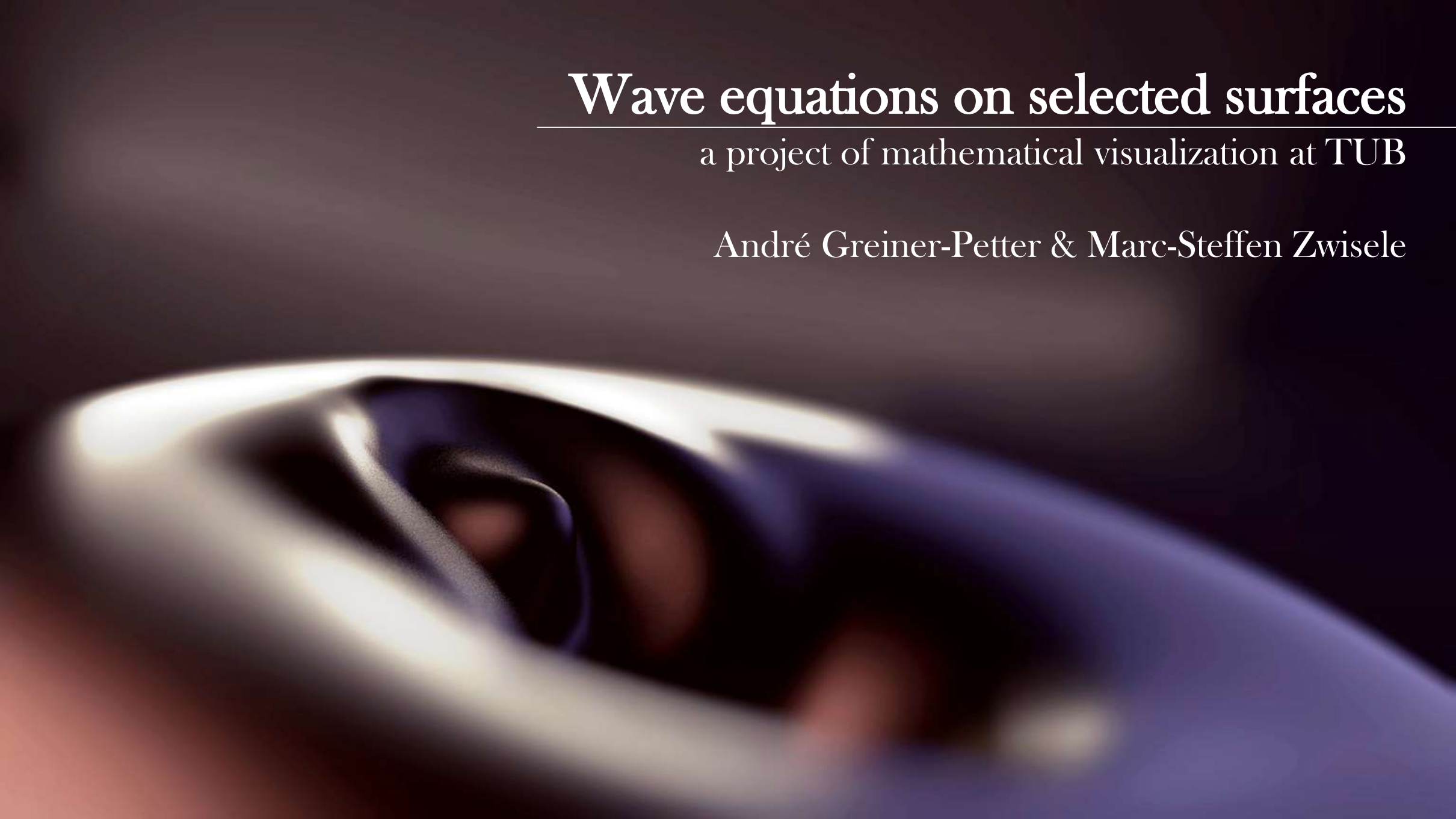


Wave equations on selected surfaces

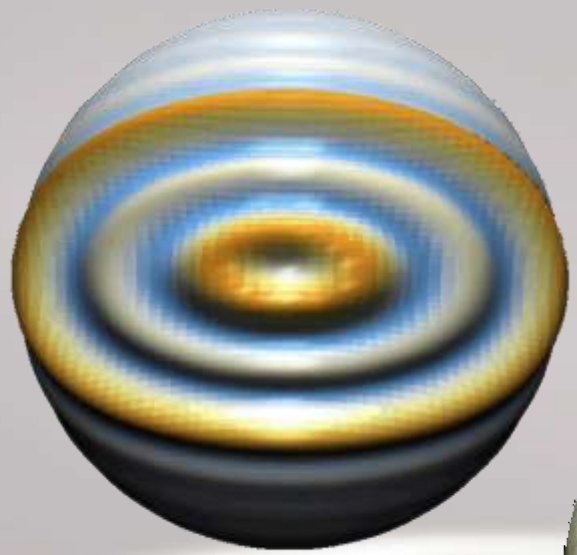
a project of mathematical visualization at TUB

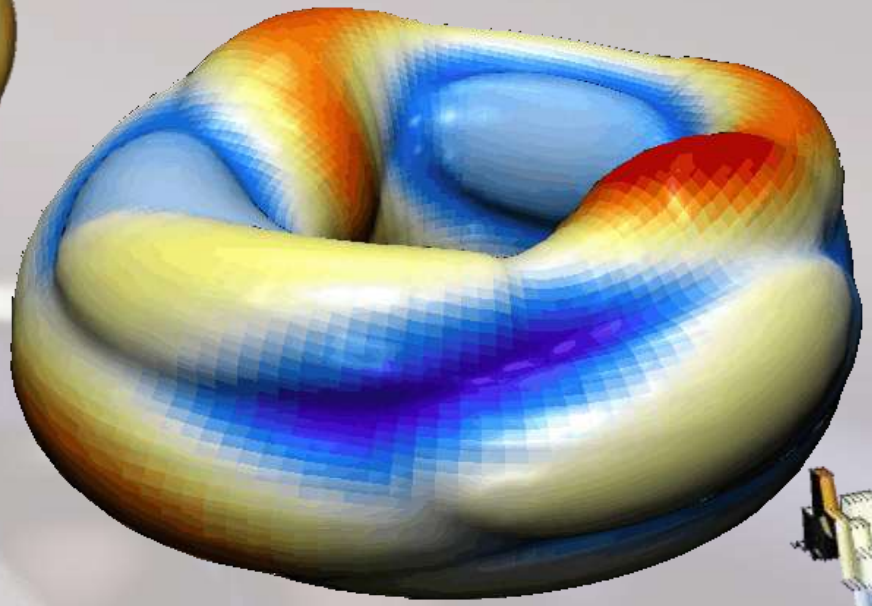
André Greiner-Petter & Marc-Steffen Zwisele

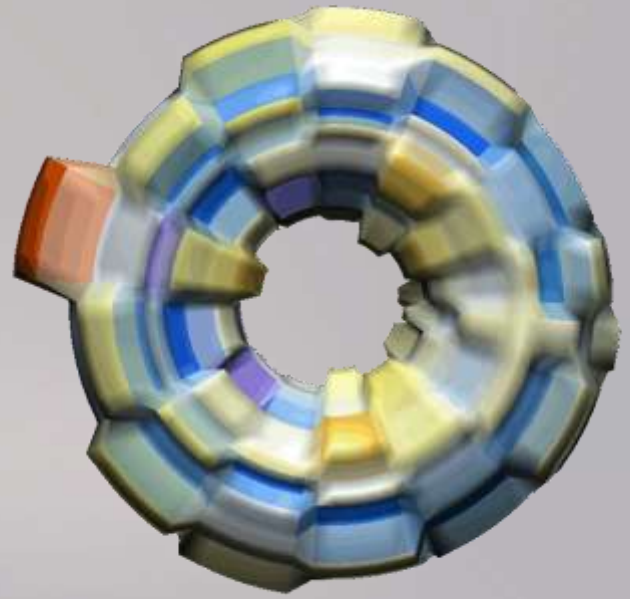
Structure

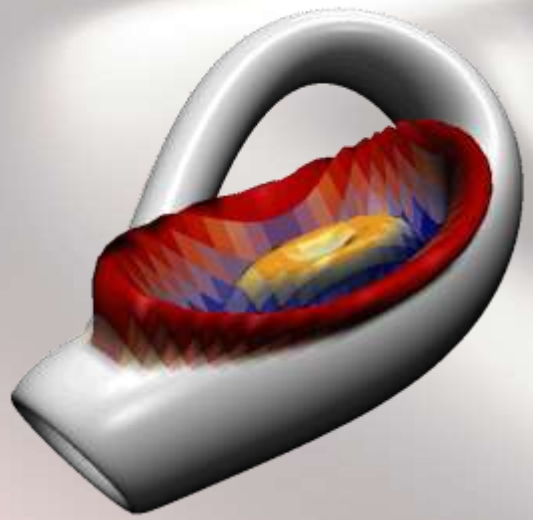
- Introduction
- Mathematical background
  - Wave equations
    - Fourier ansatz
    - Cubic splines
    - Discrete ansatz
- Software Implementations
  - Behind the scenes (start a wave)
    - WaveTool
    - Multithreading
    - Simultaneous waves
  - Wave implementations
- Run the application
- Ideas of improvements
  - Distortions of transformations
  - Other surfaces

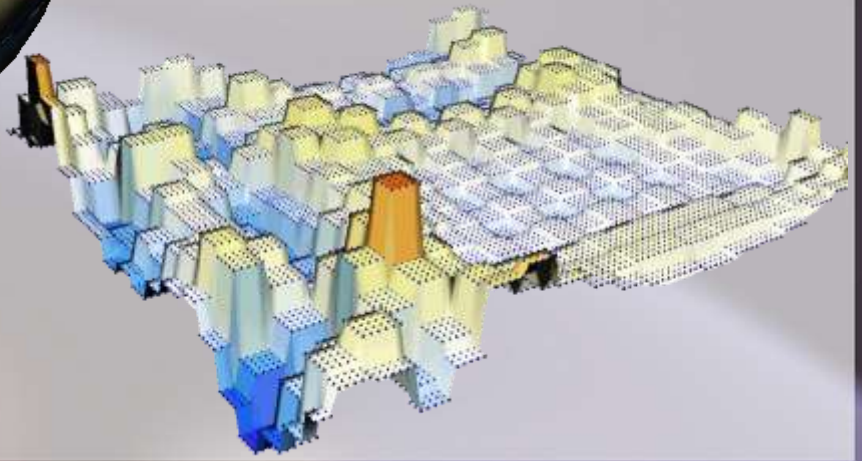
Introduction









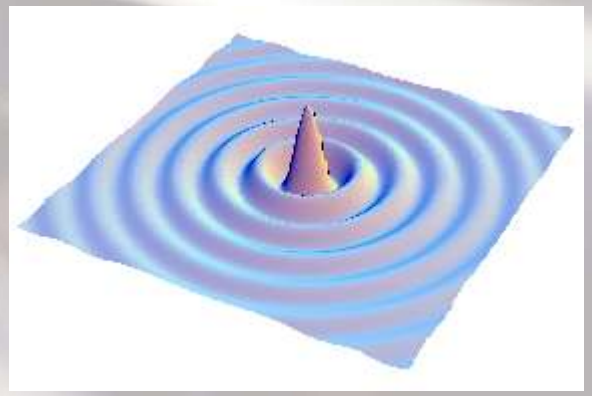


Mathematical background

- Wave equation
  - Fourier 2D
  - Cubic splines
  - Discrete wave equation

Wave equation

$$\nabla^2 u - \frac{1}{c^2} u_{tt} = 0$$



Fourier 2D

Wave equation:

$$u_{tt} = c^2 \nabla^2 u = c^2 (u_{xx} + u_{yy})$$

Dirichlet boundary conditions:

$$u(0, y, t) = u(1, y, t) = 0$$
$$u(x, 0, t) = u(x, 1, t) = 0$$

Initial functions:

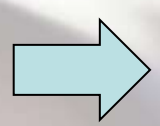
$$u(x, y, 0) = f(x, y),$$
$$u_t(x, y, 0) = g(x, y),$$

Fourier 2D

Product ansatz:

$$u(x,y,t) = X(x)Y(y)T(t)$$

Insert in the wave equation:



$$\frac{T''}{c^2 T} = A = \frac{X''}{X} + \frac{Y''}{Y}.$$

Fourier 2D

Single summand:

$$u_{mn}(x,y,t) = X_m(x)\,Y_n(y)\,T_{mn}(t)$$

$$= \sin \mu_m x \sin \nu_n y \,(B_{mn} \cos \lambda_{mn} t + B^*_{mn} \sin \lambda_{mn} t)$$

Complete Fourier Row:

$$u(x,y,t) = \sum_{n=1}^{\infty} \sum_{m=1}^{\infty} \sin \mu_m x \sin \nu_n y \,(B_{mn} \cos \lambda_{mn} t + B^*_{mn} \sin \lambda_{mn} t).$$

Fourier 2D

With the frequencies:

$$\mu_m = \frac{m\pi}{a},\ \nu_n = \frac{n\pi}{b},\ \lambda_{mn} = c\sqrt{\mu_m^2 + \nu_n^2}.$$

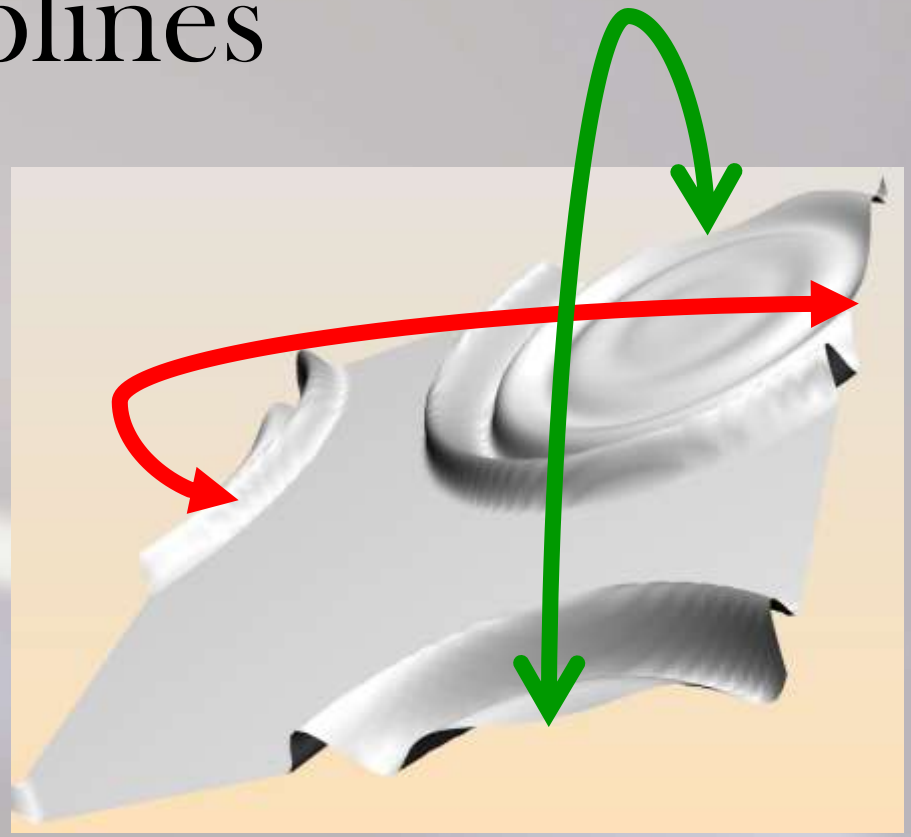
And coefficients:
(Given by the inititial functions *f,g.*)

$$B_{mn} = \frac{4}{ab}\int_0^a \int_0^b f(x,y) \sin\frac{m\pi}{a}x \sin\frac{n\pi}{b}y\, dy\, dx$$

$$B^*_{mn} = \frac{4}{ab\lambda_{mn}}\int_0^a \int_0^b g(x,y) \sin\frac{m\pi}{a}x \sin\frac{n\pi}{b}y\, dy\, dx.$$

Cubic splines

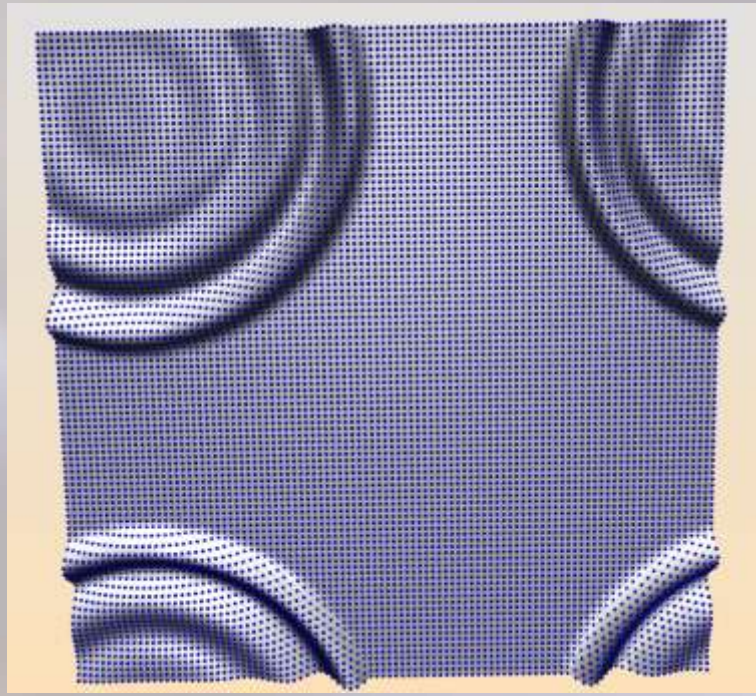
- Model a Wave with splines to
  - Have changeable parameters
  - Satisfy a kind of periodic boundary conditions



Discrete wave equations

- Periodic boundary conditions:

$$u(0,y,t) = u(1,y,t)$$
$$u(x,0,t) = u(x,1,t)$$
$$u_x(0,y,t) = u_x(1,y,t)$$
$$u_y(x,0,t) = u_y(x,1,t)$$

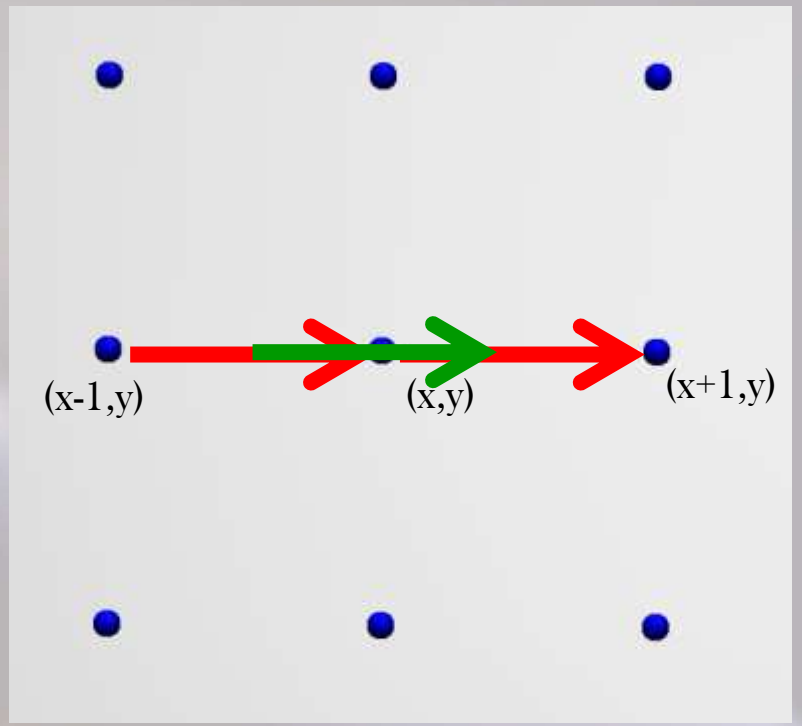


Discrete wave equations

$$u_x(x,y,t) = u_{x+1,y,t} - u_{x,y,t}$$

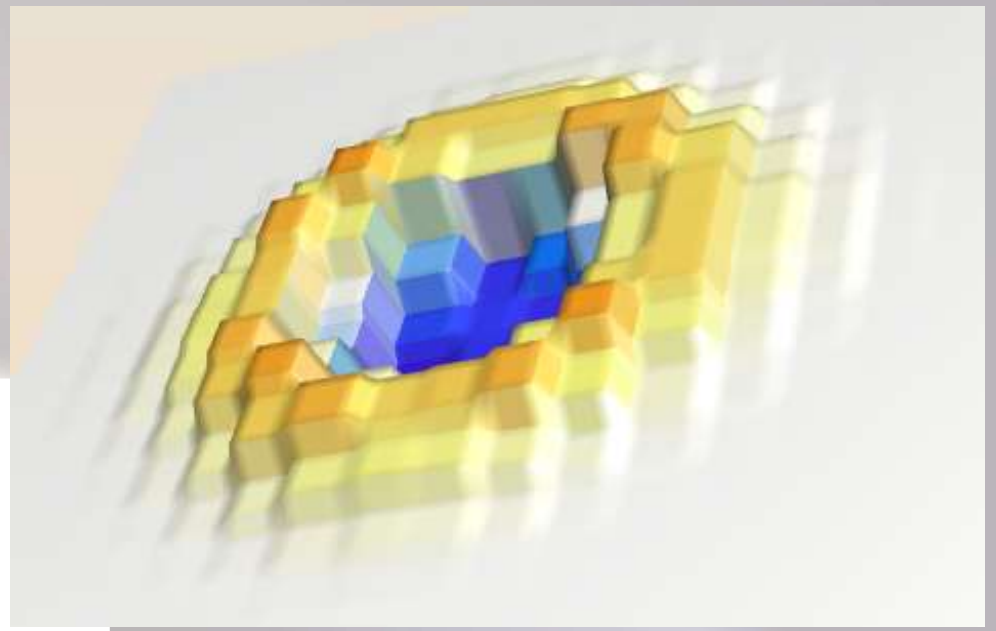
$$u_{xx}(x,y,t) = u_x(x-1,y,t) - u_x(x,y,t)$$

The same for y and t.



Discrete wave equations

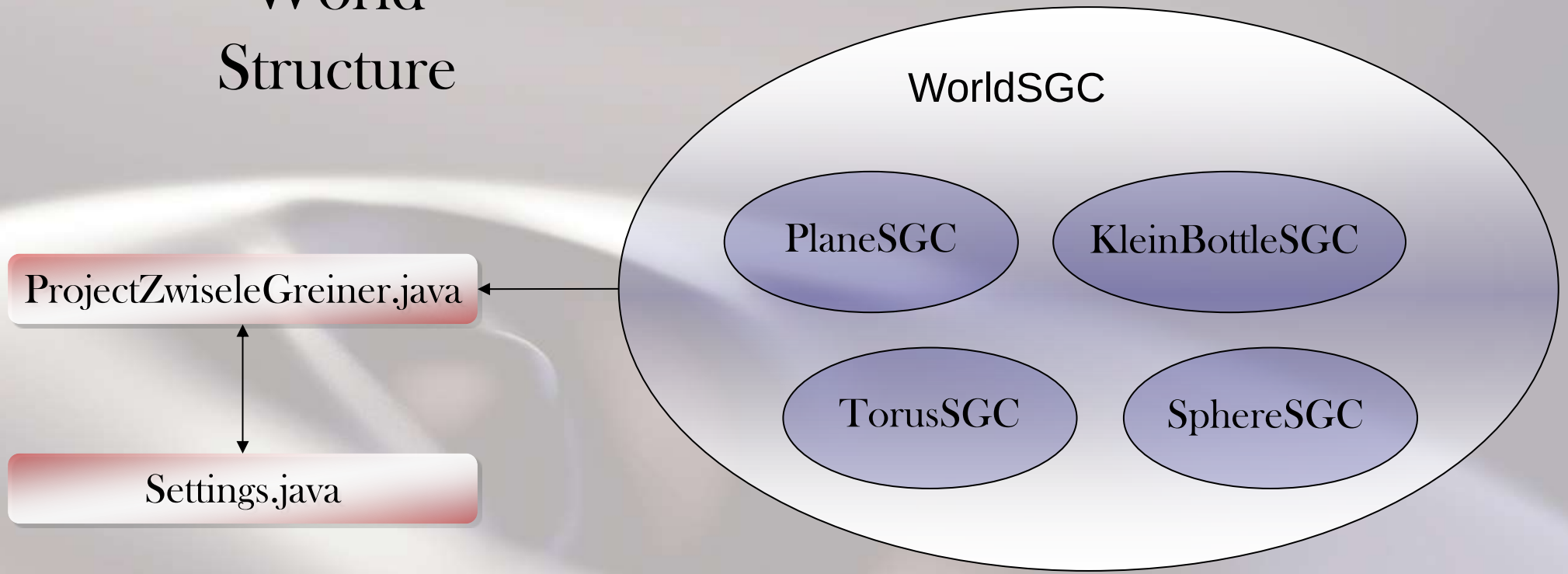
$$u_{tt} = c^2(u_{xx} + u_{yy})$$



$$\begin{aligned} & u_{x,y,t+1} + u_{x,y,t-1} - 2u_{x,y,t} \\ & = c^2(u_{x+1,y,t} + u_{x-1,y,t} \\ & + u_{x,y+1,t} + u_{x,y-1,t} - 4u_{x,y,t}) \end{aligned}$$

Software Implementations

World Structure



Program Structure

Geometry
SceneGraphComponent

Geometry

AbstractDeformableGeometry
extends IndexedFaceSetFactory
implements Deformable

- getCoordinates(int i)
- translateVertex(int i, double h)
- refresh()
- update(double[][] verts)

Plane

Torus

Sphere

Klein-Bottle

Tool

WaveTool extends DragEventTool
implements FaceDragListener

- setGeometry(AbstractDeformableGeometry deformGeo)
- faceDragStart(FaceDragEvent fde)
- setter methods for wave options

Behind the scenes

Start a wave

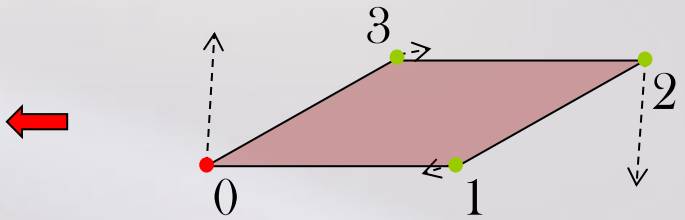

```
@Override
public void faceDragStart( FaceDragEvent event ){
    int[] faId   = event.getFaceIndices();  // start Position
    startPos     = deformGeo.getCoordinates( faId[0] );

    animation = new AnimationThread( deformGeo, timeStep );
    animation.start();
}
```

Behind the scenes

Start a wave

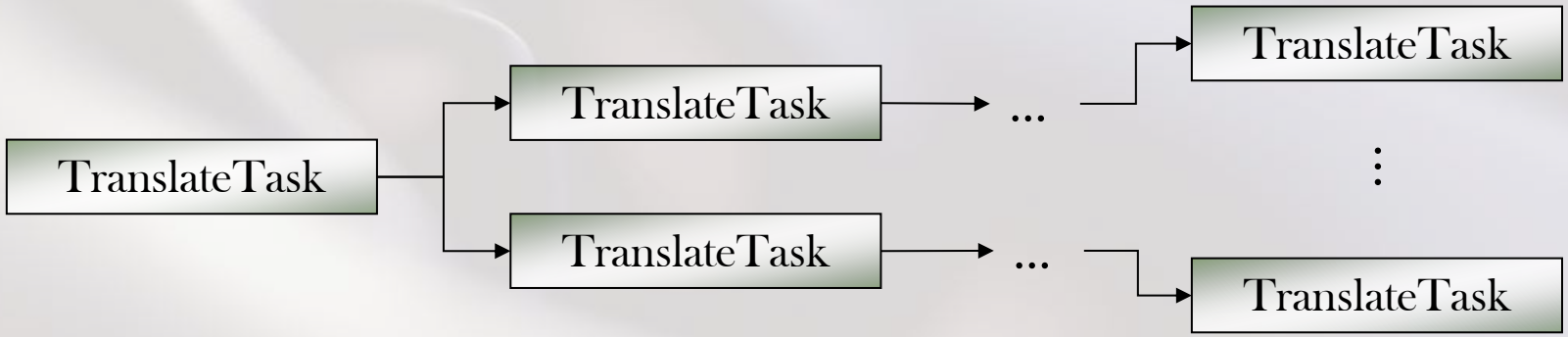

```
// method in highly simplified terms
private synchronized void animate(){
    while ( true ){
        calcThread.calculateNewVertices();
        // wait for results
        try { this.wait() } catch( InterruptedException e ){}
        deformGeo.update();
    }
}
```



Behind the scenes

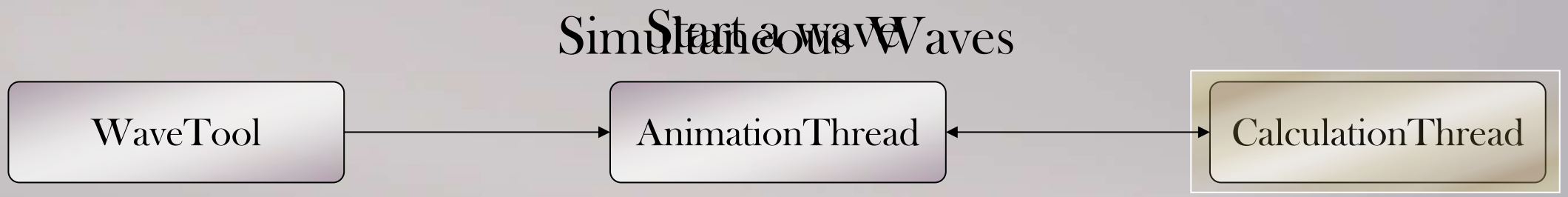
Start a wave


```
private synchronized void transCalcs(){
    // calculate high of each vertex
    translationPool.invoke( new TranslateTask( vertices, verticesColors, 0, numOfVerts - 1 ) );
    synchronized ( masterThread ) { masterThread.notify(); }
}
```



Behind the scenes

Simultaneous Waves




```
private class TranslateTask extends RecursiveAction {
    @Override
    protected void compute( ){
        for ( int index = start; index <= end; index++ ){  // if end-start < 100
            double h = wave.getHeight( position );
            vertices[ index ] = deformGeo.translateVertex( index, h );
        }
        ... // new recursive TranslateTask
    }
}
```

Behind the scenes

Simultaneous Waves


```
@Override
public void faceDragStart( FaceDragEvent event ){
    int[] faId   = event.getFaceIndices();  // start Position
    startPos     = deformGeo.getCoordinates( faId[0] );

    animation = new AnimationThread( deformGeo, timeStep );
    animation.addWave( createNewWave( startPos ) );
    animation.start();
}
```

Implementation of the waves

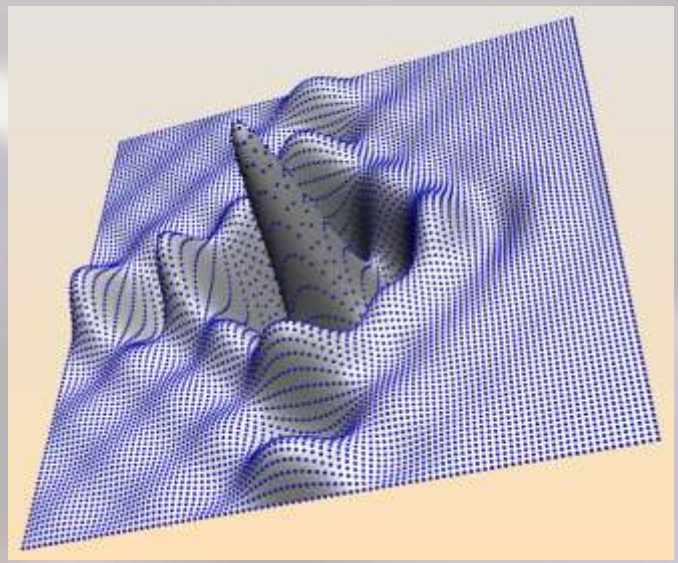
- Bounded Wave
- Simple Wave
- Discrete Wave

Bounded Wave (Fourier 2D)

- Calculate the coefficients in the constructor
- Calculate the Fourier Rows every time .getHeight is called

That is the reason

why it is so slow.

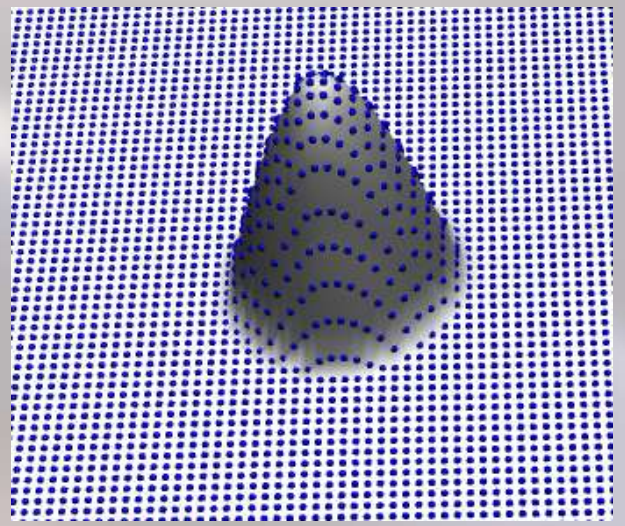


Simple Wave

- Cubic Splines, multiplied by cos(πr)+1 to smooth the wave at the end.
- Shifted in time by spline.valueAt(r-c*t)
- c is the wavespeed
- r is the distance from the initial point to (x,y)

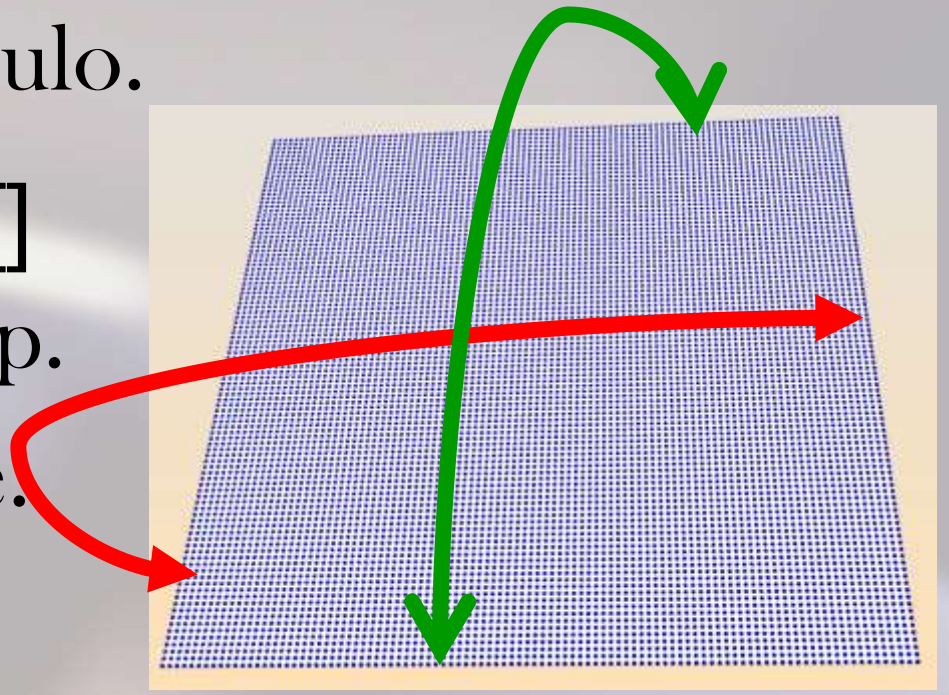
Discrete Wave

- Discretize the surface internal as a two dimensional array.
- For each time step:
  - prev, curr, next
- Calculate a initial wave



Discrete Wave

- Identify in the calculation the each of the two opposite borders using modulo.
- Update next[][] with prev[][] and curr[][] in every timestep.
- getHeight(x,y) is in notime.



Run the application!

Ideas of improvements

- Distortions of transformations
- Other surfaces (klein bottle)

Reference

http://matheplanet.com/matheplanet/nuke/html/uploads/4/1712_3.5_ABB._Amplitudenabh_ngige_Zylinderwelle.png

http://homepage.univie.ac.at/michael.puerrer/php/pics/wave2d.png

The two dimensional wave equation

Ryan C. Daileda

Trinity University

Partial Differential Equations
March 1, 2012

H.J. Oberle *Differentialgleichungen II* SoSe 2011