

# Hálózati algoritmusok

## Vizsgálati napló

---

**Téma: 5. Reconfiguration and Locomotion with Joint  
Movements in the Amoebot Model**

Kiss Marcell, Sándor Balázs, Varga Dávid István

2025.05.06.



# Tartalomjegyzék

<b>1. Amoebot Szimuláció</b>	<b>2</b>
1.1. Általános ismertető . . . . .	2
1.2. A program felépítése és elérhetősége . . . . .	2
1.2.1. A program elérhetősége . . . . .	3
1.2.2. Használat . . . . .	3
1.3. Az Amőbot modell . . . . .	5
1.4. Amőbótot Modell: Elmélet . . . . .	5
1.5. Megvalósítás . . . . .	6
1.5.1. Amoebot Osztály . . . . .	6
1.5.2. TriangleMap Osztály . . . . .	7
1.5.3. Viselkedés és Állapotok . . . . .	8
1.6. Összegzés . . . . .	8

# 1. fejezet

## Amoebot Szimuláció

Ez a projekt egy amőba-szerű robotok (Amoebot) szimulációját valósítja meg az Eötvös Loránd Tudományegyetem Informatikai Karának MSc képzése keretében. A szimuláció célja a robotok viselkedésének és interakcióinak modellezése különböző szintereken.

### 1.1. Általános ismertető

A szimuláció Pygame-re épül, és lehetővé teszi az amőbotok különböző viselkedési mintáinak tesztelését különböző szintereken. A főmenüből választhatók ki a szinterek, amelyek különböző elrendezéseket és kihívásokat kínálnak a robotok számára. A main.py fájl inicializálja a Pygame-et, létrehozza a Simulation objektumot, és elindítja a fő ciklust, amely kezeli az eseményeket, frissíti az állapotokat, és megjeleníti a grafikát.

A Scene osztály felelős a különböző szinterek beállításáért, míg a SceneLibrary osztály lehetővé teszi a szinterek külön fájlban történő kezelését, elősegítve a kód modularizálását és karbantartását.

### 1.2. A program felépítése és elérhetősége

Az alábbiakban ismertetem a program elérhetőségét és mappaszerkezetét.

### 1.2.1. A program elérhetősége

A projekt GitHub repository-ja:

<https://github.com/SandorBalazsHU/elte-ik-msc-amoebot>

### Mappastruktúra

A projekt mappaszerkezete az alábbi:

```
elte-ik-msc-amoebot/
|-- src/
|   |-- amoebot.py          - Az amobotok modellje
|   |-- behaviors.py        - A megvalósított viselkedések gyűjteménye
|   |-- config.py           - A konfigurációs változók gyűjteménye
|   |-- drawer.py           - A rajzelemeket leíró osztály
|   |-- menu_button.py      - A menüelemek gyűjteménye
|   |-- scene.py            - A színtérkezelő
|   |-- scene_library.py    - A megvalósított színterek gyűjteménye
|   |-- simulation.py       - A szimulátor főosztály
|   \-- triangle_map.py     - A háromszögrácsot és a foglaltságot kezelő os
\-- main.py                 - A főprogram
```

### 1.2.2. Használat

Telepítés: Győződj meg róla, hogy a szükséges függőségek telepítve vannak (pl. `pygame`, `pygame_menu`).

Futtatás: A szimuláció elindításához futtasd a `main.py` fájlt:

```
python main.py
```

Navigáció: Használd a menüt a különböző színterek kiválasztásához és a szimuláció beállításához.

## Főbb osztályok és funkciók

Tekintsük most át a projekt főbb osztályait és funkcióit.

## Amoebot (amoebot.py)

Az `Amoebot` osztály reprezentálja az egyes robotokat a szimulációban. Főbb jellemzői:

- **Állapotkezelés:** Aktív vagy passzív állapot.
- **Viselkedés:** Különböző viselkedési minták beállítása.
- **Mozgás:** A háromszög térképen való mozgás és kapcsolódás más robotokhoz.

## Behavior (behaviors.py)

A `Behavior` osztály és a hozzá tartozó `BehaviorType` enumeráció különböző viselkedési mintákat definiál az amőbotok számára, például:

- **RANDOM:** Véletlenszerű mozgás.
- **TO\_HEADING:** Meghatározott irányba való mozgás.
- **INTELLIGENT:** Intelligens viselkedés, például középpont keresése vagy cikcakk mozgás.

## Scene (scene.py)

A `Scene` osztály kezeli a különböző szintereket a szimulációban. Főbb funkciói:

- **Menükezelés:** Főmenü és beállítások menü megjelenítése.
- **Szinterek beállítása:** Különböző szinterek inicializálása, például véletlenszerű elrendezés, falak, meta-modulok.

## SceneLibrary (scene\_library.py)

A `SceneLibrary` osztály célja a szinterek külön fájlban történő tárolása és kezelése. Ez lehetővé teszi a `Scene` osztály karbantartásának egyszerűsítését és a kód modularizálását.

## TriangleMap (triangle\_map.py)

A `TriangleMap` osztály reprezentálja a háromszög alapú térképet, amelyen az amőbotok mozognak. Főbb funkciói:

- **Pozíciókezelés:** Ellenőrzi, hogy egy adott pozíció foglalt-e.
- **Mozgás:** Lehetővé teszi az amőbotok számára a mozgást a térképen.

## 1.3. Az Amőbot modell

Az amőbót modellező szimuláció célja az egyes amőbák mozgásának és interakcióinak modellezése egy rácsos világban. Az amőbák különböző viselkedéseket követhetnek, és reagálhatnak környezetükre, azaz egymás helyére léphetnek, ütközhetnek, és meghatározott szabályok alapján csoportosulhatnak.

A következő dokumentumban az amőbót modellező rendszer felépítését és működését írjuk le, részletesen ismertetve annak működését és a Python programozási nyelven történő implementálását.

## 1.4. Amőbót Modell: Elmélet

Az amőbák szimulációja egy klasszikus rácsos környezetben történik, ahol a világot egy mátrix vagy rács reprezentálja. Az amőbák az ezen rácshoz rendelt koordináták alapján mozognak, és képesek kölcsönhatásba lépni egymással, mint például ütközések elkerülése vagy pozíciók feloldása.

Az amőbák **\*\*állapotok\*\*** és **\*\*viselkedési típusok\*\*** szerint működnek. Minden amőba tartalmazza a következőket:

- **Pozíció** – Az amőba aktuális koordinátái a rácsban.
- **Viselkedés** – Az amőba által végzett tevékenység, mint például véletlenszerű mozgás vagy koordinált csoportosulás.
- **Lépésszámláló** – Minden amőbának van egy lépésszámlálója, amely nyomon követi, hogy hány lépést tett meg. Ez befolyásolhatja a pozíció felszabadítását és egyéb műveleteket.

A viselkedés típusok között szerepelhet például:

- **\*\*Véletlenszerű mozgás\*\*** – Az amőba egy véletlen irányba lép.
- **\*\*Célzott mozgás\*\*** – Az amőba egy konkrét irányba mozog.
- **\*\*Intelligens viselkedés\*\*** – Az amőba alkalmazkodik a környezetéhez, például a középpont felé tart.

## 1.5. Megvalósítás

A rendszer Pythonban van implementálva, és az amóbák osztálya a következő kulcsfontosságú attribútumokkal rendelkezik:

### 1.5.1. Amoebot Osztály

Az amóbák működését az `Amoebot` osztály valósítja meg. Az osztály tartalmazza a szükséges attribútumokat, mint például a pozíció, viselkedés, és a lépésszámláló.

```
class Amoebot:
    def __init__(self, triangle_map, row, col):
        self.triangle_map = triangle_map
        self.row = row
        self.col = col
        self.step_counter = 0
        self.from_pos = None
        self.behavior = None
        self.state = AmoebotState.PASSIVE
        # További inicializálás...
```

#### Lépéskezelés

Minden amóbának van egy `update` metódusa, amely kezeli a mozgását. Az amőba a `can_move_somewhere` logikát követi, hogy eldöntse, vajon képes-e mozogni a következő lépésben. Ha igen, elmenti a jelenlegi pozícióját a `from_pos` változóba, majd kiszámolja az új célpontot és végrehajtja a mozgást.

```
def update(self):
```

```
if self.can_move_somewhere(): # Ha lehetséges lépni
    self.from_pos = (self.row, self.col)
    new_row, new_col = self.calculate_target()
    self.row = new_row
    self.col = new_col
    self.triangle_map.occupy(new_row, new_col, self)
    self.step_counter += 1
    return True
return False
```

A `step_counter` változó nyomon követi az amőba lépéseit. Minden második lépésnél a pozíció felszabadítása történik meg.

### Pozíció felszabadítása

A pozíció felszabadítása akkor történik meg, ha az amőba elérte a szükséges lépésszámot, és ha a `Config.Scene.replace_pos` beállítás engedélyezi ezt. A következő kód segít abban, hogy minden második lépésnél feloldjuk az előző helyet:

```
if self.step_counter % 2 == 0:
    if self.from_pos:
        self.triangle_map.release(*self.from_pos)
        self.from_pos = None
```

### 1.5.2. TriangleMap Osztály

Az amőbák mozgása a `TriangleMap` osztály segítségével történik, amely kezeli a rácsot, a helyek lefoglalását, és a pozíciók felszabadítását. Az amőbák csak akkor léphetnek, ha a célpontjuk üres, és ha az `occupy` metódus nem foglalja el a helyet.

```
class TriangleMap:
    def __init__(self):
        self.grid = [[None for _ in range(cols)] for _ in range(rows)]

    def occupy(self, row, col, amoebot):
```



```
        if self.grid[row][col] is None:
            self.grid[row][col] = amoebot
            return True
        return False

    def release(self, row, col):
        self.grid[row][col] = None
```

### 1.5.3. Viselkedés és Állapotok

Az amőbák különböző viselkedéseket képesek végrehajtani, például véletlenszerű mozgást vagy célzott mozgást. Az `AmoebotState` és a `BehaviorType` osztályok biztosítják, hogy az amőbák állapota és viselkedése változhasson, például amikor egy amőba eléri a célját vagy új viselkedést kell alkalmaznia.

```
class AmoebotState(Enum):
    PASSIVE = auto()
    ACTIVE = auto()
    MOVING = auto()

class BehaviorType(Enum):
    RANDOM = auto()
    TO_HEADING = auto()
    INTELLIGENT = auto()
```

## 1.6. Összegzés

A szimuláció során az amőbák a rácsos világban mozognak, és különböző viselkedéseket alkalmaznak. Az amőbák mozgása és interakciói jól definiált szabályok szerint történnek, figyelembe véve a lépésszámlálást, pozíciók felszabadítását és a viselkedés típusokat. Az implementáció rugalmasan bővíthető további viselkedésekkel és interakciókkal.