Proceduralno generisanje

Seminarski rad u okviru kursa
Tehničko i naučno pisanje
Matematički fakultet

Teodora Nićković, Darinka Zobenica
mi16057@alas.matf.bg.ac.rs, mi16075@alas.matf.bg.ac.rs

Sažetak

Proceduralno generisanje (eng. procedural generation) je metoda koja se često koristi u grafičkom dizajnu, pogotovo u vizuelnim efektima za filmove i, pre svega, video igrama. Ovaj tekst objašnjava šta je to zapravo proceduralno generisanje, pokazuje na jednostavnom algoritmu kako funkcioniše, primere kako se koristi i gde je sve bila korišćena i koje su joj prednosti i mane.

1 Uvod

U računarstvu, proceduralno generisanje sadržaja je, nasuprot ručnom, metod algoritamskog kreiranja sadržaja, uz nikakav, indirektan ili ograničen korisnički uticaj. Zove se još i nasumično generisanje (eng. random generation). Sam naziv označava ono što funkcioniše uz pomoć procedura ili potprograma sa svrhom kreiranja novog sadržaja. Kako je ova definicija prilično opšta, može obuhvatiti dosta toga, od jednostavnih stvari kao što su generatori nasumičnih brojeva, do generisanja čitavih svetova u video igrama [1]. Međutim, nameće se pitanje, kako se ovo razlikuje od veštačke inteligencije? Ključna reč u ovoj definiciji je "sadržaj". Pod time podrazumevamo sve što se sadrži u nekoj igrici, pesmi, ili bilo čemu što generišemo (mape, nivoi predmeti, vozila, oružje, note, grafike, itd). Sama funkcionalnost igrice ne spada u sadržaj kao takav, pa samim tim ni algoritmi za ponašanje neigrivih likova (eng. non-playable characters). Mehanizmi za njihovu "inteligenciju" su uglavnom finiji i iza njih stoji mnogo više istraživanja i testiranja, nego u slučaju proceduralnog generisanja sadržaja.

2 Analiza nekih algoritama za proceduralnu generaciju na primerima

Svi algoritmi za proceduralnu generaciju funkcionišu , u manjoj ili većoj meri, na sličan način:

1. Nasumično postaviti sadržaj nekog tipa u neko okuženje (niz, matricu, mapu igrice...)

2. Proveriti da li prethodno postavljen sadržaj ima smisla na mestu na kom je postavljen

3. Proveriti da li taj sadržaj može da, i kako interaguje sa igračem, posmatračem...

4. Ponoviti ove korake dok se ne dobije smislena celina

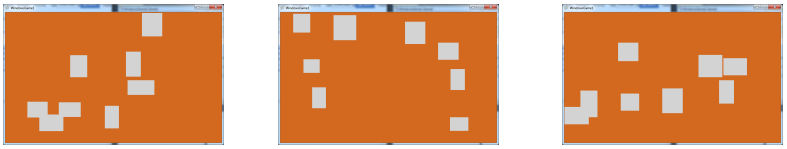
Radi pojednostavljenja izlaganja, u ovom radu fokusiraćemo se na (pojednostavljeni) algoritam koji se koristi u roguelike igrama, što je jedan od prvih žanrova u kojima se koristi proceduralno generisanje. Slični algoritmi se, sa malim izmenama, koriste i u ostalim sferama proceduralnog generisanja, o kojima će i kasnije, manje detaljno, biti reči.

Prvo ćemo kreirati sobe (pećine) koje ćemo postaviti na mapu. Postavljanje soba izvešćemo tako što ćemo ih predstaviti preko pravougaonika, a koordinate za iscrtavanje tih pravougaonika nasumično generisati, kao i dužinu i širinu pojedinačnih soba. Sada, kada smo implementirali generisanje jedne sobe, potrebno je da se ovaj proces ponavlja sve dok nemamo neki određen broj soba koje čine mapu. To postižemo jednostavnom petljom.

```
//Generisanje soba
for (j = 0; j<8; )
{
//Generisanje nasumičnih koordinata za sobe (PRVI KORAK)
    w = MIN_ROOM_SIZE + r.Next(0, MAX_ROOM_SIZE-MIN_ROOM_SIZE);
    h = MIN_ROOM_SIZE + r.Next(0, MAX_ROOM_SIZE-MIN_ROOM_SIZE);
    x = r.Next(0, WINDOW_WIDTH - w);
    y = r.Next(0, WINDOW_HEIGHT - h);

    Room room = new Room(x, y, w, h);
    rooms.Add(room);
    j++;
    }
```

Sada ćemo pokušati da nekoliko puta generišemo mapu (slika 1).



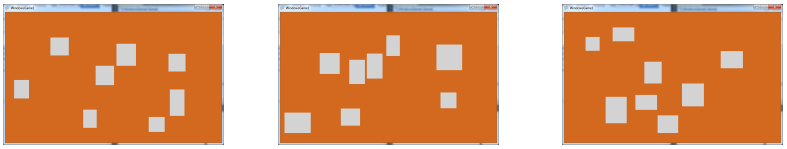
(a) (b) (c)

Slika 1: Primeri

Vidimo da ovde nastaje problem kada se neke od soba preklapaju. S obzirom da ne želimo da se to dešava, napravićemo funkciju koja, pri generisanju svake nove sobe, proverava da li se ona preklapa sa nekom od prethodnih. Ako da, bez isrtavanja te sobe prelazimo na novu iteraciju petlje, a ako ne, onda se trenutna soba iscrtava i dodaje u listu.

```
//Proveravanje da li se neke od soba preklapaju (DRUGI KORAK)
     bool failed = false;
     for (i = 0; i < rooms.Count; i++)
     {
         if (room.PreklapaSe(rooms[i]))
         {
             failed = true;
         }
     }
     if (!failed)
     {
         rooms.Add(room);
         j++;
     }
```

Pokušajmo sada ponovo da nekoliko puta generišemo mapu (slika 2).



(a) (b) (c)

Slika 2: Primeri

Napomena 1 *Da bi se postigla odgovarajuća vrsta nasumičnosti, često je potrebno nasumično generisanje izvršiti po Gausovoj raspodeli, kako generisani objekti ne bi prečesto bili preveliki ili premali ili predaleko ili preblizu. Algoritam koji ovo radi implementiran je kroz neke biblioteke u većini programskih jezika, a u suštini nalazi prosečnu vrednost opsega u kom naša vrednost treba da se nađe i raspoređuje verovatnoću da se generišu svi slučajevi od normalnog do ekstremnog po normalnoj raspodeli. U daljem tekstu kad koristimo reč "nasumično" podrazumevaćemo da se misli u ovom smislu.*

3 Video igre

Proceduralno generisanje se u video igrama često koristi, samo je pitanje u kojoj meri. Većina igara proceduralno generiše nagrade za ostvarivanje ciljeva igre, radi kreiranja elementa neočekivanosti. Još neki primeri korišćenja proceduralnog generisanja su i: tip čudovišta na koje igrački lik (eng. player character) nailazi, brojčane vrednosti vidljivije korisniku kao što je nanesena šteta ili preciznost, sledeći deo u puzlama kao što je tetris, kompletna struktura

određenog nivoa u podzemnim igrama (eng. dungeon crawler), karakteristika likova ili kompletni proceduralno generisani svet.

Mehanika igre se može fundamentalno zasnivati na proceduralnom generisanju ili ga samo koristiti kao jedan element. Tako bi igra koja generiše ceo svet (npr. teren preko poligona, strukturu podzemlja) bila primena proceduralnog generisanja na ključni element doživljaja igre, dok korišćenje proceduralnog generisanja za stvaranje neke manje varijabilnosti nije ključni deo doživljaja te igre i obično se zasniva na prilično jednostavnom generatoru brojeva, koji se eventualno adaptira na nivo igrača.

3.1 Prednosti i mane

Postoji mnogo prednosti korišćenja ove metode, ali najveća je, bez sumnje, to što stvara praktično beskonačnu količinu novog sadržaja za igrača. Od ovoga profitiraju svi, jer su igrači skloniji da kupuju ono za šta procenjuju da će im duže držati pažnju, na čemu industrija zarađuje, dok igrači zapravo dobijaju dugotrajnu zabavu za svoj novac.

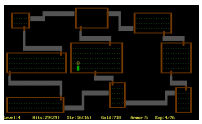
Osim ovoga, proceduralno generisanje ostavlja slobodu za razvoj igrice orijentisane specifičnom igraču. Može se dozvoliti velika sloboda igraču u smislu izbora sopstvenog stila igranja i usavršavanja istog sa neograničenim brojem mogućih strategija. Igra se čak može napraviti tako da se sama dinamički prilagođava stilu igre i sposobnostima igrača, dozvolivši neograničeno napredovanje boljim igračima ili zadovoljavajuće izazovno, ali ne preteško, iskustvo slabijim igračima, sa modifikacijama na osnovu praćenja njihovog napredovanja.

Jedna od često zanemarivanih prednosti je činjenica da ovo omogućava igrama da se više oslanjaju na procesor nego na grafičku karticu. Ukoliko se generišu grafički elementi, procesor vrši veći deo računanja nego što bi da te elemente treba povući iz memorije. Takođe je prednost što se manje toga uopšte čuva u memoriji, što smanjuje prostor koji igra zauzima i ubrzava samu igru.

Ipak, ovo ne znači da je metoda savršena. Proceduralno generisanje ne odgovara svakom tipu igara. Kada se udaljite od direktnog procesa stvaralaštva i prepustite ga računaru, samo delo izgubi mnogo toga. Često je jako teško odraditi mnoge detalje koji obogaćuju priču i pokazuju unikatnost same igre. Ironično, konstantno stvaranje novih stvari može brzo da dosadi bez odgovarajućih elemenata koji održavaju interesovanje igrača. Zato se igre koje se jako fokusiraju na naraciju obično klone proceduralnog generisanja ili ga koriste samo da učine određeni aspekt igre dinamičnijim, ali ne oslanjaju celo iskustvo na njega.

Drugi problem je što je ovo prilično skup sistem. Dobar sistem za proceduralno generisanje je teško napraviti, zahteva prilično specifičan algoritam u zavisnosti od onoga što želite da postignete i zahteva vreme. Treba se potruditi da se ne stvaraju ekstremni slučajevi previše često, što se uglavnom postiže Gausovom raspodelom. Ne treba dozvoliti preveliko skaliranje grafičkih elemenata koje narušava estetski izgled igre. Treba se pobrinuti da u slučaju kreiranja težih ili dosadnijih elemenata, kao što su jača čudovišta ili ćorsokaci, igrač bude adekvatno nagrađen. Svi nivoi treba da budu odgovarajuće težine za igrača, za šta treba uzeti u obzir mnoge faktore. Sve ovo treba i dalje da ne smanji varijabilnost rezultata vašeg generisanja do nivoa predvidivosti. Kada se sve ovo uzme u obzir, ukoliko vaš sadržaj nije dug desetine sati ili planiran tako da omogući zanimljivo igranje iznova i iznova, prosto se više isplati ručno dizajnirati sve grafički i odrediti sve vrednosti unapred.

Slika 3: Spacewar!



Slika 4: Rogue

Slika 5: Little Computer People 2

3.2 Primeri

1962. *Spacewar!* (slika 4) je jedan od ranijih primera korišćenja proceduralnog generisanja. U igri se dva igrača kreću po solarnom sistemu i treba da izbegnu sudar sa zvezdom i pogode jedan drugog torpedom. Jedan od metoda izbegavanja je prelazak u hipersvemir (eng. hyperspace), koji nakon nekoliko trenutaka stvara njihov brod na nasumičnoj lokaciji.

1975. *pedit5* je prva podzemna igra koja koristi proceduralno generisanje za nasumične susrete s protivnicima i nagrade. Ovo je takođe čini i prethodnikom "rouglike" žanra.

1980. *Rogue* (slika 5) je prva rougelike igrica, po kojoj je ceo žanr dobio i naziv. Igrač započinje igru na najvišem nivou podzemnih tamnica, koje se nasumično generišu u obliku 3x3 soba povezanih hodnicima, slično našem opisu algoritma za generisanje rougelike soba. Ponekad se sobe loše generišu i nema ih 9, te ima ćorsokaka na putu do sledeće sobe.

1985. *Little Computer People* (slika 6) je simulacija stvarnog života koja je svoj marketing zasnivala na tvrdnji da su "mali kompjuterski ljudi" stvarni i žive u našim kompjuterima, a softver ih samo prikazuje. Igrica je bila iz perspektive pogleda sa strane na kuću u koju bi se useljavao lik i radio svakodnevne stvari, koje je igrač mogao da kontroliše komandama ili igra neke mini-igrice, kao što je igranje pokera s njim. S vremena na vreme, lik bi započeo komunikaciju sam, pozvavši igrača na poker ili poslavši pismo o svojim željama i potrebama. Igra bi svaki put generisala novog, drugačijeg lika, te je svako novo igranje bilo drugačije. Igra nema krajnji cilj. Ova igra se takođe može smatrati pretkom Sim franšize.

1996. *Diablo* (slika 7), prvi deo popularne franšize, nasumično generiše svoje nivoe. Određene klase nivoa prate sličnu strukturu (npr. dugački hodnici ili puno grananja), ali svi su i dalje proceduralni.

2008. *Spore* (slika 9) je igrica koja ima pet stepena koji nude poprilično različita igračka iskustva. To su stepen ćelije (eng. Cell Stage), stepen stvorenja (eng. Creature Stage), plemenski stepen (eng. Tribal Stage), civilizacijski stepen (eng. Civilization Stage) i svemirski stepen (eng. Space Stage). Igra imitira evoluciju i igrač započinje u obliku jednoćelijskog organizma koji se kreće kroz 2D prostor, odlučuje o njegovoj ishrani i njegove navike u ishrani utiču na njegovu dalju evoluciju. Igra sama kreira stvorenja koja igrač susreće i sama na osnovu dizajna stvorenja animira njihovo kretanje, uključujući hodanje, lov, jelo, plivanje, vuču objekata, parenje, ples [3]. Dizajner, Vil Rajt (eng. Will Wright), rekao je da je za pravljenje celog stvorenja potrebno samo nekoliko kilobajta, analogno DNK

koja je u velikoj meri slična kod različitih stvorenja, ali se sekvence pozivaju različitim redosledom, te ne mora sve iznova da se pamti za svako stvorenje, već sva imaju istu bazu, a igra razvija određeni fenotip. Razne odluke igrača u toku igre izazivaju odgovore njegovog okruženja, kao što su odnosi sa drugim stvorenjima.

2008. *Left 4 Dead* (slika 8) popularizuje tzv. proceduralnu priču (eng. procedrual narrative), što je priča ili mehanika igre koja se generiše algoritamski i često prilagođava stilu igre korisnika. Igra ima veštačku inteligenciju pod imenom "Direktor" koja dinamički uređuje igru u odnosu na igračevu lokaciju u igri, sposobnosti i sl. Direktor ne kontroliše samo izazove sa kojima se igrač suočava, već proceduralno generiše i izbor muzike u odnosu na raspoloženje igrača u zavisnosti od njegovih procena situacija koje je igrač upravo iskusio.

2011. *Minecraft* je igrica čija je premisa sloboda da igrač osmisli sopstvenu igru u 3D proceduralnom svetu napravljenom od kocki. Svet se nasumično generiše u vidu različitih bioma sa različitim tipovima vegetacije, životinja i klime. Igra može da se igra u modu preživljavanja (eng. survival mode), gde igrač uglavnom gradi sebi skloništa i rudari sakupljajući materijale za svoju izgradnju, oružije, opremu ili šta god je već rešio da odredi kao svoj cilj igre i braneći se od čudovišta noću. Takođe postoji i kreativni mod (eng. creative mode) u kom je igrač oslobođen brige o preživljavanju i kretanju i može da leti i uzima bilo koje materijale dostupne za izgradnju i gradi šta želi. Iako naizgled jednostavna i neinteresantna premisa, ispostavilo se da je ova kombinacija proceduralnosti koja garantuje nove sadržaje i mešavinu nepredvidivosti i očekivanja koju imamo i u stvarnom svetu i slobode koja je data igraču u izboru kako svojih ciljeva u igri, tako i sadržaja igre između mora modova koji su korisnici kreirali, učinio Minecraft najprodavanijom PC igrom ikada i drugom najprodavanijom igrom, posle Tetrisa.

2016. *No Man's Sky* (slika 10) je igrica koja zaista kapitalizuje na proceduralnom generisanju. Cela premisa igrice je da se nalaziš na nasumičnom mestu u ogromnoj galaksiji od preko 18 triliona planeta koje je moguće istraživati. Planete se kreiraju na osnovu determinističkih algoritama, tako da kad god igrač poseti istu planetu ona će isto izgledati, ali neće se svaki aspekt svake planete čuvati u memoriji. Cilj igre je doći do centra galaksije.

4 Zaključak

Koliko god pričali o temi proceduralnog generisanja, uvek će ostati nešto što nismo ni spomenuli. Iako se ova tema istražuje već od kako su personalni računari, igrice i drugi mediji postali dostupni većini ljudi za korišćenje, ipak i dalje uspevamo da proceduralno generisanje iskoristimo na neki novi, do sada neviđen način. Mi svakako vidimo potencijal ovoga, i smatramo da će u budućnosti neki delovi ovoga što već znamo moći da budu iskorišćeni za neke stvari koje nam danas ne bi ni pale na pamet.

Literatura

[1] N. Shaker, J. Togelius, M. J. Nelson, *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*, Springer, 2016.

[2] https://en.wikipedia.org/wiki/List_of_games_using_procedural_generation

[3] http://www.gamespot.com/pc/strategy/spore/news.html?sid=6150118