




# WIREWORLD

## a cellular automata

0. A wireworld egy többszínű sejtautomata, amivel egy egyszerű módon modellezni lehet kábelekben futó elektromosságot. Néhány egyszerű szabállyal rendelkezik a következő generációra nézve, hogy melyik cella milyen állapotba kerüljön, feltéve a mostani és a környezete mostani állapotait. Ezzel elméletben bármilyen áramkört – akár egy egész számítógépet – meg lehetne valósítani.

---

4 fajta állapot van:

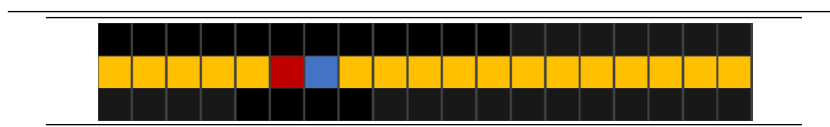
- üres -----  ----- fekete
- elektronfej -----  ----- kék
- elektronfarok -----  ----- piros
- vezeték -----  ----- sárga

---

A szabály a következő generációra nézve:

- üres → üres -----  →  -----
  - elektronfej → elektronfarok -----  →  -----
  - elektronfarok → vezeték -----  →  -----
  - vezeték → elektronfej -----  →  ----- \*ha pontosan 1 vagy 2  -vel szomszédos
  - vezeték → vezeték -----  →  ----- \*minden más esetben
- 

További, részletesebb leírás az automata [wikipedia](#) oldalán is olvasható.



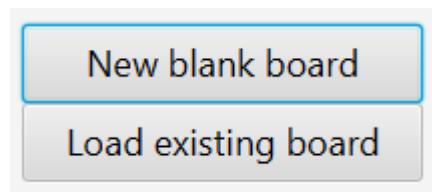
# 1. User manual

## 1.1. Főmenü

1.1.1 Indításkor megjelenik egy menü, mely két opciót tár a felhasználó elé:

New blank board:

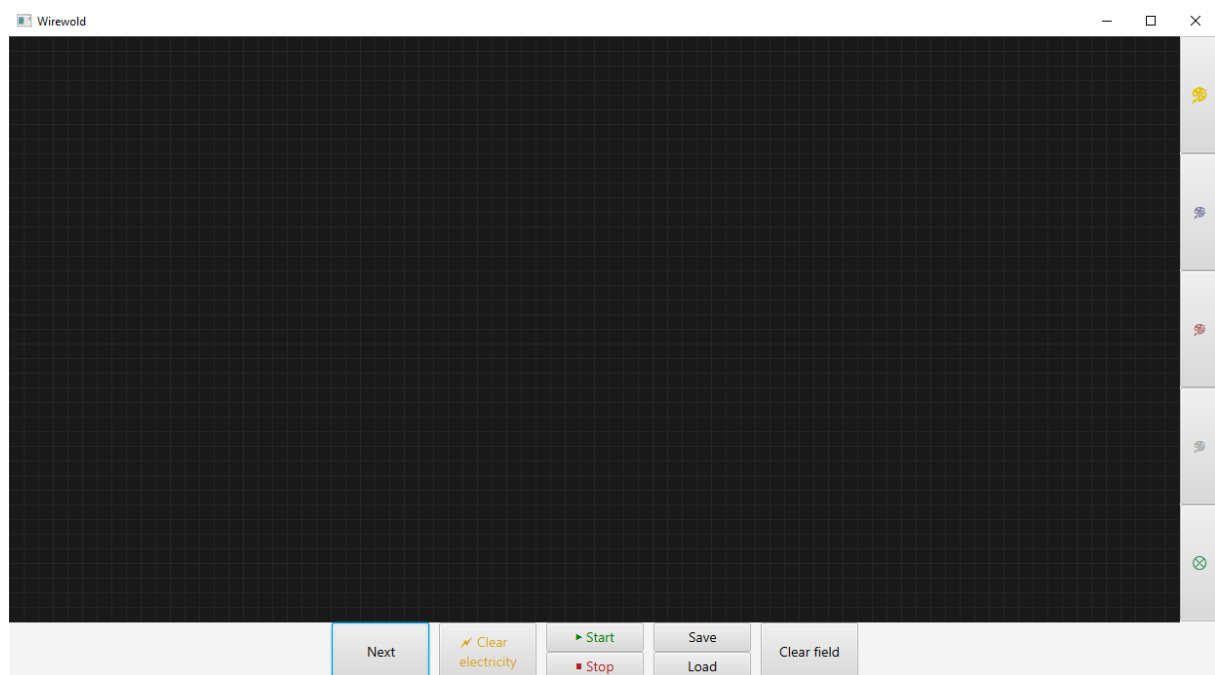
Megjeleníti az 1.2. pontban leírt játéktér, mely alapállapotban lesz.



Load existing board:







Megjeleníti az 1.4.2. pontban leírt betöltésre szolgáló jelenetet.

## 1.2. Játéktér:







1.2.1. Ez a jelenet látható a program fő funkcióinak használata közben. Megjelenik egy négyzetháló (továbbiakban Tér), melynek cellái alapállapotban mind üres (■) állapotban vannak.

1.2.2. A felhasználó a programmal interakcióba főleg ezen Tér mezőire kattintva léphet. Oldalt kiválaszthatja az „ecsetet”, mely meghatározza, milyen színűre – és ezzel együtt típusúra – kívánja módosítani majd a mezőket. Induláskor az első ecset (■) automatikusan ki van választva, a többi ecset ikonja kisebb, deszaturáltabb. Ezután, amelyik mezőre kattint, az ezt a színt és a hozzá tartozó viselkedést fogja produkálni a továbbiakban. Az aktuálisan kiválasztott ecset mindig nagyobb, élénkebb színű a többinél, megkönnyítve ezzel, hogy ránézésre látható legyen, melyik van kiválasztva. Elérhető továbbá egy másik kezelési módszer, melyet oldalt a legalsó gombbal

aktiválhatunk. Amikor ez van ki választva az összes ecset ikonja elszürkül. Ebben a módban egy mezőre ismételt kattintás a lehetséges színeken való iterálást eredményezi (  →  →  →  →  ). Ebből a módból kilépni egy ecsetszín kiválasztásával, vagy a gomb újbóli megnyomásával lehet. Utóbbi esetben az alapértelmezett (  ) ecsetszín lesz aktív.

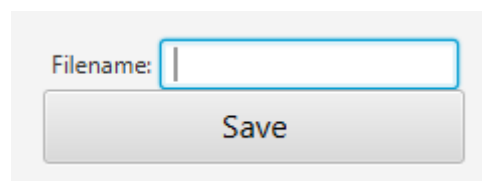
## 1.3 Gombok:

1.3.1. A Tér alatt a felhasználó további gombokat lát, melyekkel interakcióba lépve a következőket teheti:

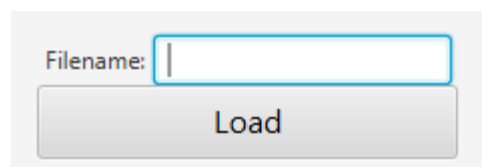
- „Ecset” gombok : Itt választható ki, hogy a fent említett funkció milyen színűre színezi majd a cellákat. (ld.: 1.2.2. pont)
- Next : A 0. pont szabályainak megfelelően egy generációval előre lépteti a Térben elhelyezkedő összes cellát.
- Clear electricity: A Térben letörli az elektronokat a kábelekről. A szabályok értelmében ez azt jelenti, hogy minden elektron komponens (  vagy  ) vezetékké (  ) alakul.
- Start : A Next pontban leírt műveletet végzi többször egymás után, ezzel egy folyamatos animációban frissíti a Teret, generáció után generációval. A Tér közben is szerkeszthető marad.
- Stop: Ha az animáció fut, leállítja azt. Ellenkező esetben nincs hatása.
- Save : Megjeleníti az 1.4.1. pontban leírt mentésre szolgáló jelenetet.
- Load : Megjeleníti az 1.4.2. pontban leírt betöltésre szolgáló jelenetet.
- Clear field: Alapállapotba helyezi a Teret. Ez azt eredményezi, hogy minden mezőt üres (  ) állapotba helyez.

## 1.4. Mentés és betöltés:

1.4.1. Mentés: a mezőbe egy szöveget beírva a [szöveg](#).txt fájlba menti a Tér jelenlegi állását a Save gomb megnyomására, majd az 1.1.1 pontban leírt főmenü jelenetre vált.

A screenshot of a 'Save' dialog box. It features a text input field labeled 'Filename:' with a cursor inside. Below the input field is a grey button with the text 'Save'.

1.4.2. Betöltés: a mezőbe egy szöveget beírva a [szöveg](#).txt fájlból betölti az abban tárolt állást a Load gomb megnyomására, majd az 1.2. pontban leírt Játéktér jelenetre vált.

A screenshot of a 'Load' dialog box. It features a text input field labeled 'Filename:' with a cursor inside. Below the input field is a grey button with the text 'Load'.

## 2. Osztályok leírása

### 2.1. MenuScene

2.1.1. Az 1.1.1 pontban bemutatott menü nézet megjelenítésére szolgáló osztály. A gombok megjelenítéséhez a 2.6. pontban leírt FormattableButton osztályból példányosított gombokat használ, melyeket a konstruktoron belül hoz létre.

2.1.2. Attribútumok:

- root: GridPane	Ez lesz a jelenethez tartozó gyökér panel, melyet jelenetváltásnál a setRoot függvény megkap.
------------------	---

2.1.3. Metódusok:

+ getRootPane(): Pane	Ezen a metóduson keresztül érjük el a gyökér panelt.
+ MenuScene()	Konstruktor, mely szerepét és implementációját a 2.1.1 pont tárgyalja bővebben.

### 2.2. InGameScene

2.2.1. Az 1.2 pontban bemutatott játéktér nézet megjelenítésére szolgáló osztály. Az megfelelő elrendezés eléréséhez GridPane és Group layoutok egymásba ágyazása használt. A gombok megjelenítéséhez a 2.6. pontban leírt FormattableButton osztályból példányosított gombokat használ, melyeket a konstruktoron belül hoz létre. A játéktér megjelenítésére egy Group layoutban létrehozott, a 2.5. pontban bemutatott Cell osztályból példányosított objektumok használt.

A gombok és layoutok létrehozása, elhelyezése és méretezése után ezek manipulációját, illetve a szerepük ellátását a gombokra kattintva lambdák vezérlik.

2.2.2. Attribútumok:

- root: GridPane	Ez lesz a jelenethez tartozó gyökér panel, melyet jelenetváltásnál a setRoot függvény megkap.
+ grid: ArrayList<ArrayList<Cell>>	Egy Cella objektumokból álló mátrix. Ez tartalmazza a Teret.
- animation: Thread	Szál, melly az animálásra szolgál.

## 2.2.3. Metódusok:

+ getRootPane(): Pane	Ezen a metóduson keresztül érjük el a gyökér panelt.
+ InGameScene(ArrayList<ArrayList<Cell>>)	Konstruktor, mely szerepét és implementációját a 2.2.1 pont tárgyalja bővebben. Amennyiben üres paramétert kap, vagy a paraméter nélküli konstruktor delegálja, feltölti a paramétert egy üres Teret reprezentáló adatokkal.
+ InGameScene()	Konstruktor, mely delegál a paraméteresre. Ennek célja, hogy ha üres Teret hozunk létre, elegendő a paraméter nélküli konstruktort hívni, de átadhatunk egy létező ArrayList<ArrayList<Cell>> típusú attribútumot is, ebben az esetben az nem lesz újra feltöltve.
- initAnimation(): void	Elindítja az animációt.
- stopAnimation(): void	Megállítja az animációt, ha az fut.
- clearElectricity(ArrayList<ArrayList<Cell>>) : void	A paraméterként kapott mátrix minden elektronrész típusú mezőjét vezetékek állapotba helyezi.
- resetField(ArrayList<ArrayList<Cell>>) : void	A paraméterként kapott mátrix minden mezőjét üres állapotba helyezi.

## 2.3. SaveScene

2.3.1. Az 1.4.1. pontban bemutatott mentés nézet megjelenítésére szolgáló osztály. A gomb megjelenítéséhez a 2.6. pontban leírt FormattableButton osztályból példányosított gombokat használ, melyeket a konstruktoron belül hoz létre. A szöveget Label, a mezőt TextField példányai valósítják meg.

Mivel az `ArrayList<ArrayList<Cell>>` típusú attribútum a Cell osztály szerializáció-implementálása mellett sem volt megfelelően betölthető, ezért `FileWriter` I/O osztály segítségével kódolja a Tér tartalmát .txt fájlba.

2.3.2. Attribútumok:

- root: GridPane	Ez lesz a jelenethez tartozó gyökér panel, melyet jelenetváltásnál a <code>setRoot</code> függvény megkap.
------------------	--

2.3.3. Metódusok:

+ <code>getRootPane(): Pane</code>	Ezen a metóduson keresztül érjük el a gyökér panelt.
+ <code>SaveScene(ArrayList&lt;ArrayList&lt;Cell&gt;&gt;)</code>	Konstruktor, mely szerepét és implementációját a 2.3.1 pont tárgyalja bővebben. Paramétereként a menteni szánt Teret kapja.

## 2.4. LoadScene

2.4.1. Az 1.4.2. pontban bemutatott betöltés nézet megjelenítésére szolgáló osztály. A gomb megjelenítéséhez a 2.6. pontban leírt `FormattableButton` osztályból példányosított gombokat használ, melyeket a konstruktoron belül hoz létre. A szöveget `Label`, a mezőt `TextField` példányai valósítják meg.

Mivel az `ArrayList<ArrayList<Cell>>` típusú attribútum a `Cell` osztály szerializáció-implementálása mellett sem volt megfelelően betölthető, ezért `FileReader` I/O osztály segítségével olvassa ki egy `.txt` fájlból, és tölt fel egy `ArrayList<ArrayList<Cell>>` objektumot, amit átad az `InGameScene` osztály konstruktorának, majd arra a jelenetre vált.

### 2.4.2. Attribútumok:

- root: <code>GridPane</code>	Ez lesz a jelenethez tartozó gyökér panel, melyet jelenetváltásnál a <code>setRoot</code> függvény megkap.
-------------------------------	--

### 2.4.3. Metódusok:

+ <code>getRootPane(): Pane</code>	Ezen a metóduson keresztül érjük el a gyökér panelt.
+ <code>LoadScene()</code>	Konstruktor, mely szerepét és implementációját a 2.4.1 pont tárgyalja bővebben.

## 2.5. Cell

2.5.1. A Rectangle osztály egy bővített, kiegészített implementációja, mely a játékban használatos mezők létrehozását támogatja.

2.5.2. Attribútumok:

- xpos: int	A cella x koordinátáját tárolja.
- ypos: int	A cella y koordinátáját tárolja.
+ nextStateType: CellType	Egy enumban tárolja a cella következő generációban milyen színű lesz. Mivel függhet a következő generáció a környező celláktól, előbb végig kell ellenőrizni mindet, csak utána frissíthetjük a jelenlegi állapotot. Ezért van szükség egy attribútumra, ami a következő állapotot is tárolni tudja.
+ brush: CellType	Egy enumban tárolja, milyen színű a kiválasztott „ecset”. Statikus.
+ boolean: cycleMode	Tárolja, hogy ciklikus-e a színezési mód. Statikus.

2.5.3. Metódusok:

- Cell(int, int)	Konstruktor mely az űsosztály konstruktorát meghívva beállítja a méretet, ezenkívül beállítja a koordinátákat tároló attribútumokat is. Privát, a createCellAt hívja, úgy van értelmezve a működése.
+ makeToColor(CellType)	Beállítja az ecset színét, a paraméterként kapott enum értékre.
+ createCellAt(int, int): Cell	Létrehoz egy cellát a paraméterként kapott koordinátákon. Ezután beállítja a formázását, a típusát (ami mindig üres cella) és a rákattintáskor történő lambda függvényt definiálja.



## 2.6. FormattableButton

2.6.1. A Button osztály kiegészítése, primitív formázási paraméterekkel létrehozható gomb.

2.6.2. Attribútumok:

-	-
---	---

2.6.3. Metódusok:

+ FormattableButton(String, String, int)	a setStyle() és textProperty().set() függvények segítségével beállítja a paraméterként átvett értékre a gomb szövegét, annak színét és betűméretét css formázással.
--	---

## 2.7. Animate

2.7.1. A Thread osztály leszármazottja, ami a Tér léptetését valósítja meg, illetve annak az adott időközönként történő meghívását, ezzel animálva azt.

2.7.2. Attribútumok:

-	-
---	---

2.7.3. Metódusok:

run(): void	250 miliszekundumonként léptetni kezdi a Teret.
nextState(ArrayList<ArrayList<Cell>>) : void	Minden cellának beállítja a nextStateType attribútumát a jelenlegi állás szerint, majd a nextStateType szerint kiszínezi a cellákat.
nearbyElectronHeads(ArrayList<ArrayList<Cell>> , int, int) : int	Megszámolja a paraméterként átvett koordináta közelében levő elektronfejeket az átadott Térben, és ennyivel tér vissza.

## 2.8. CellType

2.8.1. Enum, melyek értékei a következők lehetnek: BLANK, WIRE, TAIL, HEAD. Ez a cella lehetséges típusait sorolja fel és a Cell osztály használja.

### 3. Specifikáció

A program JavaFX platformot használ a felhasználói felület megvalósításához.

---

Egy Cella, mely a specifikáció szerinti 4 állapotot veheti fel, a `javafx.scene.shapes` csomag `Rectangle` osztályából leszármazott `Cell` osztály, melyben a Térben elfoglalt koordinátáit, és a következő generációban felvenni kívánt színét is el tudjuk tárolni, az `ő`sosztály funkcióin kívül.

---

A Tér ilyen cellák mátrixa, mely a `java.util` csomag `ArrayList` sablonosztályok használatával, cellák tömbjének tömbjével lesz megvalósítva.

---

A többi vezérlő, melyek a Teren kívül helyezkednek el a `javafx.scene.control` csomag `Button` osztályából származtatott `FormattableButton` osztállyal van megvalósítva, melyek a játék belső logikájának megfelelő módosításáért (pl. ecset színe) vagy a fájlkezelést vezérlő függvények meghívásáért felelősek. Ezen utóbbiak megfelelő szövegfájlokon tudnak dolgozni, ami csak a Tér celláinak egy adott generációját tartalmazza.

---

## 4. Diagram

