou obsazené.
Dále jsem vytvořil třídu Hrac, která reprezentuje hráče. Každý hráč má jméno a symbol, kterým hraje (obvykle X nebo O).
Poté jsem vytvořil třídu Piškvorky, která řídí celou hru. Tato třída má v sobě uložené informace o hráčích, hracím poli a aktuálním hráči, který je na řadě. Tato třída také řídí celý průběh hry.
Celý kód je napsán v C# a využívá objektově orientovaného programování. Hra je navržena tak, aby byla interaktivní a aby uživatel mohl hrát sám proti sobě nebo proti jinému hráči. Kód obsahuje kontrolu platnosti tahů a kontrolu výhry. Pokud je hra ukončena, program zobrazí vítěze nebo remízu.
Pro vylepšení projektu do budoucna bych si mohl představit vytvořit robotického soupeře, který by mohl hrát proti uživateli. Robotický soupeř by mohl používat různé strategie a být tak náročnější na poražení. Také bych mohl přemístit celý kód do Forms, což by zlepšilo uživatelské rozhraní a umožnilo by hrát hru více hráčům na jednom počítači.
V současné době je projekt funkční a umožňuje hrát piškvorky proti jinému hráči nebo proti sobě samotnému.
Pro další rozvoj tohoto projektu by bylo zajímavé zvážit vytvoření robotického soupeře, aby hráči mohli hrát i sami proti počítači.
Jako první krok by bylo potřeba implementovat umělou inteligenci, která by dokázala hrát piškvorky proti člověku. Existuje mnoho přístupů, které by se daly použít, například algoritmy Minimax nebo Alpha-Beta pruning. Tyto algoritmy by dokázaly vypočítat nejlepší tah pro robota v každé situaci, přičemž by braly v úvahu tahy hráče a snažily se minimalizovat jeho šanci na výhru.
Dalším krokem by bylo vizualizovat robota na herní desce a přidat mu nějaké grafické prvky, aby hráči viděli, jaký tah robot provedl. Může se jednat například o animovanou postavu, která se pohybuje po herní desce, nebo o jednoduchý symbol, který se zobrazuje na místě, kde robot provedl svůj tah.
Kromě toho by bylo zajímavé přemístit celý program do grafického rozhraní pomocí Forms, což by umožnilo hráčům hrát piškvorky v okně aplikace a měnit nastavení hry pomocí formulářů. To by umožnilo uživatelům snadněji ovládat hru a mohli by si také přidat další funkce, jako například možnost změnit barvu symbolů hráčů nebo pozadí herní desky.
Celkově by rozšíření projektu o robota jako soupeře a přemístění do Forms umožnilo hráčům hrát piškvorky v různých režimech - proti sobě samým, proti jinému hráči, nebo proti robotovi. To by výrazně rozšířilo možnosti a zábavu, kterou by projekt mohl nabídnout.

Jeden z možných dalších kroků by mohl být vytvořit pokročilejší algoritmy pro umělou inteligenci, které by mohly být použity k vytvoření počítačového soupeře. To by umožnilo hrát piškvorky i v případě, že nemáte nikoho, s kým byste mohli hrát, nebo pokud byste chtěli trénovat proti silnému protivníkovi. Existuje mnoho různých algoritmů, které by mohly být použity k vytvoření počítačového soupeře, jako je například Monte Carlo Tree Search.
Další možností by bylo vytvořit online verzi piškvorek, kterou by mohli hrát lidé z celého světa. Toto by mohlo být provedeno pomocí webového rozhraní, které by umožňovalo hráčům vytvářet účty a hrát proti sobě. Tento projekt by vyžadoval zvládnutí mnoha nových technologií, jako je například vývoj webových aplikací, práce s databázemi a zabezpečení aplikace.
Pokud byste chtěli dále rozvíjet projekt, můžete také zvážit, jak přidat nové funkce a prvky do hry. Například byste mohli přidat možnost hrát více hráčů najednou, nebo přidat nové herní módy, které by nabízely hráčům více možností a větší výzvy.
Další možností by bylo převést projekt do vícevrstvé architektury, kde by byly odděleny jednotlivé vrstvy aplikace, jako je například vrstva prezentace, vrstva aplikační logiky a vrstva přístupu k datům. Toto by mohlo pomoci zlepšit modularitu projektu a usnadnit další vývoj a úpravy.
Pokud byste chtěli zlepšit uživatelské rozhraní hry, můžete také zvážit použití grafického uživatelského rozhraní (GUI). Toto by mohlo zahrnovat vytvoření vlastních ikon a obrázků pro hrací desku a hráče, nebo možnost přizpůsobit vzhled aplikace podle preference uživatele.
Další možností, jak vylepšit tuto hru, by bylo přidat možnost ukládání a načítání her. Ukládání by umožnilo hráčům přerušit hru a později ji pokračovat od místa, kde skončili. Načítání by umožnilo hráčům znovu hrát předchozí hry, které si uložili.
Další vylepšení by mohlo zahrnovat vizuální prvky, jako jsou animace, zvukové efekty a různé barvy a styl pro hrací desku a symboly hráčů. Tyto prvky by mohly zlepšit celkovou zábavu a přitažlivost hry.
Mimo to by bylo možné přidat více funkcí a herních režimů, jako například možnost hrát na větší desce, změnit pravidla pro výhru nebo přidat další symboly pro hráče. Tyto prvky by mohly poskytnout hráčům více zábavy a výzvy.
Vytvoření robotického soupeře by bylo velkým krokem vpřed pro tuto hru. Když se hráč ocitne sám proti počítači, bude mít větší výzvu a možnost zlepšovat své dovednosti. Robotický soupeř by mohl mít různé úrovně obtížnosti, od začátečníka až po pokročilého hráče, což by umožnilo hráčům najít si svou vlastní úroveň výzvy.
mnoho vylepšení by mohlo být provedeno pro tuto hru, aby se stala více zábavnou a přitažlivou pro hráče. Tyto inovace by mohly zlepšit celkovou zábavu a přínos této hry pro uživatele.

Použité technologie:

```
public bool CheckWin(int row, int col, char symbol)
{

    for (int i = 0; i < Size; i++)
    {
        if (board[row, i] != symbol) break;
        if (i == Size - 1) return true;
    }

    for (int i = 0; i < Size; i++)
    {
        if (board[i, col] != symbol) break;
        if (i == Size - 1) return true;
    }

    if (row == col)
    {
        for (int i = 0; i < Size; i++)
        {
            if (board[i, i] != symbol) break;
            if (i == Size - 1) return true;
        }
    }

    if (row + col == Size - 1)
    {
        for (int i = 0; i < Size; i++)
        {
            if (board[i, Size - 1 - i] != symbol) break;
            if (i == Size - 1) return true;
        }
    }
    return false;
}
```

Metoda pro kontrolování výhry

```
2 references
public class Board
{
    public char[,] board;
    16 references
    public int Size { get; }

    1 reference
    public Board(int size)
    {
        Size = size;
        board = new char[Size, Size];
    }
```

Zadávání velikosti pole

```
1 reference
public bool Set(int row, int col, char symbol)
{
    if (board[row, col] != '\0')
    {
        Console.WriteLine("toto místo je již použité vyber si jiné");
        return false;
    }
    board[row, col] = symbol;
    return true;
}
```

Ochrana proti vybrání si již zabraného pole