Pannon Egyetem

Műszaki Informatikai Kar

[Tanszék neve]

[Szak neve]

**SZAK-/DIPLOMADOLGOZAT**

**[Dolgozat címe]**

**[Név]**

Témavezető: [Témavezető neve]

Külső/belső konzulens: [Konzulens neve]

[évszám]

Témakiírás

A szkennelt formában megkapott témakiírás beillesztése a dolgozatba.



Hallgatói nyilatkozat

Alulírott <<név>> hallgató (Neptun kód: <<SAJÁT NEPTUN KÓD>>) kijelentem, és a dolgozat feltöltésével egyidejűleg nyilatkozom, hogy a <<dolgozatcím>> című <<záródolgozatot/szakdolgozatot/diplomadolgozatot>> (a továbbiakban: dolgozat) a Pannon Egyetem Műszaki Informatikai Kar <<szervezeti egységében (tanszékén)>> készítettem a <<végzettség>> oklevél megszerzése érdekében.

Kijelentem, hogy a dolgozatban csak a megadott és hivatkozott forrásokat használtam fel, és ezekre a vonatkozó idézési szabályok szerint hivatkoztam.

Nyilatkozom, hogy a dolgozat érdemi része saját szellemi alkotásom eredménye, és azt más intézményben, szakon, vagy felsőfokú képesítés megszerzésére nem nyújtottam be. Tudomásul veszem, hogy a plágium vagy szerzői jogsértés esetén a dolgozatom elutasításra kerülhet, és ellenem fegyelmi eljárás indulhat. Tudomásul veszem továbbá, hogy szerzői jogsértés esetén az Egyetem jogosult a dolgozat elérhetőségét korlátozni, valamint eltávolítani a dokumentumot a dolgozatok tárolására szolgáló, a témát vezető szervezeti egység által meghatározott elektronikus zárt rendszerből.

Tudomásul veszem továbbá, hogy a Pannon Egyetem a dolgozat eredményeit saját céljaira eltérő írásbeli megállapodás hiányában a Pannon Egyetem Szellemi Tulajdon Kezelési Szabályzatában foglaltaknak megfelelően szabadon felhasználhatja.

Nyilatkozom, hogy a dolgozat elkészítése során mesterséges intelligencia eszközöket *használtam /nem használtam[[1]](#footnote-1).*

Nyilatkozom, hogy a dolgozat elkészítése során az alábbi táblázatban feltüntetett mesterséges intelligencia eszközöket kizárólag a kutatási, illetve fejlesztési feladat támogatására használtam fel, az érdemi munka, elemzés és következtetések teljes mértékben saját szellemi alkotásomat képezik.

Példa a táblázat kitöltésére:

|  |  |  |  |
| --- | --- | --- | --- |
| **Alkalmazott technológia** | **Alkalmazás módja** | **Előállított tartalom** | **MI használat aránya** |
| GPT-4o (OpenAI) | szöveges összefoglaló generálása | 2.2 fejezet | 80% |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Dátum: Veszprém/Nagykanizsa/Zalaegerszeg, 20.. <<hónap neve>> <<nap>>.

*<<hallgató neve>>*

Témavezetői nyilatkozat

Alulírott <<témavezető neve>> témavezető kijelentem, hogy a <<dolgozatcím>> című dolgozatot *<<*hallgató neve*>>* a Pannon Egyetem <<tanszék neve>>én készítette <<végzettség>> végzettség megszerzése érdekében.

Kijelentem, hogy a dolgozat védésre bocsátását engedélyezem.

Dátum: Veszprém/Nagykanizsa/Zalaegerszeg, 20.. <<hónap neve>> <<nap>>.

*<<témavezető neve>>*

Köszönetnyilvánítás

A hallgató köszönetet nyilvánít mindazoknak, akiktől (elméleti, gyakorlati, erkölcsi stb.) segítséget kapott.

Tartalmi összefoglaló

Tartalmi összefoglaló magyarul. Az összefoglalónak tartalmaznia kell (rövid, velős és összefüggő megfogalmazásban) a következőket:

* téma megnevezése,
* megoldott feladat megfogalmazása,
* megoldási mód,
* elért eredmények,
* kulcsszavak (4-6 darab)
* terjedelme nem lehet több 1 A4-es oldalnál.

Az összefoglalót magyar és angol nyelven kell készíteni. Sorrendben a dolgozat nyelvével megegyező kerül előrébb. A cím Title stílusú, formázása: Times New Roman, nagybetű, 14 pt, félkövér, középre igazított; az összefoglaló Normál stílusú, formázása: Times New Roman, 12 pt, sorkizárt, 1.5-ös sortávolság.

**Kulcsszavak:** [4-6 kulcsszó felsorolása, vesszővel elválasztva]

Abstract

Abstract in English

**Keywords:** [list 4-6 keywords]

Tartalomjegyzék

[Jelölésjegyzék 8](#_Toc99107193)

[1. Fejezet 9](#_Toc99107194)

[1.1. Alfejezet 9](#_Toc99107195)

[1.1.1. Alfejezet 9](#_Toc99107196)

[2. Új fejezet 10](#_Toc99107197)

[2.1. Új alfejezet 10](#_Toc99107198)

[3. Instrukciók 11](#_Toc99107199)

[Irodalomjegyzék 12](#_Toc99107200)

[Mellékletek 13](#_Toc99107201)

[Ábrajegyzék 14](#_Toc99107202)

[Táblázatjegyzék 15](#_Toc99107203)

Jelölésjegyzék

|  |  |
| --- | --- |
| AI: | Artificial Intelligence (Mesterséges Intelligencia) |
| GPU: | Graphical Processing Unit (Grafikus Processzor / Grafikus Feldolgozó Egység) |
| API: | Application Programming Interface (Alkalmazásprogramozási Felület) |
| CPU: | Central Processing Unit (Központi Feldolgozó Egység / Processzor) |
| GUI: | Graphical User Interface (Grafikus Felhasználói Felület) |
| HCI: | Human Computer Interaction (Ember-gép kapcsolat) |
| CIS: | Cognitive Information System (Kognitív információs rendszer) |

# A\* útkereső algoritmus és vizualizálása

Feladatom egy A\* útkereső algoritmus megírása és grafikus ábrázolása volt. Ezt a Python

programozási nyelven valósítottam meg, a Raylib programozási könyvtárral (Python kötéssel).

A Python egy nagyon magas szintű programozási nyelv, ez gyorssá és egyszerűvé tette a program megírását. Grafikus ábrázoláshoz a Raylib könyvtárat használtam, ami eredetileg egy C könyvtár, de készült hozzá Python kötés, lehetővé téve a nyelvvel való használatát. Egy rendkívül összetett ámde kezdőbarát motor, amiben sok előre elkészített funkció található, de hagy helyet a saját megoldásoknak, implementációknak. Nyíltforráskódú, 2D, 3D-s programok/játékok készítéséhez tökéletes, ezért esett a választásom a Raylibre.

A képen képernyőkép, Téglalap, tervezés, Betűtípus látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

*1. ábra: Raylib python verziójának a logója*

Ezek mellett rengeteg példa és segítség elérhető a használatához, a bevezető dolgoktól a haladó technikákig.

Viszont annyiban eltér más játékmotoroktól, hogy nincsen grafikus kezelőfelület, így minden a kódban történik, és ha akarunk ilyenekkel élni, magunknak kell ezeket implementálni.

## Útkereső algoritmus megvalósítása

Ehhez a bemutató programhoz az A\* algoritmust választottam, mint útkereső algoritmus.

Az A\* algoritmus egy útvonal kereső és gráfkereső algoritmus, amelynek rengeteg alkalmazási módszere van sok területen. Ezek a játékfejlesztés, robotika és navigáció csak hogy említsünk egy párat. Az A\* egy informált keresési algoritmus amely útvonal költségeket/súlyokat és heurisztikát használ a legrövidebb/legolcsóbb útvonal megtalálására, ezt úgy becsülve, hogy az adott állapot milyen közel van a célállapottól. Ez a súly egy érték, amit minden csomóponthoz/állaponthoz/Nodehoz kiszámít ezzel a módszerrel:

***f(n)= g(n) + h(n)***

ahol:

* ***g(n) =*** A kezdőponttól a jelenlegi állapotig megtett út költsége
* ***h(n) =*** A jelenlegi állapottól a célpontig hátralévő út *becsült* költsége
* ***f(n) =*** A jelenlegi állapot teljes költsége

Az algoritmus mindig a legkisebb ***f(n)*** költségű állapotot vizsgálja meg következőként.

A ***h(n)*** heurisztikus függvényünk problémától függő. Ha két állapot geometriai távolságát szeretnénk tekinteni, akkor használhatjuk az Euklideszi távolságot:

De bármilyen tetszőleges számítást használhatunk, feladatnak megfelelően.

Az algoritmus lépései:

1. A kezdőpont felvétele és a nyílt listához hozzáadása(Legelőször és egyszer hajtódik végre)
2. A nyílt listából (kiterjesztett állapotok) megvizsgáljuk és kivesszük a legolcsóbb ***f(n)*** költségű állapotot
3. Ha ez az állapot a célállapot, akkor kilép a ciklusból
4. Ha nem, akkor kiterjesztjük a jelenlegi állapotot és megvizsgáljuk a szomszédjait
5. Kiszámítjuk mindegyikre az ***f(n), g(n)*** és ***h(n)*** értékeket
6. Ha egyszer már meglátogattunk egy szomszédos állapotot, de a jelenlegi állapotból kisebb a költsége, akkor frissítjük az értékeit
7. Addig ismételjük ameddig a jelenlegi állapot megegyezik a célállapottal vagy kiürül a nyílt lista

Nyílt lista: Potenciális következő lépések listája/sora, aminek első eleme a legolcsóbb mind közül.

A képen szöveg, képernyőkép, Betűtípus, szoftver látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

*2. ábra: Részlet a költségszámítás megvalósításáról Python kódban*

Az A\* algoritmus előnyei:

* Garantáltan megtalálja a legoptimálisabb utat, feltéve, ha jó heurisztikával használjuk
* Rengeteg alkalmazása van
* Hatékony, mert a heurisztikával szűkítjük a keresési teret

És hátrányai:

* Sok állapotot kell tárolnia akárhogy is, így sok memóriát igényel
* Ha rossz heurisztikát adunk meg, lassú lehet
* Bizonyos feladatokhoz nehéz lehet heurisztikát, állapot értéket megadni

Ezeket a pontokat észbentartva választottam ki az A\*-t. Viszont ezen belül a feladat megoldásához két módszert gondoltam megpróbálni.

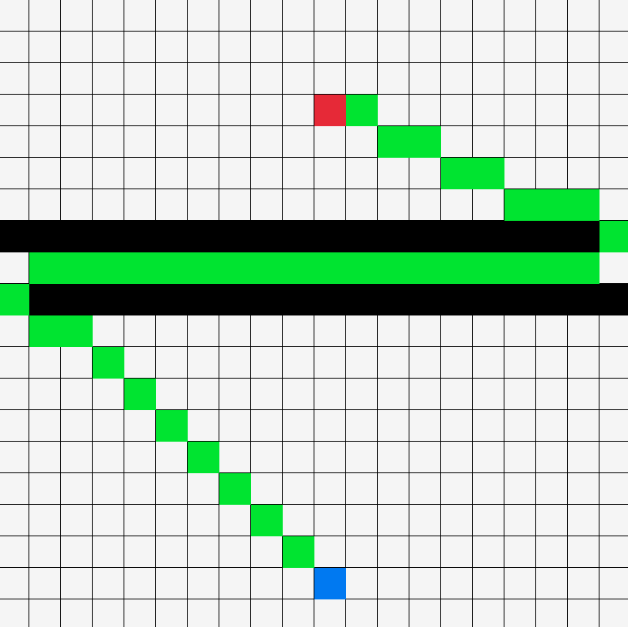
* Gridmap (négyzetháló) használata
* Utak, út szegmensek használata, mint „Node”

### Gridmap (négyzetháló)

A\* útkeresést általában egy négyzethálós rendszeren végzünk, melynek minden egyes négyzete egy lehetséges állapot. Ez lehet egy térkép mely négyzetekből áll és ezeken lépkedünk vagy egy létező terep/út amire rávetítünk egy ilyen négyzethálót és annak négyzetein vagy azokat megközelítve lépkedünk végig a keresett úton. Ezt az utóbbi metódust általában „navmeshnek” hívják.

Egy négyzetnek, „Node”-nak eme értékeit tárolhatjuk:

* A szükséges ***f, g*** és ***h*** értékek
* Ha 2 dimenziós tárolót használunk: sor és oszlop index, ha ezek által szeretnénk visszakeresni
* ***X*** és ***Y*** koordinátái az adott négyzetnek, kirajzolás, más grafikai manipulálás érdekében (**Raylib** a négyzet bal felső sarkának a koordinátáit használja kirajzoláskor)
* Szülő objektum, avagy melyik állapotból kerültünk a jelenlévőbe, hogy a kész útvonalat ezt visszavezetve könnyen összerakhassuk
* Járható-e / más költséget változtató paramérterek, abban az esetben ha nehezebben / lassabban járható cellákat vagy falakat szimulálhassunk



3. ábra: Egy négyzetrácsos pályán a kék és piros pont közötti legrövidebb út zölddel jelölve, ahol a fekete négyzetek nem átjárhatóak

Ez a megoldás egy egyszerű 2 dimenziós térképhez, amit négyzetekkel feloszthatunk tökéletes. Viszont, ha kanyargós, realisztikus autóutakat szeretnénk bejárni lehet, hogy túl pontatlan lenne.

A képen tér, Színesség, pixel, sor látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

*4. ábra: Próbálkozás egy kanyargós úttal egy négyzetrácsos pályán*

Megtehetnénk, hogy ezt a négyzethálót a bejárni kívánt útra vetítjük és a pontatlanságokat az út megtalálása után kisimíthatnánk, de szerintem fölösleges, tekintve, hogy ehhez lenne egy elegánsabb megoldás.

Út szegmensek használata

### Utak, út szegmensek használata, mint állapot

Egy másik megoldás az lehetne, hogy teljesen megkerüljük a négyzethálós megoldást, és a már elkészített utakat és azok bizonyos pontjait vesszük állapotoknak. Persze ez feltételezi, hogy valaki utakat készít és illesztéseiket egy koordináta rendszerben jelöli, szóval ebből a szempontból körülményesebb.

A képen művészet látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

*4. ábra: Úthálózat felsőnézete, láthatóak az út szegmensek összeillesztései, melyekhez koordináta is tartozik*

Az elv és az algoritmus eljárása ezzel a megoldással ugyanaz lenne:

Kezdőpont → Következő legolcsóbb pont → Azon pont szomszédos lehetőségei → Legolcsóbb kiszámítása → Ismétlés a célpontig

A heurisztikus ***h(n)*** számítása a jelenlegi pont és a célpont légvonalbeli távolságát és a feltételes következő út hosszát venné figyelembe.

Ehhez még készíteni fogok egy próba szimulációt a négyzetrácsos programhoz hasonlóan és a kettő metódust összehasonlítva fogom eldönteni melyik kerül használatra.

## Fejlesztési tapasztalatok és megvalósítás

Ahogy említettem, a Raylib könyvtár Python nyelvű verzióját használtam a bemutató program elkészítéséhez. Ez teszt célnak szolgált, hogy egy zárt környezetben tesztelni tudjam az algoritmus tulajdonságait, skálázhatóságát és általános megvalósítását. Továbbiakban majd a WeBots szimulációs programon belül fogjuk ezt egy fizikai autóval használni.

A Raylibbel már dolgoztam a korábbiakban, így könnyű döntés volt, hogy mit használjak. Az algoritmust is próbáltam már korábbiakban elkészíteni, kevesebb sikerrel.

### Program grafikus megvalósítása

Raylibbel az alábbi részlettel tudunk egy ablakot létrehozni:

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.

*5. ábra: Kódrészlet egy játékablak létrehozásához*

*A képen képernyőkép, szöveg, diagram, tervezés látható

Előfordulhat, hogy az AI által létrehozott tartalom helytelen.*

*6. ábra: A kódrészlet eredménye*

* init\_window() – Létrehozzuk az ablakot a függvénynek átadott paraméterekkel (szélesség, magasság, cím)
* while not window\_should\_close() – A program fő ciklusa, addig fut ameddig meg nem nyomódik az ablak bezárásához kiszabott gomb
* begin\_drawing() | end\_drawing() – Ezen két függvény közé íródik bármi, amit megakarunk jeleníteni
* clear\_background() – A ciklus minden egyes iterációjánál letörlődik a „vászon”
* draw\_rectangel() – Egy négyzet rajzolásának függvénye (x koordináta, y koordináta, szélesség, magasság, szín)
* close\_window() – Ha kilép a fő ciklusból, zárja be az ablakot

Ebből a példából látható a Raylib egyszerűsége, a függvény nevek lényegretörőek és az alap működési elve is könnyen felfogható. Ezután a pont után a a határ a csillagos ég a megjeleníthető formák és grafikák kapcsán, bár a könyvtár lehetőséget nyújt modellek, képek, hangok és textúrák betöltésére is.

### Tapasztalatok

A Python és Raylib együttes használata egy nagyon kényelmes és egyszerű élményt nyújt bármilyen grafikát igénylő programhoz. C és C++ verziója több helyet hagy az optimalizálásra bizonyos pontokon szóval, ha fontos a gyorsaság és feladat megoldásához akkor ne Pythonra essen a választás. Viszont a Pythonnal rendkívül gyorsan tudjuk magát a kódot megírni, fejlesztési időt megspórolva.

Az algoritmus megírása sikeresebb és könnyebb volt azáltal hogy tudtam az A\* pontos folyamatát papíros levezetések után. Megvolt a fő működési elv, így már csak implementálni kellett azt. Nehézségek egyedül az állapotok indexelésében léptek fel, ugyanis egy 2 dimenziós adattárolóból egy elemet [sor index][oszlop index] alakban hívunk meg, míg az alakokat kirajzoló függvények (x koordináta)(y koordináta) elvvel működnek, ami összezavaró lehet.

1. A megfelelő rész aláhúzandó. Amennyiben a válasza NEM, akkor a következő bekezdést és táblázatot törölje a dolgozatból. [↑](#footnote-ref-1)