Modelling of fern and horsetail using GroIMP

Katarína Smoleňová

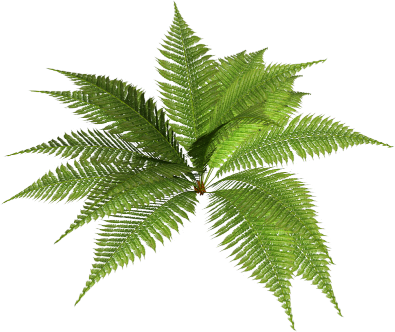
Brandenburg University of Technology at Cottbus
Chair for Practical Computer Science / Graphics Systems

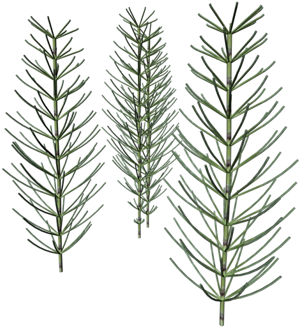
March 11, 2008

Outline

1. Motivation
2. Modelling process
   - Fern (Dryopterix filix-mas)
   - Horsetail (Equisetum arvense)
3. Conclusion
   - Fern (Dryopterix filix-mas)
   - Horsetail (Equisetum arvense)

Brandenburgische Technische Universität Cottbus





Modelling process

1. Data acquisition
2. Creating topological model
3. Texturing
4. Parameter calibration and statistical parameter distribution

Brandenburgische Technische Universität Cottbus

Fern (*Dryopterix filix-mas*)

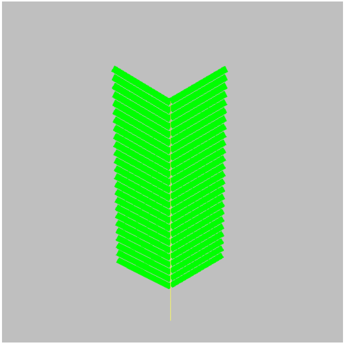
Brandenburgische Technische Universität Cottbus

Data acquisition

- leaves two times compound
  - stem
  - 20 to 25 (-35) small leaves (leaflets) on each side

Brandenburgische Technische Universität Cottbus

Creating topological model



Creating topological model

```
module Meristem(float t);
```

Creating topological model

```
module Meristem(float t);
Axiom ==>
   F(2, 0.05)
   Meristem(0);
```

Creating topological model

```
module Meristem(float t);
Axiom ==>
  F(2, 0.05)
  Meristem(0);
Meristem(t), (t < 1) ==>
  F(0.4, 0.05)
  [ RU(60)
    leaf(4, 0.05) ]
  F(0.1, 0.05)
  [ RU(-60)
    leaf(4, 0.05) ]
  Meristem(t + 0.04);
```

Creating topological model

```
module Meristem(float t);
Axiom ==>
   F(2, 0.05)
   Meristem(0);
Meristem(t), (t < 1) ==>
   F(0.4, 0.05)
   [ RU(60)
     leaf(4, 0.05) ]
   F(0.1, 0.05)
   [ RU(-60)
     leaf(4, 0.05) ]
   Meristem(t + 0.04);
```

Creating topological model

```
module Meristem(float t);
Axiom ==>
   F(2, 0.05)
   Meristem(0);
Meristem(t), (t < 1) ==>
   F(0.4, 0.05)
   [ RU(60)
     leaf(4, 0.05) ]
   F(0.1, 0.05)
   [ RU(-60)
     leaf(4, 0.05) ]
   Meristem(t + 0.04);
```

Creating topological model

```
module Meristem(float t);
Axiom ==>
   F(2, 0.05)
   Meristem(0);
Meristem(t), (t < 1) ==>
   F(0.4, 0.05)
   [ RU(60)
     leaf(4, 0.05) ]
   F(0.1, 0.05)
   [ RU(-60)
     leaf(4, 0.05) ]
   Meristem(t + 0.04);
```

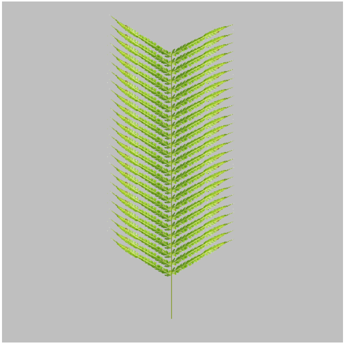
Creating topological model

```
module Meristem(float t);
Axiom ==>
   F(2, 0.05)
   Meristem(0);
Meristem(t), (t < 1) ==>
   F(0.4, 0.05)
   [ RU(60)
     leaf(4, 0.05) ]
   F(0.1, 0.05)
   [ RU(-60)
     leaf(4, 0.05) ]
   Meristem(t + 0.04);
```

Creating topological model

```
module Meristem(float t);
Axiom ==>
   F(2, 0.05)
   Meristem(0);
Meristem(t), (t < 1) ==>
   F(0.4, 0.05)
   [ RU(60)
     leaf(4, 0.05) ]
   F(0.1, 0.05)
   [ RU(-60)
     leaf(4, 0.05) ]
   Meristem(t + 0.04);
```

Textures



Textures

```
const Shader leafmat = shader("leaflet");
const Shader stem = shader("stem");
```

stem RGB(149, 186, 0)

Textures

```
const Shader leafmat = shader("leaflet");
const Shader stem = shader("stem");
Axiom ==>
   F(2, 0.05)
   Meristem(0);
Meristem(t), (t < 1) ==>
   F(0.4, 0.05)
   [ RU(60)
     leaf(4, 0.4)                         ]
   F(0.1, 0.05)
   [ RU(-60)
     leaf(4, 0.4)                         ]
   Meristem(t + 0.04);
```

stem RGB(149, 186, 0)

Textures

```
const Shader leafmat = shader("leaflet");
const Shader stem = shader("stem");
Axiom ==>
   F(2, 0.05)
   Meristem(0);
Meristem(t), (t < 1) ==>
   F(0.4, 0.05)
   [ RU(60)
     leaf(4, 0.4)                         ]
   F(0.1, 0.05)
   [ RU(-60)
     leaf(4, 0.4)                         ]
   Meristem(t + 0.04);
```

stem RGB(149, 186, 0)

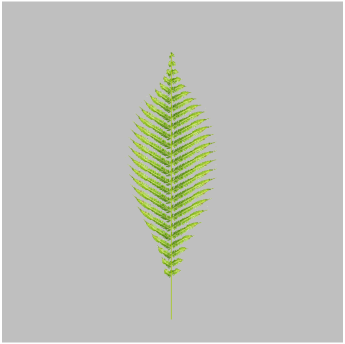
Textures

```
const Shader leafmat = shader("leaflet");
const Shader stem = shader("stem");
Axiom ==>
   F(2, 0.05).(setShader(stem))
   Meristem(0);
Meristem(t), (t < 1) ==>
   F(0.4, 0.05).(setShader(stem))
   [ RU(60)
     leaf(4, 0.4).(setShader(leafmat)) ]
   F(0.1, 0.05).(setShader(stem))
   [ RU(-60)
     leaf(4, 0.4).(setShader(leafmat)) ]
   Meristem(t + 0.04);
```

stem RGB(149, 186, 0)

Brandenburgische Technische Universität Cottbus

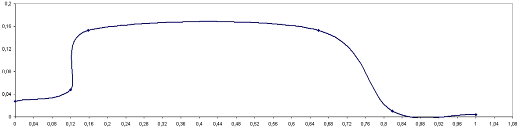
Parameter calibration and stochastical distribution 1



Brandenburgische Technische Universität Cottbus

Parameter calibration and stochastical distribution 1

```
function.func

range: 0.000000 1.000000
points: 6
0.000000 0.027821
0.120000 0.047821
0.158400 0.152621
0.658400 0.152621
0.818800 0.010000
1.000000 0.003821
```



Brandenburgische Technische Universität Cottbus

Parameter calibration and stochastical distribution 1

```
const Function radius = function ("function");
```

Brandenburgische Technische Universität Cottbus

Parameter calibration and stochastical distribution 1

```
const Function radius = function ("function");
Meristem(t), (t < 1) ==>
   F(0.4, 0.05).(setShader(stem))
   [ RU(60)
     leaf(              , 0.4).(setShader(leafmat)) ]
   F(0.1f, 0.05f).(setShader(stem))
   [ RU(-60)
     leaf(              , 0.4).(setShader(leafmat)) ]
   Meristem(t + 0.04);
```

Parameter calibration and stochastical distribution 1

```
const Function radius = function ("function");
Meristem(t), (t < 1) ==>
   F(0.4, 0.05).(setShader(stem))
   [ RU(60)
     leaf(              , 0.4).(setShader(leafmat)) ]
   F(0.1f, 0.05f).(setShader(stem))
   [ RU(-60)
     leaf(              , 0.4).(setShader(leafmat)) ]
   Meristem(t + 0.04);
```

Parameter calibration and stochastical distribution 1

```
const Function radius = function ("function");
Meristem(t), (t < 1) ==>
   F(0.4, 0.05).(setShader(stem))
   [ RU(60)
     leaf(radius[t] * 20, 0.4).(setShader(leafmat)) ]
   F(0.1f, 0.05f).(setShader(stem))
   [ RU(-60)
     leaf(radius[t] * 20, 0.4).(setShader(leafmat)) ]
   Meristem(t + 0.04);
```

Parameter calibration and stochastical distribution 2

Brandenburgische Technische Universität Cottbus

Parameter calibration and stochastical distribution 2

```
Meristem(t), (t < 1) ==>
   F(0.4, 0.05).(setShader(stem))
   [

     leaf(radius[t] * 20, 0.4).(setShader(leafmat)) ]
   F(0.1f, 0.05f).(setShader(stem))
   [

     leaf(radius[t] * 20, 0.4).(setShader(leafmat)) ]


   Meristem(t + 0.04);
```

Parameter calibration and stochastical distribution 2

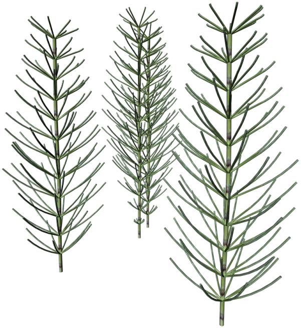
```
Meristem(t), (t < 1) ==>
   F(0.4, 0.05).(setShader(stem))
   [

     leaf(radius[t] * 20, 0.4).(setShader(leafmat)) ]
   F(0.1f, 0.05f).(setShader(stem))
   [

     leaf(radius[t] * 20, 0.4).(setShader(leafmat)) ]


   Meristem(t + 0.04);
```

Brandenburgische Technische Universität Cottbus

Parameter calibration and stochastical distribution 2

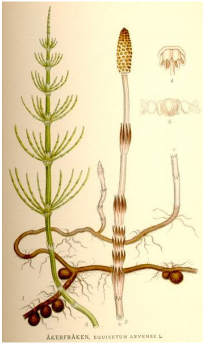
```
Meristem(t), (t < 1) ==>
   F(0.4, 0.05).(setShader(stem))
   [ RU(random(60, 70))
     RH(random(0, 20))
     leaf(radius[t] * 20, 0.4).(setShader(leafmat)) ]
   F(0.1f, 0.05f).(setShader(stem))
   [ RU(random(-60, -70))
     RH(random(-20, 0))
     leaf(radius[t] * 20, 0.4).(setShader(leafmat)) ]
   RL(1)
   RU(-0.5)
   Meristem(t + 0.04);
```

Horsetail (*Equisetum arvense*)

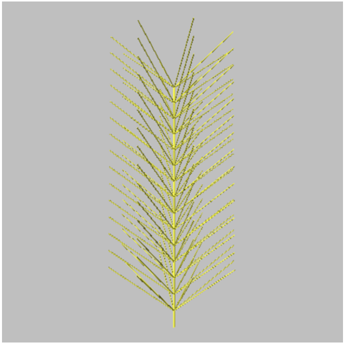


Data acquisition

- two types of a stem
    - fertile non-green one (spring)
    - sterile green one (summer)
- sterile stem
    - main stem (up to 20 segments)
    - branches growing in whorls from nodes



Creating topological model



Creating topological model

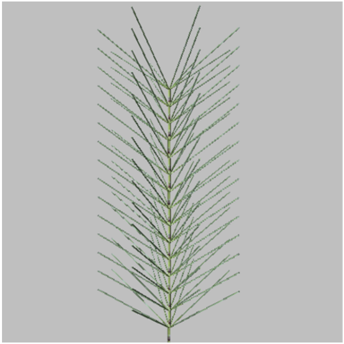
```
module Meristem(float t);
module BranchBud(float length, float width, int counter);

const int numOfBranches = 8;
const float angleOfBranches = 55;

Axiom ==>
   Meristem(0);
```

Brandenburgische Technische Universität Cottbus

Creating topological model

```
{ double angle = 360/numOfBranches; }
Meristem(t), (t < 1) ==>
   F(0.3, 0.05)
      for((0:numOfBranches))(
         RH(angle)
         [ RL(angleOfBranches)
           F(0.3, 0.02)
           BranchBud(0.3, 0.02, 0) ]
      )
   Meristem(t + 0.07);
BranchBud(l, w, c), (c < 4) ==>
   F(l, w)
   BranchBud(l, w, c+1);
```

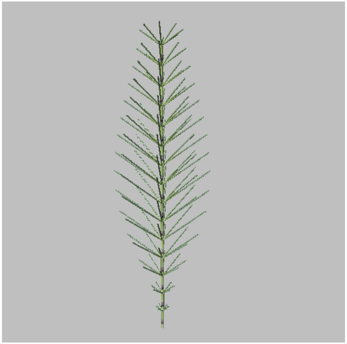
Textures



Textures

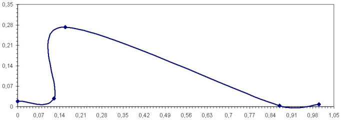
```
const ShaderRef stem = shader("stem");
const ShaderRef branch = shader("branch");
const ShaderRef branch0 = shader("branch0");
Meristem(t), (t < 1) ==>
   F(0.3, 0.05).(setShader(stem))
      for((0:numOfBranches))(
         RH(angle)
         [ RL(angleOfBranches)
           F(0.3, 0.02).(setShader(branch0))
           BranchBud(0.3, 0.02, 0) ] )
   Meristem(t + 0.07);
BranchBud(l, w, c), (c < 4) ==>
   F(l, w).(setShader(branch))
   BranchBud(l, w, c+1);
```

Parameter calibration and stochastical distribution 1



Parameter calibration and stochastical distribution 1

```
function.func

range: 0.000000 1.000000
points: 5
0.000000 0.018221
0.120000 0.027821
0.158400 0.272621
0.868800 0.003821
1.000000 0.008621
```

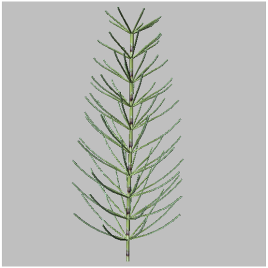


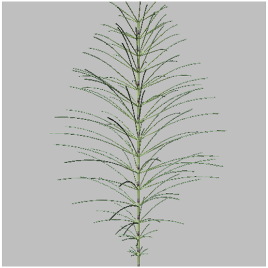
Parameter calibration and stochastical distribution 1

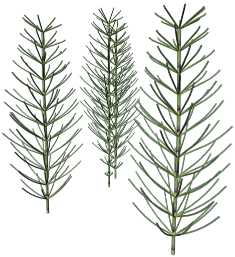
```
const Function radius = function ("radius");

Meristem(t), (t < 1) ==>
   F(0.3, 0.05).(setShader(stem))
      for((0:numOfBranches))(
         RH(angle)
         [ RL(angleOfBranches)
           F(radius[t*0.5] * 3, 0.02).(setShader(branch0))
           BranchBud(radius[t], 0.02, 0) ]
      )
   Meristem(t + 0.07);
```

Brandenburgische Technische Universität Cottbus

Parameter calibration and stochastical distribution 2



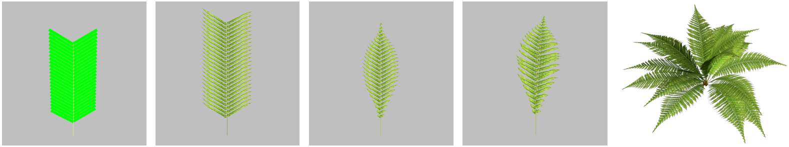




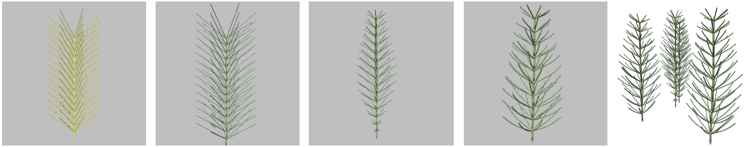
Parameter calibration and stochastical distribution 2

```
Meristem(t), (t < 1) ==>
   F(0.3, 0.05).(setShader(stem))
      for((0:numOfBranches))(
         RH(random(angle - 10, angle + 10))
         [ RL(angleOfBranches, angleOfBranches + 10)
           F(radius[t*0.5] * 3, 0.02).(setShader(branch0))
           BranchBud(radius[t], 0.02, 0) ]
      )
   Meristem(t + 0.07);
```

Modelling process of fern



Modelling process of horsetail



Thank you for your attention.

Brandenburgische Technische Universität Cottbus