How to model a daisy in 1/2 hour

Reinhard Hemmerling, Katarína Smoleňová

University of Göttingen, Göttingen, Germany

September 27th 2010

Overview

Where to find data ?

- Collect images
- Read books
- Go into the nature !

Daisy data

- 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

Define the parts

```
module Leaf;
module Stem;
module Bract;
module FlowerBase;
module Flower;
```

Define parameters

```
module Leaf(float length, float diameter);
module Stem(float length, float diameter);
module Bract(float lenth, float diameter);
module FlowerBase(float length, float diameter);
module Flower(float length, float diameter,
    int color);
```

Assign shape to parts (1)

Different ways to assign a shape to a part:

- Derivation
- Instantiation

Assign shape to parts (2)

```
module Leaf(float length, float diameter)
    ==> leaf(length, diameter);
module Stem(float length, float diameter)
    extends Cylinder(length, diameter/2);
module Bract(float lenth, float diameter)
    ==> leaf(length, diameter);
module FlowerBase(float length, float diameter)
    ==> Cone(length, diameter/2);
```

Assign shape to parts (3)

```
module Flower(float length, float diameter,
    int color)
    ==> if (color == YELLOW)
            (Cylinder(length, diameter/2))
        else if (color == WHITE)
            (leaf(length, diameter));
```

Use parts to derive topology

- Derive flower from an initial symbol, in this case `Axiom`
- Parameter values will be guessed, can be adjusted later on
- Connectivity information is important in this step

Derive leaves

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

Derive stem

```
...

// create the stem, 70 mm long,
// diameter 2 mm
Stem(70, 2)

...
```

Derive bracts

```
...

// create 13 bracts,
// each 9 mm long, 2 mm in width
for (int i:1:13)
( [
    M(-1)
    RH(360 * i / 13)
    RU(bractAngle)
    RH(90)
    Bract(9, 2)
] )

...
```

Derive flower base

```
...

// create flower base,
// 6 mm long, 5 mm diameter
FlowerBase(6, 5)

...
```

Derive white flowers

```
...

// create white flowers around flower base
for (int i:1:50)
( [
    M(-6 + i * 0.02)
    RH(i * 13.7)
    RU(whiteFlowerAngle)
    RH(90)
    Flower(11, 2, WHITE)
] )

...
```

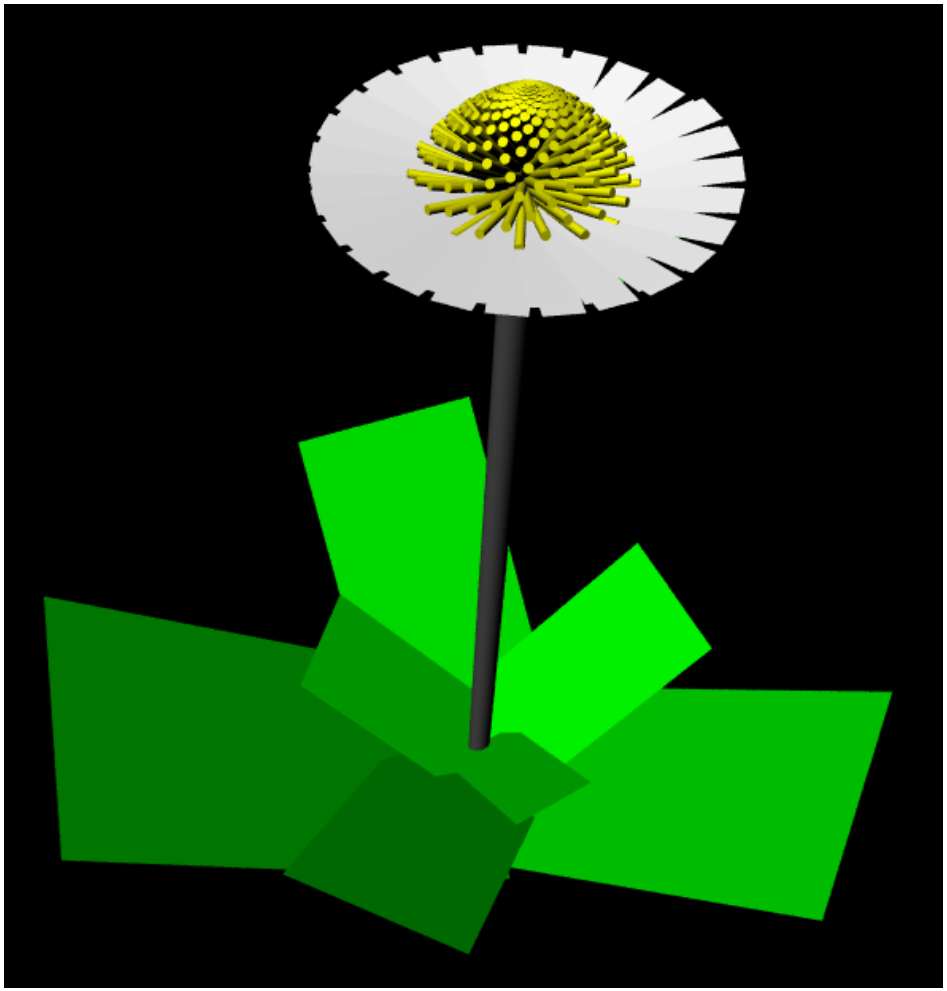
Derive yellow flowers

```
...

// create yellow flowers around flower base
for (int i:1:250)
( [
    { float h = i * 0.02; }
    M(-h)
    RH(i * 137.5)
    Translate(h * k, 0, 0)
    RU(whiteFlowerAngle * i / 250)
    Flower(1.0 + 3.0 * (h / 5.0), 0.5, YELLOW)
] )

...
```

Result of topology creation



Obtain textures

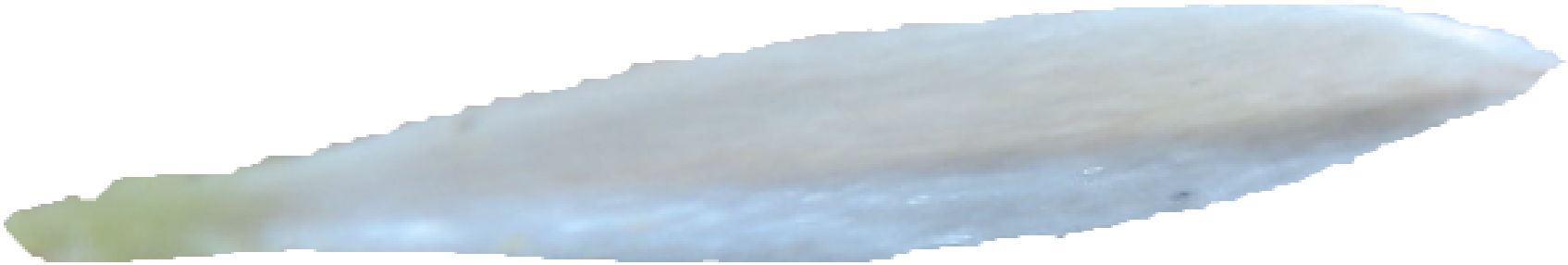
- From (digital) camera
- From scanner
- From the internet
- Any kind of image will do

Prepare textures

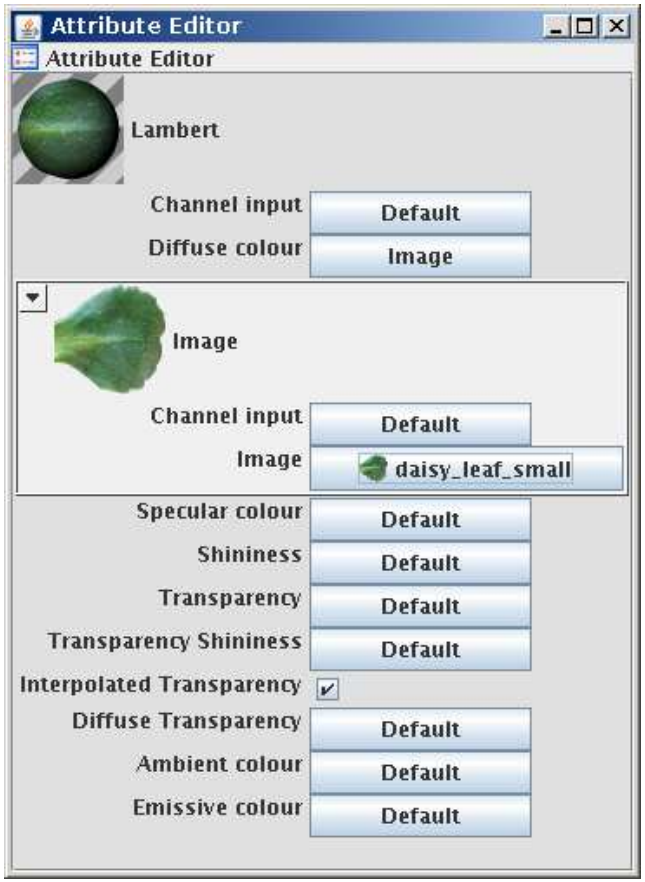
- Adjust lighting
- Cut out, make background transparent
- Resize (no 100MB textures)

Prepared daisy textures



Importing textures into GroIMP



Apply texture to object

```
// obtain reference to named shader
ShaderRef leafShader = shader("leafShader");

// set shader during interpretation
module Leaf(float length, float diameter)
    ==> leaf(length, diameter).(
        setShader(leafShader));
```

Result of texturing

Adjust parameters of the model

- Make plant look more natural by generating values (angle, length, diameter, ...) randomly
- Perform statistical analysis of the model followed by parameter adjustment until the model fits the observed data
- Perform statistical analysis of real plants to obtain mean and variance for stochastic generation of parameter values

Result with stochastic distribution

That's all

Thank you for your attention.