Procedurell generering av nivåer i datorspel

Anton Backman 2001139
Kandidatavhandling i Datateknik
Handledare: Mats Aspnäs
Fakulteten för naturvetenskaper och teknik
Åbo Akademi
2023

Innehåll

Kapitel 1

Introduktion

Spel skapar möjligheten för oss att utforska nya världar och planeter som vi aldrig kunde uppleva i den verkliga världen på ett nytt sätt. Datorspel har en distinkt skillnad från andra underhållingsformer så som filmer och böcker eftersom du själv kan påverka händelserna i världen. Möjligheten för spelaren att påverka hur spelet slutar bidrar till att inte alla upplevelser är samma, men endast för några genomspelningar. Hur kan man då skapa unika upplevelser gång på gång utan att måste skapa tusentals världar för hand? Ett av sätten är att procedurellt generara världarna.

Procedurell generering är en populär metod inom spelutveckling som tillåter spelutvecklarna att skapa stora, komplexa och mångfaldiga världar med unika spelupplevelser utan att manuellt skapa varje detalj och element. Ett område där procedurell generering visat sig vara speciellt effektiv är genereringen av nivåer i datorspel. Genom att använda sig av algoritmer och slumptal, kan spelutvecklarna skapa nivåer som är unika, utmanande och oförutsägbara. Det här gör att spelarna inte blir uttråkade av samma nivåer gång på gång eftersom ingen genomspelning kommer vara exakt samma som någon annan. *Rogue* är ett av de tidigaste spelen som utnyttjat procedurell generering för sina nivåer och föremål. Detta inspirerade många entusiaster av spelet att göra sina egna förbättringar som de sedan delade online. På grund av *Rogue* och dess popularitet skapades många andra spel som tar inspiration av dess procedurella generering, till exempel *Minecraft*, som har sålt flera kopior än något annat datorspel.[1]

Fördelarna med procedurell generering är många. En fördel är att det sparar utvecklarna enorma mängder tid och resurser, eftersom de inte måste spendera hundratals timmar på att skapa, balansera och testa alla nivåerna för hand. Det finns så klart även vissa nackdelar med procedurell generering. Att skapa algoritmer som skapar både roliga och balanserade nivåer är en svårt och tidskrävande process som kräver mycket testande och finslipande. Om det inte finns ett sätt att påverka nivåns omgivning måste man säkerställa att nivån inte är omöjlig att klara av. En annan fallgrop är att nivåerna skapade av proce-

durell generering kan kännas som att de saknar samma personlighet som en handskapad nivå har. Därför använder spelutvecklare sällan endast procedurell generering, utan en blandning av procedurell generering och handskapade element som placeras ut i nivån. Världsgenereringen i *Minecraft* använder sig av färdiga element, som till exempel hus, ruiner, övergivna tempel och andra strukturer man kan utforska vilket får världen att kännas mycket mera engagerande och spännande än om det bara skulle finnas skogar och berg.

Procedurell generering kan skapa nya föremål och data enligt restriktioner som utvecklaren satt. De här restriktionerna är till för att genereringen inte ska gå snett, så att världarna är både spelbara och roliga för spelaren. Man kan procedurellt generera många saker i ett spel. Det första man tänker på är terrängen, men man kan även slumpmässigt generera vapnen, fienderna, de icke-spelbara karaktärerna, uppdrag, belöningarna man får från uppdragen och mycket mera. För att få fungerande egenskaper hos dessa kan man till exempel använda sig av färdiga modeller och slumptal för att skapa otaliga unika karaktärer och föremål.

I den här avhandlingen kommer metoderna som används för procedurell generering att analyseras för att bättre förstå varför vissa saker fungerar eller inte fungerar. Finns det olika sätt man kan kombinera metoder för att skapa världar som inte liknar någon annat värld? Vilka element utöver terrängen kan använda sig av procedurell generering som man kanske inte tänkt på? Det finns både för- och nackdelar med procedurell generering och det är viktigt att känna till dem, så man kan skapa nivåer av hög kvalitet.

Kapitel 2

Metoder för att procedurellt generera terrängen

Metoderna man använder för att procedurmässigt generera nivåer är i teorin ganska simpla algoritmer. De här algoritmerna kan dessutom kombineras för att skapa mycket komplexa nivåer med ytterst lite kod, vilket sparar på utrymmet spelet tar upp på datorn. Det här var speciellt viktigt för många år sen då datorminne var relativt dyrt jämfört med idag.

2.1 Utrymmespartitionering

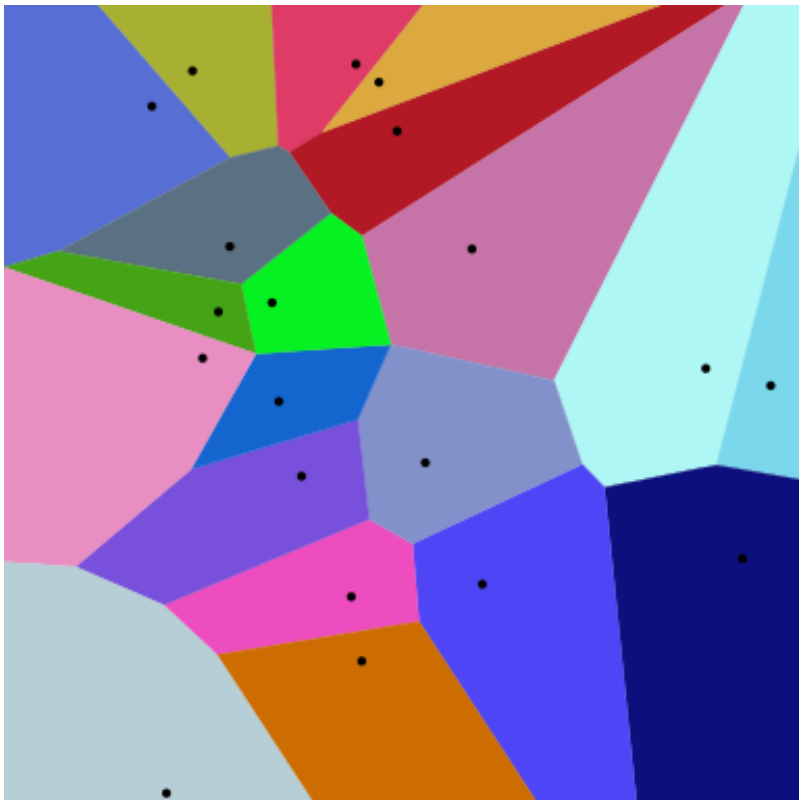
Om man vill dela upp ett tomt utrymme finns det några olika sätt att göra det på. Man måste anpassa metoderna till sitt ändamål, till exempel kan binär utrymmespartitionering passa bra för att skapa rum och voronoi diagram för att tilldela landområden på en världskarta.

2.1.1 Binär utrymmespartitionering

Binär utrymmespartitionering(eng. Binary Space Partitioning) innebär att man tar ett utrymme och delar det i två slumpmässigt stora delar, sätter en dörr mellan de skapade utrymmen, sedan upprepar man processen tills man har det önskade antalet utrymmen. Genom att sätta en dörr mellan de två skapade utrymmen säkerställer man att alla utrymmen kan nås av spelaren. Det finns även små förbättringar man kan göra genom att sätta begränsningar på hur utrymmen delas i två, så att ett utrymme inte blir för litet vilket skulle se konstigt ut. Man kan även vidareutveckla algoritmen så att man placerar ett rum inuti de skapade utrymmen för att få nivån att se mindre fyrkantig ut. [2, p. 293-295]

2.1.2 Voronoi diagram

Voronoi diagram skapas genom att ta x antal punkter på en karta och tilldela området som ligger närmast till en punkt till punkten. Sättet man bestämmer distansen till en punkt kan göras på olika sätt och då kommer indelningen av områden att se lite olika ut.



Figur 2.1: Exempel på indelning med hjälp av voronoi diagram

(testbild för att kolla hur dom fungerar i latex)

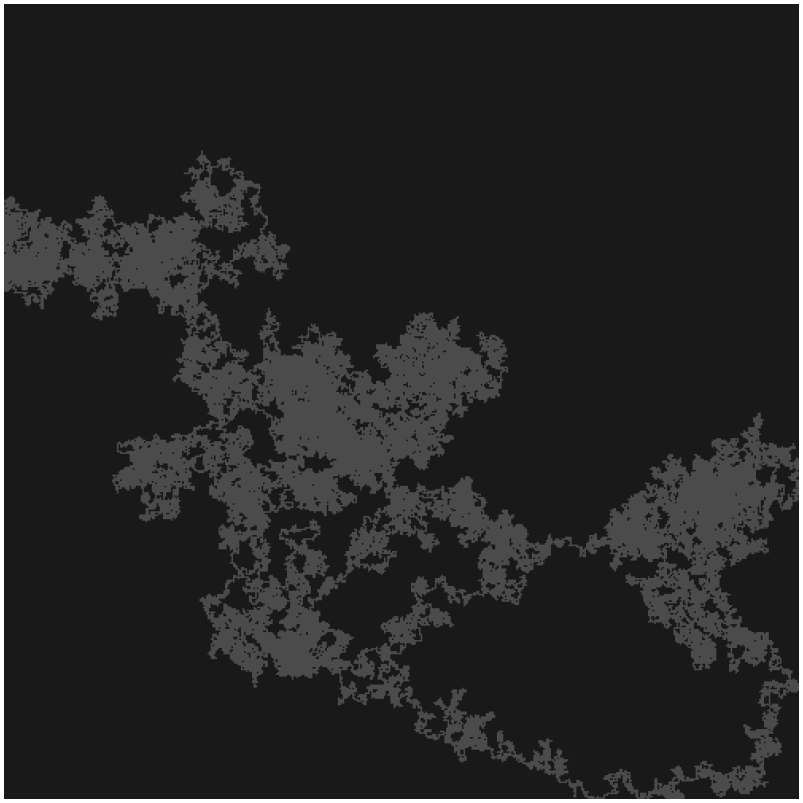
2.1.3 Dijkstra maps

2.2 Fyllning av utrymme

2.2.1 Slumpmässig gång

Slumpmässig gång(eng. Random Walk) är när man startar från en punkt och går slumpmässigt i olika riktningar. Man kan använda sig av slumpmässig gång i en eller flera dimensioner. Man kan använda sig av slumpmässig gång i en dimension, där man kan gå upp eller nedåt för varje steg man åt höger man tar för att generera berg och dalar. Om man gör en slumpmässig gång i två dimensioner kan man till exempel generera grottsystem, stigar, flodar och mycket mera. Man behöver bara modifiera algoritmen lite för att

få den önskade effekten, om man till exempel vill generera en labyrint kan man avsluta den slumpmässiga gången varje gång man korsar en gammal stig, och börja på nytt ända tills alla rutor är besökta. [2, p. 286-288] En till fördel med slumpmässig gång är att varje ruta på kartan kommer att vara åtkomlig av spelaren från startpunkten då man inte gör några hopp över någon ruta.



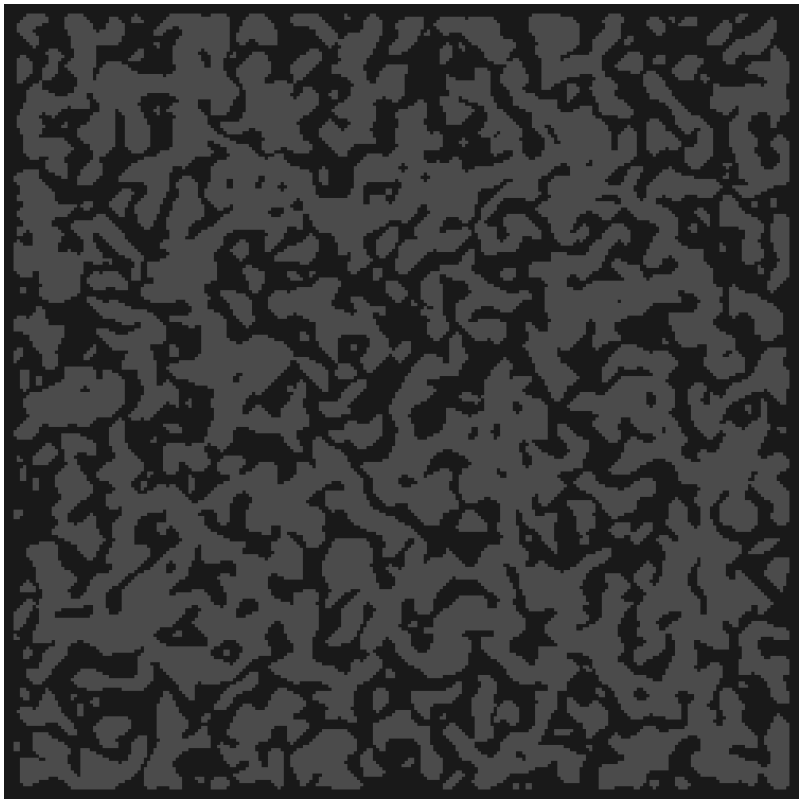
Figur 2.2: Exempel på ett grottsystem genererat med en slumpmässig gång

2.2.2 Cellulära automata

Cellulära automata är varje cell, eller ruta i ett rutnät, diskret till exempel en vägg eller luft. Cellens tillständ ändras baserat på dess närliggande cellers tillständ baserat på givna regler. De här reglerna tillämpas i tur och ordning på varje cell i rutnätet i så många iterationer tills man får det önskade resultatet.

2.3 Höjdkartor

Höjdkartor används i stor grad för att skapa terrängen för spelnivåer. I verkliga världen används de för att beskriva terrängens höjdskillnader, vilket kan översättas till spel. Det går då procedurellt att generera en nivå genom att slumpmässigt skapa spelens höjdkartor.



Figur 2.3: Exempel på ett grottsystem genererat med cellulära automata

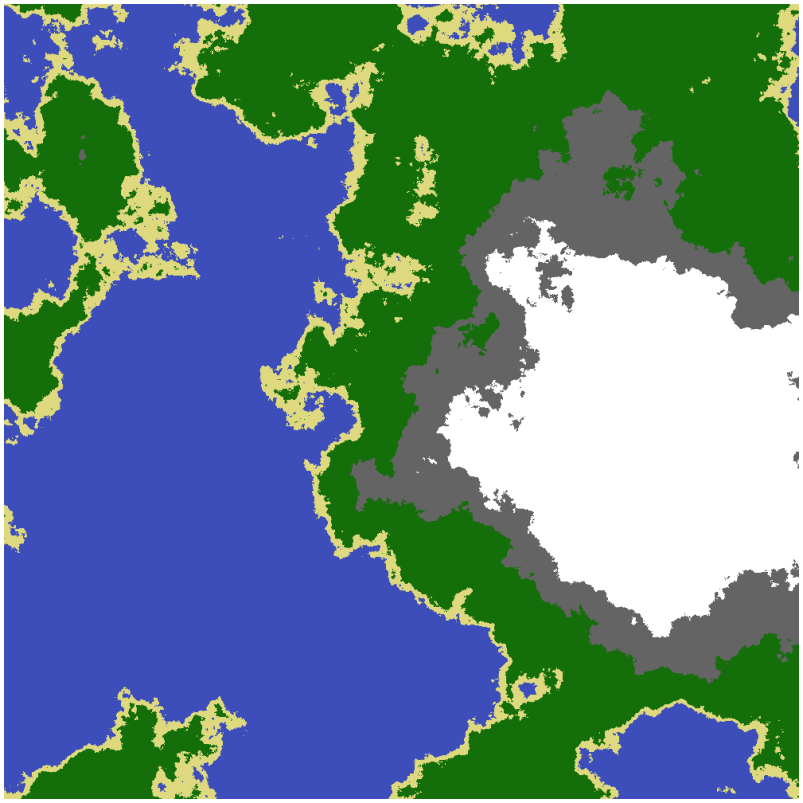
Andra områden i spel som också kan använda sig av höjdkartor är till exempel väder, temperatur och fuktighet. Om man kombinerar höjdkartorna kan man skapa världar som ser väldigt naturliga ut med jämna övergångar i väder, klimat och biomer. De två vanligaste metoderna för att slumpmässigt skapa höjdkartor är Perlin- och simplexbrus.

2.3.1 Perlinbrus

Perlinbrus är en deterministisk algoritm, det vill säga att man får samma höjdkarta om man använder sig av samma frö(eng. seed). För att då skapa unika höjdkartor väljer man slumpmässigt ett frö, i *Minecraft* väljer man detta slumpmässiga frö bland 2^{64} andra frön, så det är osannolikt att man får samma värld som någon annan. Perlinbrus skapar ett rutnät med toppar och dalar med värden från 1 till -1. I dess mest grundläggande form skapar Perlinbrus mjuka vågor som är placerade jämnt över rutnätet. För att skapa mer intressanta höjdkurvor tar man flera lager av brus, med olika amplituder och frekvenser så att man får höjdkurvor utan några geometriskt igenkännbara drag.

2.3.1.1 Hur Perlinbrus används i *Minecraft*

(Här finns engelska uttryck och namn som ännu borde översättas)



Figur 2.4: Exempel på karta genererad med Perlinbrus

Minecraft är ett sandlådespel(eng. sandbox game), det vill säga att spelaren får fritt välja vad man vill göra. Världen är gjord av block och är 60 miljoner block från en sida till den andra. Det skulle vara omöjligt att lagra en hel värld på en vanlig dator då en hel värld ungefär skulle ta upp 97000 terabytes. Genom procedurell generering behöver inte spelet lagra en hel värld, utan skapar den vartefter spelaren rör sig.

Procedurella genereringen i *Minecraft* sker i olika skeden. Först formas terrängen, så man vet vilka block som är sten eller luft. Sedan räknar man ut vilka block som ska fyllas med vatten, sedan byter man ut översta lagret till andra sorts block så som gräs eller sand. Till sist lägger man till detaljer som strukturer, träd och mineraler. Allt det här bestäms med algoritmer så att varje värld är unik, men om man använder samma frö kommer världen alltid att se lika ut. Huvudsakligen används perlinbrus för den procedurella genereringen, men för att terrängen ska se intressantare ut används flera lager av brus. För terrängen används huvudsakligen tre olika perlinbrus med olika oktaver och frekvenser som de kallar *Continetalness*, *Erosion* och *Peaks and Valleys*. Från dom här brusen räknar man sedan ut hur världen kommer se ut. Om *Continentalness* och *Peaks and Valleys* har höga värden och *Erosion* är lågt värde kommer det bildas höga berg och om *Erosion* är högt kommer landskapet att vara ganska platt oavsett vad dom andra värden är.

Ett till kännetäcken i världen är grottorna. Det finns två sorters grottor i *Minecraft*,

swiss cheese grottor och *spaghetti* grottor. Båda genereras med perlinbrus men på lite olika sätt. *Swiss cheese* grottor genererar luft där värdet på perlinbruset är högt, då skapas stora runda hål i världen, namnet kommer också därefter. *Spaghetti* grottor skapas när perlinbruset är nära noll vilket skapar långa slingrande grottor.

Världen i *Minecraft* har också olika biomer som ändrar väldigt drastiskt på utseendet. För att bestämma vilken biom som placeras var använder man ytterligare två brus *Humidity* och *Temperature*. Baserat på de alla olika brusen bestämmer man från tabeller vilken biom som kommer placeras var. Det här är dock en väldigt grov förenkling av hur det verkligen fungerar och det finns flera optimeringar för att snabba upp koden, men i grund och botten fungerar det så här.

2.3.2 Simplexbrus

2.3.3 Någon till metod

Kapitel 3

Unika fiender med procedurell generering

3.1 Slumpmässigt skapa fiender

3.2 Slumpmässigt beteende

3.3 Fiendevågor(?)

Kapitel 4

Vilka spel använder slumpmässig generering?

Det här kapitlet kommer högst troligt sammanslås med kapitlen om metoderna som exempel för att bättre förklara hur dom fungerar och hur slutresultatet av metoderna kan se ut.

4.1 Conway's Game of Life

4.2 Minecraft

4.3 No Man's Sky

4.4 Spore

4.5 Borderlands

4.6 Stardew Valley

Kapitel 5

Diskussion

Källförteckning

[1] N. Brewer, ”Computerized Dungeons and Randomly Generated Worlds: From Rogue to Minecraft,” 2017.

[2] T. X. Short och T. Adams, *Procedural Generation in Game Design*. Taylor & Francis Group, LLC, 2017.