

Vyhledávání nejkratší cesty v reálném terénu

Autor: Petr Holec
Vedoucí práce: Jiří Jeníček

Abstract

The work deals with problems of artificial intelligence and three-dimensional environment rendering. It will focus on the interpretation of the environment for artificial intelligence units and the algorithms needed to search the shortest path. The result is application for indoor navigation.

Úvod

Aplikace pro navigaci v silničním provozu jsou dnes běžnou věcí, pro interiéry jako jsou nemocnice, školy nebo muzea už nejsou tak časté. Úloha je komplexnější a je předmětem tohoto projektu.

Cíle

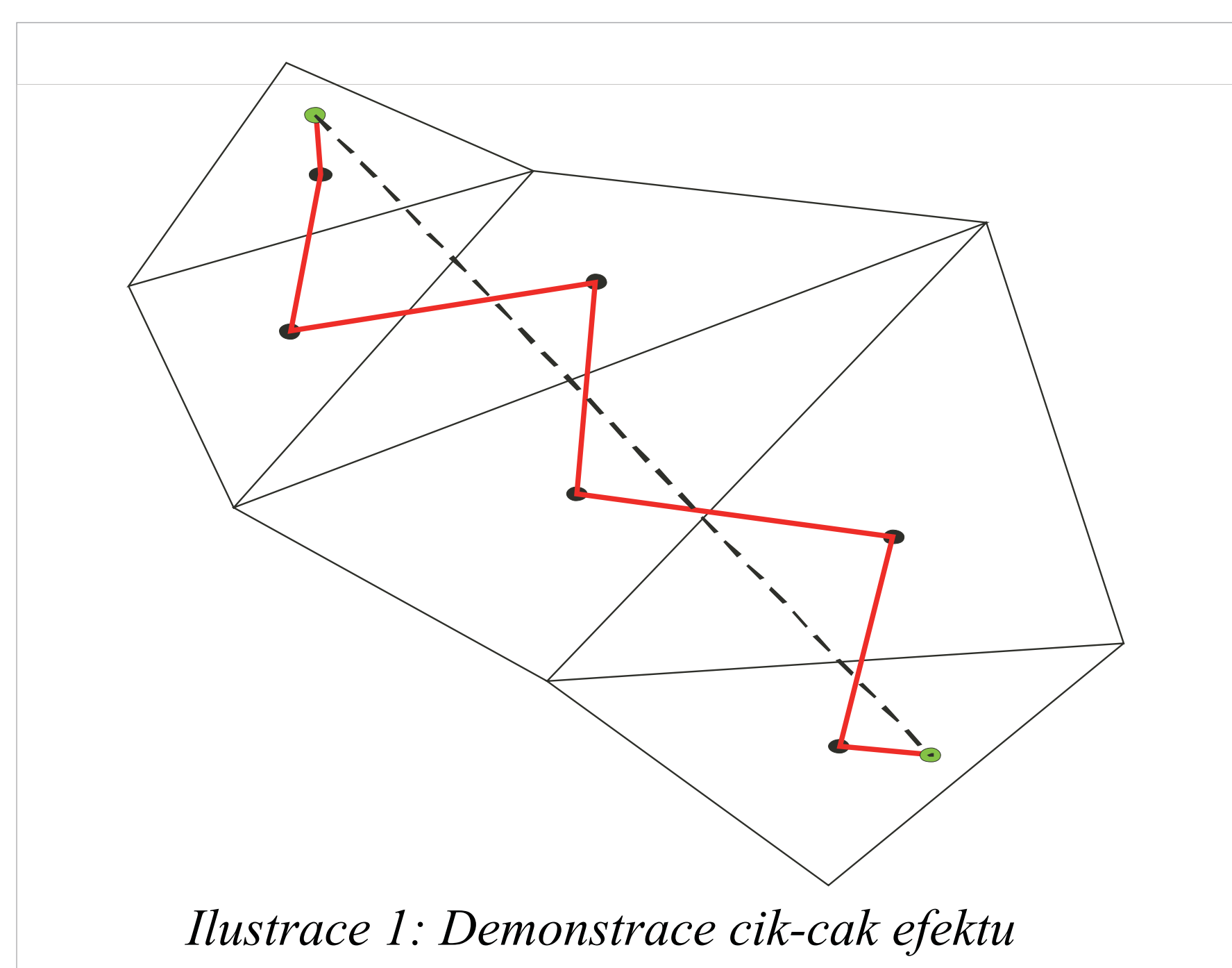
Hlavním úkolem je tedy vytvoření aplikace pro efektivní a interaktivní navigaci komplexy. Hlavními součástmi je volný pohyb po budovách a vyhledávání tras.

3 kroky k nalezení cesty

Pro interpretaci prostředí se využívá *navigation mesh*. Jedná se o soustavu navzájem spojených konvexních polygonů a jako taková v sobě skrývá potřebné detaily o prostředí.

Krok 1.

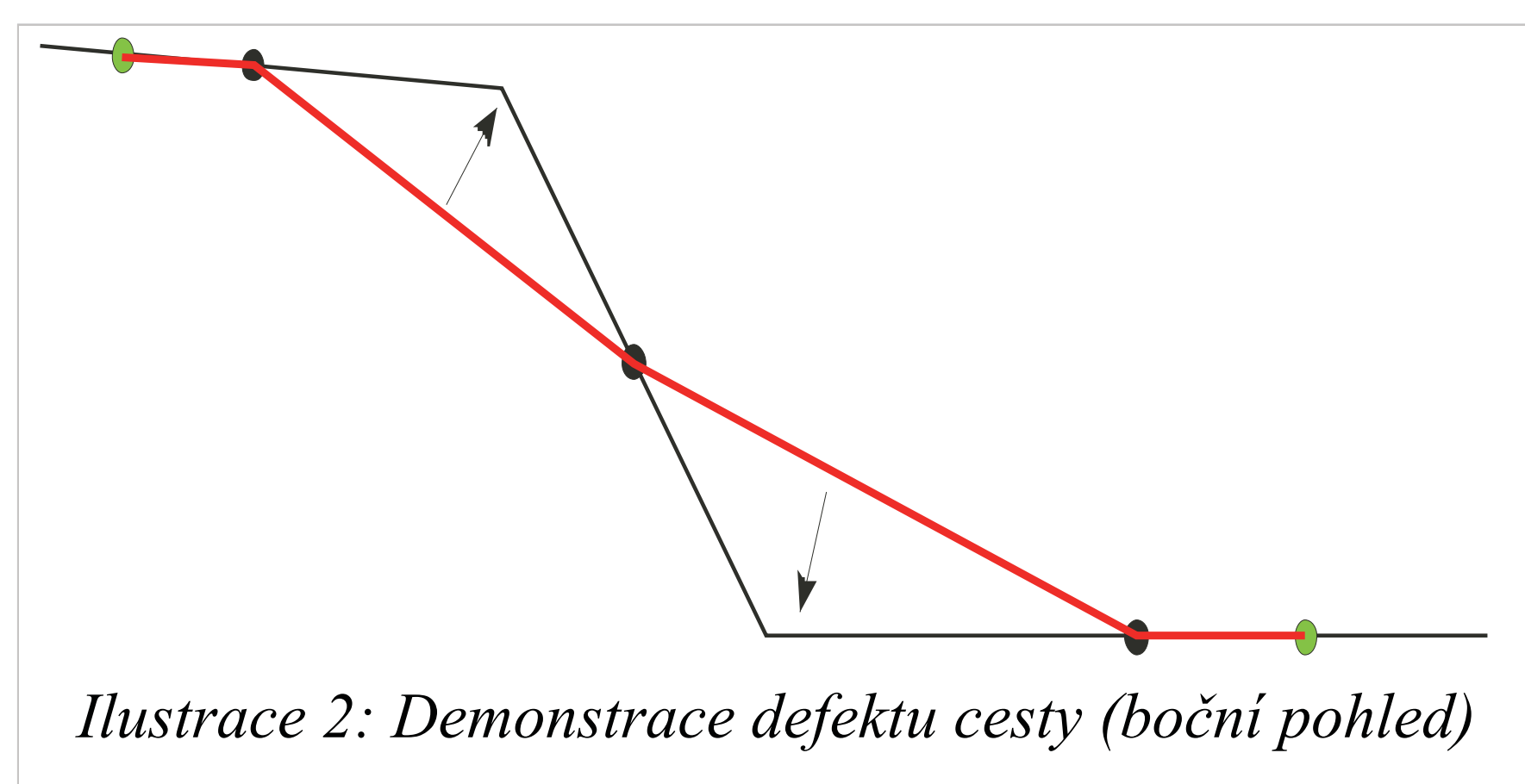
Vyhledání nejkratší cesty přes grafový Dijkstrův algoritmus (nebo jeho verze A*). Výsledkem prvního kroku je *hrubá* cesta. Je sice nejkratší v rámci grafu, ale dochází k cik-cak efektu.



Ilustrace 1: Demonstrace cik-cak efektu

Krok 2.

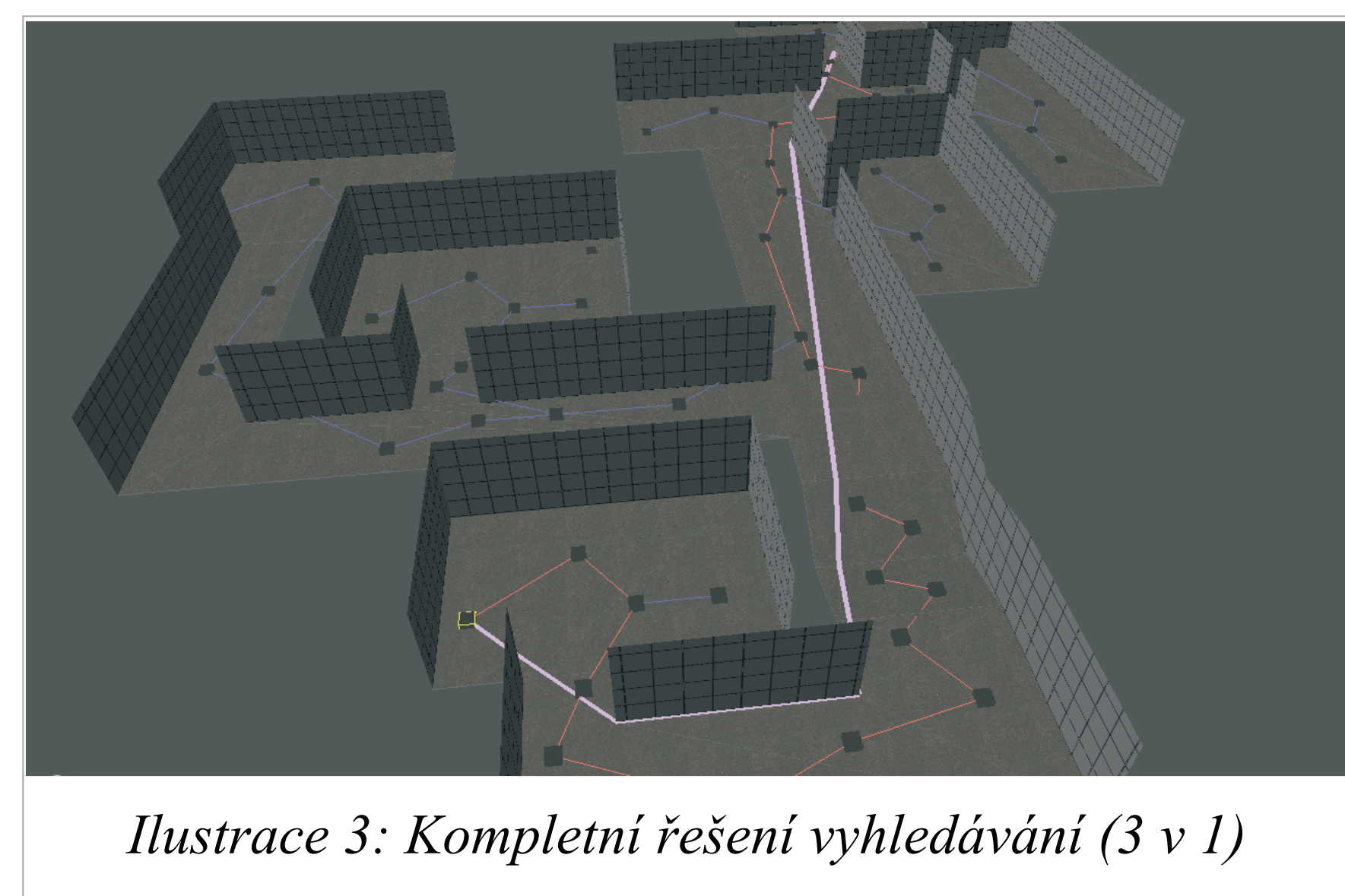
Tento krok odstraňuje cik-cak efekt a zajistí optimální cestu v rámci kanálu. Kanál je prostor v navigation mesh vytyčený prvním krokem a vyhledání probíhá pomocí *funnel algoritmu* (trychtýřový alg.).



Ilustrace 2: Demonstrace defektu cesty (boční pohled)

Krok 3.

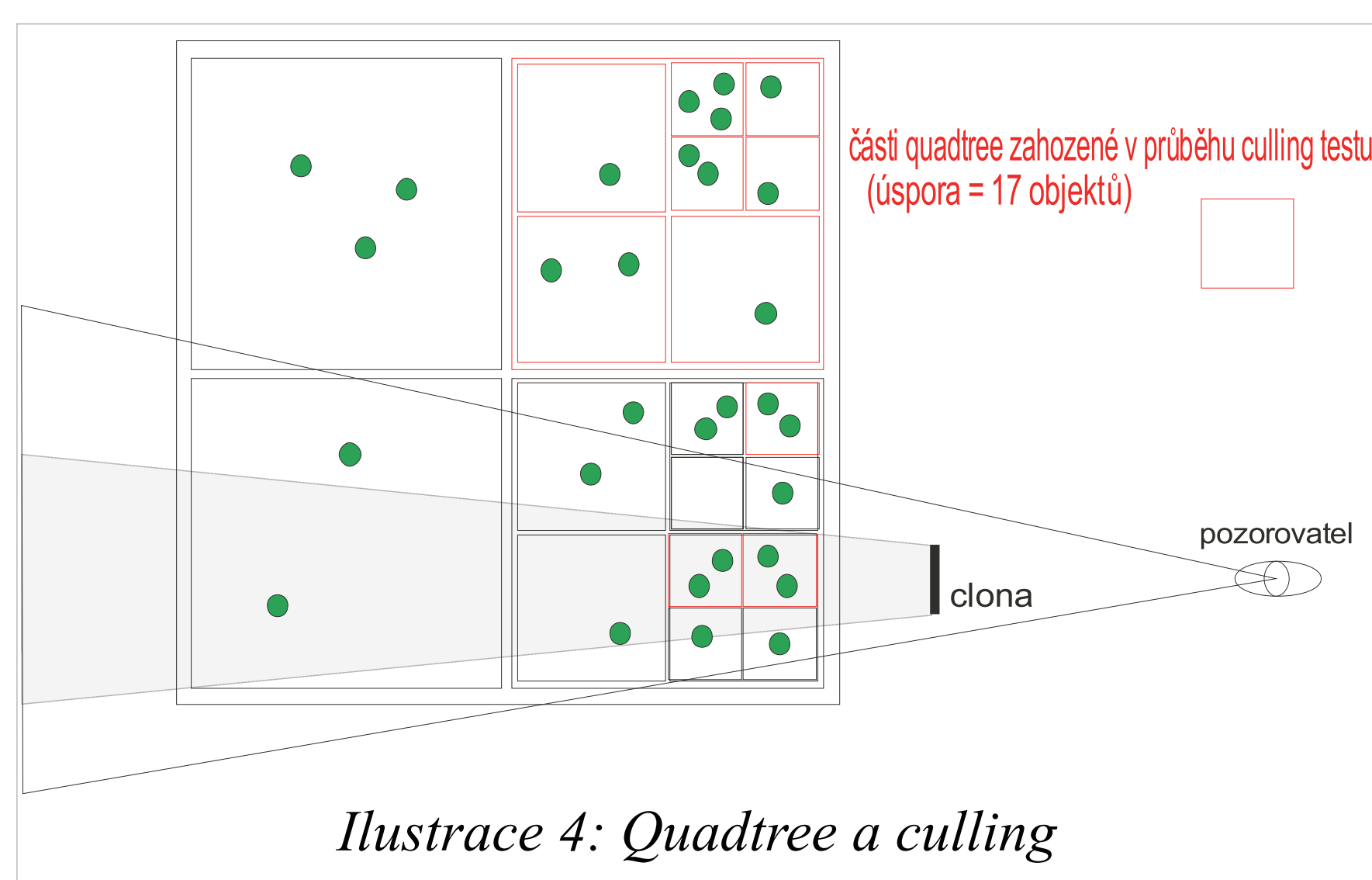
Posledním krokem je korekce cesty. Dochází k chybám, kdy cesta vede *vzduchem* a nebo prochází *zemí*. Tento defekt se odstraňuje promítnutím cesty do navigation mesh (resp. kanálu).



Ilustrace 3: Kompletní řešení vyhledávání (3 v 1)

Rychlejší vykreslování

Významnou částí grafického engine jsou optimalizace pro vykreslování. Často se *zahazují* objekty (resp. skupiny), které nejsou viditelné (PVS) a to ze dvou hlavních důvodů – nejsou v zorném poli, nebo jsou zakryté clonou (view frustum a occlusion culling). Objekty se seskupují pomocí quadtree (resp. octree).



Ilustrace 4: Quadtree a culling

Rychlejší vyhledávání

Pomocí víceúrovňového vyhledávání lze dosáhnout eliminování některých uzlů z vyhledávacího algoritmu. Nejprve se nalezne cesta na nejabstraktnější úrovni, postupně se konkretizuje.

Výsledky

Na následující hierarchii je graficky znázorněno, které části jsou v jaké fázi vývoje – **částečně**, **plně** a **vůbec** neimplementováno.

- **Načtení grafických dat**
 - načtení ze souboru
- **vygenerování vstupní navigace**
- **Interaktivní průchod**
 - ovládání klávesnice / myš
 - detekce kolizí a fyzikální zákony
- **Interaktivní prostředí**
 - grafické uživatelské rozhraní (GUI)
 - použití „Point of Interest“
 - přepínání různých komplexů
- **Vyhledávání nejkratší cesty**
 - Dijkstrův algoritmus
 - Funnel algoritmus
- **Systém harmonogramu**
 - komunikace přes internet a xml
 - vyhledávání podle osoby a času
- **Optimalizace**
 - optimalizace vyhledávání
 - optimalizace vykreslování

Závěr

Nyní je aplikace v jazyku C# a platformu .NET s použitím API OpenGL. Pro lepší přenositelnost je v plánu přepis do jazyku Java. V úvahu připadá i portování na mobilní telefony.

Reference

- [1] Demyen, D., *Efficient Triangulation-Based Pathfinding*, (2007).
- [2] Fixing Pathfinding Once and For All, <http://www.ai-blog.net/archives/000152.html>