Számítógépes szimulációk VI.: Sejtautomata

Pál Balázs*

*Eötvös Loránd Tudományegyetem

2019. május 10.

Abstract

A *Számítógépes szimulációk* laboratórium hatodik és egyben utolsó gyakorlatának alkalmával a sejtautomaták kérdéskörébe tettünk bepillantást. Feladatunk volt, hogy létrehozzuk az elhíresült Conway-féle Életjáték megvalósító kódot, mely tetszőleges kihalási szabály szerinti működést képes produkálni. Ezután létre kellett hoznunk egy homokdombautomatát és meg kellett határoznunk annak skálázási törvényében szereplő exponens értékét.

1. BEVEZETŐ

A labor utolsó gyakorlatán a sejtautomaták témakörével foglalkoztunk. Ezeket populárisan híressé a Conway-féle Életjáték tette, azonban már évtizedekkel előtte is ismertek voltak a számításelmélet területén, melynek alapjait Neumann János fektette le.
A sejtautomaták mechanizmusának szakirodalma tág, azonban a kurzus során ebben nem kellett a témába mélyebben belemennünk, így kizárólagosan a Conway-féle Életjáték alapvető szabályaival, valamint az Abel-, vagy más néven Bak–Tang–Wiesenfeld-féle homokdomb modell megvalósításával foglalkoztunk.

2. FELADATOK ÉS ELMÉLET

A mostani gyakorlat alkalmával - az előzőhöz hasonlóan - szintén magunknak kellett megírjunk minden forráskódot, előre csak a feladatokat ismertető leírás volt megadva. A két megvalósítandó rendszer a fentebb is említett Életjáték nevű sejtautomata, valamint a homokdomb modell volt.

2.1. ÉLETJÁTÉK

Az Életjátékot 1970-ben John Horton Conway, brit matematikus találta fel. A játék alapvető felépítését egy klasszikus, 2-dimenziós, végtelen négyzetrácson működő sejtautomata alkotta, mely működésére specifikus szabályok vonatkoztak. Az játékterület négyzetrácsán minden négyzet két állapotot vehet fel, melyeket a 0 és 1 értékekkel, vagy a `Halott` és `Élő` kifejezésekkel különböztetünk meg. Egy játék kezdetén a négyzterács minden cellájának kijelölünk valamilyen értéket a fenti kettő közül, a játék pedig ebből az általunk választott tetszőleges kezdőpozícióból indul. Ezt követően a négyzetrács az adott szabály szerint lépésekben frissül. A Conway-féle Életjáték standard módon megfogalmazott szabályai a következőek:

1. Ha egy cella `Élő`, és `Élő` szomszédjainak száma 0, vagy 1, akkor a következő körben `Halott`á változik. (Alulnépesedés)
2. Ha egy cella `Élő`, és `Élő` szomszédjainak száma 2, vagy 3, akkor a következő körben is `Élő` marad. (Fajfenntartás)
3. Ha egy cella `Halott`, és `Élő` szomszádjainak száma pontosan 3, akkor a következő körben `Élő`vé változik. (Reprodukció)
4. Ha egy cella `Élő`, és `Élő` szomszédjainak száma több, mint 3, akkor a következő körben is `Halott`á változik. (Túlnépesedés)
5. Minden más esetben a cella `Halott`.

A program megírása során ezeket az értékeket kontrollálhatóvá kellett megírjuk. Ez azt takarta, hogy tetszőleges, terminálból megadható N érték esetén történjen a játékban reprodukció a fenti 3 helyett, valamint $N-1$ és N értéknél pedig maradjon a cella `Élő`, ha eddig is `Élő` volt. A másik két szabály hasonlóan ehhez kellett ilyenkor igazodjon automatikusan.
A másik megkötés az volt, hogy a programot tetszőlegesen, különböző kezdőfeltételekkel indíthassuk (végtelen, periodikus, konstans élő határ és konstans random határ), hasonlóan terminálból szabályotható módon.

2.2. ABEL-FÉLE 2D HOMOKDOMB MODELL

A 2 dimenziós homokdomb modell a nevéből is érthetően „homokszem-tornyok” 2 dimenziós síkon vett ábrázolása. A síkot felosztva egy négyzetráccsal, minden négyzetet megfeleltetünk 1-1 homokszem-toronynak, melyben N darab homokszem tartózkodik. A rendszert egy H mátrixxal írhatjuk ilyenkor le, melynek minden eleme az adott négyzetben tartózkodó homokszemek számát jelöli. Az Abel-féle definícióban egy ilyen tornyot instabilnak hívunk, ha benne több, mint 3 homokszem található. Ha egy torony eléri ezt az instabil állapotot, akkor egy lavina alakul ki, mely a következőt jelenti:

$$\text{Ha } M_{i,j} > 3 \rightarrow \begin{cases} M_{i,j} = M_{i,j} - 4 \\ M_{i\pm1,j} = M_{i\pm1,j} + 1 \\ M_{i,j\pm1} = M_{i,j\pm1} + 1 \end{cases} \tag{1}$$

A gyakorlat során ezen modell megvalósítása volt a feladatunk. Ezután ellenőriznünk kellett annak skálázási törvényét. Megfigyelés alapján az $N(n_t)$ eloszlást az alábbi függvény adja meg:

$$N(n_t) = \frac{1}{n_t^b} \tag{2}$$

3. MEGVALÓSÍTÁS

Az alapvető forráskódokat a kötelező előírásnak megfelelően `C++` nyelven implementáltam. Mind a sejtautomata, mind pedig a homokdombmodell egy-egy külön forrásfileban található. A forráskódokat az eddigiekhez hasonlóan egy saját batch file segítségével, benne a `clang` fordító felhasználásával fordítottam. A futtatható `exe` programokat egy Jupyter Notebook-ban futó Python 3 kernel segítségével indítottam, a szimuláció a kezdőfeltételeket szintén ebből a környezetből várja, bemenő paraméterek formájában.

A kimenet minden esetben egy `.dat` file, melyben a szimulációt lépéseit illusztráló 2D mátrixok egyszerűen egymás alatt szerepelnek. A gyakorlaton tárgyalt feladatokhoz néhány animációt is készítettem, amik közül azonban a homokdomb esetén nem az eredeti forráskódot, hanem egy könnyebben meghívható és léptethető python-verziót használtam. Az animációk elkészítéséhez így két különböző kódot használtam. A sejtautomatát animáló kód egy, az eddigiekhez is hasonló, az `imageio` könyvtárat és az `ffmpeg` szoftver használtam. A homokdomb esetében egy teljesen más megközelítést alkalmaztam. Egy `class SandpileAnimation` nevű python classt írva és a `matplotlib.animation` library-t felhasználva egy tetszőlegesen léptethető és videót generáló kódot valósítottam meg, mely képes arra, hogy pontosan a homokdomb egyensúlyi pozíciójának beálltáig animáljon, így optimalizálva egy szimulációt.

A sejtautomata programnak kétféle bemeneti módja létezik. Az első esetében egy tetszőleges méretű és pozíciójú, random generált életet helyezhet el a játékos a játéktéren. A második verzió esetén egy `.dat` file-ból adhat be neki egy 0 és 1 értékekből álló, tetszőleges méretű mátrixot, melyet aztán szintén általa választott pozicióban helyezhet el a játéktéren. A kiértékelés közben kiderült, hogy az MS Paint program megfelelő ilyen tetszőleges kezdeti állapotok létrehozására, hisz benne tetszőleges módon rajzolhatunk fekete és fehér pixelekből felépített képeket, melyeket aztán 0 és 1 értékekként értelmezve, szintén beadhatunk a programnak.

A homokdomb modell esetén két fajta kezdőfeltételt lehet váalsztani, azonban a peremfeltételeket minden esetben nyíltként értelmeztem. Az első kezdőfeltételt választva a homokszemek száma az egyes tornyokban véletlenszerűen választódik. A homokszemek száma a $[0, n]$ intervallumból sorsolódik ahol n egy terminálból beadható paraméter. A második kezdőfeltétel választása esetén minden toronyban egyaránt n darab homokszem fog elhelyezkedni kezdetben, a szimuláció ebből az állapotból indul. Ezen utóbbi a homokdomb modell szimulációk szokásos kezdőfeltétele, ahol n értékét 7-nek szokták megválasztani.

A végleges forráskódok és a programokat futtató Notebook file mind elérhető GitHub-on (Pál, 2019).

4. KIÉRTÉKELÉS

4.1. SEJTAUTOMATA

A sejtautomaták viselkedését legjobban működésének folyamatában vizsgálhatjuk. Emiatt több különböző esetéről animációt is készítettem, melyeket a youtube-on elérhetővé is tettem, és melynek linkje a hivatkozások között elérhető. Az animációkon számtalan ismert automatát szimuláltam, különböző határfeltételek mellett. Ezek között sorrendet nem figyelve szerepel pl. a *Gosper glider gun*, a *Copperhead* űrhajó, vagy pl. a *Kox's galaxy* névre hallgató oszcillátor. Ezen felsorolt hármat a jegyzőkönyvben is ábrázoltam és a (1) - (3) ábrákon tekinthetőek meg.
Abban a két esetben, amikor a határokat valamilyen konstans Élő, vagy random módon sorsolt elemeknek választjuk meg, a határokról az azonnal ki fog terjedni, ha a (2.1)-es részben szereplő N paraméter értéke $N \leq 3$. Minden más esetben nem változhatnak a határok ebből fakadóan, hisz a határon található sejtek melleti cellák mindegyike pontosan 3 határnégyzettel szomszédos.
Komplexebb formákat magasabb rendű esetekben azonba nagyon nehéz kialakítani. $N = 8$ esetben pl. az egyetlen stabil forma az egész teret egyenletesen kitöltő `Élő` cellák halmaza, ez is kizárólag konstans élő, vagy periodikus határfeltételek mellett. Minden más esetben a sejtccsoport elhal.
Az $N \in [4, 7]$ intervallumban már lehetséges létrehozni olyan formákat, amik stabilak. Ilyen pl. a $N = 5$ esetben periodikus határfeltételek mellett sakktáblaszerűen kitöltött játéktér, megfelelő oldalhosszakkal megválasztva.

4.2. HOMOKDOMB MODELL

A megvalósított homokdomb modellt különböző nagyságú játékterületre futtatva, mértem a stabil állapot eléréséhez szükséges lépések számát. Minden futtatás esetén a kezdőfeltételt úgy választottam meg, hogy minden toronyban azonosan $n = 7$ homokszem tartózkodjon kezdetben. A szimulációt ezután a stabil állapot beálltáig futtattam, a kapott végleges állapotokat pedig a (4) - (7) képeken ábrázoltam, különböző játéktér méretekre. A mintázat minden esetben nagyon hasonló és tökéletesen visszaadja a (Gonsalves, 2004) leírásban is látható képet. Ehhez hasonlóan próbálkoztam olyan modellek elkészítésével is, melyek nem négyzet, hanem tetszőleges téglalap alaőúak voltak. Ezek külsőre nagyon hasonlóak voltak a négyzet alapúakhoz, így külön nem részletezem őket, hisz nem produkáltak semmiféle egyéb érdekességet.
A 120×120 méretű játéktérrel rendelkező esetről egy animációt is készítettem, melyet a youtube-on elérhetővé is tettem, és melynek linkje a hivatkozások között elérhető.

5. DISZKUSSZIÓ

A laborgyakorlat előtt már volt tapasztalatom a sejtautomaták működését és programozását illetően, azonban most még több ismeretre tehettem szert a gyakorlat feladatainak megoldása közben. Emellett megismerkedhettem a homokdomb modellel is és újabb módszereket próbálhattam ki annak animálásától motiválva. Amennyire időm engedte a szakdolgozat írása és a vizsgákra való készülés mellett, megpróbáltam kellő mélységben belemenni az utolsó gyakorlat feladatainak megvalósításába is. Több-kevesebb sikerrel, egyáltalán nem 100%-osan minden feladatot kidolgozva, de végül időre végeztem elégséges módon vele.

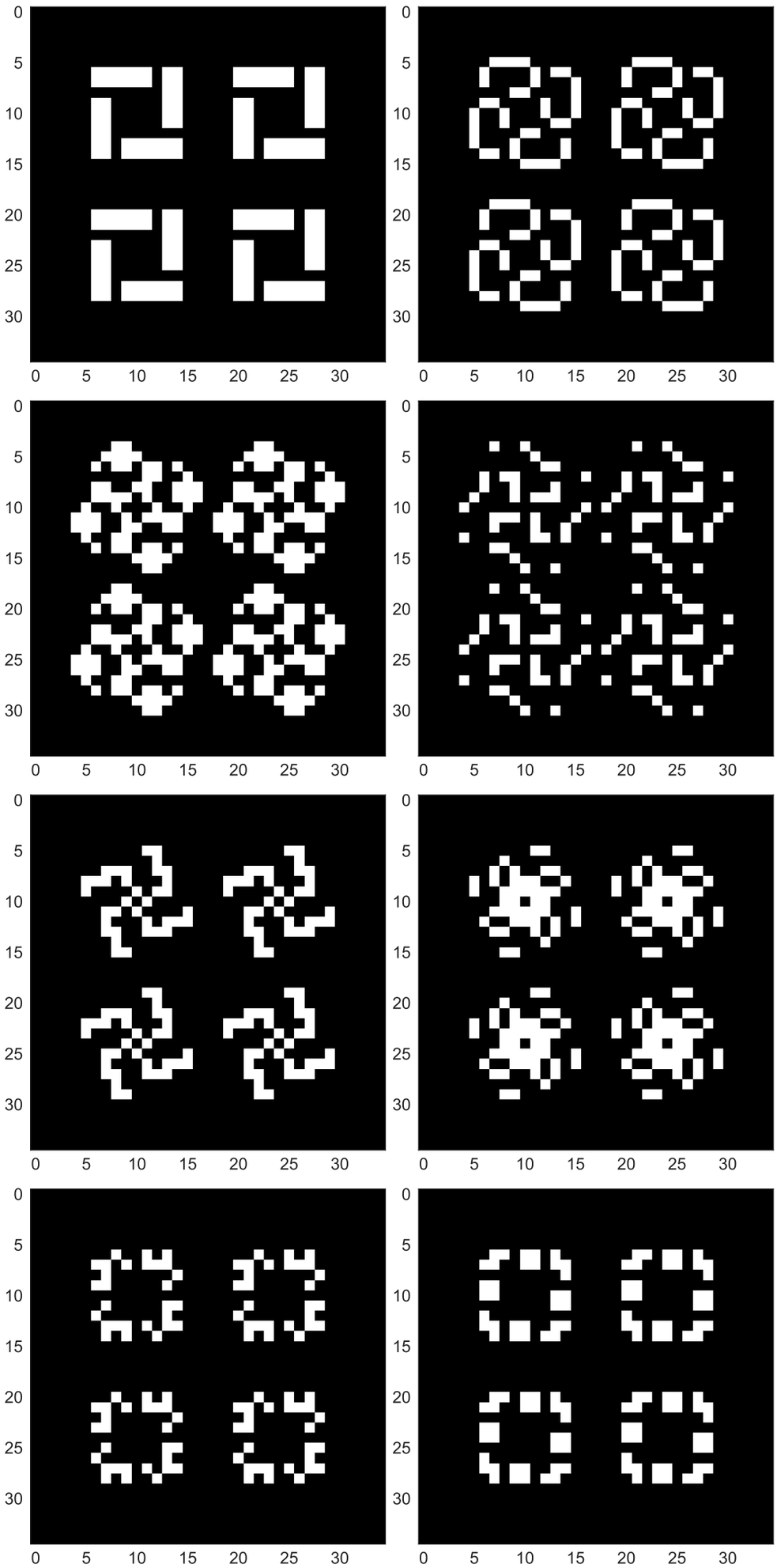
---

[1] Balázs Pál. *ELTE Computer Simulations 2019 — GitHub*. 2019. URL: https://github.com/masterdesky/ELTE_Comp_Simulations_2019. [Online; opened at May 6, 2019].

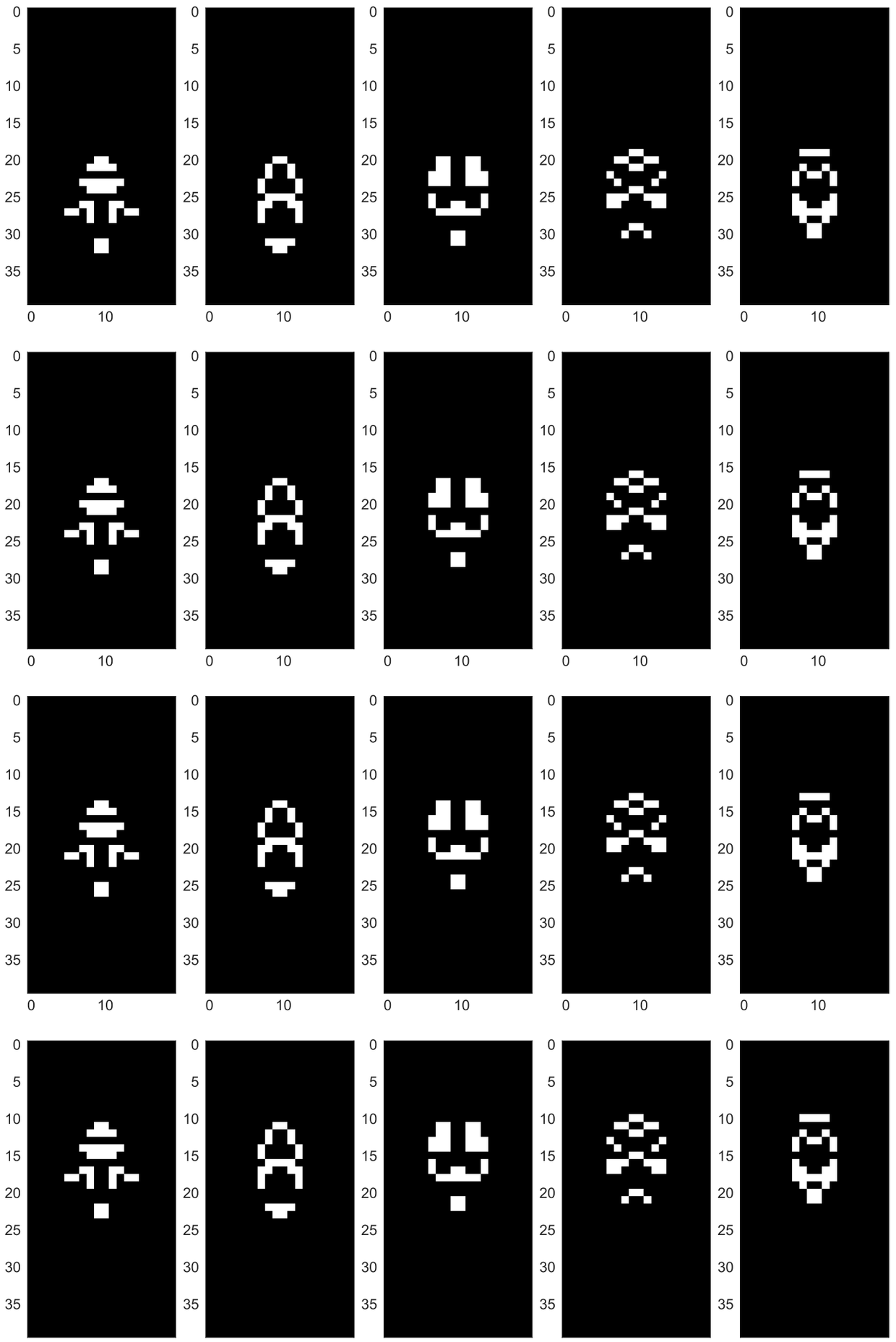
[2] *Pál Balázs's Channel — YouTube*. 2019. URL: https://www.youtube.com/channel/UCBDSB7PdQ3E9l9WSBsTy7cQ. [Online; opened at May 6, 2019].

[3] Richard J. Gonsalves. *Two dimensional sandpile automaton*. 2004. URL: http://csabai.web.elte.hu/http/szamszim/lecture7/topic6-lec3.pdf. [Online; opened at May 6, 2019].

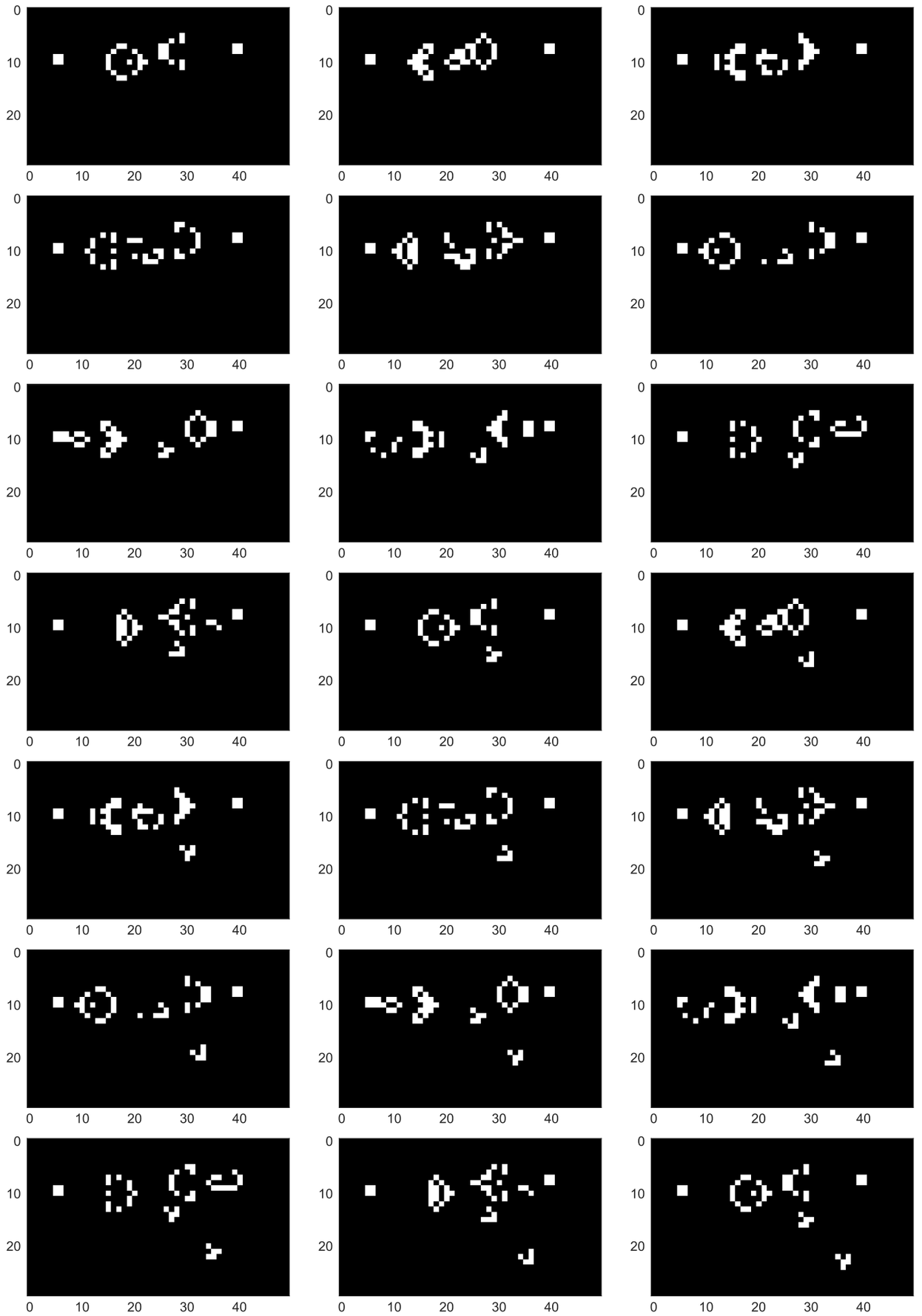
APPENDIX A - ÁBRÁK



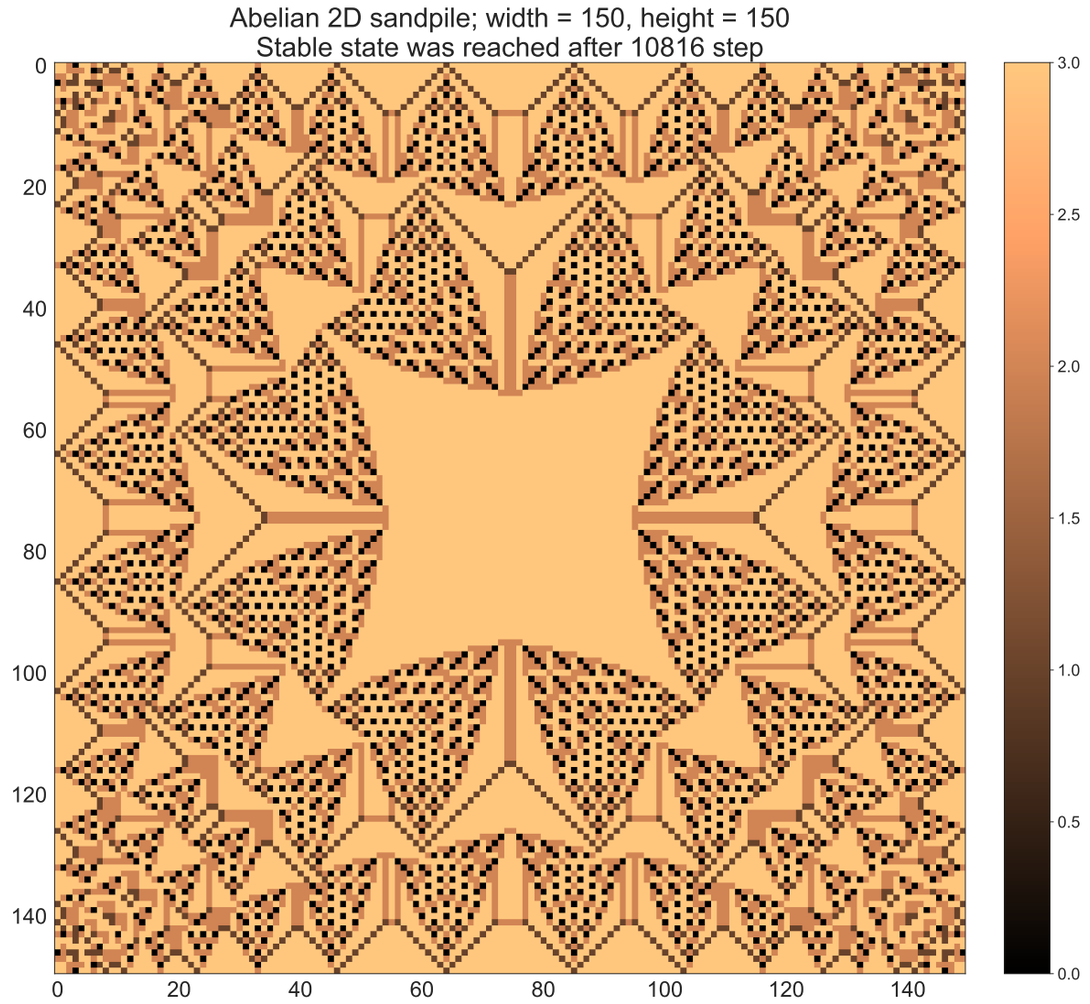
1. ábra. A Conway-féle Életjáték egyik oszcillátora, a *Kok's galaxy*. A játék ebben az esetben a standard, $N = 3$ szabályt használja. A rendszer egy nyolc-állapotú oszcillátor, mely egy galaxis-szerű forgást imitál.



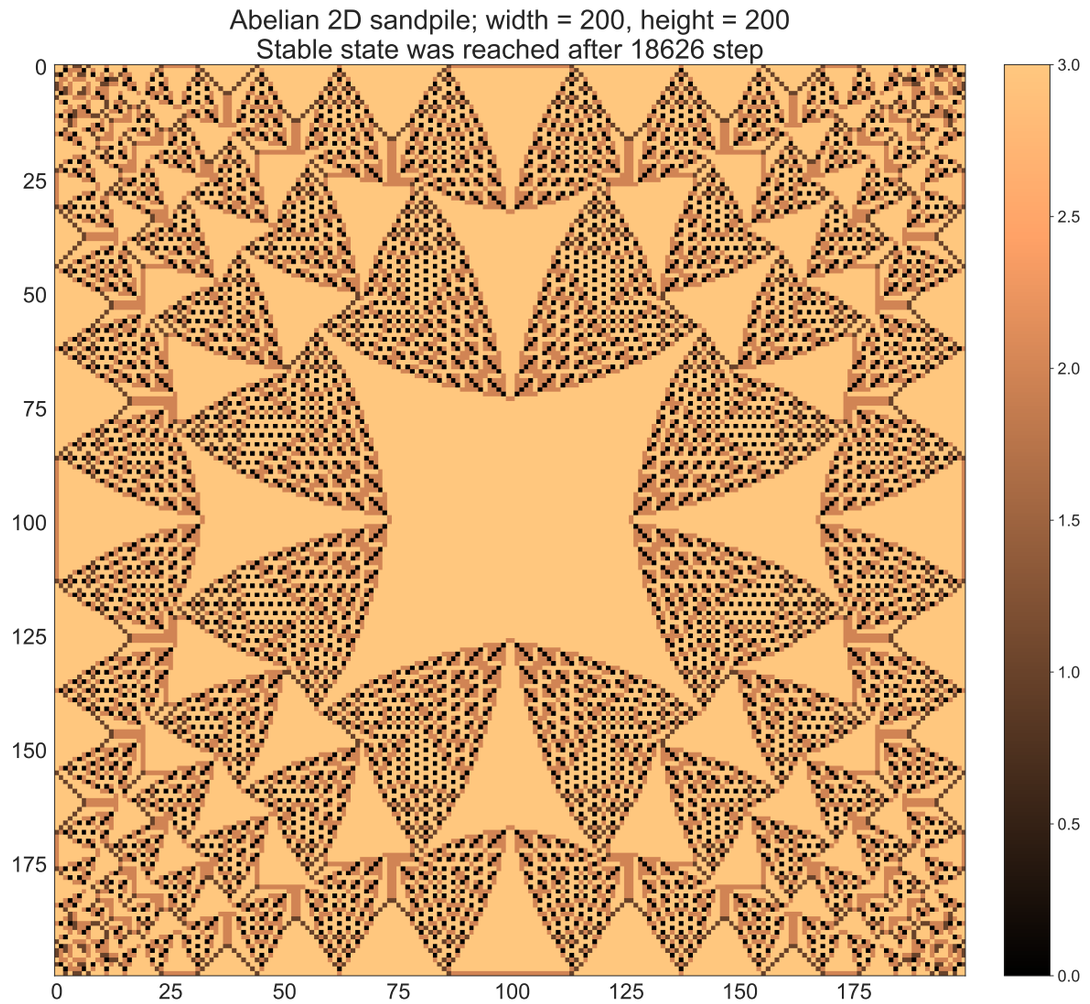
2. ábra. A Conway-féle Életjáték egyik űrhajója, a *Copperhead*. A játék ebben az esetben a standard, $N = 3$ szabályt használja. A Copperhead űrhajó $c/10$ sebességgel halad folyamatosan felfelé a játékmezőn. Az ábrán csak minden 6. lépés illusztráltam, hogy mozgásának iránya látható legyen.



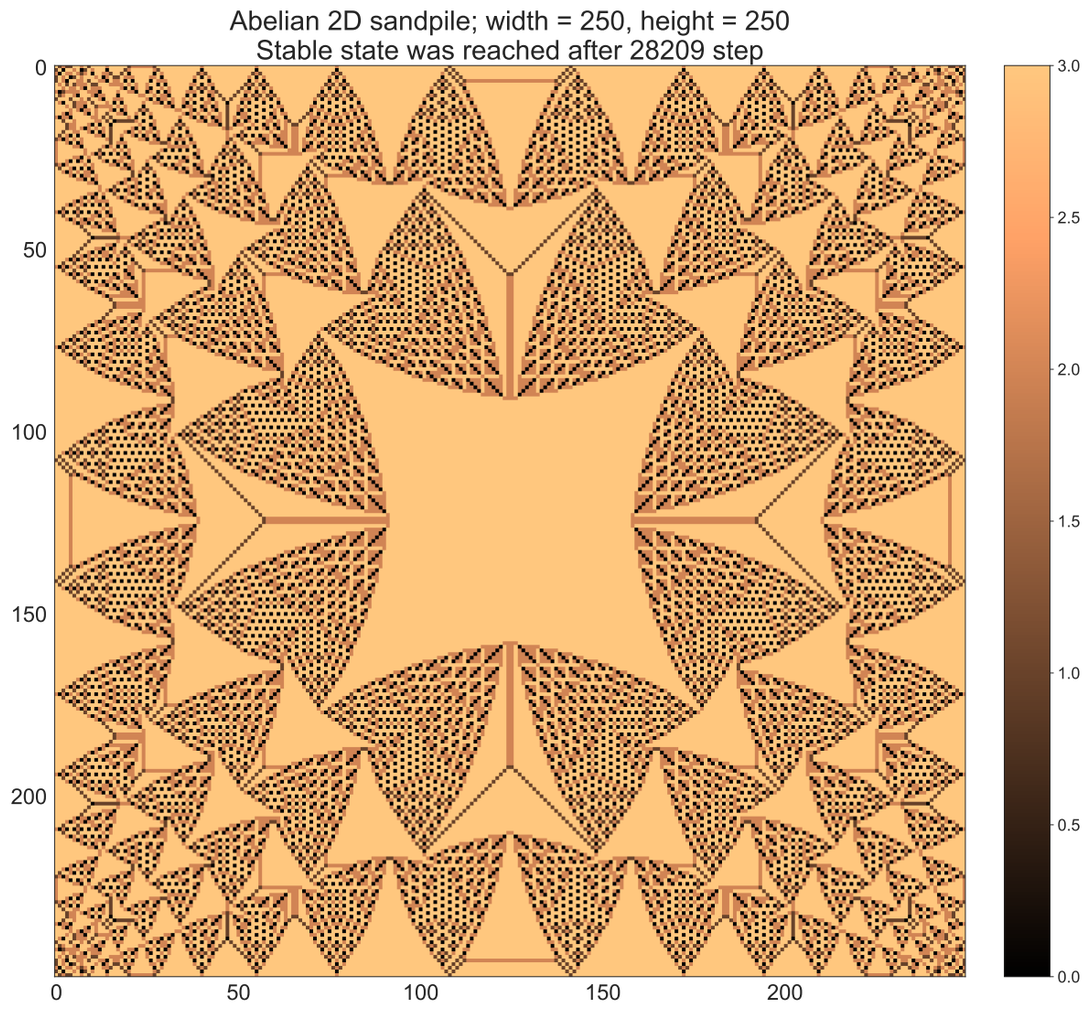
3. ábra. A Conway-féle Életjáték egyik *gun*-ja, a *Gosper glider gun*. A játék ebben az esetben a standard, $N = 3$ szabályt használja. A legelső ismert *gun* valójában két darab, egy-egy blokkal stabilizált *Queen bee shuttle*, melyek így minden periódusukban kibocsájtani egy $c/4$ sebességgel, diagonálisan haladó *Glider*t. Az ábrán csak minden 3. lépés illusztráltam, hogy a Gliderek kilövése és a forma oszcillációja jól látható legyen.



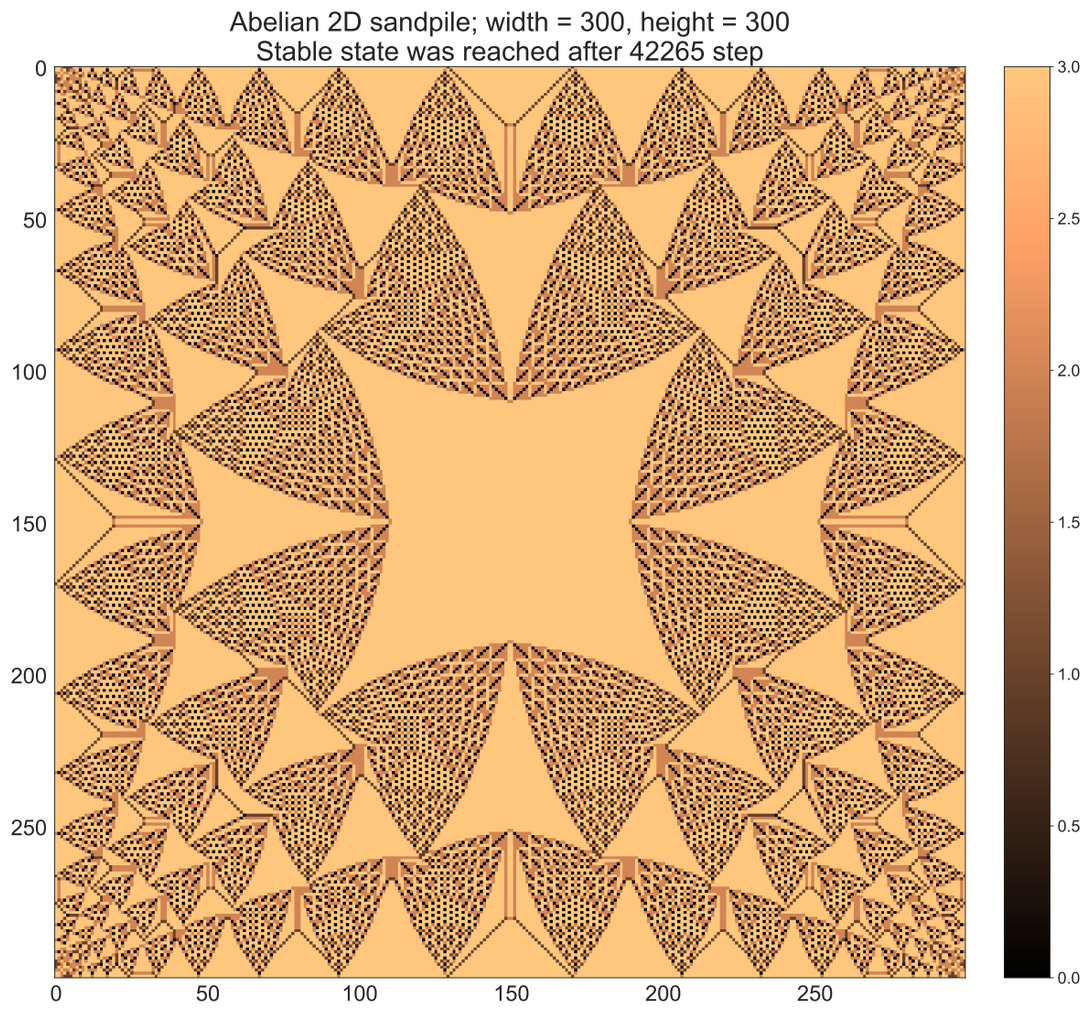
4. ábra. Az Abel-féle 2D homokdomb modell stabil állapota 150×150 nagyságú területre, $s = 7$ maximális toronymagasság esetén.



5. ábra. Az Abel-féle 2D homokdomb modell stabil állapota 200×200 nagyságú területre, $s = 7$ maximális toronymagasság esetén.



6. ábra. Az Abel-féle 2D homokdomb modell stabil állapota 250×250 nagyságú területre, $s = 7$ maximális toronymagasság esetén.



7. ábra. Az Abel-féle 2D homokdomb modell stabil állapota 300×300 nagyságú területre, $s = 7$ maximális toronymagasság esetén.