Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Řešení bludiště pomocí A\* algoritmu

Bc. Nikola Jačková

Semestrální práce

2021

**Obsah**

[Seznam obrázků 3](#_Toc89865410)

[Seznam zkratek 4](#_Toc89865411)

[Úvod 5](#_Toc89865412)

[1 A\* algoritmus 6](#_Toc89865413)

[1.1 Popis algoritmu 6](#_Toc89865414)

[2 Projekt NNUI1\_SemestralWork 7](#_Toc89865415)

[2.1 Labyrinth Finding Path 7](#_Toc89865416)

[2.1.1 Labyrinth 7](#_Toc89865417)

[2.1.2 Search 8](#_Toc89865418)

[2.2 Labyrinth Finding Path Tests 9](#_Toc89865419)

[2.3 Labyrinth GUI 9](#_Toc89865420)

[3 Uživatelská příručka 11](#_Toc89865421)

[3.1 Spuštění aplikace 11](#_Toc89865422)

[3.2 Ovládání aplikace 11](#_Toc89865423)

[3.2.1 Horní menu 11](#_Toc89865424)

[3.2.2 Levý panel 12](#_Toc89865425)

[3.2.3 Pravý panel 14](#_Toc89865426)

[Závěr 15](#_Toc89865427)

[Přílohy 16](#_Toc89865428)

Seznam obrázků

[Obrázek 1: Struktura projektu LabyrinthFindingPath 7](#_Toc89865385)

[Obrázek 2: Implementace algoritmu A\* 9](#_Toc89865386)

[Obrázek 3: Uživatelské prostředí aplikace 11](#_Toc89865387)

[Obrázek 4: Horní menu aplikace 12](#_Toc89865388)

[Obrázek 5: Levý panel aplikace 13](#_Toc89865389)

[Obrázek 6: Zobrazení bludiště v aplikaci 14](#_Toc89865390)

Seznam zkratek

WPF Windows Presentation Foundation

WinForms Windows Forms

Úvod

Semestrální práce z předmětu NNUI1 se zabývá programovým řešením nalezení nejefektivnější cesty bludištěm ze zadaného bodu A do zadaného bodu B. Nalezení a ohodnocení této cesty využívá algoritmus A\*. Řešení bylo realizováno pomocí vyššího programovacího jazyka C#. Vytvořený program by měl umožňovat načítat obrázek černobílého bludiště a zadat počáteční a koncový bod. Jeho výstupem je nalezení efektivní cesty z počátečního bodu do koncového.

# A\* algoritmus

## Popis algoritmu

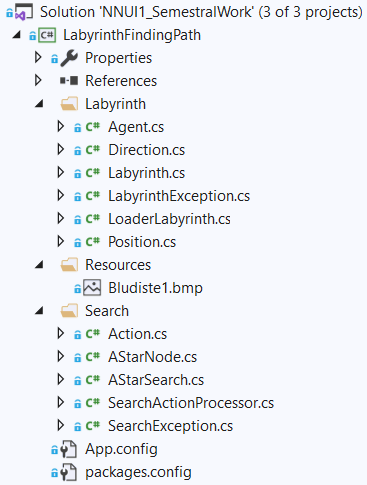
A\* algoritmus je algoritmus, který se využívá pro vyhledávání optimálních cest v kladně ohodnoceném grafu. Optimální cestou se rozumí nejkratší, nejrychlejší, nejlevnější apod. cesta v závislosti na ohodnocení hran grafu. A\* algoritmus pro hledání cesty využívá funkci *f(x)* skládající se z dalších dvou funkcí: *f(x) = g(x) + h(x)*. Funkce *g(x)* představuje vzdálenost mezi aktuálním a počátečním vrcholem. Funkce *h(x)* představuje heuristickou funkci odhadující délku nejkratší, nejrychlejší, nejlevnější apod. cesty z aktuálního vrcholu do cíle. Efektivita algoritmu je tedy závislá na volbě heuristické funkce. Samotný algoritmus pak ke svému procházení využívá prioritní frontu. Čím je hodnota *f(x)* menší, tím má větší prioritu. V každém kroku je pak daný vrchol odebrán z fronty a jsou vypočítány hodnoty pro jeho sousední uzly.

# Projekt NNUI1\_SemestralWork

Program, zabývající se řešení nalezení cesty bludištěm, s názvem NNUI1\_SemestralWork je rozdělen na tři další projekty: Labyrinth Finding Path, Labyrinth Finding Path Tests a Labyrinth GUI. V celém programu je použita angličtina.

## Labyrinth Finding Path

Projekt Labyrinth Finding Path se dělí na další dvě složky: Labyrinth a Search. Tento projekt také obsahuje obrázek příkladu bludiště. Struktura souborů projektu je zobrazena obrázku níže.



Obrázek : Struktura projektu LabyrinthFindingPath

### Labyrinth

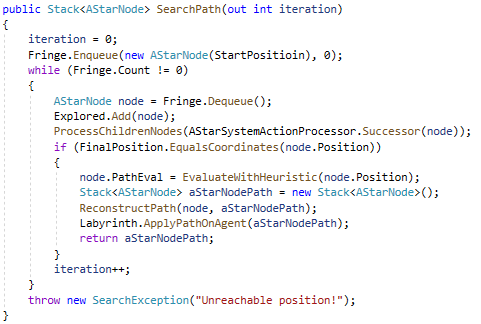
Třídy v této složce se zabývají načítáním a realizací bludiště. Bludiště je v programu reprezentováno dvourozměrným polem typu boolean. Černá barva v bludišti má hodnotu FALSE a bíla barva má hodnotu TRUE. Načítání bludiště z obrázku do pole je realizováno třídou LoaderLabyrinth. Hlavní třída Labyrinth reprezentuje instanci bludiště. Každé bludiště má svého agenta, který je reprezentován třídou Agent. Ta obsahuje informace o pozici agenta, o nákladech jednotlivých akcí a událost reagující na změnu pozice agenta. Po nalezení optimální cesty algoritmem A\* jsou na instanci agenta, resp. na jeho pozici postupně aplikovány všechny kroky. Třída Position obsahuje metody pro změnu směru a pozice. Složka ještě obsahuje třídu LabyrinthException, reprezentující vyvolané výjimky související s bludištěm, a výčet Direction, který představuje jednotlivé směry (North, South, East, West).

### Search

Složky Search obsahuje třídy, které zajišťují nalezení řešení optimální cesty pomocí algoritmu A\*. Pro spuštění algoritmu je nutné znát bludiště, počáteční a cílový bod a přípustné akce. Přípustnými akcemi jsou otočení o 90° doprava a doleva, tato akce je oceněna hodnotou 2, otočení o 180°, tato akce je oceněna hodnotou 3 a krok dopředu, který je ohodnocen hodnotou 5. Při aplikování otáčecí akce se zároveň provede krok vpřed v daném směru. Akce otočení a provedení kroku vpřed byly sloučeny z důvodu snížení času vykonávání algoritmu přibližně o polovinu. Jako heuristická funkce byla použita vzdálenost do cíle v manhattanské metrice. Alternativou by mohla být vzdálenost do cíle v Euklidovské metrice. Tato funkce by byla vhodná při možnosti pohybů diagonálně.

Samotný algoritmus pak využívá jednu prioritní frontu. Jelikož .NET Framework verze 4.8 nenabízí implementovanou prioritní frontu, byl využit NuGet balíček Optimized Priority Queue. A\* algoritmus implementovaný ve třídě AStarSearch pak pochází jednotlivé prvky dle jejich priority reprezentované třídou AStarNode. Z každého prvku se vytvoří nové prvky aplikováním jednotlivých platných akcí a vypočítá se jejich celková cena. Ověření a aplikování akce na prvky zajišťuje třída SearchActionProcessor. Akce jsou definovány ve výčtu Action. Nově vzniklé prvky jsou vloženy do prioritní fronty. Prvky, které již byly zpracovány se ukládají do seznamu, aby se zajistilo, že tento prvek již nebude znovu zpracováván. Algoritmus končí, pokud je nalezen cílový stav, nebo pokud je prioritní fronta vyprázdněna, pak daný problém nemá řešení. Složka Search ještě navíc obsahuje třídu SeachException sloužící pro zachycování výjimek při hledání řešení bludiště.

Na následujícím obrázku je zobrazena stěžejní metoda celé aplikace. Tato metoda se nachází ve třídě AStarSearch. Má jeden výstupní parametr, který představuje počet iterací algoritmu, tzn. počet prvků, které algoritmus zpracoval. Metoda vrací proměnou typu *Stack<AStarNode>*, tzn. zásobník s prvky cesty. Po nalezení cílového stavu je cesta bludiště zkonstruována a aplikována na instanci agenta v bludišti. Při neřešitelném zadání je vyvolaná výjimka třídy SearchException.



Obrázek : Implementace algoritmu A\*

## Labyrinth Finding Path Tests

Tento projekt obsahuje jednu třídu PositionTests s jednotkovými testy testující metody související se změnou pozice (pohyb vpřed a otočení). Všechny testy mají úspěšný výsledek.

## Labyrinth GUI

Uživatelské prostředí aplikace bylo tvořeno pomocí architektury uživatelského rozhraní WPF. Alternativou k této architektuře je knihovna tříd WinForms. WPF byla zvolena, protože je modernější a také z důvodu rozšíření znalostí autorky, jelikož při výuce na Fakultě elektrotechniky a informatiky se využívá pouze knihovna WinForms.

Grafické uživatelské prostředí se skládá z jednoho okna Labyrinth Path Finding. Prostředí využívá ke své funkčnosti tři třídy. Třída GuiUtility zajišťuje pohyb a vzhled startovního a cílového bodu a celé cesty. Pro reprezentaci obou bodů se využívá třída PointOfInterest, která obsahuje samotný tvar bodu (čtverec) a jeho pozici v bludišti. Třída FindingPath pak obsahuje mj. instanci bludiště, vyhledávacího algoritmu a načtenou cestu. Slouží pro zajištění hledání řešení v bludišti.

Uživatelské prostředí dále ještě obsahuje dvě částečné třídy. Třída MainWindow obsahuje metody reagující na uživatelské vstupy a třída App slouží jako výchozí vstupní bod aplikace.

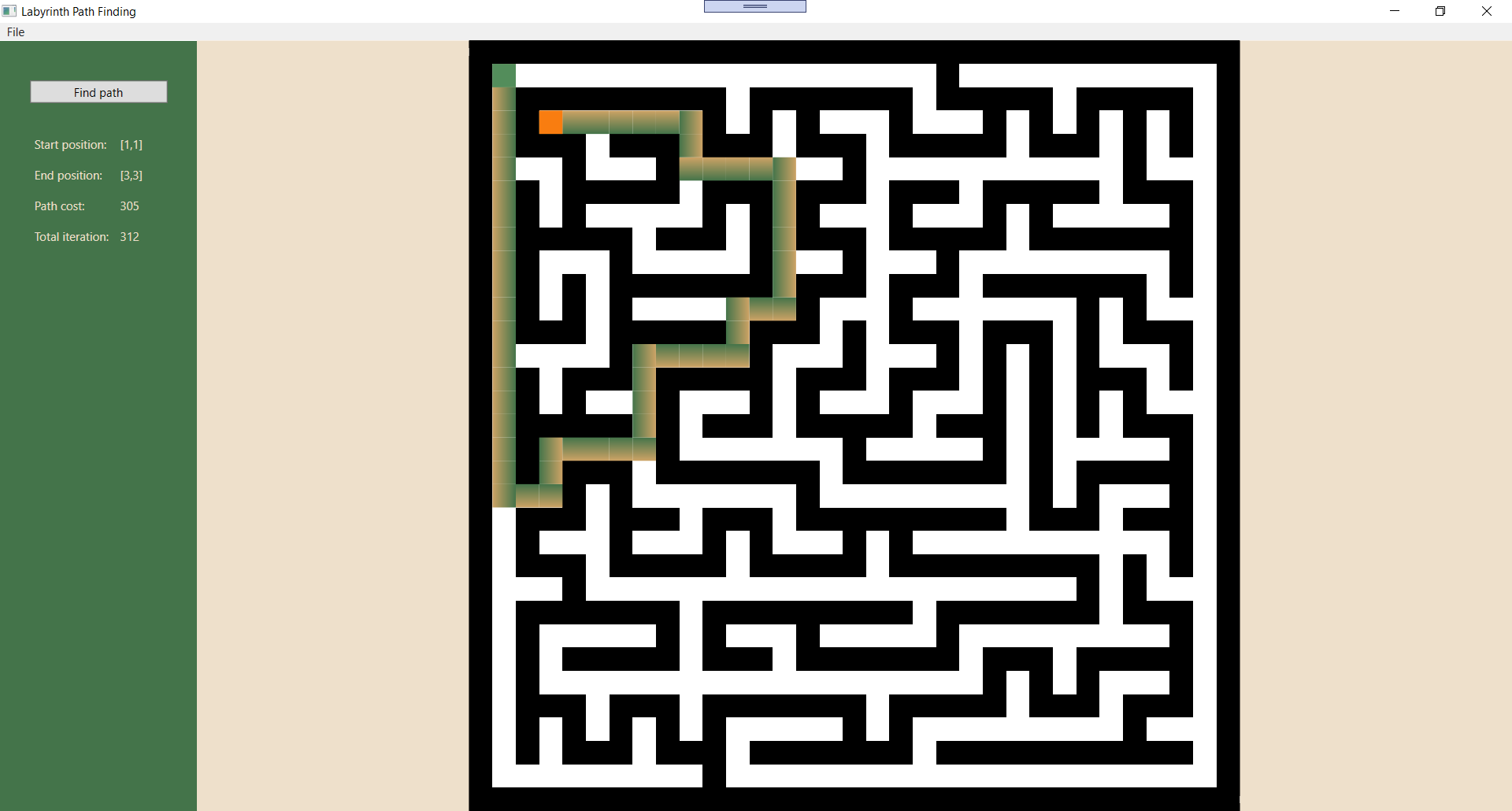
# Uživatelská příručka

## Spuštění aplikace

Aplikaci lze spustit otevřením souboru LabyrinthGUI.exe, který je součástí přílohy A.

## Ovládání aplikace

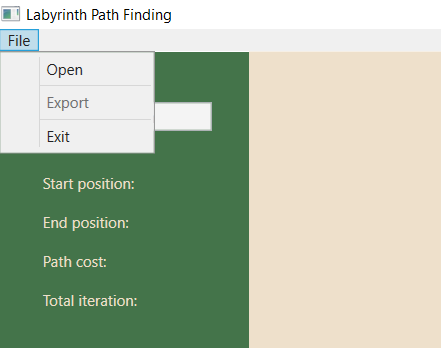
Aplikace se skládá z jednoho okna, které je rozděleno na tři části: horní menu, levý panel a pravý panel.



Obrázek : Uživatelské prostředí aplikace

### Horní menu

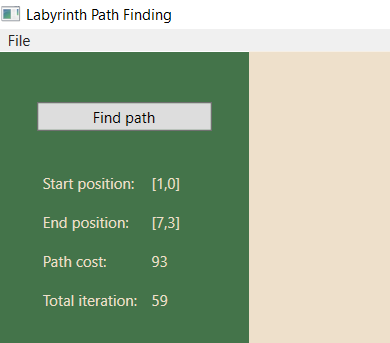
Horní menu aplikace obsahuje tři položky. Položka Open otevře dialog pro výběr obrázku bludiště. Povolené typy obrázku jsou .bmp a .png. Položka Export exportuje nalezenou cestu bludiště do textového souboru *Path.txt* do adresáře aplikace. Pro povolení této funkce je nutné načíst bludiště, počáteční a koncový bod a spustit algoritmus hledání. Položka Exit ukončí aplikaci.



Obrázek : Horní menu aplikace

### Levý panel

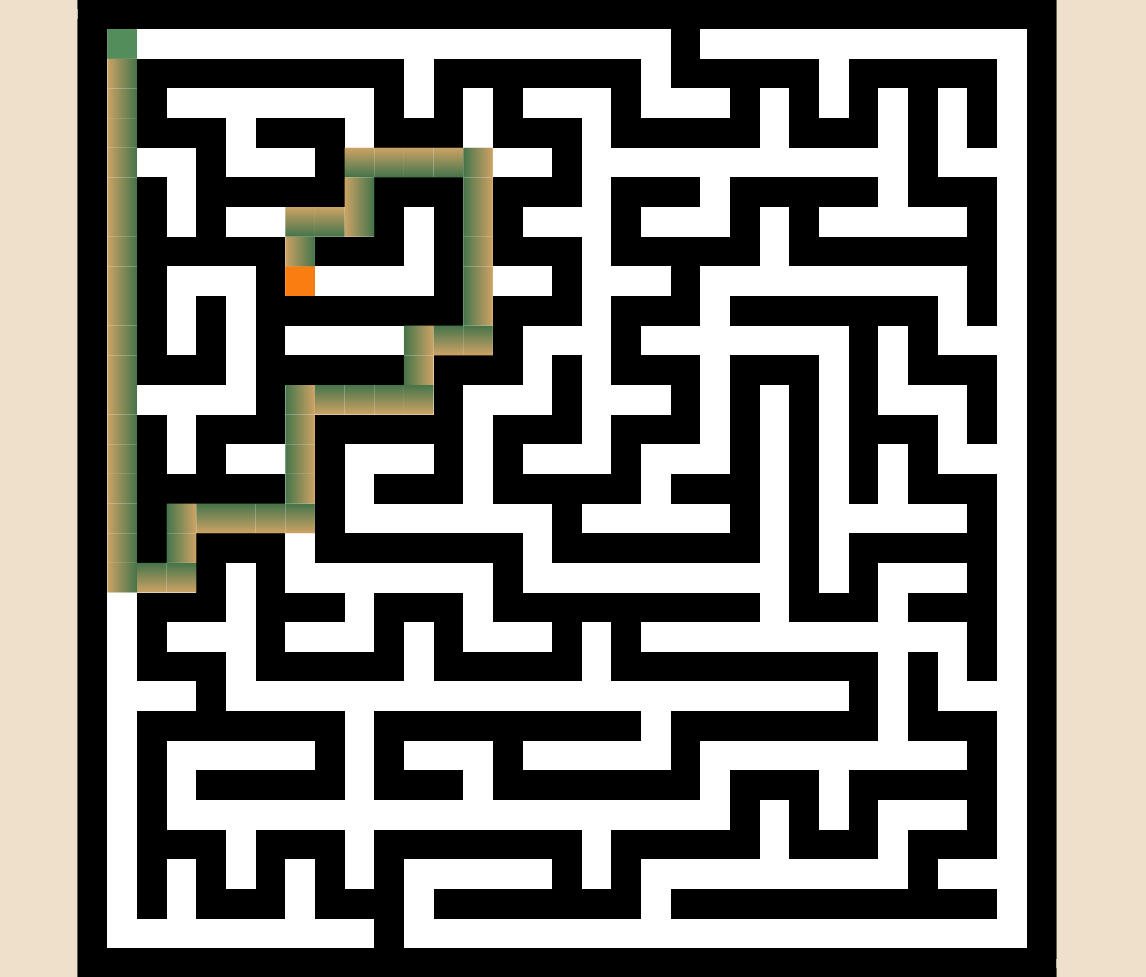
Levý panel obsahuje tlačítko pro spuštění hledání řešení a několik statických textů. Tlačítko Find path spustí algoritmus A\*. Pro povolení tlačítka je nutné mít načtené bludiště a zadaný počáteční a koncový bod. Panel také zobrazuje souřadnice obou bodů. Po nalezení cesty je zde také zobrazena cena cesty a celkový počet iterací.



Obrázek : Levý panel aplikace

### Pravý panel

Pravý panel zobrazuje obrázek bludiště a vykresluje do něj nalezenou cestu. Do bludiště lze zadat počáteční bod levým kliknutím myši a koncový bod pravým kliknutím myši. Následně se zpřístupní tlačítko Find path a v bludišti se vykreslí nalezená cesta. Pro vykreslení cesty byla použita výplň s barevným přechodem z důvodu názorné ukázky otáčení se při průchodu bludištěm. Oba body mohou být opakovaně zadávány.



Obrázek : Zobrazení bludiště v aplikaci

Závěr

Semestrální práce řeší nalezení optimální cesty bludištěm pomocí algoritmu A\*. Aplikace umožňuje vyřešit černobílé bludiště s jakoukoli platnou pozicí počátečního a koncového bodu. Čas potřebný k vyřešení úlohy závisí na velikosti a složitosti bludiště a na umístění obou bodů. Aplikace nabízí také vizualizaci nalezeného řešení a exportování celé cesty do textového souboru.

Přílohy

[Příloha A – NNUI1\_SemestralWork\_JackovaNikola.ZIP 17](#_Toc89865429)

Příloha – NNUI1\_SemestralWork\_JackovaNikola.ZIP

Přiložený .zip soubor obsahuje následující složky:

* Aplikace – obsahuje spustitelný soubor,
* Dokumentace – obsahuje soubor s dokumentací této práce,
* Příklady bludiště– obsahuje soubory s dvěma příklady bludiště,
* Zdrojové soubory – obsahuje zdrojové kódy aplikace.