SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Računalna grafika

3D Stanični automat

Ante Žužul

3. laboratorijska vježba

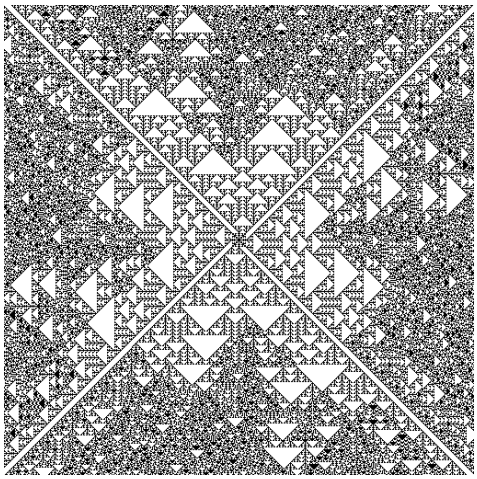
Zagreb, siječanj 2021.

Sadržaj

1. Teorija

Stanični automat diskretan je računski model i kao takav dio teorije automata. Stanični automat sastoji se od mreže stanica (čelija) od kojih se svaka nalazi u nekom od konačnog broja stanja. Mreža može biti proizvoljnih, ali konačnih dimenzija.

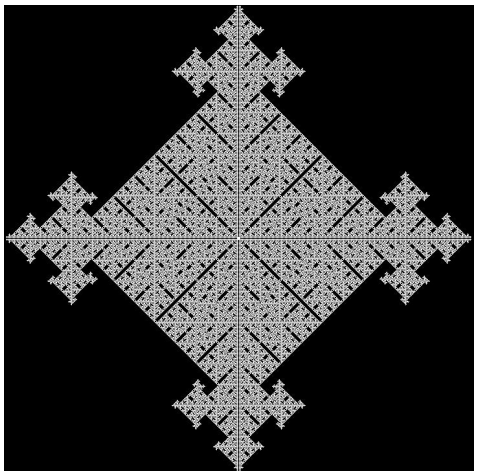
3D stanični automat proširenje je poznatijih 1D staničnog automata (Slika 1.1) i 2D staničnog automata (Slika 1.2). Umjesto da se provjerava susjedstvno s čelijama na X i Y osi, uvodi se i Z os.



Slika 1.1: Slika generirana 1D staničnim automatom.

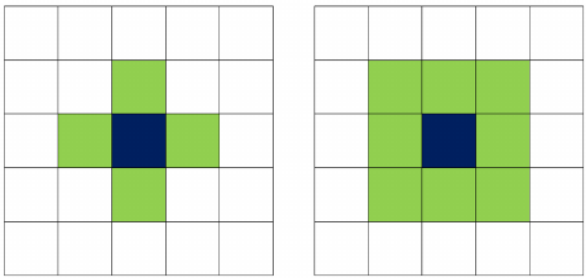
Za svaku stanicu definiran je skup susjednih stanica. Razlika susjednih stanica u 1D i 2D može se vidjeti na (Slika 1.3). Susjedstvo može biti definirano na razne načine, a o broju susjeda ovisi rađanje, umiranje i raspadanje stanice. Najčešći tipovi susjedstava stanica u staničnim automatima su Mooreovo [1] i Von Neumannovo [3] susjedstvo. Njihove razlike u 2D i 3D mogu se vidjeti na slikama (Slika 1.4) i (Slika 1.5). Očito je da je maksimalan broj susjeda za Von Neumanna 4 u 1D i 6 u 3D, dok je za Moorea maksimalan broj susjeda 8 u 2D te čak 26 u 3D. Naravno, minimalan broj susjeda za oba tipa je 0.

Iz načina određivanja susjedstva proizlaze pravila na kojima se temelji stanični automat. Ovisno o zadanom pravilu, a i o određenom broju susjeda, neka stanica će preživjeti, roditi se ili početi s raspadom dok konačno ne nestane.

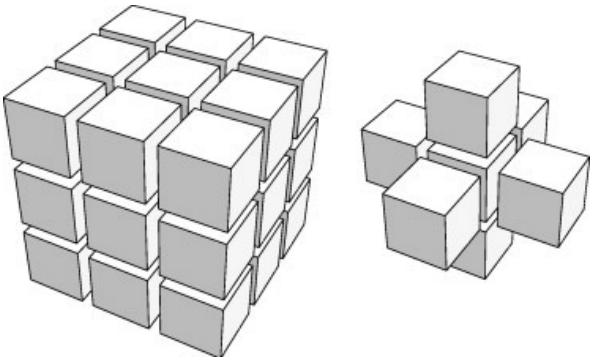


Slika 1.2: Slika generirana 2D staničnim automatom.

Slika 1.3: Usporedba susjedstva 1D (lijevo) i 2D (desno) staničnog automata.



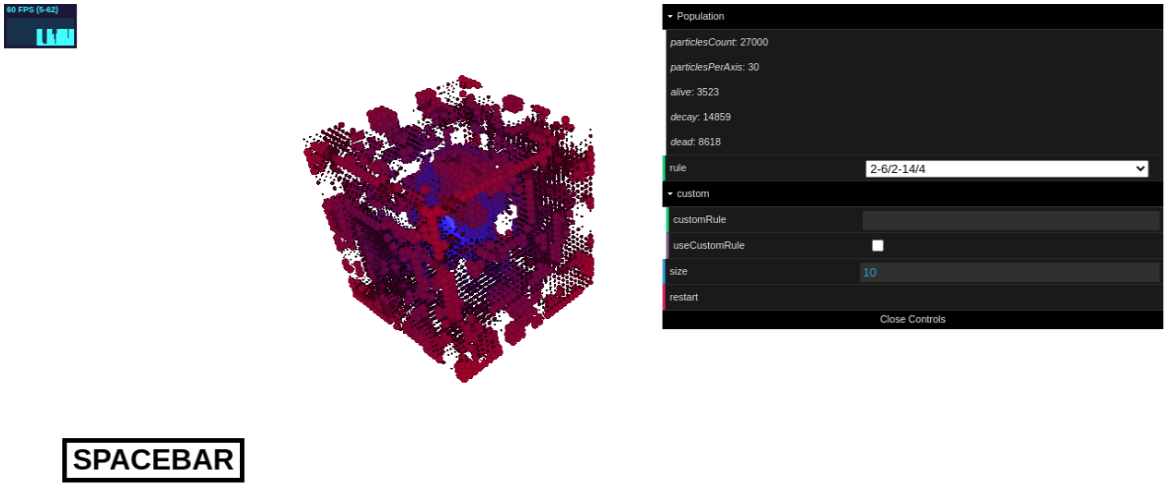
Slika 1.4: Usporedba Von Neumannovog (lijevo) i Mooreovog (desno) susjedstva u 2D.



Slika 1.5: Usporedba Mooreovog (lijevo) i Von Neumannovog (desno) susjedstva u 3D.

2. Funkcionalnost

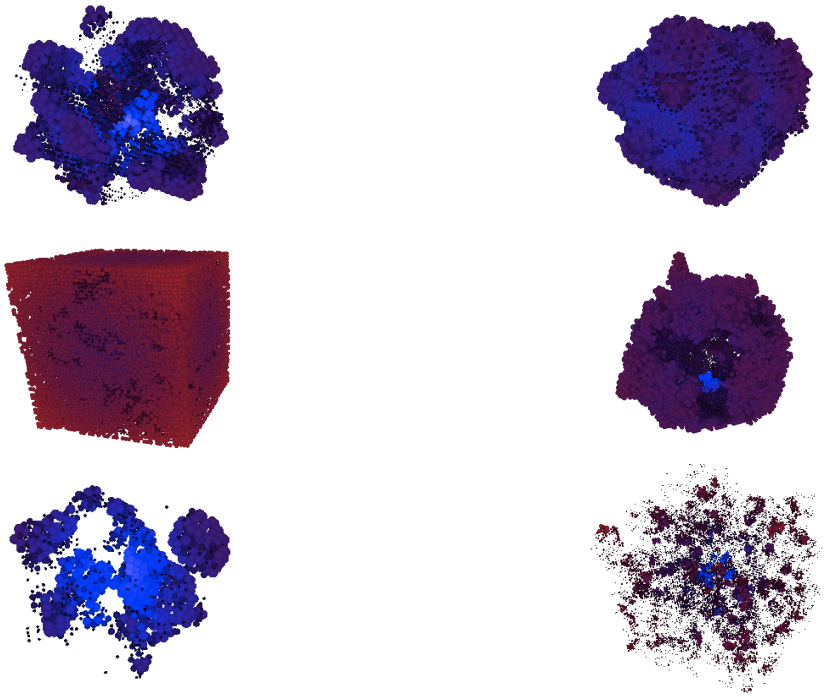
Na (Slika 2.1) može se vidjeti korisničko sučelje ostvareno **three.js** [2] - Javascript 3D bibliotekom. Sučelje omogućava korisniku odabir veličine početne populacije, odabir jednog od unaprijed određenih pravila, dodavanje vlastitiog pravila te resetiranje populacije. Također, korisniku je u svakom trenutku dostupan pregled broja živih, raspadajućih te mrtvih stanica, kao i broj ukupan broj stanica u sustavu i broj stanica po koordinatnoj osi. Preživljavanje, rađanje i umiranje stanica odvija se korak po korak, a prijelaz u novi korak korisnik može pokrenuti pritiskom na 'Space' (SPACEBAR).



Slika 2.1: Prikaz korisničkog sučelja

Početna populacija je nasumična. Iako je cijeli sustav deterministčan, broj mogućih oblika poprilično je velik zbog te nasumičnosti i zbog pravila koja mogu biti raznolika. Neke od zanimljivijih generiranih populacija prikazane su u nastavku (Slika 2.2).

Pravila su definirana u obliku $P/R/S$ [4] gdje P predstavlja potreban broj susjeda kako bi stanica preživjela, R potreban broj susjeda kako bi stanica oživjela, a S predstavlja broj stanja u kojima stanica može biti. Vrijednost za P ili S može biti broj (x) ili interval (x-y) ili kombinacija više njih odvojenih zarezom (x,y-z), npr. 9-26/5-7,12-13,15/5. Potrebno je upamtiti da je zbog Mooreovog susjedstva maksimalan broj susjeda 26.



Slika 2.2: Prikaz nekih od najzanimljivijih generiranih oblika.

Boje su odabrane tako da kod stanica najbližh središtu dominira plava boja, a što je stanica udaljenija od središta to je njena boja crvenija. Također, boja stanice ovisi i o stanju u kojem se nalazi, odnosno o njezinom skaliranju. Stanica će biti tamnija što više propada, odnosno što je više manja. Postupak određivanja boje jasno je vidljiv u kodu sjenčara fragmenata 2.3.

```
precision highp float;

uniform sampler2D map;

varying vec2 vUv;
varying float vScale;
varying float distance;

void main() {
    vec4 diffuseColor = texture2D( map, vUv );
    gl_FragColor = vec4(
        ( ( 0.01 / distance) * diffuseColor.x + 1.2 * distance * diffuseColor.x ) * vScale,
        ( ( 0.01 / distance) * diffuseColor.y + 0.0 * distance * diffuseColor.y ) * vScale,
        ( ( 0.1 / distance ) * diffuseColor.z + 0.0 * distance * diffuseColor.z ) * vScale,
        diffuseColor.w
    );
    if ( diffuseColor.w < 0.5 ) discard;
}
```

Slika 2.3: Prikaz GLSL koda sjenčara fragmenata.

3. Upute za pokretanje

Izvorni kod ovog projekta i vezane dokumentacije može se pronaći ovdje.

GitHub repozitorij može se klonirati ili preuzeti kao zip arhiva (u tom slučaju treba arhivu nakon preuzimanja i raspakirati). Implementacija se može pokrenuti na dva načina:

1. **NE PREPORUČA SE** Otvaranjem **index.html** datoteke u web pregledniku
   Ovaj način se ne preporuča jer je vrlo vjerojantno potrebno omogućiti web pregledniku dodatan pristup lokalnim datotekama što predstavlja mogući sigurnosni rizik.

2. Pozicioniranjem u direktorij preuzetog projekta te pokretanjem jednostavnog lokalnog servera
   Iako možda zvuči komplicirano, vrlo je jednostavno. Mnogi programski jezici nude ugrađene HTTP servere. Predlažem Python server.
   Nakon pozicioniranja u navedeni direktorij pokrenuti

   ```
   //Python 2.x
   python -m SimpleHTTPServer

   //Python 3.x
   python -m http.server
   ```

   što će posluživati datoteke iz trenutnog direktorija na adresi

   ```
   http://localhost:8000/
   ```

   te ćete unošenjem te adrese u web preglednik pokrenuti implementaciju. Oko dodatnih uputa i nejasnoća prilikom pokretanja **three.js** aplikacija preporučam ovaj link.

4. Literatura

[1] *Moore neighborhood*. URL https://en.wikipedia.org/wiki/Moore_neighborhood.

[2] *three.js*. URL https://threejs.org/.

[3] *Von Neumann neighborhood*. URL https://en.wikipedia.org/wiki/Von_Neumann_neighborhood.

[4] Softology's Blog. *3D Cellular Automata*. URL https://softologyblog.wordpress.com/2019/12/28/3d-cellular-automata-3/.