Procedurell generering av städer (WIP)

Alexander Norrvik 38789

Handledare: Mats Aspnäs

Kandidatavhandling i datateknik

Abstrakt

#TODO

Innehållsförteckning

1 Inledning

Att skapa städer har intresserat forskare inom datorgrafik under årtionden, dessutom har behovet av 3D genererade miljöer enbart ökat. Inom media som film och datorspel samt stadsplanering har 3D miljöer och städer gått från nyhet till norm. Även inom stadsplanering används modeller av städer för att få en bättre överblick och för att kunna simulera effekten av planerade förändringar.

Att skapa modeller av städer är arbetsdrygt och därför används procedurell generering för att underlätta delar eller hela processen. Att skapa städer procedurellt kan verka komplext och det kan vara svårt att veta var processen borde börja. Syftet med avhandlingen är därför att lyfta fram stegen och metoder för procedurell generering av städer och sålunda klarlägga processen.

För att avhandlingen skall hållas koncis sätts några begräsningar på omfattningen. Generering av miljö eller terräng för staden kommer uteslutas helt. En hel kandidatavhandling kunde skrivas om enbart tekniker för generering av terräng. Texten skulle bli mycket längre utan att bidra stort till centrala syftet. För generering av städer finns många metoder vilka inte alla kan behandlas i texten. Avsikten är att klarlägga grunderna till förståelse för procedurell generering av städer. I detta syfte belyses några metoder som ger insikt i olika tillvägagångssätt för stadsgenereringen.

Efter att metoderna har behandlats kommer användningsområden för 3D stadsmodeller lyftas fram. Avslutningsvis kommer en diskussion om framtiden för procedurella städer.

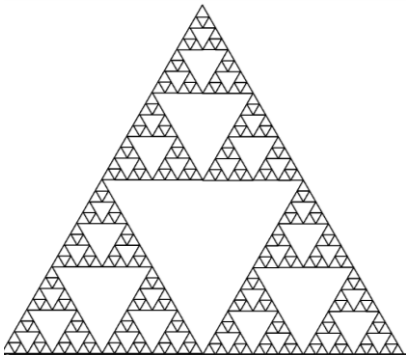
2 Bakgrund

2.1 Vad betyder procedurell generering?

Procedurell generering är en mycket vid term. Att något genereras procedurellt betyder att data skapas av ett program i stället för manuellt. Med andra ord går procedurell generering ut på att data skapas med hjälp av regler, algoritmer och program. Inom datorgrafik brukar termen användas för att särskilja texturer och modeller som inte gjorts manuellt. Procedurell generering används ofta i samarbete med manuellt gjorda modeller.

Ett klassiskt exempel på procedurell generering är fraktaler, vars relativt enkla basform upprepas för att nå högre komplexitet. Dessa upprepningar kallas för iterationer.

För att rita upp en Sierpinsky triangel (figur 1) så kunde man spara information om varenda linje som bör ritas för önskad komplexitet och om flera olika iterationer krävs så behövs också den informationen lagras. Detta kunde lätt bli mycket information ifall många olika eller väldigt detaljerade iterationer krävs. Alternativt kan enkla regler definieras som sedan kan upprepas för att få mer detaljer.



Figur 1 5te iterationen av en Sierpinsky triangel ritat med ett L-system.

2.2 L-system

Lindenmayer system (L-system) är en sorts formell grammatik som används för att beskriva parallellt växande fenomen. Dessa system introducerades 1968 av Aristid Lindenmayer för att beskriva tillväxten på de bakterier och svampar som han studerade [1]. Efter deras introduktion har L-system används för att beskriva många olika parallellt växande system, bland annat alger, träd, rötter och även

vägnätverk [1, 2, 3]. L-system fungerar genom att skriva om textsträngar och består av ett starttillstånd och en eller flera produktioner. Starttillståndet beskriver vilka symboler som strängen ursprungligen har och produktionerna beskriver hur strängen skall skrivas om.

För att tydliggöra hur L-system ser ut och tolkas presenteras ett exempel ur boken L-systems [1, p. 9].

ω: a

p_1: $a => b$

p_2: $b => ba$

Figur 2 L-system för Fibonacci serien [1, p. 9]

Startordet benämns med ω och kallas även för axiomet. ω används även för att visa vilket ord man befinner sig på, exempelvis ω_0 för startordet och ω_1 för första omskrivningen och så vidare. Produktionerna numreras inte alltid men då de numreras indikeras produktionerna med p och dess nummer (p_1, p_2, p_3…).

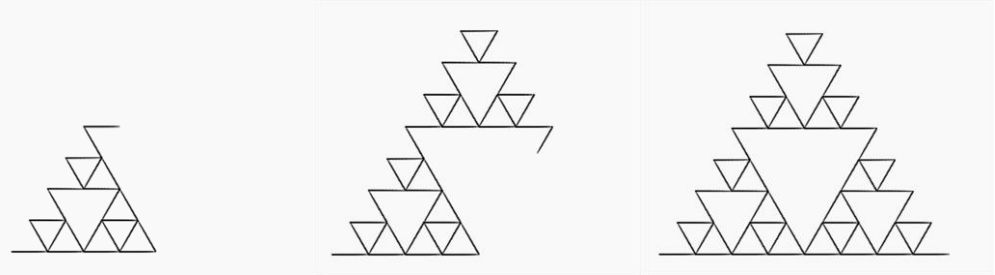
För att få 7:de ordet ur systemet i Figur 2 skrivs strängen om med hjälp av produktionerna. *a* skrivs om till *b* och *b* skrivs om till *ba*. Genom denna omskrivning fås serien *a,b,ba,bab,babba,babbabab,babbababbabba* och så vidare. Då ordlängden räknas fås serien 1,1,2,3,5,8,13 vilken kan kännas igen som Fibonacci serien.

Fraktaler kan representeras i 2D eller 3D med L-system och Sköldpaddas grafik (Turtle graphics). Sköldpaddas grafik ritar linjer genom att definiera en riktning och längd, nästa linje fortsätter där den tidigare tog slut. För att rita upp en Sierpinsky triangel används L-systemet i Figur 3.

$$\omega: F - G - G$$
$$p_1: F \rightarrow F - G + F + G - F$$
$$p_2: G \rightarrow GG$$

Figur 3 L-system för Sierpinsky triangel

Både F och G betyder att Sköldpaddan ritar rakt framåt. Plustecknet (+) innebär en rotering till höger och minustecknet (-) betyder rotering till vänster. Vinkeln för rotationen är i detta fall 120° för att bilda liksidiga trianglar. Figur 4 visualiserar uppritningen av L-systemets tredje ord (F-G+F+G-F-GG+F-



Figur 4 Sierpinsky triangel i olika skeden av uppritningen

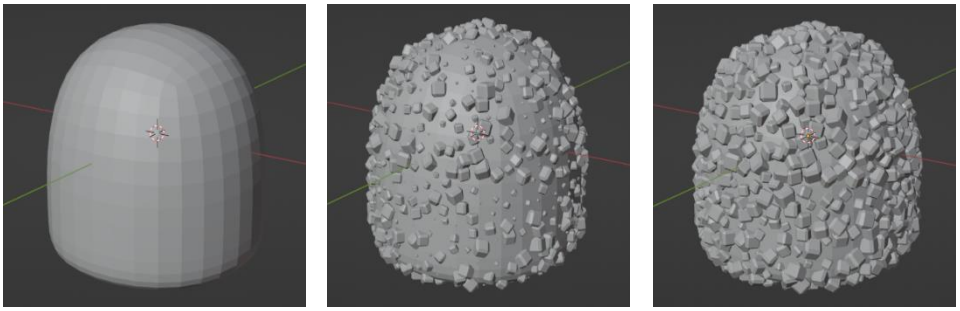
G+F+G…).

2.3 Öppna L-system

2.4 Fördelar med procedurell generering

En av fördelarna med att generera data procedurellt under exekvering (eng. runtime) är att man inte behöver spara data i minnet, utan endast vid behov skapa data med den upplösning som krävs för ändamålet. Till exempel om man vill rita upp en sinusvåg på skärmen kan man antingen ha färdigt uträknade punkter för den största upplösning som kan behövas eller så räknar man under exekvering ut värdet för sinus i varje punkt som krävs för att nå den önskade resolutionen. I praktiken är problemen eller funktionerna som ska räknas ut sällan så enkla som en sinusfunktion, så en balans måste nås beroende på hur snabbt resultatet behövs jämfört med hur mycket minne som finns tillgängligt.

Procedurell generering kan även förenkla arbetsflödet på två olika sätt. Först och främst så kan man automatisera delar av arbetet [2, pp. 1-2]. Speciellt monotona eller enkla arbetsskeden som är tidskrävande kan med fördel automatiseras med procedurell generering. I Figur 5 visas ett exempel på en enkel men tidskrävande process. Målet är att skapa en rundad godis med, platt botten och som är dekorerad med socker. Grundformen går snabbt att göra manuellt (Figur 5a) men att sedan placera ut sockerkristaller av varierande storlek och rotation skulle kräva mycket arbete. Genom att skriva regler för hur sockerkristallerna ska placeras ut på ytan av modellen så kan den önskade effekten genereras procedurellt. Denna effekt visas i Figur 5b



Figur 5 godis med sockerkristaller. Från vänster: (a) utan sockerkristaller, (b) med sockerkristaller, (c) med sockerkristaller och modifierade parametrar

Den andra orsaken till att procedurell generering underlättar arbetsflödet är att man kan göra parametrar av relevanta egenskaper hos modellen [2, pp. 1-2]. Genom att ändra på parametrarna ändras även slutresultatet. I Figur 5c används samma modell som i Figur 5b men med ökad medelstorlek på sockerkristallerna och mindre varians i storleken. Om man gör stora förändringar, som att byta ut basmodellen, skulle allt arbete vara bortkastat utan procedurell generering eftersom de utplacerade sockerkristallerna var placerade i relation till ytan på den tidigare modellen. Med procedurell generering kan basmodellen bytas ut eller parametrar ändras och det nya resultatet fås i realtid.

3 Procedurella städer

En stad består av många sammankopplade delar. För procedurell generering är de tre viktigaste byggstenarna vägar, uppdelning i tomter och generering av hus. Terräng är en viktig del av en stads uppbyggnad och karaktär, men de tekniker som används för att generera terräng kommer inte behandlas. Därför antas

terrängen vara färdigt genererad, till exempel en genererad miljö, eller en miljö baserad på geografiska data av en verklig plats.

3.1 Generering av vägar

Gemensamt för alla artiklar om generering av städer var att processen började med att generera ett vägnätverk (ifall terrängen bortses) [3, 4, 5, 6]. Likhet kan dras kan dras till byggandet av verkliga hus, där vägarna byggs först för att kunna köra material för husbyggnaden. För väggenerering finns olika tillvägagångssätt.

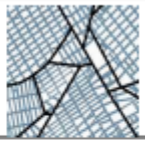
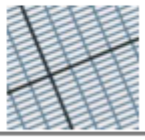
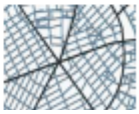
3.1.1 L-system

I Parish & Müller 2001 används ett L-system för generering av vägnätverket som styrs med olika kriterier [4]. Dessa kriterier fås från kartor vilka i artikeln delas in geografiska och sociala kartor.

Vägarna generas av ett L-system och modifieras av globala och lokala begräsningar. Globala begränsningar modifierar vägarna så att de följer vägmönstret som valts för staden samt prioriterar vägar till områden med större befolkningstäthet. Lokala begränsningar förhindrar vägar som går till förbjuden terräng, som parker eller vattendrag. Vägarna modifieras även av lokala begränsningar så att de inte går över varandra utan att skapa korsningar. Korsningarna tillåts inte skapas för nära existerande korsningar.

Första steget för generering av vägnätverket är att generera motorvägar mellan områden med hög befolkningstäthet (vilka fås från kartorna). Varje ändpunkt av motorvägarna strålar ut linjer och skapar nya vägar till de punkterna som har den största befolkningstätheten. Om vägen skulle korsa ett vattendrag så kan systemet kodas så att en väg tillåts ifall avståndet till andra sidan är tillräckligt kort och systemet kan då även skapa broar.

Efter att motorvägarna är skapade kan genereringen av resten av vägnätverket börja. I vissa punkter förgrenas vägarna och de nya vägarna skapas baserat på de globala och lokala begräsningarna. I Parish & Müller 2001 används 4 olika

Pattern name	Pattern	Example
Basic	No superimposed pattern.	
New York	Rectangular Raster	
Paris	Radial to center	

Tabell 1 - Olika vägmönster för procedurell generering. Tabellen är från Parish & Müller 2001 [3]

vägmönster för att generera städer. Det första kallar de för enkelt mönster (basic pattern), och det går ut på att man inte sätter restriktioner på vägmönstret utan låter befolkningstätheten bestämma mönstret. Detta leder till ett relativt naturligt mönster och påminner om gamla städer som byggts utan genomgående stadsplanering. Det andra mönstret som tas upp kallas New York mönster och bildar vägnätverk med ett rutmönster. Med detta mönster kan man efterlikna vägnätverket i många nyare städer som byggts med planerade vägar och kvarter i ett rutmönster.

För att generera vägnätverk för existerande städer kan även färdiga data användas. Dessa är antingen baserat på satellitbilder i vilka man kan identifiera vägar eller med vägnätverksinformation från exempelvis Open Street Maps (OSM) [7]. I Zhang et al. 2020 används neurala nätverk som tolkar satellitbilder i kombination med OSM för att få så noggrann representation av verkligheten som möjligt [5]. Här behövs inte vägnätverket skapas utan det krävs bara att konvertera existerande information till ett ändamålsenligt format.

En annan metod för väggenerering är att låta en person skapa vägnätverket. Denna metod används i CityGen i vilket man interaktivt skapar huvudvägarna och programmet kan sedan generera mindre vägar för att ge husen tillgång till vägnätverket [8]. Trots att det krävs mycket mer arbete att manuellt göra upp

vägstruktur kan det vara användbart exempelvis då man vill uppnå en viss konstnärlig vision.

3.2 Uppdelning I tomter

Efter att vägnätverket har genererats börjar uppdelningen av områden runt vägarna i tomter. Detta steg är viktigt eftersom tomten påverkar både tillhörande husets placering samt storlek. För att dela upp dessa områden i tomter finns olika metoder.

Figur 6 uppdelning av kvarter i mindre tomter. Från vänster (a) genererade vägnätverket, (b) utvidga vägarna, (c) dela upp kvarteren i tomter. taget från Parish & Müller 2001 [3]

Vägnätverket bildar kvarter som sedan kan delas upp i mindre tomter. Både Parish & Müller och CityGen använder sig av en rekursiv algoritm som delar på de sidor som är längst och ungefär parallella [4, 8]. Uppdelningen fortsätter tills tomterna är tillräckligt små. Maximala storleken på tomterna är en av parametrarna som kan justeras för att ändra på slutresultatet. Den minsta tillåtna storleken på tomterna kan också justeras. Efter att uppdelningen är klar så förkastas alla tomter som är för små eller inte har tillgång till en väg.

Algoritmen av CityGen har fördelen att även konkava tomter tillåts. Detta möjliggör mer komplexa former av tomter än jämförelsevis med algoritmen av Parish & Müller 2001. Eftersom formen av tomten påverkar hurudana hus som kan genereras så möjliggör detta också mer komplexa husmodeller.

Tomter kan även skapas med data av verkliga städer på samma sätt som vägarna. För detta har LIDAR (Light Detection and Ranging), satellitbilder och flygbilder använts. Zhang et al. [5] använder satellitbilder i kombination med ett neuralt nätverk som uppskattar medelstorleken av tomterna i ett kvarter. En algoritm delar sedan upp kvarteret i tomter på två olika sätt.

3.3 Generering av hus

4 Användningsområden

4.1 Media spel/film/konst

4.2 Simulering/modellering

5 Sammanfattning

Framtid

Referenser

[1] L. Kari, G. Rozenberg och A. Salomaa, ”L Systems,” i *Handbook of Formal Languages*, Springer Berlin Heidelberg, 1997, p. 253–328.

[2] D. S. Ebert, Texturing and Modeling A Procedural Approach, Elsevier Science & Technology Books, 2014, p. 332.

[3] A. Emilien, A. Bernhardt, A. Peytavie, M.-P. Cani och E. Galin, ”Procedural generation of villages on arbitrary terrains,” *The Visual Computer*, vol. 28, p. 809–818, April 2012.

[4] Y. I. H. Parish och P. Müller, ”Procedural modeling of cities,” i *Proceedings of the 28th annual conference on Computer graphics and interactive techniques,* 2001.

[5] X. Zhang, A. Shehata, B. Benes och D. Aliaga, ”Automatic Deep Inference of Procedural Cities from Global-Scale Spatial Data,” *ACM Trans. Spatial Algorithms Syst.,* vol. 7, October 2020.

[6] C. A. Vanegas, T. Kelly, B. Weber, J. Halatsch, D. G. Aliaga och P. Müller, ”Procedural Generation of Parcels in Urban Modeling,” *Computer Graphics Forum,* vol. 31, p. 681–690, May 2012.

[7] OpenStreetMap Foundation, ”Open Street Map,” [Online]. Available: https://www.openstreetmap.org/.

[8] G. Kelly och H. McCabe, ”Citygen: An Interactive System for Procedural City Generation,” 2007.

[9] T. Niese, S. Pirk, M. Albrecht, B. Benes och O. Deussen, ”Procedural Urban Forestry,” *ACM Transactions on Graphics,* vol. 41, p. 1–18, March 2022.

[10] A. Mustafa, X. W. Zhang, D. G. Aliaga, M. Bruwier, G. Nishida, B. Dewals, S. Erpicum, P. Archambeau, M. Pirotton och J. Teller, ”Procedural generation of flood-sensitive urban layouts,” *Environment and Planning B: Urban Analytics and City Science,* vol. 47, p. 889–911, November 2018.

[11] R. Měch och P. Prusinkiewicz, ”Visual models of plants interacting with their environment,” i *Proceedings of the 23rd annual conference on*

Computer graphics and interactive techniques - SIGGRAPH \textquotesingle96, 1996.

[12] E. Galin, A. Peytavie, N. Maréchal och E. Guérin, ”Procedural Generation of Roads,” *Computer Graphics Forum*, vol. 29, p. 429–438, May 2010.

[13] D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin och S. Worley, Texturing and Modeling A Procedural Approach, Taylor & Francis Group, 2003, p. 687.

[14] Kocaman, S. and Zhang, L. and Gruen, Armin and Poli, Daniela, "3D City Modeling from High-resolution Satellite Images," 2006.