How to model a daisy in 1/2 hour

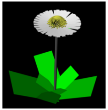
Katarína Streit (Smoleňová), Michael Henke

ksmolen@gwdg.de
Departement Ecoinformatics, Biometry and Forest Growth
Georg-August-University Göttingen

September 16, 2013 / Prague

“Modelling of Ecosystems by Tools from Computer Science”

```
Axiom ==>
 // create rosette of 7 leaves
 for (int i:(1:7)) (
   [
     RH(i * 137.5)
     M((i - 1) / 2)
     RU(LEAF_ANGLE)
     RH(90)
     { double r = 50 - i * 5; }
     Leaf(r, r * 0.5)
   ]
 ) ...
;
```



Common daisy (*Bellis perennis*)

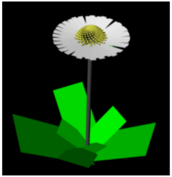
Modelling of real plants

How?

Where to start?

Outline

Data acquisition



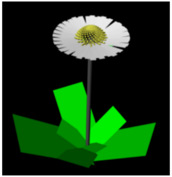
Creating topology

Texturing

Parameter calibration

Outline

Data acquisition



Creating topology

Texturing

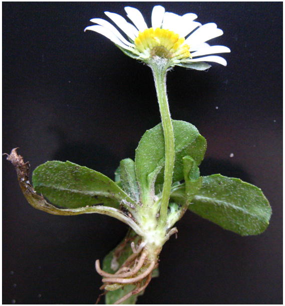
Parameter calibration

pure structural (static) model → no growth & no functioning included

Where to find data?

- Books, journals
- Internet
- Nature
- . . .

Collected data about daisy



- Small rounded or spoon-shaped evergreen leaves, 2-5 cm long, close to the ground, rosulate arrangement
- Leafless stem, 2-10 cm long
- Green bracts in two rows, usually 13
- Flower base, conical shape, 6 mm long, 5 mm in diameter
- White flowers, 11 mm long, 2 mm wide
- Yellow disc flowers
- Short rhizomes

Specify types of organs (modules)

Only above-ground organs considered here:

```
module Stem;

module Leaf;

module Bract;

module FlowerBase;

module Flower;
```

Add parameters to modules

```
module Stem(float length, float diameter);

module Leaf(float length, float width);

module Bract(float length, float width);

module FlowerBase(float length, float diameter);

module Flower(float length, float diameter, int color);
```

How to assign shape to modules

2 possibilities:

- Derivation from an existing geometry object:

```
module Stem extends Cylinder(1, 0.1);

module Stem(super.length, super.radius)
    extends Cylinder(length, radius);

module Stem(super.length, float diameter)
    extends Cylinder(length, diameter/2);
```

- Using instantiation rules (more in the talk by W. Kurth!):

```
module Stem(float length, float diameter)
==> Cylinder(length, diameter/2);
```

Assign shape and colour to modules

```
module Stem(super.length, float diameter)
    extends Cylinder(length, diameter/2);

module Leaf(float length, float width)
==> leaf(length, width);  // leaf returns a green parallelogramm

module Bract(float length, float width)
==> leaf(length, width);

module FlowerBase(super.length, float diameter)
    extends Cone(length, diameter/2);

module Flower(float length, float diameter, int color)
==> if (color == YELLOW_COLOR) (
        P(YELLOW_COLOR) Cylinder(length, diameter/2)
    ) else if (color == WHITE_COLOR) (
        P(WHITE_COLOR) Parallelogram(length, diameter)
    );

const int YELLOW_COLOR = 14;
const int WHITE_COLOR = 15;
```

Create topology

Model parameters (values can be adjusted later on):

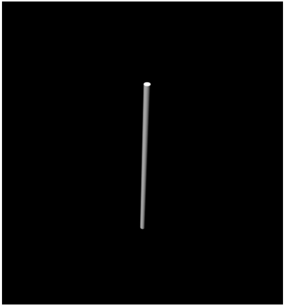
```
const float LEAF_ANGLE = 50;
const float BRACT_ANGLE = 75;
const float YELLOW_FLOWER_ANGLE = 80;
const float WHITE_FLOWER_ANGLE = 80;
```

Plant will be derived from an initial symbol, the Axiom, inside the init method (connectivity information is important in this step):

```
protected void init()
[
    Axiom ==>
        // daisy structure
        ...
    ;
]
```

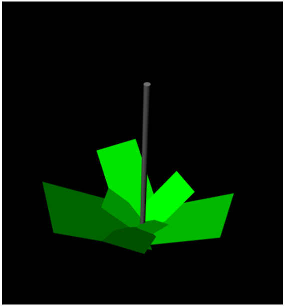
Create stem

```
// create the stem,
// 70 units long, diameter 2 units

Stem(70, 2)
```



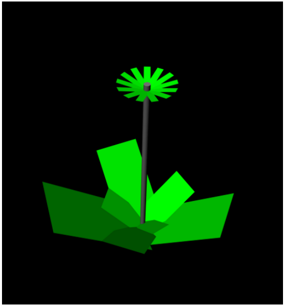
Create leaves

```
// create rosette of 7 leaves,
// diameter half of the length

for (int i:1:7) (
    [
        RH(i*137.5)
        M(-70 + (i-1)/2)
        RU(LEAF_ANGLE) RH(90)
        // alternative: RL(LEAF_ANGLE)
        { double r = 50 - i*5; }
        Leaf(r, r*0.5)
    ]
)
```



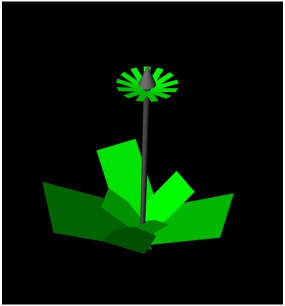
Create bracts

```
// create 13 bracts,
// each 9 units long, 2 units in width

for (int i:1:13) (
    [
        M(-2)
        RH(i*137.5)
        RU(BRACT_ANGLE) RH(90)
        Bract(9, 2)
    ]
)
```



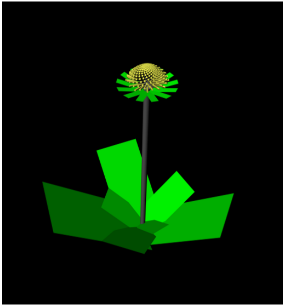
Create flower base

```
// create flower base,
// length 6 units, diameter 5 units

FlowerBase(6, 5)
```



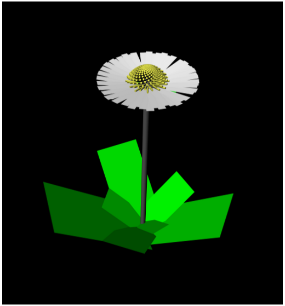
Create disc florets

```
for (int i:1:300) (

    {
        float h = i * 0.02;
        float s = 0.2 * Math.sqrt(i);
    }

    // create yellow flowers
    // around flower base

    if (i <= 250) (
        [
            M(-h)
            RH(i*137.5)
            Translate(s, 0, 0)
            RU(YELLOW_FLOWER_ANGLE * i/250)
            Flower(0.1 + 3.0*(h/5.0),
                0.5, YELLOW_COLOR)
        ]
    )
```



Create ray florets

```
        // create white flowers
        // around flower base
        // and below yellow flowers

        else (
            [
                M(-h)
                RH(i*137.5)
                Translate(s, 0, 0)
                RU(WHITE_FLOWER_ANGLE) RH(90)
                Flower(11, 2, WHITE_FLOWER)
            ]
        )
    )
```



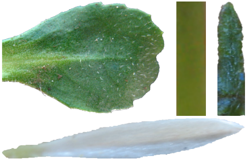
Using textures to improve the visual plausibility

- Sources: digital camera, scanner, . . .
- Preprocessing of textures:
  - Adjust lighting
  - Cut out, make transparent background
  - Resize (prevent too big textures)

- Prepared daisy textures:

Using textures to improve the visual plausibility

- Sources: digital camera, scanner, . . .
- Preprocessing of textures:
  - Adjust lighting
  - Cut out, make transparent background
  - Resize (prevent too big textures)
- Prepared daisy textures:

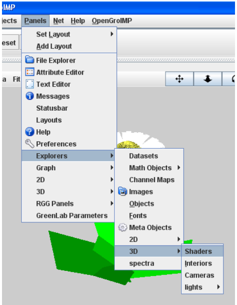


How to import textures into GroIMP

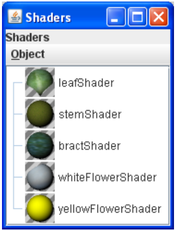
Menu: Panels → Explorers → 3D → Shaders

Shaders: Object → New → Lambert (2 clicks or F2 to rename)

Attribute Editor: Diffuse colour → Surface Maps → Image; Image → From File

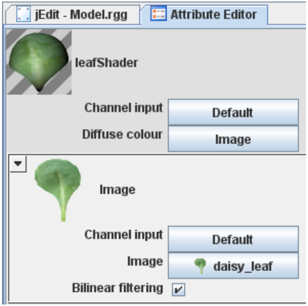


→



leafShader
stemShader
bractShader
whiteFlowerShader
yellowFlowerShader

→



leafShader

Channel input: Default
Diffuse colour: Image

Image

Channel input: Default
Image: daisy_leaf
Bilinear filtering ☑

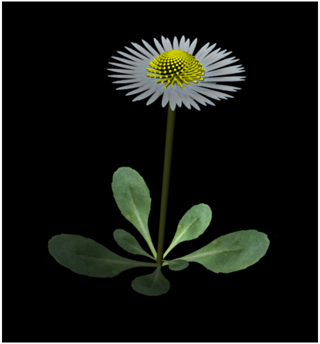
How to apply texture to module

```
// obtain reference to named shader

const ShaderRef leafShader = shader("leafShader");


// set shader during interpretation

module Leaf(float length, float width)
==> leaf(length, width).(setShader(leafShader));
```

Apply textures to daisy modules

```
module Stem(super.length, float diameter)
    extends Cylinder(length, diameter/2).(setShader(stemShader));

module Leaf(float length, float width)
==> leaf(length, width).(setShader(leafShader));

module Bract(float length, float width)
==> leaf(length, width).(setShader(bractShader));

module Flower(float length, float diameter, int color)
==> if (color == YELLOW_FLOWER) (
        Cylinder(length, diameter/2).(setShader(yellowFlowerShader))
    ) else if (color == WHITE_FLOWER) (
        Parallelogram(length, diameter).(setShader(whiteFlowerShader))
    );

const ShaderRef leafShader = shader("leafShader");
const ShaderRef stemShader = shader(“stemShader");
const ShaderRef bractShader = shader("bractShader");
const ShaderRef whiteFlowerShader = shader("whiteFlowerShader");
const ShaderRef yellowFlowerShader = shader("yellowFlowerShader");
```

Result of texturing

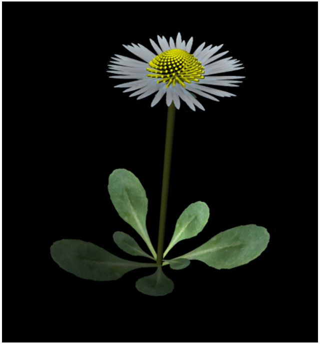


Adjust parameters and variability of the model

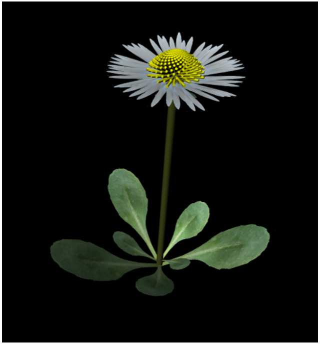
- Generally a complex and time-consuming process!
- Statistical analysis of the model followed by parameter adjustment until the model fits the observed data
- Statistical analysis of real plants to obtain mean and variance for stochastic generation of parameter values
- More realistic look of the plant
  by generating values (angle, length, diameter, ...) randomly

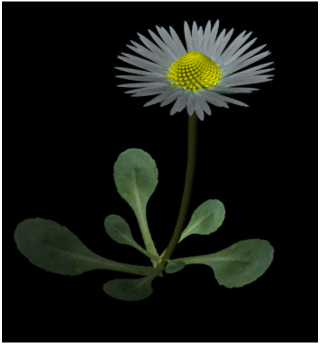
```
RU(random(WHITE_FLOWER_ANGLE - 5, WHITE_FLOWER_ANGLE + 5))
```

Result with stochastic distribution



Result with stochastic distribution





thank you