Dokumentace Quantik

Uživatelská příručka

Cíl projektu

V rámci projektu byla vytvořena implementace hry Quantik. Tato implementace umožňuje uživateli hrát proti jinému uživateli na stejném počítači či proti vytvořeném počítačovém protihráči. Aplikace je vytvořena ve dvou verzích s odlišným uživatelským rozhraním. Prvním je jednoduché konzolové rozhraní a druhé je pokročilé grafické rozhraní, které nabízí přívětivější ovládání hry a přehlednější zobrazení aktuálního stavu.

Pravidla hry

Quantik je čistě strategická hra. Hraje se s herním plánem 4x4 políčka. Herní plán obsahuje 4 sloupce, 4 řádky a je rozdělen na 4 čtverce o velikosti 2x2. Každý hráč má k dispozici 8 figurek, kde figurky mají 4 různé tvary a každý tvar má hráč k dispozici tedy právě dvakrát. Cílem hry je být první hráč, který umístí čtvrtý unikátní tvar do řádku, sloupce či čtverce.

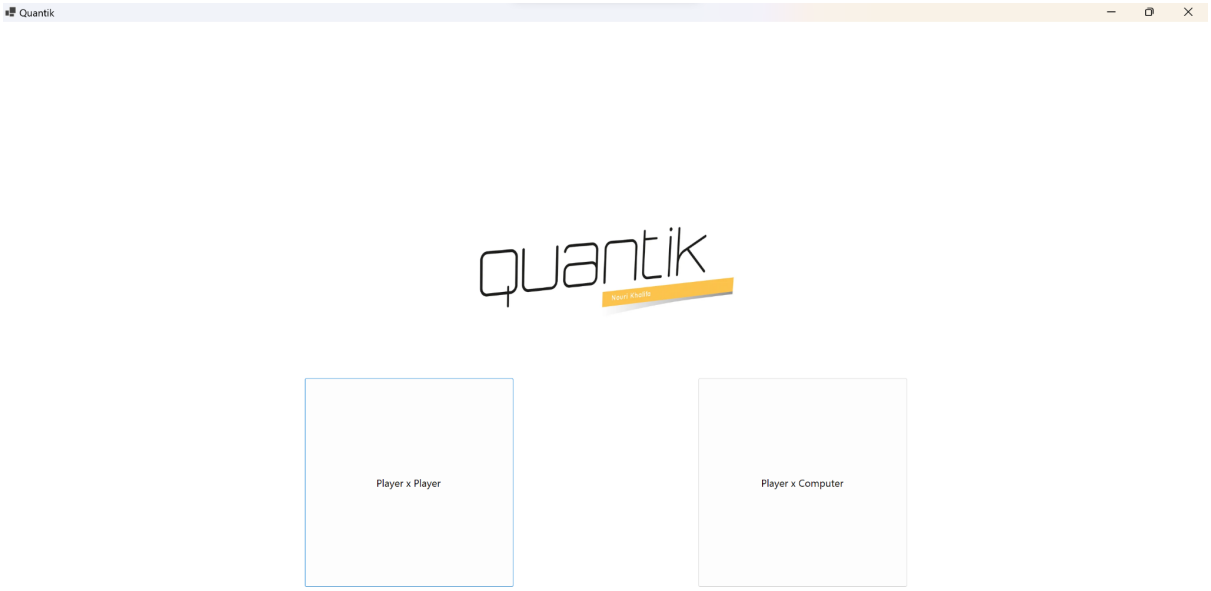
Hráči se střídají a během tahu hráč, který je zrovna na tahu umístí jeden z jejích dílků na hrací plán. Je zakázáno umístit tvar do řádku, sloupce či čtverce, ve kterém už tento tvar je umístěn oponentem. Pokud tento tvar byl umístěn hráčem, který je zrovna na tahu, tak ten může umístit i druhý stejný do řádku, sloupce či čtverce. Hráč který jako první umístí čtvrtý unikátní tvar do řádku, sloupce či čtverce okamžitě vyhrává bez ohledu na to kdo umístil zbývající tvary.

Konzolová aplikace

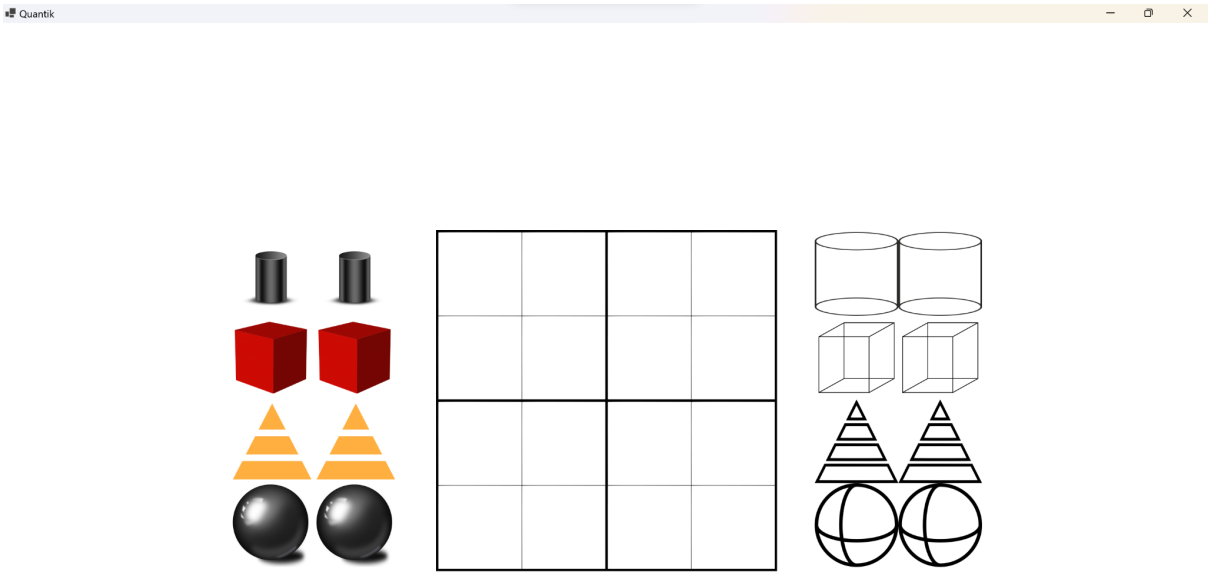
Ve složce Release_console se nachází soubor Quantic_console.exe. Po jeho spuštění je uživatel dotázán zda chce hrát proti počítači či proti druhému uživateli. Poté je dotázán na souřadnice, na které chce umístit jeho figurku. První souřadnice vybere řádek a druhá souřadnice vybere sloupec. Souřadnice jsou počítány z levého horního rohu od nuly. Dále jsou uživateli zobrazeny dostupné figurky a je dotázán na tvar, který umístěná figurka má mít, načež dochází ke kontrole tahu a pokud je tah možné zahrát, je hra aktualizována a uživateli je zobrazen aktuální stav hrací plochy. Během tahu počítače jsou vypsány pomocné informace, což odpovídá tomu, že tato aplikace je zamýšlena spíše pro testování. Pomocná informace je například ohodnocení tahu. Pokud některý z hráčů vyhraje je o tom uživatel informován a hra je ukončena.

Aplikace s grafickým rozhraním

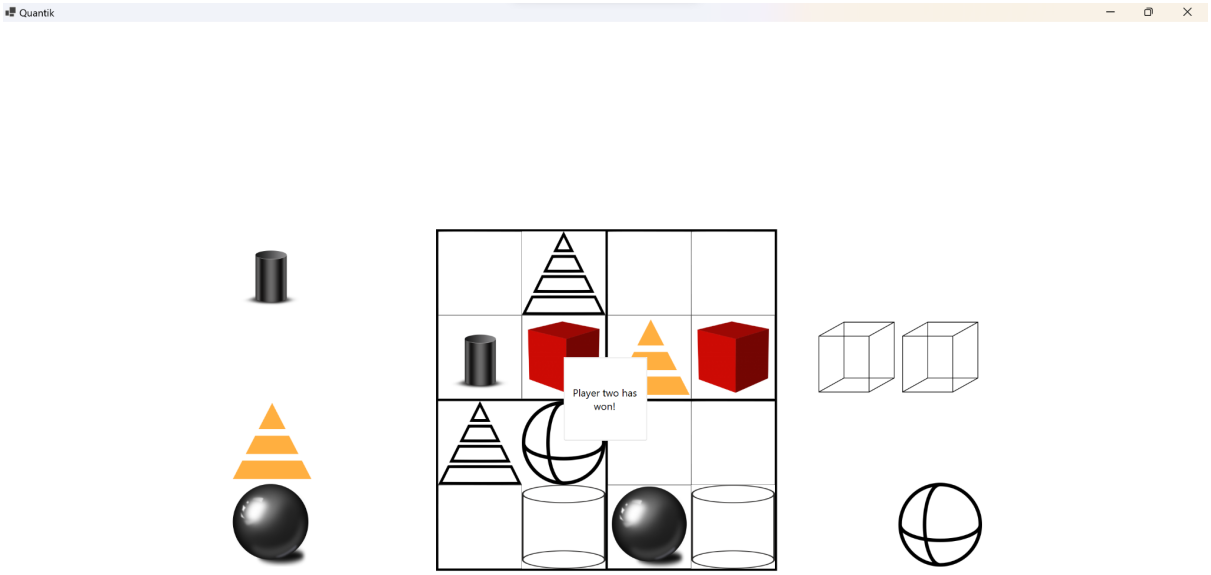
Ve složce Release_console se nachází soubor Quantic_gui.exe. Po jeho spuštění jsou uživateli zobrazeny dvě tlačítka viz obrázek 1. Prvním tlačítkem se spustí hra hráče proti hráči a druhým hra hráče proti počítači. Počáteční situace je zachycena na obrázku 2. První na tahu je hráč, který ovládá barevné figurky. Pro umístění figurky stačí jednoduše kliknutím vybrat požadovanou figurku a poté kliknutím vybrat políčko pro její umístění. Pokud jeden z hráčů vyhraje je uživateli zobrazeno tlačítko s informací kdo vyhrál a po jeho stisknutí je hra znovu spuštěna viz obrázky 3 a 4.



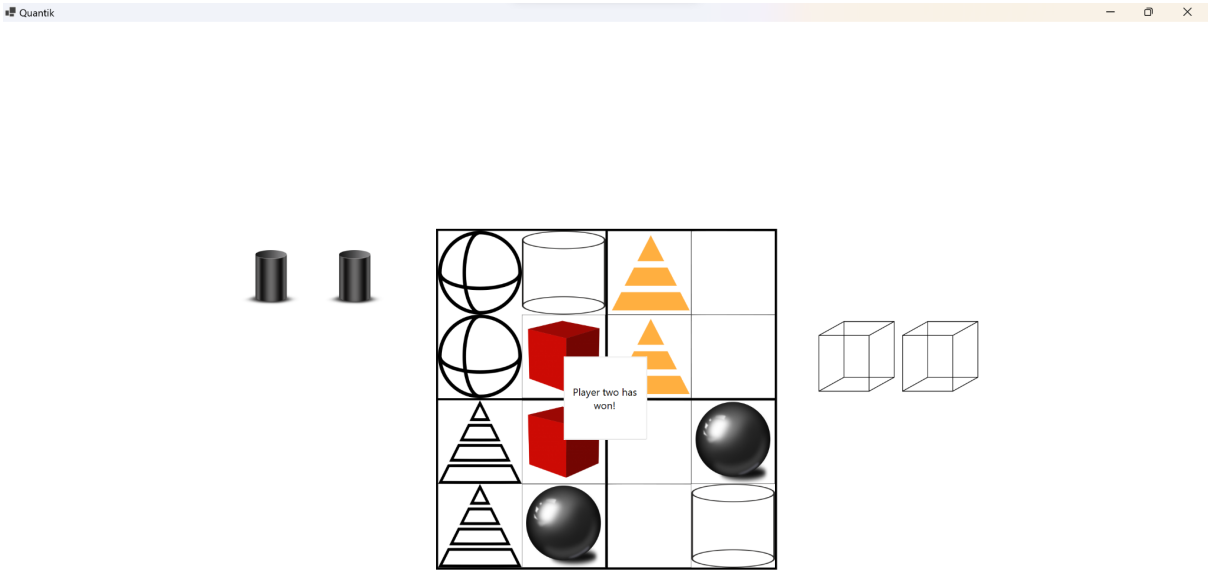
Obrázek 1 - Úvodní obrazovka



Obrázek 2 - Počáteční stav hry



Obrázek 3 - Hráč 2 vyhrál, hra je ukončena a kliknutím na tlačítko bude spuštěna odznova



Obrázek 4 - Hráč 1 je na tahu, ale nemá žádný možný tah a tedy prohrává.

Programátorská dokumentace

Počítačový protihráč prohledává stavový prostor hry pomocí algoritmu minimax do hloubky 4 tahy, což reprezentuje čtvrtinu hry. Pro efektivní prohledávání je používáno vícevláknové programování, konkrétně technologie Task. Minimax byl také obohacen o alfa beta prořezávání pro větší výkon.

Třída Game

Třída Game je hlavní třídou, která se stará o herní dynamiku. V rámci konstruktoru této třídy je vytvořen hrací plán, hráči a v případě přání uživatele je nastaven druhý hráč na počítačového protivníka.

public void SetComputerPlayer(Logger logger)
Nastaví druhého hráče tak aby byl ovládán počítačem. Uvažované tahy tohoto počítačového protivníka budou zaznamenávány pomocí poskytnuté reference na třídu logger a umožňují tedy oddělit skutečně zahrané tahy od těch, které byly pouze uvažovány.

public void GetPlayers()
Zjistí od uživatele pomocí konzole zda chce hrát proti druhému uživateli či proti počítači.

public void PlayGame(BoardViewer viewer)
V konzolové verzi se tato metoda stará o průběh hry - zjišťuje tahy od hráčů a pomocí herní logiky (třída GameLogic) kontroluje tahy a aktualizuje herní plán

Třída GameLogic

Tato třída obsahuje pravidla hry a je tedy takovým rozhodčím. Obsahuje metody pro kontrolu validity tahů, zahrání tahů a kontroly zda některý z hráčů vyhrál (umístěním posledního tvaru do řádku, sloupce či oblasti 2x2) či zda některý z hráčů prohrál (nemá žádné tahy). Udržuje pro oba hráče seznam jejich možných tahů.

public static void GetAllPossibleMovesAtTheStartOfTheGame(Dictionary<Piece.ShapeType, HashSet<Coordinates>> possibleMoves)
Tato metoda naplní danou Dictionary všemi 64 tahy, které má na začátku hry hráč k dispozici.

public List<Move> GetCurrentPossibleMoves(Player player,HashSet<Piece.ShapeType>? shapes)
Tato metoda vrátí seznam aktuálně dostupné tahy pro zadaného hráče s množinou zadaných tvarů.

public HashSet<Coordinates> GetPossibleMovesWithSpecifiedShape(Player player, Piece.ShapeType shape)
Tato metoda vrátí množinu souřadnic, na které lze umístit zadaný tvar daným hráčem.

public static HashSet<Piece.ShapeType> GetPlayersShapes(Player player)
Tato metoda vrátí množinu tvarů, které má hráč k dispozici.

public static bool CheckMove(Move? move, Board board, Player player)
Tato metoda zkontroluje, zda daný hráč může zahrát daný tah na daném plánku.

private static bool CheckTurn(Move move, Piece.PlayerID player)
Zkontroluje, zda je hráč právě na tahu.

private static bool CheckEmptiness(Move move, Board board)
Zkontroluje, zda políčko na které chce hráč umístit figurku je prázdné.

private static bool CheckColumn(Move move, Board board)
Zkontroluje, zda protivník nemá stejný tvar v sloupci tahu.

private static bool CheckRow(Move move, Board board)
Zkontroluje, zda protivník nemá stejný tvar v řádku tahu.

private static bool CheckArea(Move move, Board board)
Zkontroluje, zda protivník nemá stejný tvar v 2x2 oblasti tahu.

private static bool CheckAvaibility(Move move, Player player)
Zkontroluje, zda má hráč tento tvar k dispozici.

public void MakeMove(Move move, Board? board, Player player)
Provede tah a aktualizuje herní plán a dostupné figurky hráče.

public void RemoveNotValidMoves(Move move, Player player)
Odstraní tahy, které se po zadaném tahu již nedají zahrát.

private static void RemoveShape(Piece.ShapeType shape, Dictionary<Piece.ShapeType, HashSet<Coordinates>> possibilities, Player player)
Odstraní tahy hráče s daným tvarem, pokud již zahrál svou druhou figurku tohoto tvaru.

private static void RemoveRow(Piece.ShapeType shape, Dictionary<Piece.ShapeType, HashSet<Coordinates>> possibilities, Move move)
Odstraní protivníkovi tahy s daným tvarem ve stejném řádku.

private static void RemoveColumn(Piece.ShapeType shape, Dictionary<Piece.ShapeType, HashSet<Coordinates>> possibilities, Move move)
Odstraní protivníkovi tahy s daným tvarem ve stejném sloupci.

private static void RemoveArea(Piece.ShapeType shape, Dictionary<Piece.ShapeType, HashSet<Coordinates>> possibilities, Move move)
Odstraní protivníkovi tahy s daným tvarem ve stejné 2x2 oblasti.

private static void RemoveCoordinates(Coordinates coordinates, Dictionary<Piece.ShapeType, HashSet<Coordinates>> possibilities)
Odstraní všechny tahy na dané souřadnice. Voláno pro oba hráče.

public static bool CheckWin(Board board,Move move)
Zkontroluje, zda je daný tah vítězný.

public bool CheckLoss(Piece.PlayerID player)
Zkontroluje, zda daný hráč prohrál kvůli absenci možných tahů.

private static bool CheckColumnForWin(Move move, Board board)
Zkontroluje, zda tah doplnil sloupec čtvrtým unikátním tvarem.

private static bool CheckRowForWin(Move move, Board board)
Zkontroluje, zda tah doplnil řádek čtvrtým unikátním tvarem.

private static bool CheckAreaForWin(Move move, Board board)
Zkontroluje, zda tah doplnil 2x2 oblast čtvrtým unikátním tvarem.

Třída ArtificialPlayer

Tato třída využívá C# konstruktu Tasks, aby pomocí algoritmu minimax s alfa beta prořezáváním paralelně prohledala stavový prostor hry a nalezla nejlepší možný tah.

public override Move SelectMove(GameLogic gameLogic, Board board,Player current, Player other)
Nalezne nejlepší tah, dle pravidel gameLogic, s aktuálním stavem hracího plánu pro aktuálního hráče. Metoda zjistí použité tvary na hracím plánu a poté zavolá metodu MiniMaxDepthZero,která reprezentuje první úroveň minimaxu a liší se od ostatních, že tahy uvažuje pomocí Tasks paralelně, místo sériově.

private static HashSet<Piece.ShapeType> GetUsedShapes(Board board)
Vrátí tahy použité na herním plánu. Počítač totiž nemusí umožňovat například všechny tři nepoužité tvary, ale místo toho jeden zástupný, který je vybrán v metodě DetermineWhichShapesToConsider. Na další úrovni minimaxu jsou tyto tvary aktualizovány dle změny situace.

private static HashSet<Piece.ShapeType> DetermineWhichShapesToConsider(HashSet<Piece.ShapeType> usedShapes)
Dostane množinu tvaru vyskytujících se na herním plánku a přidá k nim ještě jeden pokud je to možné. Tyto tvary budou použity pro hledání nejlepšího tahu. Na další úrovni minimaxu jsou tyto tvary aktualizovány dle změny situace.

private async Task<MinimaxResult> MinimaxDepthZero(GameLogic gameLogic, Board board, Player currentPlayer, Player otherPlayer, HashSet<Piece.ShapeType> usedShapes, double alpha, double beta)
První krok minimaxu. V rámci tohoto kroku jsou tahy vyhodnocovány paralelně pomocí Tasks.

private void WorkerMinimax(GameLogic gameLogic, Player currentPlayer,
  Player otherPlayer, Board board, Move move, HashSet<Piece.ShapeType> usedShapes, MinimaxInfo alphaBeta, MinimaxResult result)
Provedení jednoho tahu a jeho vyhodnocení s minimaxem v rámci jednoho vlákna.

private MinimaxResult MiniMax(GameLogic gameLogic,Board board, int depth, Player currentPlayer, Player otherPlayer,HashSet<Piece.ShapeType> usedShapes, double alpha, double beta)

Minimax pro další úrovně. Větvení je provedeno sekvenčně, na jednom vlákně.

private double EvaluateMove(GameLogic gameLogic,Player currentPlayer, Player otherPlayer,
  Board board, Move move, int depth,HashSet<Piece.ShapeType> usedShapes, double alpha, double beta)
Provede daný tah, zkontroluje zda některý z hráčů vyhrál a zavolá minimax další úrovně.

private static double EvaluatePosition(GameLogic gameLogic,Player currentPlayer,Player otherPlayer)
Ohodnocení situace v největší hloubce minimaxu. Evaluační funkce bere v potaz počet možných tahů aktuálního hráče a jeho oponenta.

Další třídy

- Board
  Tato třída reprezentuje hrací plán a obsahuje 4 x 4 tabulku políček (třída Square)
- Square
  Reprezentuje jedno políčko hracího plánu. Může či nemusí obsahovat figurku.
- Coordinates
  Pomocné třída pro předávání x-ové a y-ové souřadnice pomocí reference.
- BoardViewer
  Abstraktní předek pro zobrazování stavu hry uživateli. Potomci jsou ConsoleBoardViewer a GUIBoardViewer.
- Logger
  Tato třída slouží k uložení informace o hře (zahrané a uvažované tahy, výsledek hry) do souborů ve složce logs
- MinimaxInfo, MinimaxResult
  Pomocné třídy pro minimax. MinimaxInfo uchovává informace o aktuální hodnotě alfy a bety a je synchronizovaná pomocí zamykání na úrovni 0 minimax. MinimaxResult obsahuje nejlepší tah a jeho hodnocení a je také synchronizovaný pomocí zamykání na úrovni 0 minimaxu.
- Move
  Obsahuje informace o x-ové a y-ové souřadnici, tvaru a hráči
- Piece, Placed Piece
  Obsahuje informaci o tvaru a majiteli figurky, placedPiece je používáno pro typovou kontrolu během umísťování figurek a pro zalogování konečného stavu hry.
- Player
  Abstraktní předek pro uživatele či počítačového hráče. Obsahuje metody pro výběr tahu a obsahuje seznam dostupných figurek.
- User
  Hráč hraný uživatelem. Implementuje metodu pro výběr tahu.
- Program
  Vstupní třída programu.