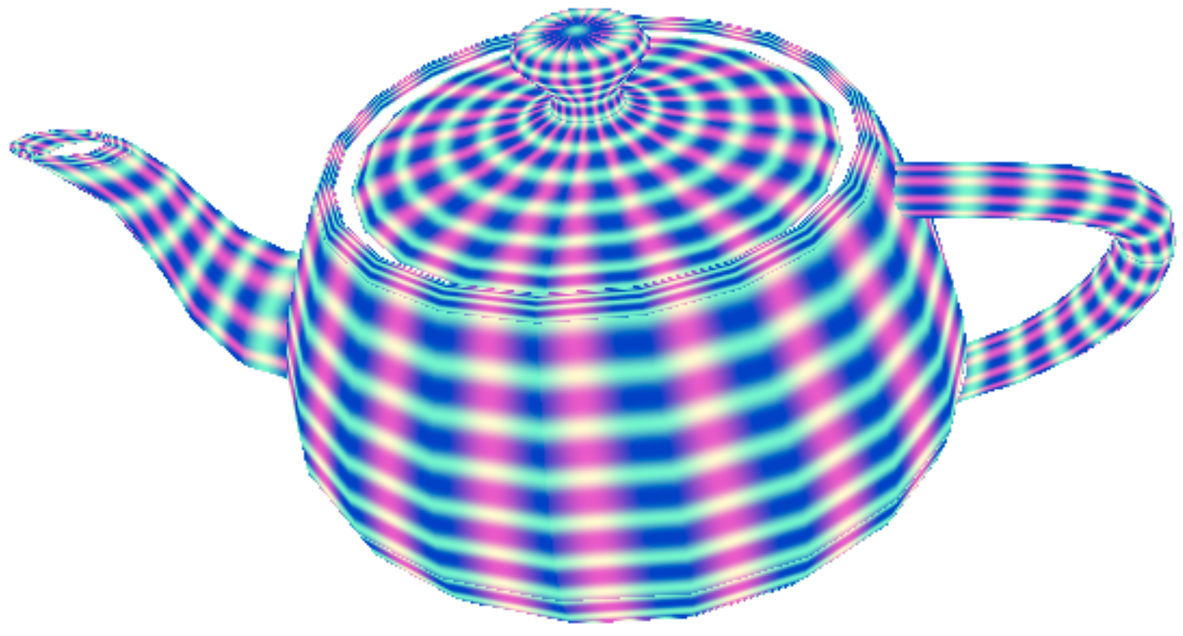
Information Coding / Computer Graphics, ISY, LiTH

TNM084
Procedural images

Ingemar Ragnemalm, ISY



Lecture 1

- Course goals and contents
- Lectures, labs, projects
- Time plan
- Examination
- Interactive poll
- Base functions and basic procedural generation

Who am I?

Lecturer
Researcher
Hacker (in the legal sense)
Game designer
Geocacher

Who are you?

Students in 4th or 5th year
Most likely in the Media Technology program
Considerable knowledge
but maybe so much graphics lately

Procedural *Images*?

We will produce images, but not only that.

Images usually means *textures*.

- Simple procedural images
- Noise-based images
- Procedural 2D animations
- Procedural 3D shapes
- Fractals

More generally speaking: Procedural generation of *assets*.

Learning goals

1) Knowledge of the principles for procedural generation of assets, including noise functions and fractals, plus shader programming

2) Practical experience by laborations

3) Deeper knowledge of a chosen specific problem as a project

Everybody should learn something new!

The different parts of the course

Basic procedural image/
texture generation

Random numbers

Noise
functions

Shaders

Fractals

3D geometry

Real-time

News 2021

Pretty damn *everything*!

• Replacing Stefan Gustavson as course leader. (He is still involved as advisor.)
• Much more participants than usual
New course model based on TSBK07 and TSBK03 (because they work).

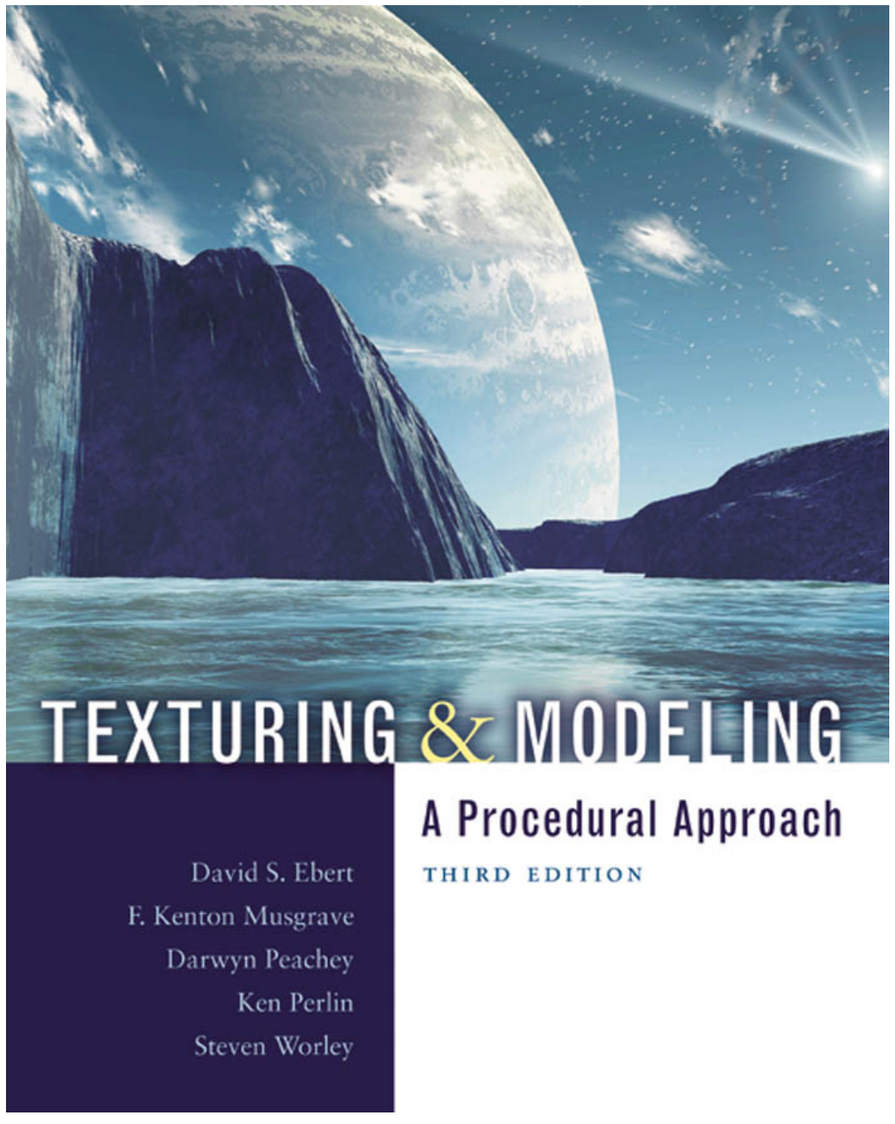
• All lectures were made from scratch
• New labs, 99% new
• Additional examination on the labs
• Projects are to be presented orally (by video) at the end

News last year

• Number of participants back to normal.
• Projects do not have to be solo this time.
• Projects are planned to be presented on location. (This did happen.)

News this year

• One more lab assistant (Måns Gezelius)



Course book

Out of print, NOT available as physical copies but available on-line.

Course page

https://www.computer-graphics.se/TNM084/

Brand new page 2021!

My adress:

ingemar.ragnemalm@liu.se
ingemar@ragnemalm.se

Linköping B-house, corridor A, between entrance 25 and 27 upper floor.
(The door with all the comics!)

Overview

- 10 lectures
- 4 laborations
- Project

Same model as my other courses

Start with the lectures as a base

Practice with given tasks: laborations

Projects: Considerable freedom, go deeper into something you really want to do!

Lecture plan (preliminary)

1: Intro, simple procedural images

2: Randomness, Perlin noise

3: OpenGL and GLSL

4: More noise functions, Voronoi noise, anti-aliasing

5: OSL, Open Shading Language

6: Procedural geometry, fractals

7: Procedural grammars, fractal terrains, Fractal Brownian Motion

8: Instancing, Geometry and tesselation shaders

9: Particle systems, billboards, FBOs

10: Repetition, project discussions, loose ends, guest lecture

Examination

- Laborations demonstrated to the lab assistants
- Short tests ("duggor") at start of laboration 1-4 + retake at 5th
- Projects are presented orally, on location, and by a report

Note: I want your reports to be public, I want to upload them to the course page so everybody can check out what the others did.

Suggestion: Can we use "opt-out" so I can assume that they are public unless you say otherwise?

Possibly even videos, if you can make them.

Laborations

1: Generation of images on CPU and GPU

2: Generation of images with OSL

3: Procedural geometry generation (terrains, trees)

4: Geometry and tesselation shaders (procedural planet)

X1: Extra lab 1 + retake
X2: Extra lab 2

We almost fit in one lab but we have two, no booking needed. Work in pairs.

You may use your own computer if you prefer that.

OpenGL 3.2+ = Modern OpenGL!

We are using OpenGL for much of the course (everything except OSL). OpenGL 3.2 is our "baseline", where modern OpenGL starts. You are free to use newer versions as you please. Lab 4 needs OpenGL 4.2 or higher.

Old OpenGL = Upp till 2.1.
Modern = 3.2 och upp.
Latest = 4.6.

Modern = Always shaders, always VAOs, old calls are taken out. Better performance, more flexible, the math is more visible.

Time plan for projects

Tight! You need to make progress fast.

During the lectures: Search for project ideas.

1/12: Last lecture: Your project plan should be handed in and approved. Earlier if possible.

12-16/12: Half-time check (informal).

In january, before VT1: Project presentations and report.

About the project

- Choose from your interests and ambition level
- No fixed list with projects to choose from (since it would limit you)
- Focus on some interesting problem/algorithm
- Feel free to go beyond the book/course material.
- You may base your work on the lab material, using C or C++, but you are allowed to use other material.

Concerning external material, language, platform

Anything goes! (Essentially.)

You should use tools that makes you as productive as possible.

You can use code and libraries found on-line.

Beware: Only use things you are sure will help. External libraries will sometimes cost you more than you gain!

but my general message about external sources is:

Cheating is allowed if it is properly documented!

I must know what you did, and what you got from someone else.

Connection to other projects? Allowed, if it is clear what belongs here and what belongs in the other project. (If that other project is not in another of my courses, you need OK from the other examiner.)

Literature search

Google and Wikipedia are not the best sources! Try this:

dl.acm.org

If you Google use Google Scholar!

Grades = project + tests at labs (duggorna)!

• Up to 50 points
• Limits: 20, 30, 40

• 20 points on duggor (max 5p each)
• 30 points on the project

• "Duggor" are not related to being allowed to do the lab. Everybody can do the labs regardless of score.

Scores

(Rip-off of TSBK03)

5p presentation (prepared, nice, understandable, nice demo)

5p report (neat, readable, proper references)

10p technical quality (was it interesting, beyond course base?)

8p goal fulfllment

2p on time

Note that top grade is not guaranteed only on technical quality - but it helps.

Total hand-in material

Important!

Your total hand-in consists of

- Initial project plan

and at the end of the course

- Report (PDF, filename should include LiU ID)
- Source code (repository link)
- *Presentation* with theory and demo, max 15 minutes.

Questions

on the course, grades, projects etc...?

Online poll

I need to know who you are, what pre-knowledge you have, how you work!

We can not adjust everything on-the-fly but we can adapt to known needs.

Help us to make a good course that fits you!

Use web browser:

https://ragnemalm.se/quiz/

Functions

Important tools that we are going to use.

- abs, min, max
- sin, cos
- floor, ceiling, mod
- clamp
- step
- smoothstep
- splines
- fract

Harmonic functions

Our dear sin and cos

Floor, ceiling, mod

Well known standard functions

Floor = integer part

Ceiling = integer, but up

mod = modulo

Matrix multiplication

Translation

Rotation

Scaling

Mirroring

Color transformations

Translation and scaling

2D

Translation: $T(t_x, t_y) = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$

Scaling: $S(s_x, s_y) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Mirror: $M_x = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

3D

Translation: $T(t_x, t_y, t_z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Scaling: $S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Mirror: $M_x = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Rotation

2D

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

+ rotation around arbitrary axis

3D

Around the x axis: $R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Around the y axis: $R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Around the z axis: $R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Mix

Linear interpolation

mix(rand(i), rand(i + 1.0), f);

```
float mix(float a, float b, float f)
{
    f = clamp(f, 0, 1);
    return f*a + (1-f)*b;
}
```

Fract

Fractional part

Take away the integer part

fract(a) = a - (int)a = a - floor(a)

(Same only for positive numbers)

fract(12.34) = 0.34

Abs, max, min

Well known standard functions

```
float abs(float a)
{
  if (a < 0)
    return -a;
  else
    return a;
}
```

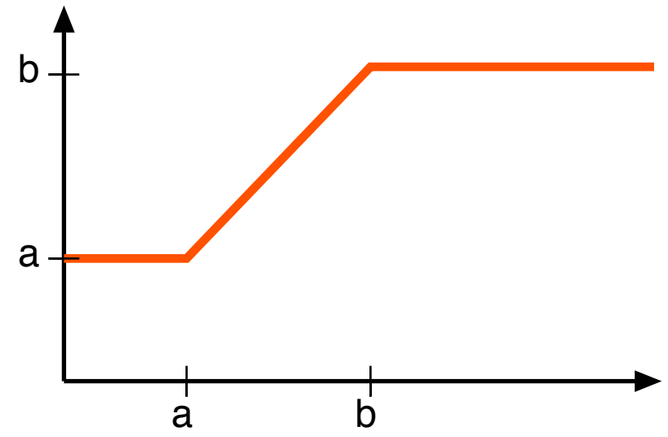
```
float max(float a, float b)
{
  if (a < b)
    return b;
  else
    return a;
}

float min(float a, float b)
{
  if (a < b)
    return a;
  else
    return b;
}
```

Clamp

Limit the output to a range

```
float clamp(float x, float a, float b)
{
  return (x < a ? a:  (x > b ? b : x));
}
```



or a bit more readable

```
float clamp(float x, float a, float b)
{
  if (x < a)
    return a;
  else if (x > b)
    return b
  else
    return x;
}
```

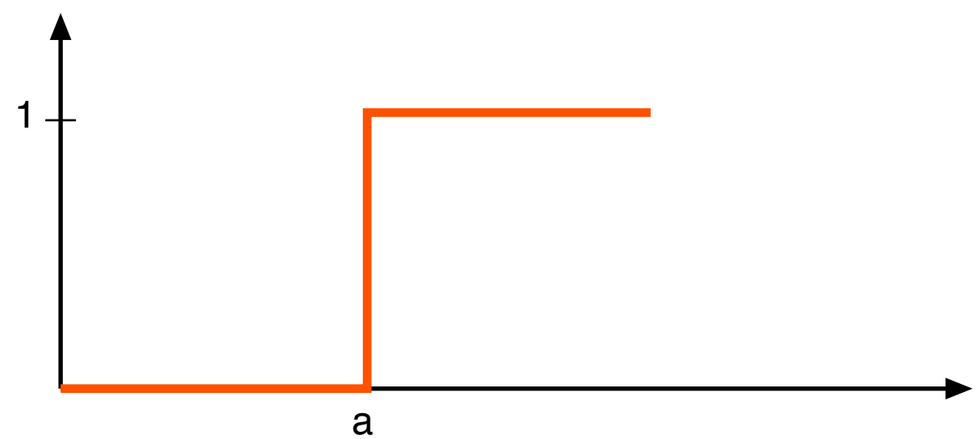
or simply

```
float clamp(float x, float a, float b)
{
  return min(max(x, a), b);
}
```

Step (Heaviside step function)

A simple 0-1 transistion

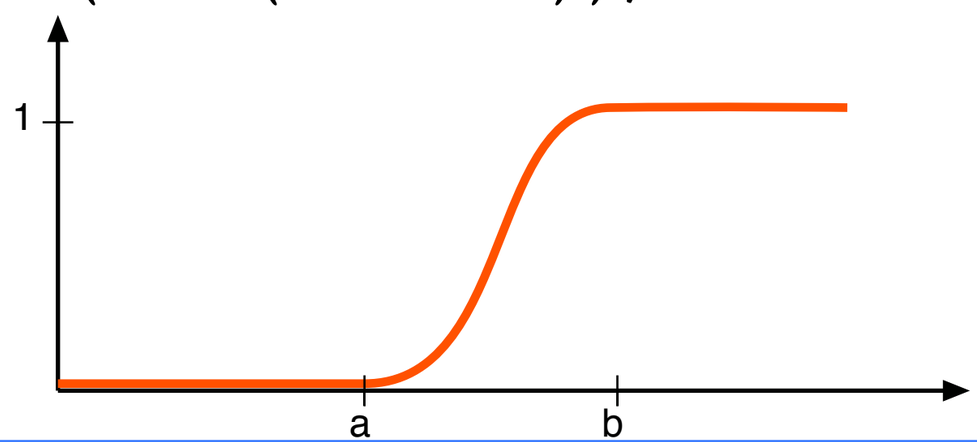
```
float step(float a, float x)
{
    return (float) x >= a;
}
```



Smoothstep

A smooth 0-1 transistion

```
float step(float a, float b, float x)
{
    if (x < a)
      return 0;
    if (x > b)
      return 1;
    x = (x - a) / (b - a);
    return (x*x*(3 - 2*x));
}
```



Smoothstep is really a spline!

Spline = a curve defined by a polynomial

Usually 3rd or 4th degree polynomial

Common splines:

Bézier (3r or 4th degree)

Catmull-Rom

Splines

Mathematically described curves

Commonly used for 2D drawings

Also useful for describing movement paths

Can be combined to surfaces

Bézier splines

2nd or 3rd degree polynomials

3 or 4 control points per section

Does not pass through all control points

Catull-Rom splines

3rd degree polynomials

4 control points per section

Passes through all control points

More on these later

Regular patterns

Sequential patterns

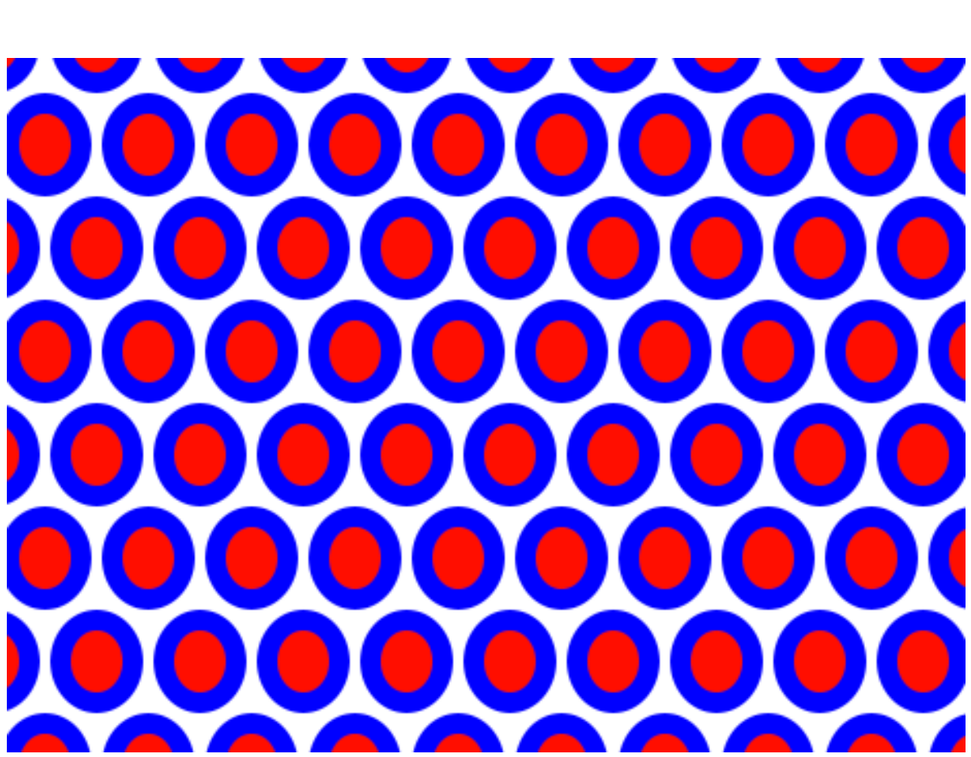
Parallell patterns

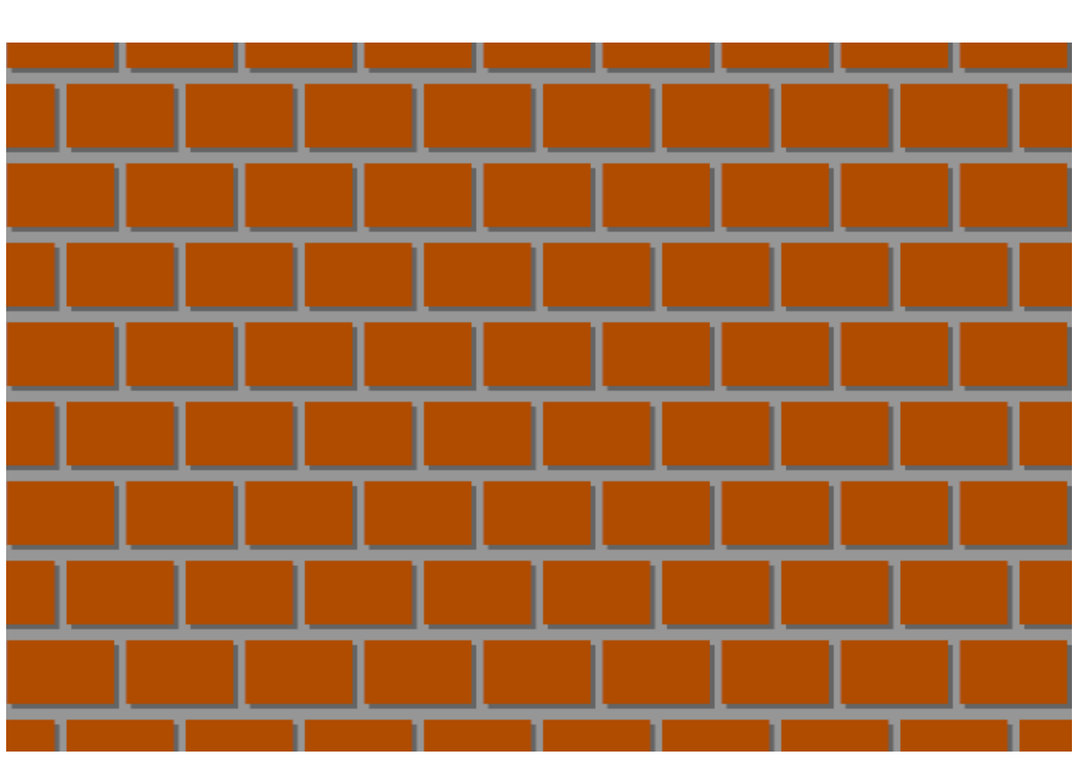
Animated patterns

Regular patterns

Repeated patterns

No randomness





Sequential patterns

Generate a patterns by sequentially drawing objects

Intuitive

Easy to design

Inefficient

Inflexible

Sequential pattern, examples

Checkerboard

Circles placed in selected locations

Bars drawn the screen in 2 directions

Brick wall

Based on placing shapes

Parallel patterns

Calculates each pixel individually

Based on functions of the location

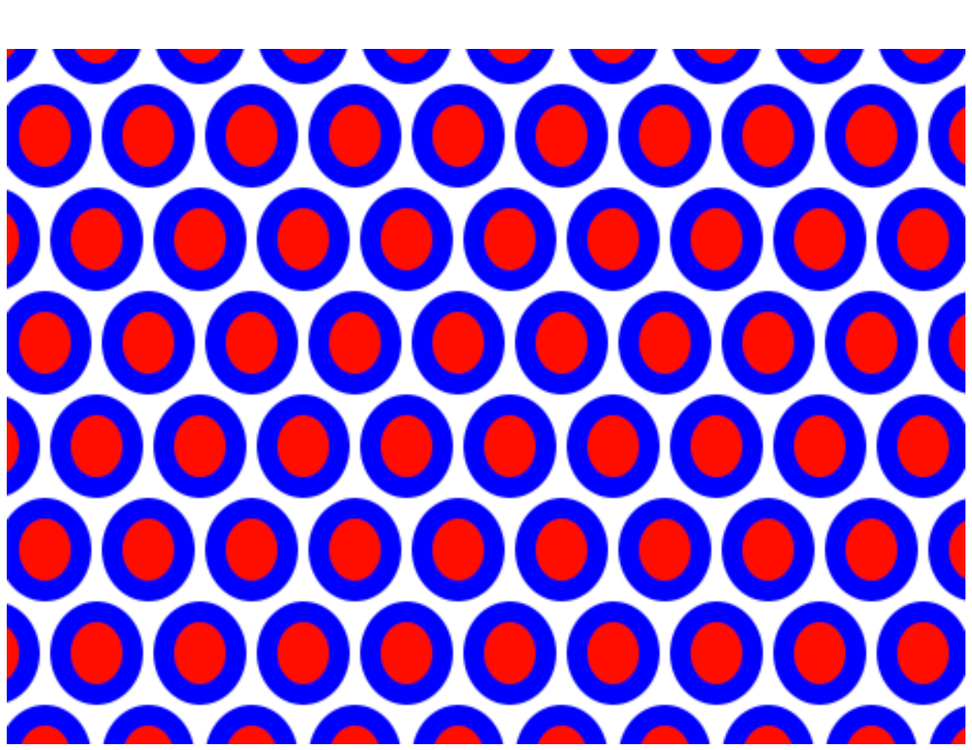
Independent calculations

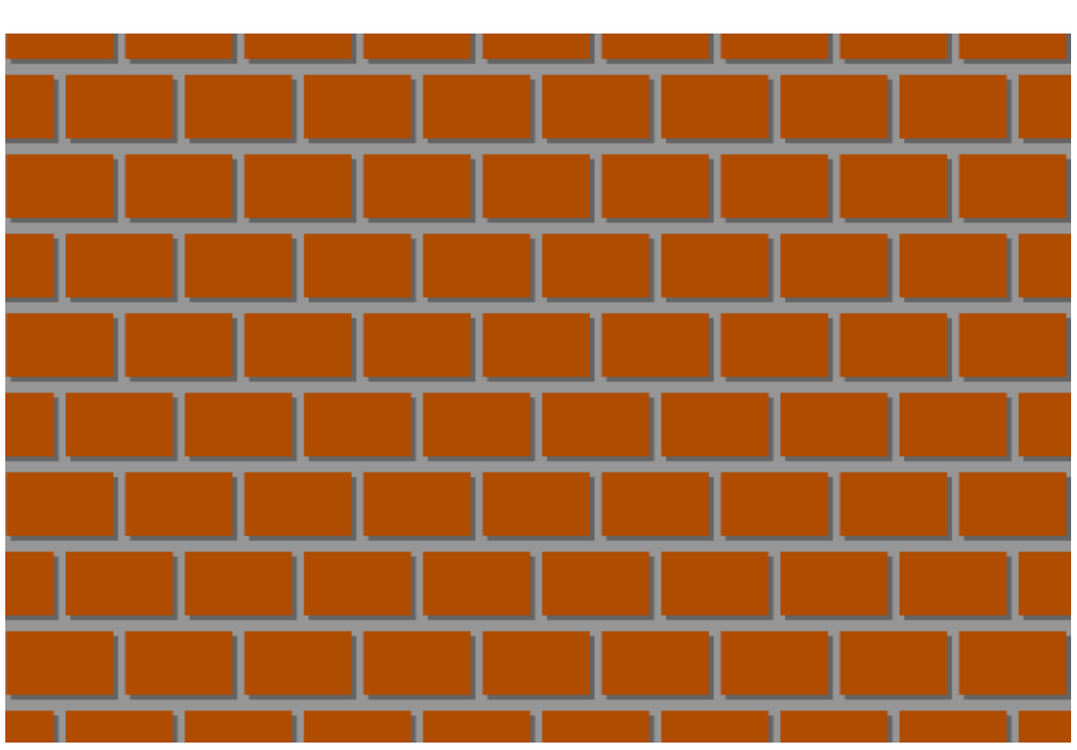
Suitable for parallel implementation

= GPU friendly!

Parallel patterns, examples

Brick wall, circles - same as before but described differently





Animated patterns

Add a time variable

Affect some variable, color, location...

CPU:

```
glUniform1f(glGetUniformLocation(program, "time"),
glutGet(GLUT_ELAPSED_TIME) / 1000.0);
```

GPU:

```
uniform float time;

if (sqrt(xx*xx + yy*yy) > 0.55 + sin(time)/4.0)
```

Regular patterns, conclusion

Good for fixed, manual designs

Hard to make interesting and varied

Should be based on location, parallell friendly

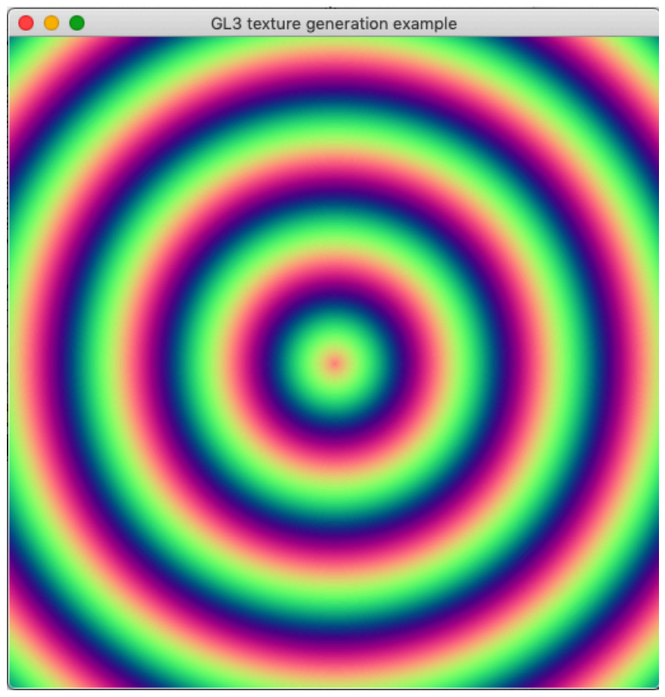
Next: Add randomness!

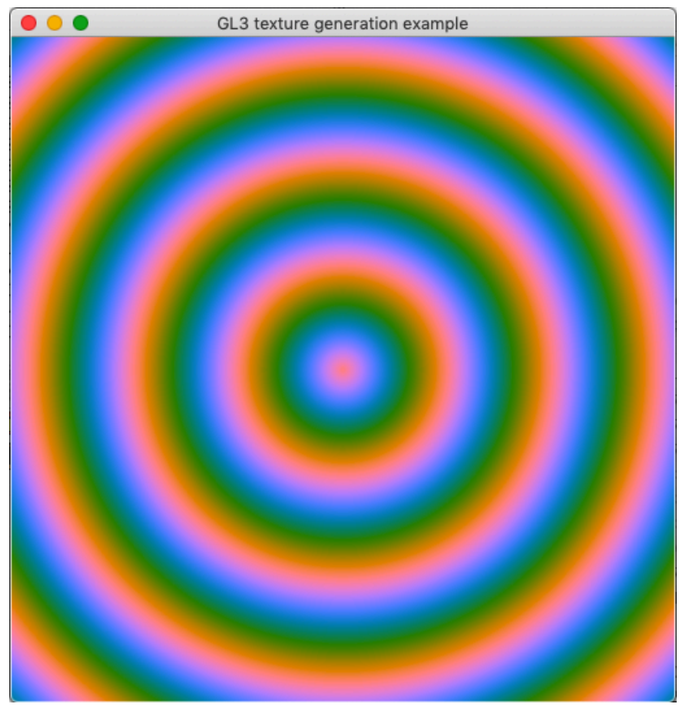
Information Coding / Computer Graphics, ISY, LiTH

Parallel patterns, GPU vs CPU

Lab material for lab 1

Initial: Trivial patterns for both CPU and GPU





End of lecture 1

Questions?