

# POOL SIMULATION *RAPPORT*

AV ALEXANDER PETROVSKI

ALEPE261 810205

Kurs: Procedurella Bilder

Kurskod: TNM022

Examinator: Stefan Gustavsson, [stegu@itn.liu.se](mailto:stegu@itn.liu.se)

## BAKGRUND

*”Procedurell generering av bilder är ett mycket användbart och ofta använt verktyg inom modern datorgrafik. Kursen avser att ge kunskaper om klassiska och moderna metoder för procedurell generering av bilder, speciellt texturer som avser att efterlikna komplicerade och visuellt detaljerade fenomen i naturen.”<sup>1</sup>*

I kursen är det förutom obligatoriska moment i form av laborationer, en examination med ett avslutande individuellt projekt. Projektarbetet skall enligt direktiv avrapporteras skriftligt i form av en kortfattad rapport och en kort muntlig framställning på ett seminarium. Dessutom hålls en individuell presentation för examinator. Projektet ger tre poäng och skall inklusive aktiva studier ta 120h att genomföra. Själva rapportskrivningen får beräknas ta högst ett par, tre dagar.

## IDÉN

Som grund för projektidén låg en laboration under kursens gång där RenderMan Shading Language skulle användas för att skapa en scen. Resultatet blev en poolbotten där så kallade caustics<sup>2</sup> kunde urskiljas på en botten av kakel. En fascination för vattens dynamik och dess fysikaliska egenskaper ledde till valet av en pool där vattenvågor och någon form av caustics på botten simulerades i realtid.

## UTFÖRANDET

Hur utförandet av pool med vattenvågor samt någon form av caustics<sup>3</sup> skulle simuleras, var klart på några punkter. Det skulle programmeras i C++, samt att något s.k. framework<sup>4</sup> skulle användas. Tiden för större delen av projektarbetet förlades under jullovet vilket innebar att utrustningen som användes var personlig.

### Tekniska detaljer

Utvecklingsmiljön var Dev-C++ 4.9.8.0<sup>5</sup> tillsammans med multimedialbiblioteket SDL 1.2<sup>6</sup> under operativsystemet Windows XP Professional. Datorn bestod av Pentium 4 2.26 MHz överklockad till 2721 MHz samt 640 MHz buss och 512Mb DDR 333 tillsammans med ett grafikkort av modell Nvidia Geforce 4Ti 4200 64Mb överklockat till 315 MHz (Core Clock Frequency) och 615 MHz på minnet (original 250/500).

### Simulera vattenvågor

För att simulera vattenvågor användes en TLM, Transmission Line Matrix – som är en metod för att simulera en vågs fortplantning och spridning från en valfritt lokaliserad källa.

---

<sup>1</sup> Citat hämtat från kursplanen

<sup>2</sup> Även kallad brännyta eller kaustika – kommer nu att skrivas **caustics** då rapporten är i datorgrafiksammanhang

<sup>3</sup> Ljuseffekt, ofta i form av en skimrande ring, som uppstår när ljus reflekteras från en ojämn yta eller passerar genom till exempel ett vattenglas. Ordet brännyta anknyter till brännpunkt.

<sup>4</sup> Med framework menas en ramverk innehållande fördefinierade funktioner att t.ex. skriva ut pixlar på skärmen, hantera muslick etc.

<sup>5</sup> Dev-C++ är en fri utvecklingsmiljö för C/C++ som inkluderar Mingw kompilatorn. Finns att hämta på <http://www.bloodshed.net/devcpp.html>

<sup>6</sup> SDL står för Simple DirectMedia Layer och är ett multimedialbibliotek med lågnivå-användning av mus, ljud, 2D, 3D med mera.

Transmission Line Matrix är en numerisk simuleringsteknik lämplig att användas för att lösa elektromagnetiska problem eller vågekvationen. Den är baserad på impulser som fortplantas i ett nätverk av transmissionslinjer. Forskning har visat att den kan användas för att lösa olika typer av problem men har mestadels använts inom elektromagnetismen.

Eftersom elektromagnetiska vågor och vattenvågor fortplantar sig på samma sätt, kan metoden användas för att simulera vattenvågors fortplantning. Den mest använda TLM version är uppbyggd av ett system av fyra portar i fyra olika riktningar.

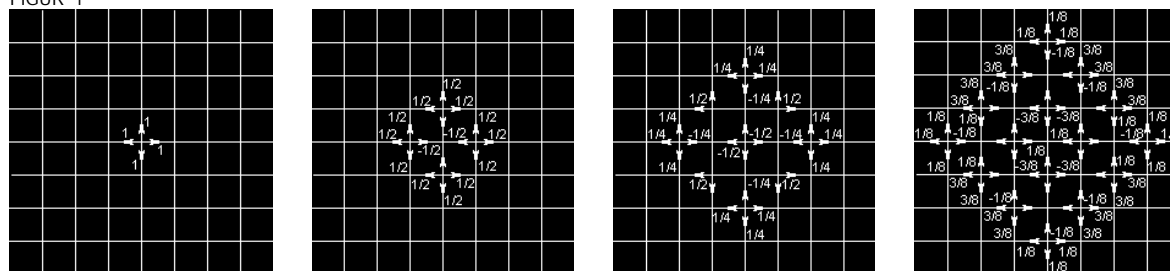
TLM baserar sig på Christian Huygens princip. Christian Huygen (1629-1695) menar att hans insikter inom vågrörelselära erhöles genom studerandet av vågorna i en kanal. Christian Huygens princip är:

*”Varje punkt i en vågfront kan tänkas som en ny punktkälla för vågor genererade i den riktigt som vågen färdas eller fortplantas i.”*

Med ovanstående menas att vid en tidpunkt,  $t=0$ , sprider en punktkälla en våg. Efter  $t=0$ , skall varje punkt på vågen ses som ett derivat från tidigare punktkällor.

Modelleringsprincipen är att sampla ytan och representera den med en kropp av transmissionslinjer. Utbredningen av vågen är i elektromagnetismen modellerad som volt och ström.

FIGUR 1



Bilderna representerar energiutbredning i tidsdomänen vid fyra olika efterföljande tidpunkter. Bilderna ovan visar en 2 dimensionellt TLM-nätverk bestående av ett system av fyra portar fyra olika riktningar.

Beräkningarna för vågen fås genom att räkna ut nedanstående matris. Varje nod kräver exakt en matrismultiplikation. I programmet motsvarar varje pixel en nod. Nedanför visas utbredningsmatrisen.

$$\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}_{k+1} = \frac{1}{2} \begin{bmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}_k$$

P står för pressure och s ovanför vektorn till vänster står för scattering (utbredning). I:et ovanför höger vektor står för incident (infallande) våg. K anger antalet iterationer eller vilken beräkning som multiplikationen åsyftar. Ovanstående matrismultiplikation motsvarar funktionen *scatter()* i programmets kod.

När utbredningsmatrisen är klar är det dags att utbyta energin. Hela tiden håller programmet på att räkna på detta sätt, för varje punkt. Det sker också en reflektion (funktionen *handleboundary()*) då vågen når ”poolkanten”. Den sker på motsvarande sätt som ovan är beskrivet. Denna

reflektion multipliceras med en faktor  $\alpha$ , som skall motsvara en energiförlust då vågen reflekteras tillbaka.

### Simulera caustics

För att simulera caustics används Ken Perlins 3D noise. Två lager av animerat noise i olika storlek läggs samman tillsammans med vågen. Lagren rör sig med olika hastighet åt höger samtidigt som de ändras i Z-led med olika hastighet. På håll ser detta bra ut men det är inget som tål närmare granskning. Dock är det tänkt att tillsammans med vågrörelserna, skall detta efterlikna fenomenet av en pool som har vågor och caustics. I denna simulering finns inget samband mellan caustics och vågor. Rörelse på ytan krävs för att simuleringen visuellt bäst skall komma till sin rätt.

### Akkumulering av caustics och våg per frame

Det finns ingen som helst vetenskaplig grund för hur simuleringens olika caustics samt våg läggs ihop. Istället har konstanter valts på ett sådant sätt att simulering skall ge bästa möjliga visuella intryck.

## MANUAL

Med vänster musknapp går det att göra enstaka klick för att skapa impulser.

Genom att hålla in vänster musknapp och dra sakta får man intrycket av att dra i vattnet.

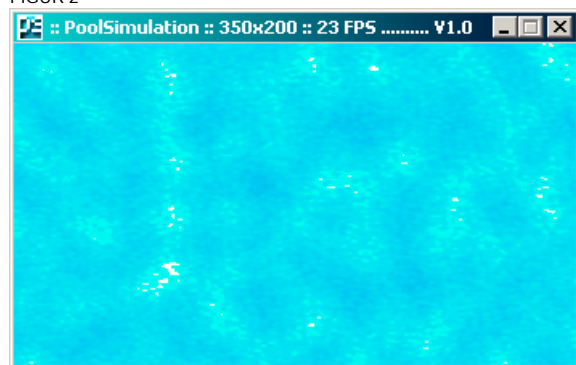
Genom att trycka på 'r' går det att starta en sinusformad impuls. Frekvens ändras med '+' eller '-'.

Hela tiden spelas muskoordinater in om ingen av knapparna är nedtryckta. För att "släppa" impulser på dessa koordinater skall höger musknapp klickas.

## RESULTATET

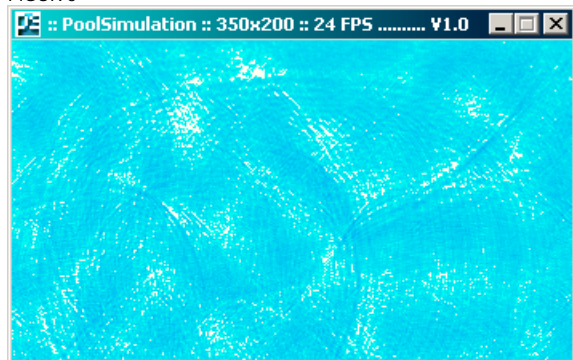
Nedan följer några bilder tagna ur programmet för simuleringen. Bilderna har inget samband inbördes utan är tagna spontant vid olika tidpunkter. De representerar olika former av vågrörelser som skulle kunna tänkas inträffa i en pool. Vyn är tänkt vara belägen rakt uppifrån, på ungefär 15-30 meters avstånd. För att kunna köra simuleringen i någorlunda takt, krävs det en processor på minst 2Ghz och rekommenderat är 3Ghz.

FIGUR 2



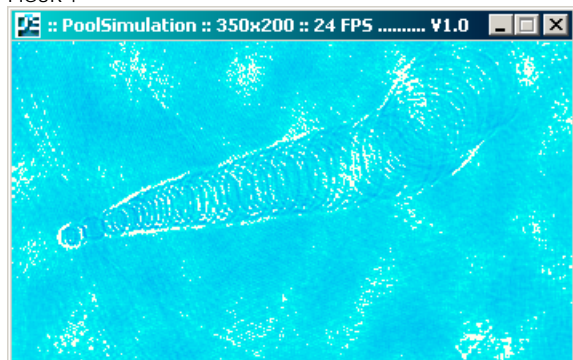
Här syns caustics utan vågor i vattnet.

FIGUR 3



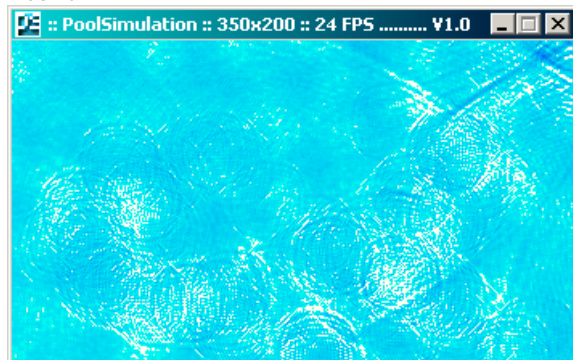
Poolen inkluderat allmänt liv på ytan.

FIGUR 4



Poolen tillsammans med ett farande objekt

FIGUR 5



Poolen med "impulser" i vattnet på olika ställen.

## SLUTORD

Vatten är fascinerande men relativt svårt att göra bra om hänsyn till volymen skall tas. Delvis därför valdes en helt rektangulär pool. I denna simulering har den tredje dimensionen lagts åt sidan med anledning av att det krävdes mer arbete och inte hinns med inom projektets tidsram. Hade en tredje dimension nyttjats, utan GLSL, så hade det behövs en displacement-map, bump-map etc. Detta är som bekant något som tar tid längre att implementera. Egentligen var tanken inledningsvis att använda delvis en GLSL-shader vilket då markant hade underlättat. Problemet som då uppstod var att projektplatsen var belägen på hemorten, vilket innebar att universitetets datorer med GLSL-grafikkort inte kunde nyttjas. Det grafikkort som stod till förfogande då var

ett Geforce4Ti 4200 med stöd för endast OpenGL 1.3. Utan hårdvarustöd hade det blivit svårt att hinna rendera tillräckligt snabbt och att själv hinna med att programmera inom projektets tidsram.

Vidare hade det varit betydligt intressantare att kunna modellera och animera fysikaliskt korrekta caustics som beror på ytans form eller dynamik. Vågorna i sig rör sig på ett tillfredställande sätt och lämnar inte mycket fler önskemål, förutom den tredje dimensionen yvmässigt sett och möjligheten att kunna ställa in hastigheter, i alla fall under 2 dimensioner. Att använda sig av TLM i 3 dimensioner (TLM och inte bara vyn) hade varit intressant men i dagsläget troligtvis omöjligt på av dagens begränsade beräkningskapacitet.

Sammanfattningsvis har projektet varit stimulerande ur teknisk och kreativ synvinkel. Mycket av simuleringen kan användas i framtida projekt där olika komponenter kan återanvändas. Förutom det, har stor portion respekt för Ken Perlins noise ackumulerats fram. Den skepsis som tidigare funnits är nu helt försvunnen. Det går att visualisera olika former av texturer och geometrier av riktigt hög kvalité till helt skilda ändamål. Jag är nöjd med resultatet som erhöles och tycker det ser relativt bra ut när rörelse finns i poolen.

## REFERENSER

<http://oldeee.see.ed.ac.uk/SaS/Thesis/Ma/thesis.pdf> [2005-01-2]  
<http://www.jlherring.f9.co.uk/tlm/> [2005-01-2]  
[http://www-ibt.etec.uni-karlsruhe.de/postscript/encf\\_14.pdf](http://www-ibt.etec.uni-karlsruhe.de/postscript/encf_14.pdf) [2005-01-2]

F. Kenton Musgrave et. al. (2003) Texturing & Modelling - a procedural approach kap. 16