Gymnázium
Veľká okružná 22, 010 01 Žilina

Botanica - simulátor rastlín na báze celulárnych automatov

Stredoškolská odborná činnosť

Č. odboru: 11

Riešitelia: František Knapec, Michael Sklenka, Marek Beňo
Ročník štúdia: 4.

Mesto: Žilina
Rok: 2025

Gymnázium
Veľká okružná 22, 010 01 Žilina

Botanica - simulátor rastlín na báze celulárnych automatov

Stredoškolská odborná činnosť

Č. odboru: 11

Riešitelia: František Knapec, Michael Sklenka, Marek Beňo
Ročník štúdia: 4.
Školiteľ: PaedDr. Jana Pekárová, PhD.
Konzultant: Ing. Tomáš Milet, PhD.

Mesto: Žilina
Rok: 2025

Čestné vyhlásenie:

Prehlasujeme, že sme prácu na tému Botanica - simulátor rastlín na báze celulárnych automatov vypracovali samostatne s použitím literatúry uvedenej v zozname použitej literatúry. Zároveň prehlasujeme, že sme predloženú písomnú prácu neprihlásili a ani neprezentovali v žiadnej inej súťaži, ktorá je pod gestorstvom MŠMVVaŠ SR. Sme si vedomí zákonných dôsledkov, ak v nej uvedené údaje nie sú pravdivé.

Obsah

0 Úvod

1 Problematika a prehľad literatúry

Táto práca sa zaoberá implementovaním rastlín do digitálnej podoby a následným simulovaním ich správania relatívne k podmienkam, v ktorých sa nachádzajú, ako aj k iným rastlinám v ich okolí. Do tejto práce patrí aj vyobrazenie rastlín v trojdimenzionálnom priestore. Pôjde aj o schopnosť dlhodobého pozorovania ekosystému naprieč časom.

Ľudia k problému tvorenia digitálnych rastlín postupovali rôznymi spôsobmi, avšak väčšina z nich len realistické rastliny vykresľovali, ako je napríklad známy príklad L-systémov,[1] ktorý je založený na opisovaní štruktúry rastlín sériou pravidiel, ako je napr. otočenie, vytvorenie úsečky alebo vrátenie sa na určitú pozíciu. Ďalej sa štruktúra zdokonaľuje za pomoci algoritmu, ktorý nahrádza časti jej série pravidiel za iné, dopredu určené a detailnejšie série. Tento postup dokáže vytvoriť celkom realistické rastliny ako je znázornené na obr. 1.

Obr. 1: Príklad tráv vygenerovaných s použitím L-systému v 3D
(zdroj: https://en.wikipedia.org/wiki/L-system)

Toto riešenie a jemu podobné však dané rastliny len vytvárajú, no našim cieľom je ich aj simulovať.

1.1 Spôsoby simulovania rastlín

Kôli samotnej simulácii vzniklo viacero metód simulovania s vlastnými kladmi, ale aj zápormi. Medzi takéto metódy patria:

- **Fyzikálne založené modelovanie** využívajúce body prepojené pružnými štruktúrami, na ktoré pôsobia rôzne sily. Aj keď realistické, modelovanie neumožňuje rastliny tvoriť, iba simulovať.

[1] Lindenmayerov systém - https://en.wikipedia.org/wiki/L-system

- **Simulácia pomocou strojového učenia**[2] využíva štatistické algoritmy, pomocou ktorých sa počítač dokáže počas tréningu učiť vzťahy medzi vstupmi a výstupmi.

- **Celulárne automaty**[3] je názov pre matematický model a nástroj pre simuláciu. Jedná sa o starší koncept, ktorý obsahuje štvorcovú sieť so „zafarbenými“ políčkami. Každá farba je určitý stav, ktorý má špecifické správanie závisiace od okolitých políčok.

Fyzikálne založené modelovanie, aj keď mocné, vyžaduje aby rastliny boli vygenerované dopredu, pred začatím simulácie, a po jej spustení nie sú tieto rastliny schopné sa ďalej vyvíjať. Navyše spôsob vyžaduje fyzikálne založenú simuláciu, ktorá je príliš komplikovaná pre túto prácu.

Na druhú stranu, strojové učenie sa stáva čím ďalej tým využívanejšie pre riešenie všetkých možných problémov. Avšak, tento spôsob je náročný na čas a výpočtovú techniku.

Nakoniec Celulárne automaty ponúkajú jednoduchosť a možnosť ich jednoducho modifikovať, čo otvára pole možných využití. Vďaka týmto výhodám je simulácia v tejto práci založená na princípe celulárnych automatoch.

1.2 Celulárne automaty

CA sa skladá z týchto základných častí: mriežka (často štorcová sieť) a bunky tejto mriežky, ktoré majú vlastný stav, pričom každý stav obsahuje určité pravidlá, ktoré zapríčiňujú správanie buniek, a tým celého systému. Tieto pravidlá sa aplikujú každú generáciu, každé simulačné kolo. Vďaka týmto vlastnostiam sú CA systém, ktorý je jednoduché prispôsobiť ktorýmkoľvek požiadavkám.

Klasickým príkladom takéhoto systému je Conwayova hra života.[4] V tejto hre nadobúda každá bunka štvorcovej siete jednu z dvoch hodnôt: buď je bunka živá, alebo mŕtva. Conwayova hra života má nasledovné pravidlá:

1. Každá živá bunka s menej ako dvoma živými susedmi umrie.

2. Každá živá bunka s dvomi alebo tromi živými susedmi prežie.

[2] Strojové učenie - https://en.wikipedia.org/wiki/Machine_learning

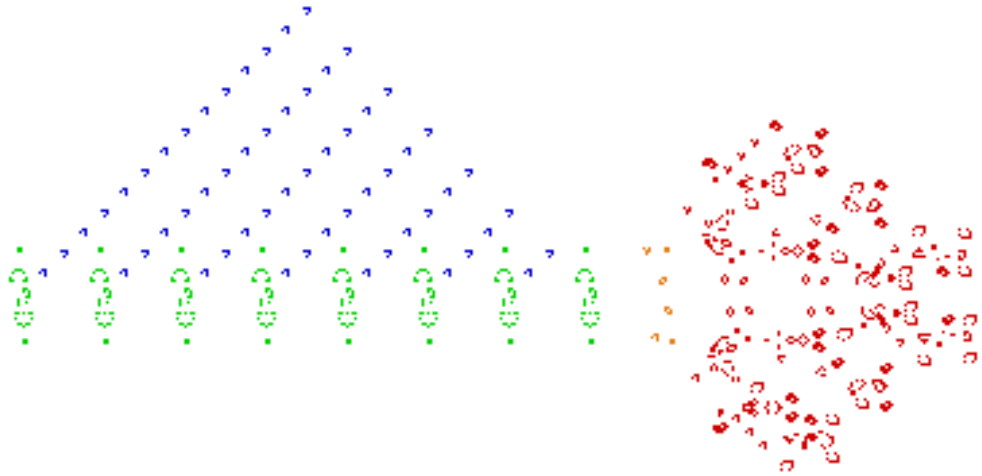
[3] Celulárne automaty - https://en.wikipedia.org/wiki/Cellular_automaton

[4] Conwayova hra života - https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

3. Každá živá bunka s viac ako tromi živými susedmi umrie.

4. Každá mŕtva bunka s presne tromi živými susedmi ožije.

V hre života sa nachádza veľa stabilných configurácií (glidery, továrne, oscilátory, atď.) ako je vidno na obr. 2. Vďaka týmto štruktúram je Conwayova hra života turingovo kompletná (môže simulovať hociaký počítač, čiže aj sama seba[5]).



Obr. 2: Na obrázku je znázornená pohyblivá štruktúra (červená), ktorá za sebou zaneháva továrne (zelená) produkujúce glidery (modrá), jednoduché pohybujúce sa štruktúry.

1.2.1 Príklady CA zamerané na simuláciu rastlín

„Simulation of root forms using cellular automata model“ - zameriava sa na korene rastlín a dopad premenných ako napr. živiny v pôde, voda a prekážky v raste. Jedná sa o prácu, ktorá je viac zameraná na matematický rast koreňov, avšak my sme rast koreňov simulovali genetikou.

„A Model Based on Cellular Automata for the Simulation of the Dynamics of Plant Populations“ - zameriava sa na 2D simuláciu heterogénnej populácie rastlín, zameraná na konkurenciu medzi druhmi, rýchlosť rastu a interakcie s prostredím.

[5]Life in life - https://www.youtube.com/watch?v=xP5-iIeKXE8

2 Ciele práce

Cieľom tejto práce je vytvorenie aplikácie schopnej vytvárania a simulácie pseudo-realistických rastlín v trojrozmernej mriežke pomocou celulárnych automatov a implementovanej genetiky. Tieto rastliny by mali byť schopné reagovať na prostredie v reálnom čase a adaptovať svoje správanie, tvar a iné charakteristiky v závislosti od podmienok, v ktorých sa nachádzajú.

Výsledkom tejto práce je softvér, Botanica, ktorý umožňuje pozorovanie správania a vývinu simulovaných rastlín vďaka schopnostiam:

1. generovať časť 3D voxelového sveta za pomoci Perlinovho šumu,
2. simulovať rastliny vo vytvorenom prostredí pomocou vlastného algoritmu inšpirovaného celulárnymi automatmi,
3. vykresľovať túto simuláciu za pomoci grafického API OpenGL.

Veríme, že metódy využité v Botanice sú využiteľné ako na vytváranie dynamického a nerepetetivného prostredia vo video-hrách, tak aj na viac vedecké účely ako pozorovanie správania rastlín vo virtuálne vytvorených podmienkach a ich ideálne charakteristiky za daných podmienok.

3 Aplikácia (Materiál a metodika)

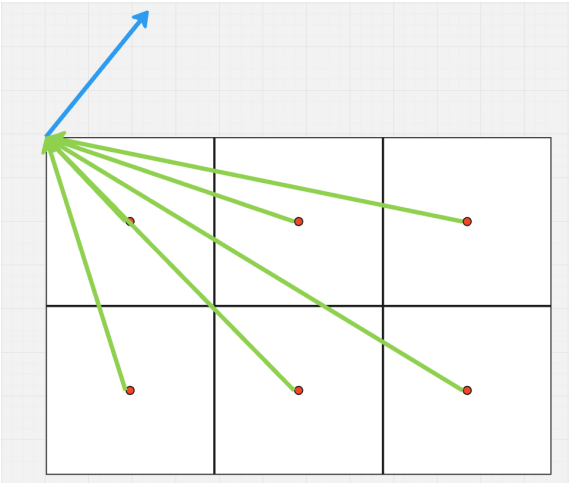
Na realizovanie algoritmu sme vyvinuli jednoduchú aplikáciu, ktorá má všetky potrebné prostriedky na demonštráciu všetkých aspektov tohto algoritmu. Aplikácia umožňuje vizualizáciu rastu rastlín na základe celuárnych automatov a poskytuje možnosť meniť parametre simulácie na dosiahnutie idealného vzhľadu rastlín.

3.1 Svet

Svet aplikácie pozostáva z mriežky buniek o rozmeroch 32x32x32 voxelov. Každý voxel nadobúda určitý stav. Môže sa jednať o časti rastlín (stonka, list, koreň alebo ovocie), ale aj o časti terénu (vzduch, voda, pôda).

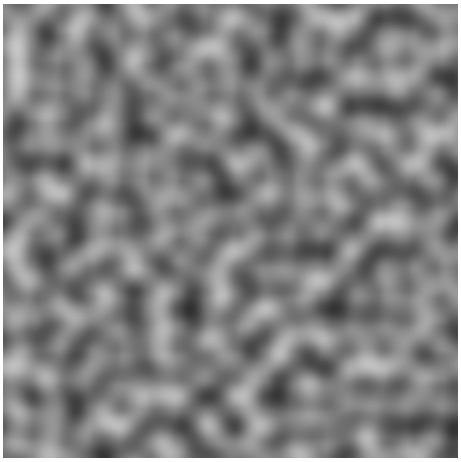
3.1.1 Perlin noise

Terén na určenie svojej výšky, aby nebol len rovina, využíva funkciu zvanú Perlin noise. Perlin noise je spôsob generovania plynule sa meniacich náhodných hodnôt, ktoré sa dajú skvelo využiť na generovanie realistického terénu. Čo sa týka algoritmu na jeho generovanie (obr. 3), v určitých miestach sa vytvoria vektory s náhodným smerom (modrý). Z bodov (červené), ku ktorým chceme pripísať hodnoty, sa vytvoria vektory smerujúce k týmto miestam (zelené). Z náhodného vektoru a vektoru smerujúceho z bodu sa vytvorí skalárny súčin a tým dosiahneme hodotu v danom bode.



Obr. 3: Znázornenie generovania Perlinovho šumu za pomoci 1 náhodného vektoru pre 6 buniek.

Avšak väčšinu času sa hodnoty nepočítajú len pre jeden náhodný vektor, ale viacero, najčastejšie 4 najbližšie, a hodnoty ich skalárnych súčtov sa spriemerujú. Často sa využíva aj viacero oktáv[6], pričom každá oktáva využíva viacej náhodných vektorov, ktoré ovplyvňujú menej buniek, ako predchádzajúca oktáva. Avšak tieto oktávy vplývajú menej na výslednú hodnotu. Vyjadrenie hodnôt bodov farebnou škálou kde vyššie hodnoty sú viac biele a nižšie viac čierne popisuje obrázok 4.



Obr. 4: Textúra Perlinovho šumu.
(zdroj: https://en.wikipedia.org/wiki/Perlin_noise)

Naša aplikácia, vďaka jej obmedzenej veľkosti, využíva 4 vektory umiestnené na rohoch simulácie a iba 1 oktávu.

3.1.2 Tvorenie terénu

Tvorenie terénu sa začne vytvorením Perlin noisu, poďla ktorého sa určí jeho výška. Pre každú dvojicu súradníc X a Z sa vytvorí hodnota Perlin noisu. Táto hodnota bude ovplyvňovať os Y. Každý voxel, ktorého hodnota súradnice Y je nižšia alebo rovná hodnote noisu v X,Z bude voxel terénu. Inak sa bude jednať o voxel vzduchu. Na vytvorenie voxelov vody sa nastaví hodnota, ktorá bude označovať hladinu vody a ak hocaký voxel vzduchu má hodnotu Y menšiu alebo rovnú ako táto hodnota, zmení sa na vodu. Táto hodnota nie je spojená s noisom, ide o jedno číslo univerzálne pre celú simuláciu. Zoberme si príklad: máme zmenšenú simuláciu a pre každú dvoju X,Z pripadajú len 4 voxely [0,1,2,3]. Tieto voxely majú rovnaké súradnice X,Z ale odlišné Y, ktorá korešponduje s ich číslom. Vytvoríme Perlin noise, ktorý bude mať pre danú dvoju X,Z hodnotu 1. To znamená, že

[6]Oktáva je v tomto prípade mapa vytvorená Perlinovím šumom.

voxely s hodnotou Y 0 a 1 sa premenia na voxely pôdy a voxely 2,3 na voxely vzduchu. Ďalej hladina vody má hodnotu 2, takže voxel 2, ktorý bol pred tým vzduch sa premení na vodu.

3.1.3 Živiny

Simulácia obsahuje rôzne živiny, vodu a vzduch, tieto veci rastlina potrebuje získavať aby rástla a prežila. Voda a vzduch majú zastúpenie v podobe ich voxelov. Pôda obsahuje vodu a živiny. Každá živina napomáha rôznym procesom rastliny, ako napríklad fotosyntézovať, čerpať vodu alebo získavať viac živín. Ich nadbytok má kladný vplyv na rastliny, avšak ich nedostatok má zase negatívny. Toto pridáva na komplexnosti, realizme a dovoľuje nám získať rôzne typy rastlín, podľa toho, ku ktorým živinám majú alebo nemajú prístup. Napríklad ak rastlina nemá dostatok dusíka, ktorý je v simulátore dôležitý na fotosyntézu, rastlina to musí kompenzovať viacerými listami, inak zanikne. Živiny sú 3, dusík, draslík a fosfor, pričom bola snaha o to, aby napomáhali realistickým precesom, ktoré tieto prvky vyžadujú. Draslík je nápomocný pri absorpcii vody, takže ak má rastlina nadbytok draslíka, efektívnejšie získava vodu. Využitie dusíka je široké, ale v simulátore je jeho funkcia obmedzená len na zlepšenie fotosyntézy, keďže je hlavnou zložkou chlorofylu. Fosfor je rovnako ako dusík dôležitý v rôznych častiach, avšak v simulátore je zameraný na absorpciu živín.

3.2 Rastliny

Rastliny sú v tomto simulátore objekty so spoločnou „triedou.“ Trieda je predloha premenných a špeciálnych funkcií zvaných metód, ktoré využívajú jednotlivé entity, taktiež známe ako objekty. Takáto šablóna nám umožňuje jednoducho a usporiadane vytvárať rastliny, pretože nemusíme písať samostatný kód pre každú rastlinu.

3.2.1 Premenné v rastlinách

Rastliny majú viacero premenných. Ich hlavnou úlohou je opísať stav rastliny. Medzi ne paria údaje o skonzumovaných živinách za toto kolo, zoznam všetkých voxelov, v ktorých sa nachádzajú časti rastliny a ich gény. V premenných taktiež máme aj bonusy pre živiny. Hovoria nám o nadbytku alebo nedostatku živín. Čo tieto bonusy robia je rozbrané v kapitole „Živiny“.

3.2.2 Rastlinné voxely

Fyzické zobrazenie rastlín sa skladá zo špecializovaných voxelov, ktoré v mriežke reprezentujú ich časti. Tieto voxely taktiež voláme aj bunky. Existujú 4: bunky koreňov, stonky, listov a ovocia. Každá skupina buniek plní vlastnú úlohu. Koreň má za úlohu rastline získať živiny a vodu, to dosahuje odoberaním týchto živín z okolitých voxelov. Koreň môže vzniknúť iba na miestach, kde je pôda. Úlohou listu je fotosyntéza, tú dosiahne tým, že skontroluje, či políčka nad ním sú vzduch. Predpokladáme, že na voxeli vzduchu priamo svieti slnko. Môže nahradiť políčka kde je vzduch. Stonka uskladňuje živiny pre rastlinu. Rastie iba zvislo. Ovocie vytvára nové rastliny s rovnakým genetickým kódom ktorý trochu zmutuje. Na pár kôl nahrádza 1 list, ktorý sa po premenení ovocia na novú rastliny znova zmení na list.

3.2.3 Genetika

Na základe genetiky rastliny rozhodujú, aké časti majú rásť (listy, stonka…) a v prípade listov a koreňov sú gény, ktoré hovoria, akým smerom majú rásť. Takže existujú 3 gény: čo má rásť, kde majú rásť korene a kde majú rásť listy. Gén „čo má rásť“ je vo všetkých prípadoch list so 4 číslami. Sú 4, pretože sú 4 typy buniek, ktoré môže rastlina rásť: korene, listy, stonku a ovocie. Čísla v týchto listoch označujú, ako moc chce daná rastlina

danú časť rásť. Takže ak má rastlina v časti označujúcu stonku 2 a koreň 10, daná rastlina bude chcieť koreň rásť 5-krát viac, ako stonku. Ktorú časť bude rastlina naozaj rásť sa rozhoduje náhodne. Takže ak sa vrátime k nášmu príkladu dokopy 2+10=12, takže to je ako keby naša rastlina hádzala s 12-stenou kockou. Ak padne číslo, ktoré je väčšie ako 2, rastie koreň, inak stonka. Gény „kde majú rásť korene“ a „kde majú rásť listy“ sú rovnaké, len ako ich názov hovorí, jeden sa zaoberá koreňmi a druhý listami. Sú to listy s 26 číslami. Je ich 26 pretože bunka má v trojrozmernom priestore 26 susedov, ak rátame aj diagonály. Zvyšok funguje rovnako ako gén „čo má rásť“. Takže sú tam čísla, ktoré označujú pravdepodobnosť rastu do toho smeru. Ak sa tieto 2 gény zavolajú vyberie sa náhodná bunka (list alebo koreň, podla toho, ktorý gén sa zavolá) a ak nemôže rásť do toho smeru, tak sa vyberie iná bunka, ak žiadna nemôže rásť, vyberie sa iný smer.

3.3 Algoritmus

Ako bolo spomínané, simulácie nebeží súvisle, ale v časových „krokoch“. Za každý krok sa vykonajú určité akcie, tzv. cyklus. Tieto akcie slúžia na beh simulácie a starajú sa o veci ako je rast rastlín, ich vzniknutie, smrť atď. Algoritmus za týmto môžeme rozdeliť do nasledujúcich častí, ktoré sú spísané chronologicky, čiže postupne, ako ich simulácia vykonáva.

3.3.1 Vznik rastlín

Je jediná časť algoritmu, ktorá nebeží v krokoch, ale iba raz, a to na začiatku simulácie. Po vytvorení terénu sa vyberie náhodné miesto na jeho povrchu, kde je pôda, a tam vznikne rastlina. Toto sa dá opakovať viackrát, na vytvorenie viacerých rastlín. Takéto rastliny sa skladajú z 1 koreňa, 1 stonky a 1 listu. Ich genetika je náhodná, keďže ju nemôžu dediť z materínskych rastlín, ktoré neexistujú.

3.3.2 Redistribúcia živín

Je prvá akcia, ktorá sa vykonáva v cykle, každý krok simulácie. Živiny sú totiž ukladané v dvoch listoch, jeden permanentný, ktorý drží informácie o počte živín a vody zo začiatku simulácie a druhý dočasný, z ktorého rastliny čerpajú. V tomto bode sú odobrané všetky živiny, ktoré sa udržiavajú v rastlinách a list dočasných živín je znovunastavený na hodnoty rovné permanentnému listu. Toto umožňuje rastlinám znovu brať živiny v novom cykle.

3.3.3 Zasadenie rastlín

Ak má rastlina ovocie, ktoré existuje po určitú dobu, tak sa z neho stane nová rastlina. Rozdiel vo vzniku a zasadení je, že tentokrát genetický kód nie je vygenerovaný náhodne, ale ho rastlina zdedí zo svojej materinskej rastliny, pričom v každej časti genetiky jednu hodnotu náhodne zmení, teda „zmutuje“.

3.3.4 Produkcia

V tejto fázy rastlina získava živiny vďaka svojím bunkám, konkrétne listom a koreňom. Listy sa pozrú na bunky horizontálne nad nimi a podla toho, koľko z nich sú vzduch, „pro-

dukujú“ svetlo. Korene zase pozrú bunky v ich susedstve a podla toho generujú živiny a vodu. Koľko z okolitých buniek dokážu listy a korene dostať je určený samozrejme, počtom živín v bunkách terénu, z ktorého rastlina čerpá, ale rastlina nezoberie jednoducho všetky živiny v okolitom teréne. Maximálny počet živín, ktoré dokáže bunka získať je stanovený konštantou a jej bonusom zo živín. V stonkách sa uskladňujú živiny, čiže rastlina nemôže skonzumovať viac, ako jej to limit odvodený z počtu buniek stoniek dovoľuje. Na príklad: chceme fosfor, je koreň s 5 okolitými bunkami, pričom každá obsahuje 15 jednotiek fosforu. Konštantu produkcie máme 10, povedzme že máme bonus pre fosfor krát 1.2, čiže z každej dokážeme brať až 12 jednotiek. To máme do kopy 60 jednotiek, ktoré dokáže daný koreň získať, ale povedzme, že kapacita stoniek je len 50, takže náš koreň nakoniec vyprodukuje len 50 jednotiek fosforu.

3.3.5 Smrť

Ak rastlina má nedostatok jednej zo živín, zomrie. Koľko živín rastlina potrebuje na prežitie je odvodené od jej veľkosti, čiže počtu jej voxelov a premennej určenej na začiatku simulácie zvanej „obtiažnosť“. Ako príklad dajme rastlinu s 10 bunkami, obtiažnosťou 5 a 45 jednotkami živiny, ktorú pozorujeme ako napr. fosfor. Vynásobíme počet buniek a obtiažnosť, takže 10*5=50. To znamená, že rastlina potrebuje najmenej 50 jednotiek zo všetkých živín, aby prežila. Keďže má len 45 jednotiek fosforu, zomrie na jeho nedostatok.

3.3.6 Kalkulácia bonusov zo živín

Podobne ako smrť, sa odvíja od veľkosti rastliny a konštanty. Pre každý bonus sa ráta zvlášť a keďže tento krok je až na konci cyklu v úplne prvom cykle simulácie majú všetky rastliny bonusy rovné 1.

3.4 Renderovanie

Aby nebola simulácia len v podobe čísiel v súbore, je simulovaný svet vykresľovaný na obrazovku. Renderovanie je prevedené na grafickú kartu, processor má tak viac času venovať sa simulácii. Na renderovaie bolo vybrané grafické API OpenGL pre jeho jednoduchosť a kompatibilitu medzi operačnými systémamy.

Ako už bolo spomenuté v časti 3.1, svet je zložený z malých kociek (voxelov) v sieti 32x32x32. Voxely môžu byť tovrené rôznymi materiálmi, čo je symbolizované číselnou hodnotou pre každý jeden voxel. Tieto hodnoty sú po každej zmene sveta nahrané na grafickú kartu pomocou grafického API OpenGL.

Grafická karta následne vygeneruje sieť bodov (vertexov) s farebnými a polohovými hodnotami, ktoré sú následne každý snímok vykreslené na obrazovku.

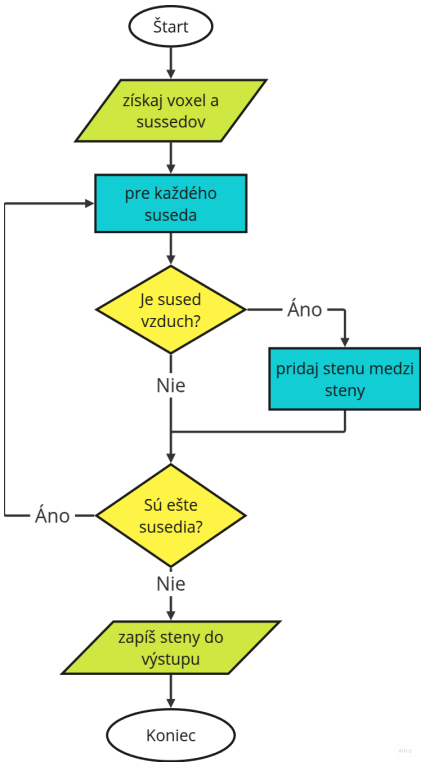
3.4.1 Tvorenie vertexov z voxelových dát sveta

Na začiatok treba povedať, že grafické karty spracovávajú dáta paralelne, čo im umožňuje "prehrýzť"sa obrovským množstvom dát za okamih. Je to niečo, na čo treba myslieť pri programovaní výpočtových programov (compute shaderov).

Shader je spustený niekoľkokrát, raz na každý voxel. Jeho výstupom je neprerušená sieť vertexov, ktoré sú následne zaslané na renderovanie. To je dosiahnuté globálnym atomickým počítadlom.[7]

Dáta sú shaderu poslané v jednom veľkom bloku pamäti, čiže shader potrebuje vedieť, kde sa jemu priradený voxel v tejto pamäti nachádza. Preto má každá invokácia (každý spustený shader) priradené poradové číslo, ktoré je použité ako index voxelu v pamäti, z ktorej je vytiahnutá nielen hodnota jemu priradenému voxelu, ale aj každého susedného voxelu. Tie sú následne využité na tvorenie stien (obr. 5).

[7] Atomické počítadlo je číslo, ktoré môže zvýšiť iba jeden program naraz. Táto vlastnosť zabraňuje prepisovanie už zapísaných dát.

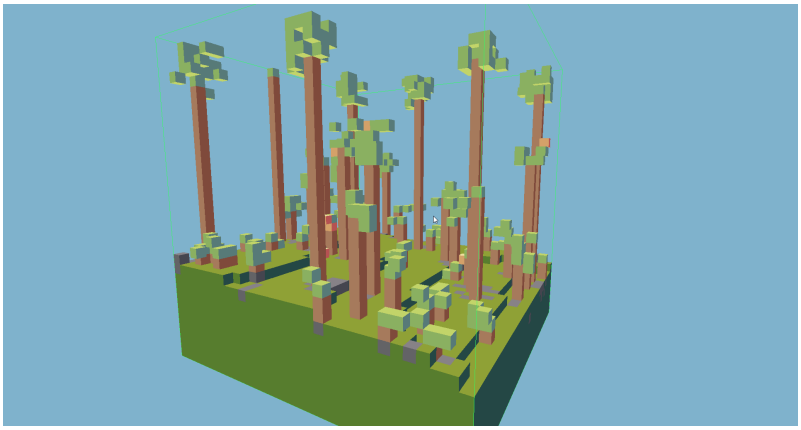


Obr. 5: Znázornenie algoritmu generujúceho steny (4 vertexy) okolo priradeného voxelu.

3.4.2 Vykresľovanie vygenerovaných dát

Keď sú dáta vygenerované, nie je zložité ich vykresliť pomocov jednoduchého programu pozostávajúceho z dvoch častí:

- **vertex shader**, počíta otáča a posúva body pomocou dát kamery, premieta 3D body na 2D obrazovku,
- **fragment shader**, vyfarbuje trojuholníky tvorené vertexmi.



Obr. 6: Vyrenderovaný snímok simulácie.

4 Výsledky práce

5 Diskusia

6 Závery práce

7 Zhrnutie