Modelling morphological features of trees in XL

Katarína Smoleňová

Georg-August University of Göttingen
Chair for Computer Graphics and Ecological Informatics

September 28, 2010

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Outline

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

What will you see in the tutorial?

- Terms and pictures:
  - D. Barthélémy and Y. Caraglio. **Plant Architecture: A Dynamic, Multilevel and Comprehensive Approach to Plant Form, Structure and Ontogeny**. *Annals of Botany*, 99(3): 375–407, 2007.
- Examples:
  - of morphological characteristics (summarized in the paper above) created with **XL**[a] in **GroIMP**[b]
- E.g. leafy axis

[a] eXtended L-system language
[b] Growth-grammar related Interactive Modelling Platform

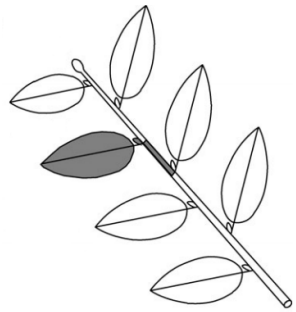
GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

What will you see in the tutorial?

- Terms and pictures:
  - D. Barthélémy and Y. Caraglio. **Plant Architecture: A Dynamic, Multilevel and Comprehensive Approach to Plant Form, Structure and Ontogeny**. *Annals of Botany*, 99(3): 375–407, 2007.
- Examples:

of morphological characteristics (summarized in the paper above) created with **XL**[a] in **GroIMP**[b]

- E.g. leafy axis



[a] eXtended L-system language
[b] Growth-grammar related Interactive Modelling Platform

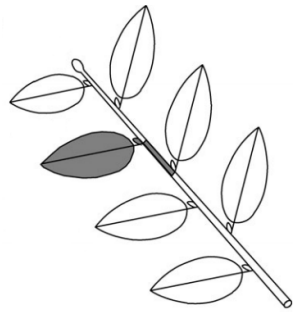
GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

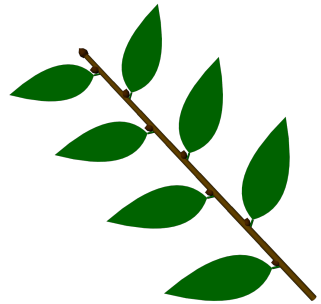
What will you see in the tutorial?

- Terms and pictures:
  - D. Barthélémy and Y. Caraglio. **Plant Architecture: A Dynamic, Multilevel and Comprehensive Approach to Plant Form, Structure and Ontogeny**. *Annals of Botany*, 99(3): 375–407, 2007.
- Examples:

  of morphological characteristics (summarized in the paper above) created with **XL**[a] in **GroIMP**[b]

- E.g. leafy axis





[a] e**X**tended **L**-system language

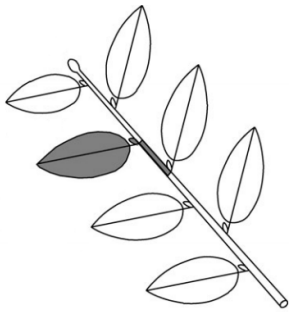
[b] **Gro**wth-grammar related **I**nteractive **M**odelling **P**latform

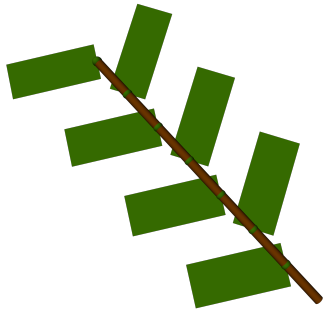
GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

What will you see in the tutorial?

- Terms and pictures:
  - D. Barthélémy and Y. Caraglio. **Plant Architecture: A Dynamic, Multilevel and Comprehensive Approach to Plant Form, Structure and Ontogeny**. *Annals of Botany*, 99(3): 375–407, 2007.
- Examples:
  - **Simplified examples** of morphological characteristics (summarized in the paper above) created with **XL**[a] in **GroIMP**[b]

- E.g. leafy axis



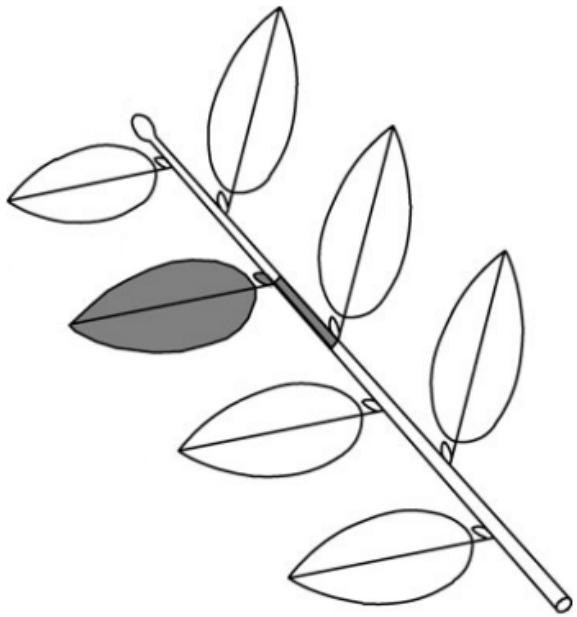


[a] e**X**tended **L**-system language

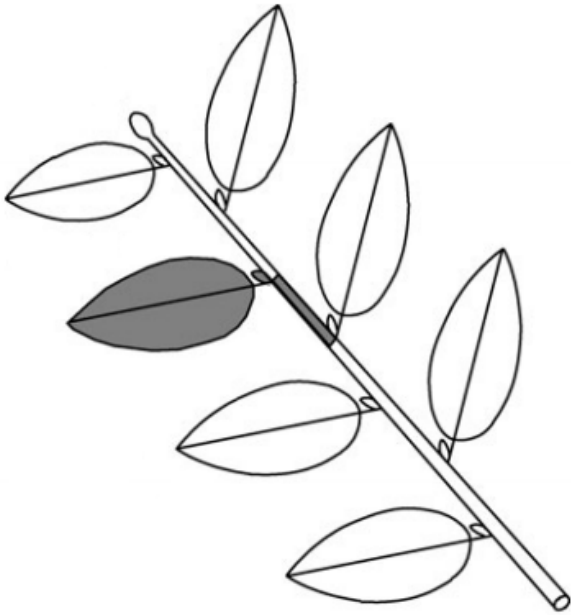
[b] **Gro**wth-grammar related **I**nteractive **M**odelling **P**latform

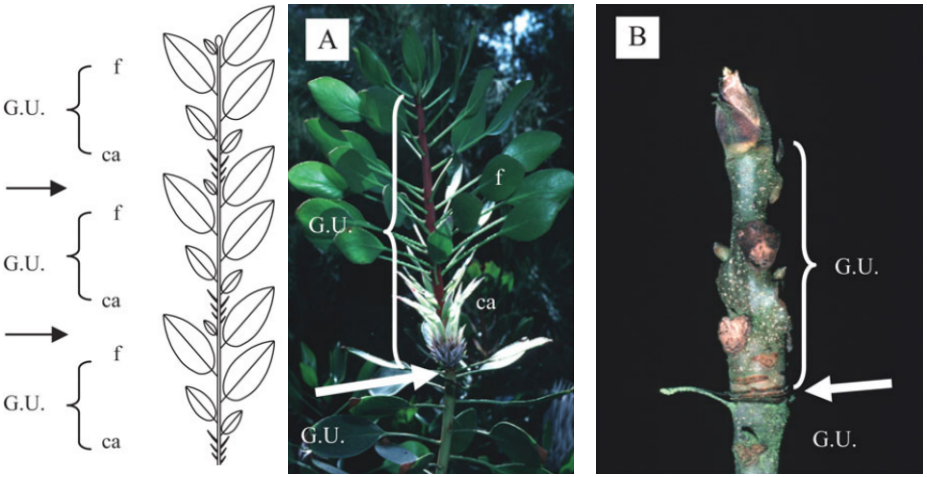
GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Shoot / growth unit / annual shoot

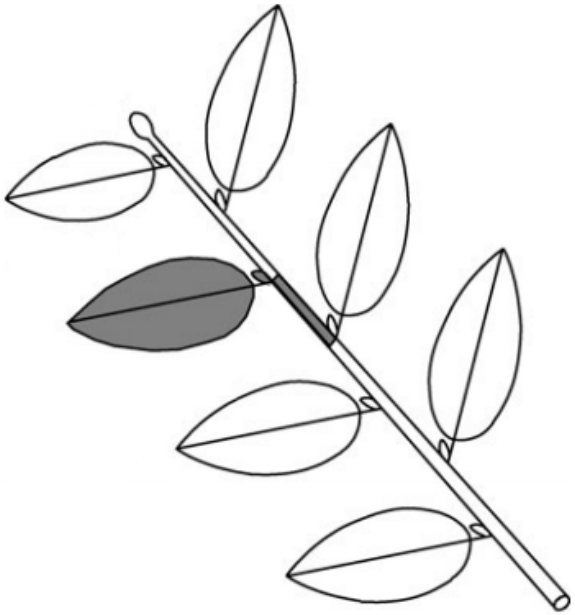


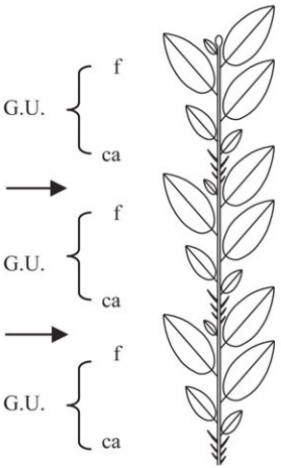
Shoot / growth unit / annual shoot

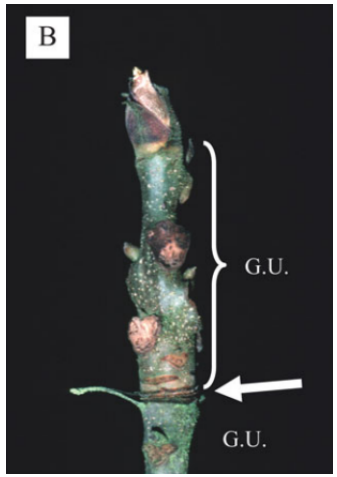


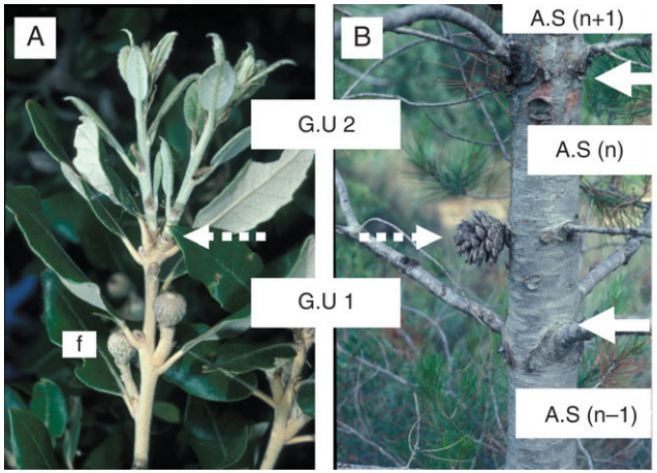


Shoot / growth unit / annual shoot









GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Shoot / growth unit / annual shoot (with XL) 1

```
module ApicalBud;

module LateralBud;

module Internode;

module Leaf;
```

red = newly added code

Shoot / growth unit / annual shoot (with XL) 1

```
module ApicalBud extends Sphere(0.1);

module LateralBud extends Sphere(0.08);

module Internode extends Cylinder(1, 0.05);

module Leaf extends Parallelogram(1, 0.5);
```

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Shoot / growth unit / annual shoot (with XL) 1

```
module ApicalBud extends Sphere(0.1)
{
   { setColor(0x336600); setTransform(0, 0, 0.1); }
}
module LateralBud extends Sphere(0.08)
{
   { setColor(0x336600); setTransform(0, 0, 0.1); }
}
module Internode extends Cylinder(1, 0.05)
{
   { setColor(0x663300); }
}
module Leaf extends Parallelogram(1, 0.5)
{
   { setColor(0x336600); }
}
```

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Shoot / growth unit / annual shoot (with XL) 1

```
module ApicalBud extends Sphere(0.1)
{
   { setColor(0x336600); setTransform(0, 0, 0.1); }
}
module LateralBud extends Sphere(0.08)
{
   { setColor(0x336600); setTransform(0, 0, 0.1); }
}
module Internode extends Cylinder(1, 0.05)
{
   { setColor(0x663300); }
}
module Leaf extends Parallelogram(1, 0.5)
{
   { setColor(0x336600); }
}
protected void init()
[
   Axiom ==> ApicalBud;
]
```

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Shoot / growth unit / annual shoot (with XL) 2

```
public void grow()
[
   ab:ApicalBud ==>


   ;
]
```

Shoot / growth unit / annual shoot (with XL) 2

```
public void grow()
[
   ab:ApicalBud ==>
      Internode
      [ RL(40) LateralBud ]
      [ RL(60) Leaf ]
   ;
]
```

Shoot / growth unit / annual shoot (with XL) 2

```
public void grow()
[
   ab:ApicalBud ==>
      Internode
      [ RL(40) LateralBud ]
      [ RL(60) Leaf ]
   ;
]
```

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Shoot / growth unit / annual shoot (with XL) 2

```
const float BRANCH_ANGLE = 40;
const float LEAF_ANGLE = 60;

public void grow()
[
   ab:ApicalBud ==>
      Internode
      [ RL(40) LateralBud ]
      [ RL(60) Leaf ]
   ;
]
```

Shoot / growth unit / annual shoot (with XL) 2

```
const float BRANCH_ANGLE = 40;
const float LEAF_ANGLE = 60;

public void grow()
[
   ab:ApicalBud ==>
      Internode
      [ RL(BRANCH_ANGLE) LateralBud ]
      [ RL(LEAF_ANGLE) Leaf ]
   ;
]
```

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Indeterminate growth

Indeterminate growth (with XL)

```
const float BRANCH_ANGLE = 40;
const float LEAF_ANGLE = 60;


public void grow()
[  ab:ApicalBud ==>
      Internode
      [ RL(BRANCH_ANGLE) LateralBud ]
      [ RL(LEAF_ANGLE) Leaf ]


   ;
```

```
]
```

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

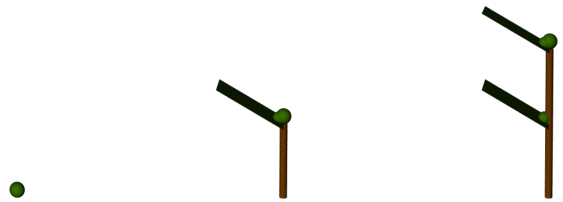
Indeterminate growth (with XL)

```
const float BRANCH_ANGLE = 40;
const float LEAF_ANGLE = 60;


public void grow()
[  ab:ApicalBud ==>
      Internode
      [ RL(BRANCH_ANGLE) LateralBud ]
      [ RL(LEAF_ANGLE) Leaf ]

      ab
   ;
```

```
]
```

Indeterminate growth (with XL)

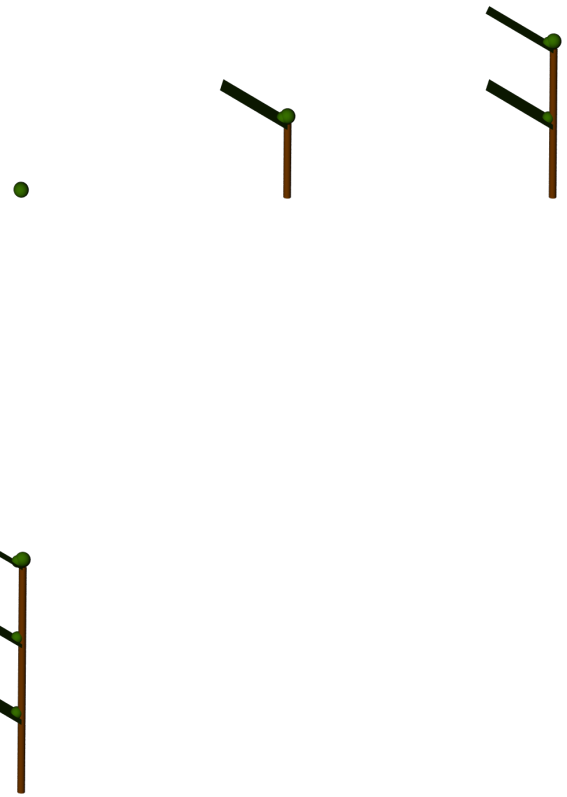
```
const float BRANCH_ANGLE = 40;
const float LEAF_ANGLE = 60;


public void grow()
[  ab:ApicalBud ==>
      Internode
      [ RL(BRANCH_ANGLE) LateralBud ]
      [ RL(LEAF_ANGLE) Leaf ]

      ab
   ;
```



```
]
```

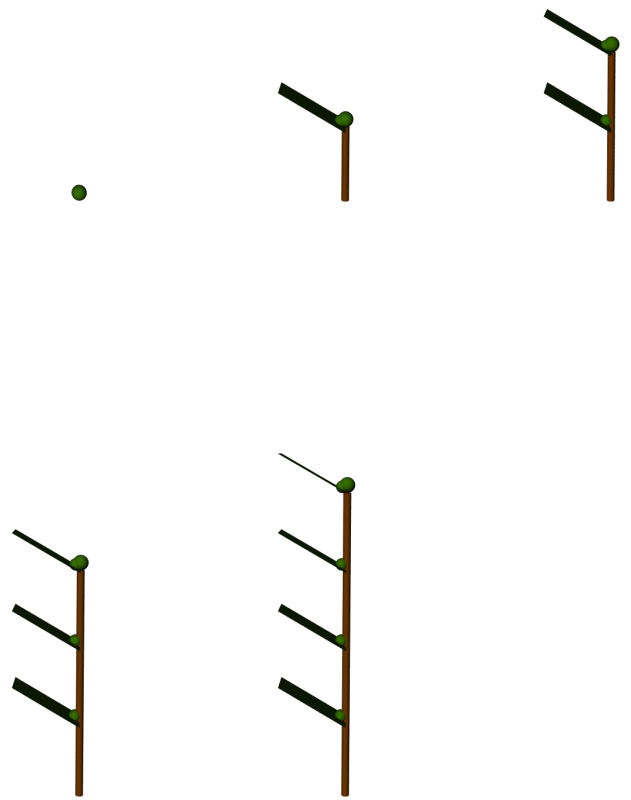
Indeterminate growth (with XL)

```
const float BRANCH_ANGLE = 40;
const float LEAF_ANGLE = 60;


public void grow()
[  ab:ApicalBud ==>
      Internode
      [ RL(BRANCH_ANGLE) LateralBud ]
      [ RL(LEAF_ANGLE) Leaf ]

      ab
   ;




]
```



Indeterminate growth (with XL)

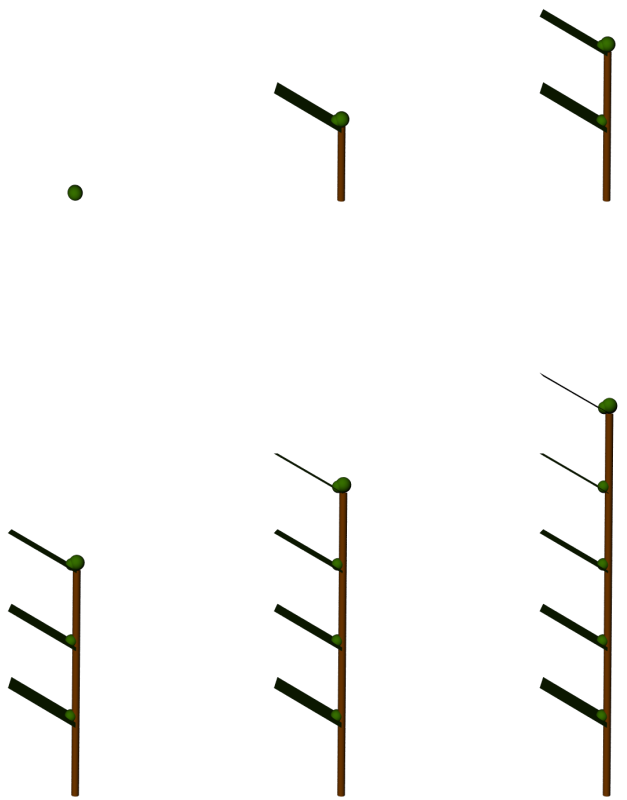
```
const float BRANCH_ANGLE = 40;
const float LEAF_ANGLE = 60;


public void grow()
[  ab:ApicalBud ==>
      Internode
      [ RL(BRANCH_ANGLE) LateralBud ]
      [ RL(LEAF_ANGLE) Leaf ]

      ab
   ;




]
```



GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Indeterminate growth (with XL)

```
const float BRANCH_ANGLE = 40;
const float LEAF_ANGLE = 60;


public void grow()
[  ab:ApicalBud ==>
      Internode
      [ RL(BRANCH_ANGLE) LateralBud ]
      [ RL(LEAF_ANGLE) Leaf ]

      ab
   ;




]
```



GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Indeterminate growth (with XL)

```
const float BRANCH_ANGLE = 40;
const float LEAF_ANGLE = 60;
const float PHYLLOTAXIS = 180;

public void grow()
[  ab:ApicalBud ==>
      Internode
      [ RL(BRANCH_ANGLE) LateralBud ]
      [ RL(LEAF_ANGLE) Leaf ]

      ab
   ;
]
```

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Indeterminate growth (with XL)

```
const float BRANCH_ANGLE = 40;
const float LEAF_ANGLE = 60;
const float PHYLLOTAXIS = 180;

public void grow()
[  ab:ApicalBud ==>
      Internode
      [ RL(BRANCH_ANGLE) LateralBud ]
      [ RL(LEAF_ANGLE) Leaf ]
      RH(PHYLLOTAXIS)
      ab
   ;
]
```

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Indeterminate growth (with XL)

```
const float BRANCH_ANGLE = 40;
const float LEAF_ANGLE = 60;
const float PHYLLOTAXIS = 180;

public void grow()
[  ab:ApicalBud ==>
      Internode
      [ RL(BRANCH_ANGLE) LateralBud ]
      [ RL(LEAF_ANGLE) Leaf ]
      RH(PHYLLOTAXIS)
      ab
   ;
   itn:Internode ::> {
      itn[length] :+= 0.2;
      itn[radius] :+= 0.01;
   }



]
```

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Indeterminate growth (with XL)

```
const float BRANCH_ANGLE = 40;
const float LEAF_ANGLE = 60;
const float PHYLLOTAXIS = 180;

public void grow()
[  ab:ApicalBud ==>
      Internode
      [ RL(BRANCH_ANGLE) LateralBud ]
      [ RL(LEAF_ANGLE) Leaf ]
      RH(PHYLLOTAXIS)
      ab
   ;
   itn:Internode ::> {
      itn[length] :+= 0.2;
      itn[radius] :+= 0.01;
   }
   lf:Leaf ::> {
      lf[length] :+= 0.2;
      lf[axis][x] :+= 0.2;
   }
]
```

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

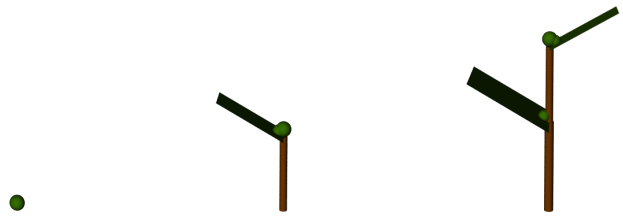
Indeterminate growth (with XL)

```
const float BRANCH_ANGLE = 40;
const float LEAF_ANGLE = 60;
const float PHYLLOTAXIS = 180;

public void grow()
[  ab:ApicalBud ==>
      Internode
      [ RL(BRANCH_ANGLE) LateralBud ]
      [ RL(LEAF_ANGLE) Leaf ]
      RH(PHYLLOTAXIS)
      ab
   ;
   itn:Internode ::> {
      itn[length] :+= 0.2;
      itn[radius] :+= 0.01;
   }
   lf:Leaf ::> {
      lf[length] :+= 0.2;
      lf[axis][x] :+= 0.2;
   }
]
```

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Indeterminate growth (with XL)

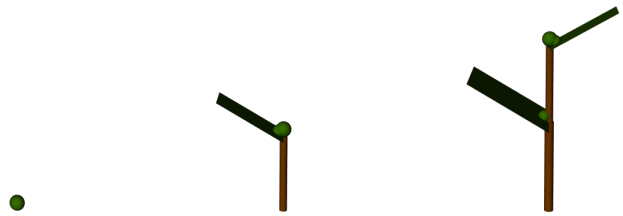
```
const float BRANCH_ANGLE = 40;
const float LEAF_ANGLE = 60;
const float PHYLLOTAXIS = 180;

public void grow()
[  ab:ApicalBud ==>
      Internode
      [ RL(BRANCH_ANGLE) LateralBud ]
      [ RL(LEAF_ANGLE) Leaf ]
      RH(PHYLLOTAXIS)
      ab
   ;
   itn:Internode ::> {
      itn[length] :+= 0.2;
      itn[radius] :+= 0.01;
   }
   lf:Leaf ::> {
      lf[length] :+= 0.2;
      lf[axis][x] :+= 0.2;
   }
]
```

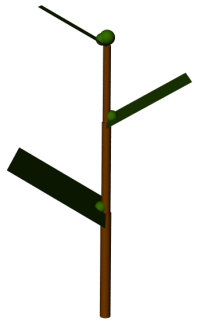


GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Indeterminate growth (with XL)

```
const float BRANCH_ANGLE = 40;
const float LEAF_ANGLE = 60;
const float PHYLLOTAXIS = 180;

public void grow()
[  ab:ApicalBud ==>
      Internode
      [ RL(BRANCH_ANGLE) LateralBud ]
      [ RL(LEAF_ANGLE) Leaf ]
      RH(PHYLLOTAXIS)
      ab
   ;
   itn:Internode ::> {
      itn[length] :+= 0.2;
      itn[radius] :+= 0.01;
   }
   lf:Leaf ::> {
      lf[length] :+= 0.2;
      lf[axis][x] :+= 0.2;
   }
]
```

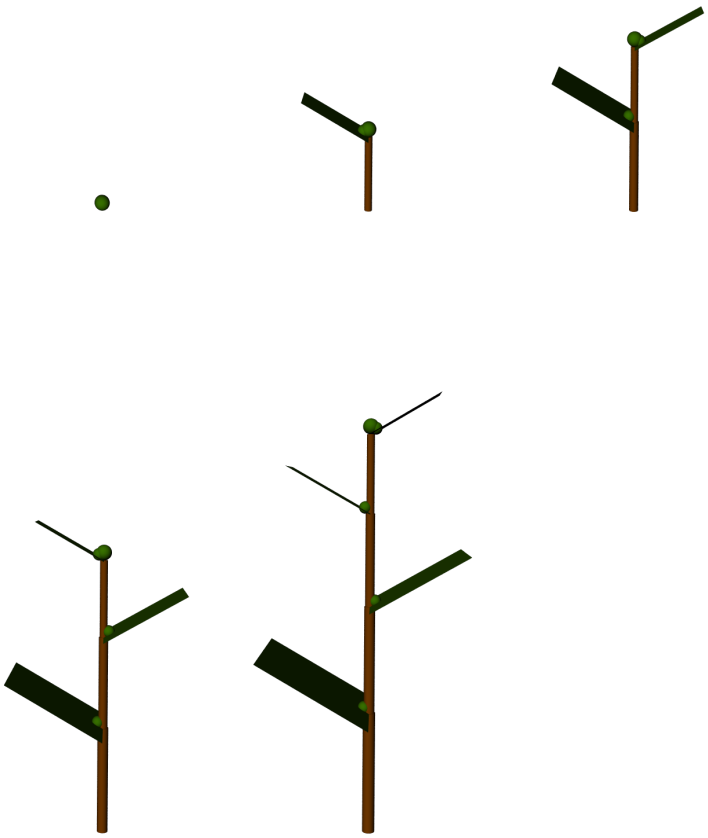




GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Indeterminate growth (with XL)

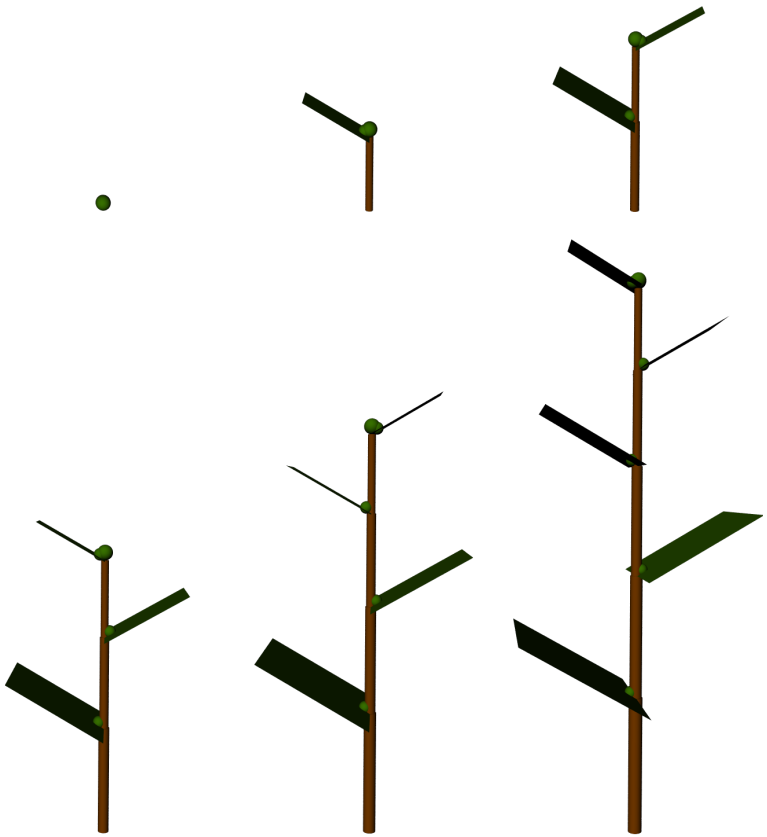
```
const float BRANCH_ANGLE = 40;
const float LEAF_ANGLE = 60;
const float PHYLLOTAXIS = 180;

public void grow()
[  ab:ApicalBud ==>
      Internode
      [ RL(BRANCH_ANGLE) LateralBud ]
      [ RL(LEAF_ANGLE) Leaf ]
      RH(PHYLLOTAXIS)
      ab
   ;
   itn:Internode ::> {
      itn[length] :+= 0.2;
      itn[radius] :+= 0.01;
   }
   lf:Leaf ::> {
      lf[length] :+= 0.2;
      lf[axis][x] :+= 0.2;
   }
]
```



GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

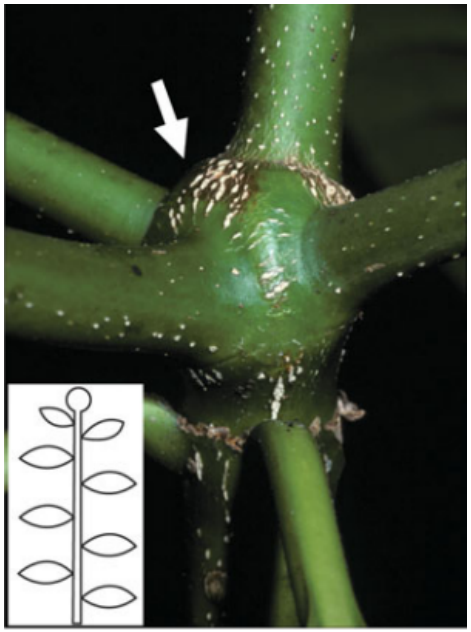
Indeterminate growth (with XL)

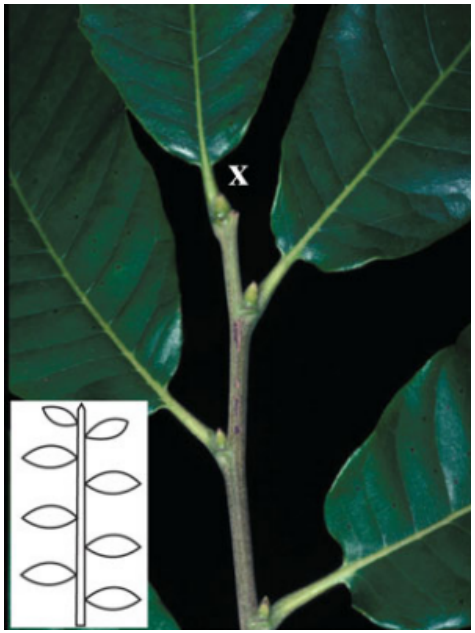
```
const float BRANCH_ANGLE = 40;
const float LEAF_ANGLE = 60;
const float PHYLLOTAXIS = 180;

public void grow()
[  ab:ApicalBud ==>
      Internode
      [ RL(BRANCH_ANGLE) LateralBud ]
      [ RL(LEAF_ANGLE) Leaf ]
      RH(PHYLLOTAXIS)
      ab
   ;
   itn:Internode ::> {
      itn[length] :+= 0.2;
      itn[radius] :+= 0.01;
   }
   lf:Leaf ::> {
      lf[length] :+= 0.2;
      lf[axis][x] :+= 0.2;
   }
]
```



GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Determinate growth





GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Determinate growth (with XL)

```
protected void init()
[
   Axiom ==> ApicalBud;
]
public void grow()
[  ab:ApicalBud ==>

         Internode
         [ RL(BRANCH_ANGLE) LateralBud ]
         [ RL(LEAF_ANGLE) Leaf ]
         RH(PHYLLOTAXIS)
         ab



   ;
   ...

]
```

Determinate growth (with XL)

```
int time;
protected void init()
[  { time = 0; }
   Axiom ==> ApicalBud;
]
public void grow()
[  ab:ApicalBud ==>
      if (time < 4) (
         Internode
         [ RL(BRANCH_ANGLE) LateralBud ]
         [ RL(LEAF_ANGLE) Leaf ]
         RH(PHYLLOTAXIS)
         ab
      ) else (

      )
   ;
   ...
   { time :+= 1; }
]
```

Determinate growth (with XL)

```
module Flower extends Sphere(0.2) { {setShader(RED);} }
int time;
protected void init()
[  { time = 0; }
   Axiom ==> ApicalBud;
]
public void grow()
[  ab:ApicalBud ==>
      if (time < 4) (
         Internode
         [ RL(BRANCH_ANGLE) LateralBud ]
         [ RL(LEAF_ANGLE) Leaf ]
         RH(PHYLLOTAXIS)
         ab
      ) else (

      )
   ;
   ...
   { time :+= 1; }
]
```


GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Determinate growth (with XL)

```
module Flower extends Sphere(0.2) { {setShader(RED);} }
int time;
protected void init()
[  { time = 0; }
   Axiom ==> ApicalBud;
]
public void grow()
[  ab:ApicalBud ==>
      if (time < 4) (
         Internode
         [ RL(BRANCH_ANGLE) LateralBud ]
         [ RL(LEAF_ANGLE) Leaf ]
         RH(PHYLLOTAXIS)
         ab
      ) else (
         Internode Flower
      )
   ;
   ...
   { time :+= 1; }
]
```


GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

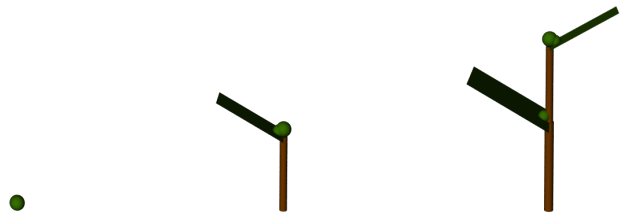
Determinate growth (with XL)

```
module Flower extends Sphere(0.2) { {setShader(RED);} }
int time;
protected void init()
[  { time = 0; }
   Axiom ==> ApicalBud;
]
public void grow()
[  ab:ApicalBud ==>
      if (time < 4) (
         Internode
         [ RL(BRANCH_ANGLE) LateralBud ]
         [ RL(LEAF_ANGLE) Leaf ]
         RH(PHYLLOTAXIS)
         ab
      ) else (
         Internode Flower
      )
   ;
   ...
   { time :+= 1; }
]
```

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

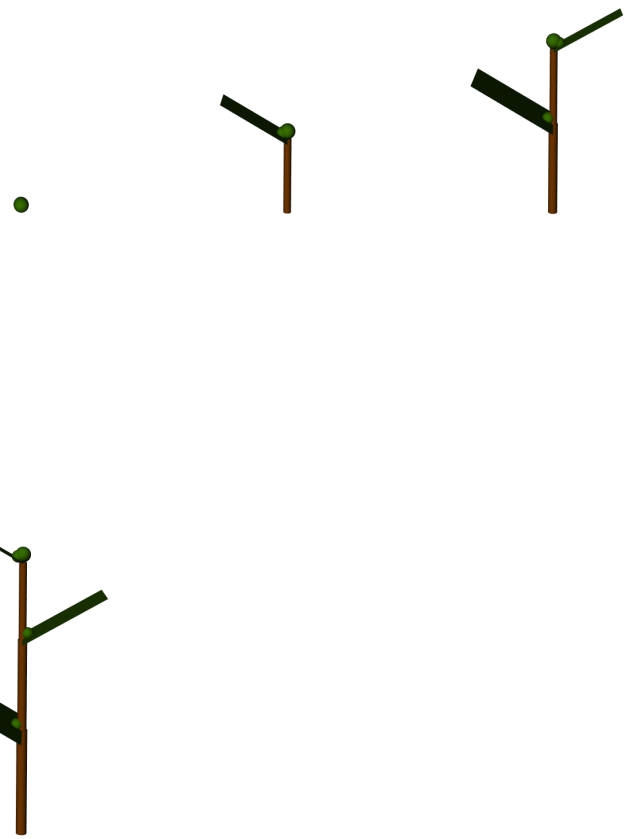
Determinate growth (with XL)

```
module Flower extends Sphere(0.2) { {setShader(RED);} }
int time;
protected void init()
[  { time = 0; }
   Axiom ==> ApicalBud;
]
public void grow()
[  ab:ApicalBud ==>
      if (time < 4) (
         Internode
         [ RL(BRANCH_ANGLE) LateralBud ]
         [ RL(LEAF_ANGLE) Leaf ]
         RH(PHYLLOTAXIS)
         ab
      ) else (
         Internode Flower
      )
   ;
   ...
   { time :+= 1; }
]
```



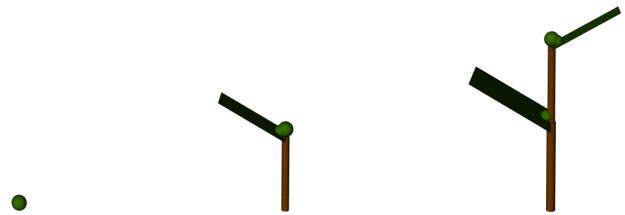
Determinate growth (with XL)

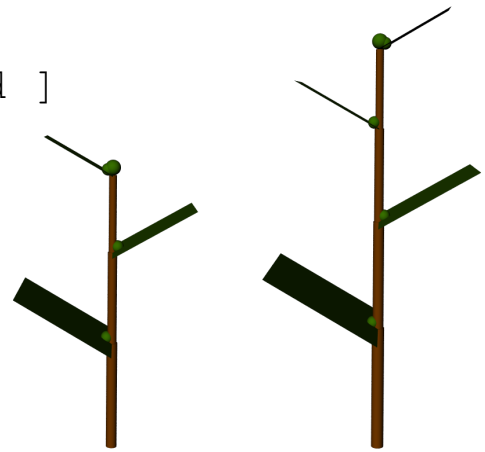
```
module Flower extends Sphere(0.2) { {setShader(RED);} }
int time;
protected void init()
[  { time = 0; }
   Axiom ==> ApicalBud;
]
public void grow()
[  ab:ApicalBud ==>
      if (time < 4) (
         Internode
         [ RL(BRANCH_ANGLE) LateralBud ]
         [ RL(LEAF_ANGLE) Leaf ]
         RH(PHYLLOTAXIS)
         ab
      ) else (
         Internode Flower
      )
   ;
   ...
   { time :+= 1; }
]
```



Determinate growth (with XL)

```
module Flower extends Sphere(0.2) { {setShader(RED);} }
int time;
protected void init()
[  { time = 0; }
   Axiom ==> ApicalBud;
]
public void grow()
[  ab:ApicalBud ==>
      if (time < 4) (
         Internode
         [ RL(BRANCH_ANGLE) LateralBud ]
         [ RL(LEAF_ANGLE) Leaf ]
         RH(PHYLLOTAXIS)
         ab
      ) else (
         Internode Flower
      )
   ;
   ...
   { time :+= 1; }
]
```

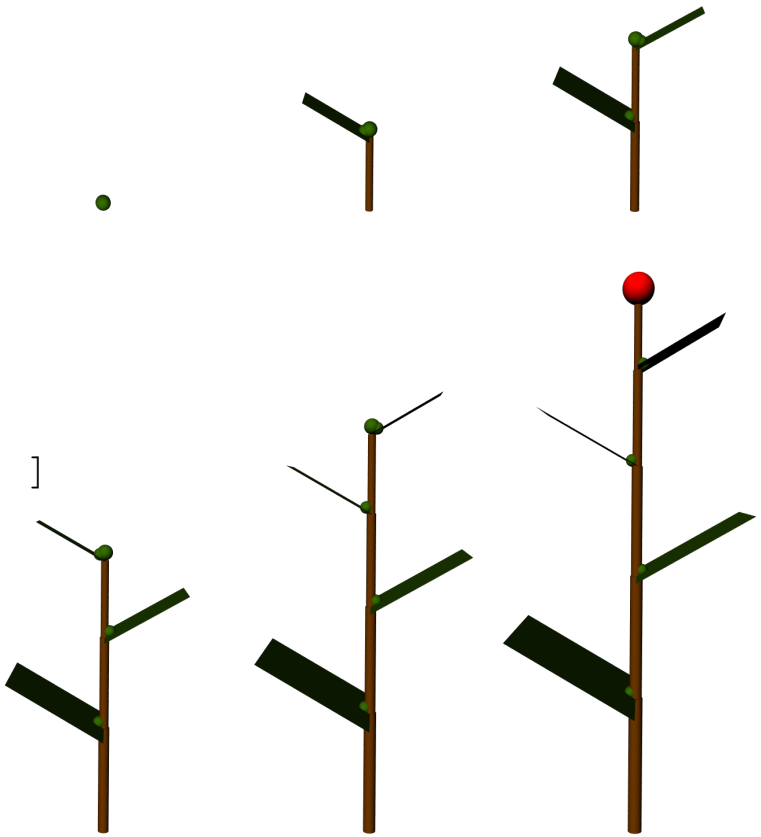




GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

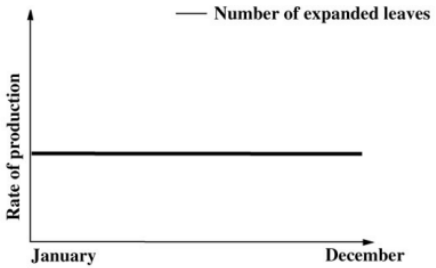
Determinate growth (with XL)

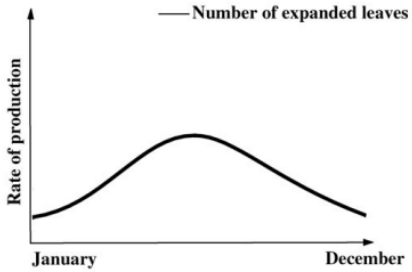
```
module Flower extends Sphere(0.2) { {setShader(RED);} }
int time;
protected void init()
[  { time = 0; }
   Axiom ==> ApicalBud;
]
public void grow()
[  ab:ApicalBud ==>
      if (time < 4) (
         Internode
         [ RL(BRANCH_ANGLE) LateralBud ]
         [ RL(LEAF_ANGLE) Leaf ]
         RH(PHYLLOTAXIS)
         ab
      ) else (
         Internode Flower
      )
   ;
   ...
   { time :+= 1; }
]
```

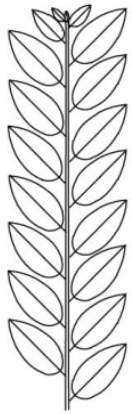


GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Continuous growth / rhythmic growth

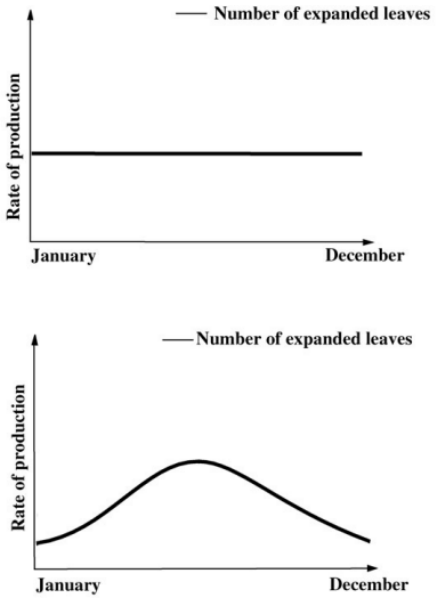


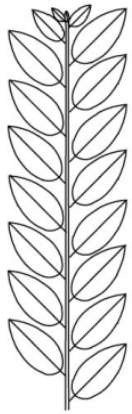


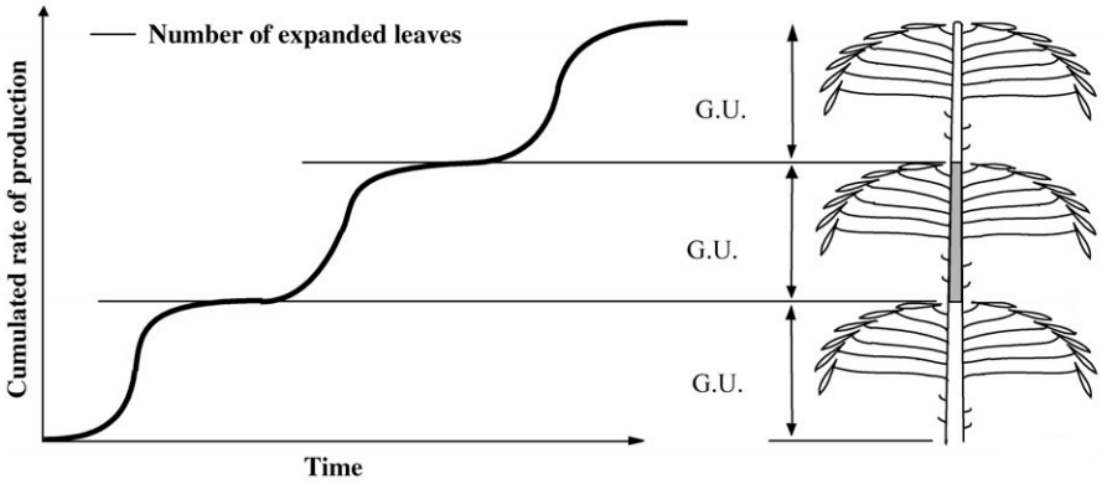


GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

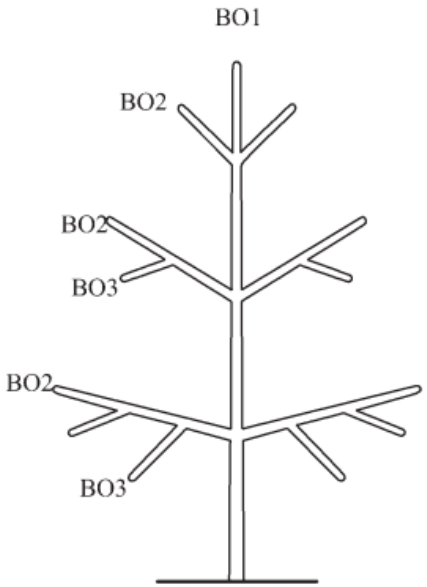
Continuous growth / rhythmic growth







Monopodial branching and branching order



GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

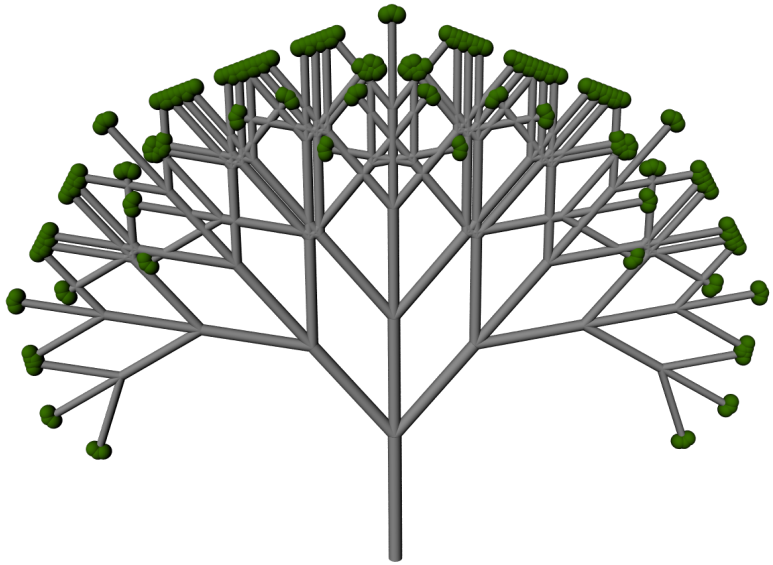
Monopodial branching and branching order (with XL)

```
module Bud extends Sphere(0.1)
   { { setColor(0x336600); } }

module Internode extends Cylinder(1, 0.05);

const float BRANCH_ANGLE = 40;

protected void init()
[
   Axiom ==> Bud;
]

public void grow()
[  b:Bud ==>
      Internode
      [ RL(BRANCH_ANGLE) Bud ]
      [ RL(-BRANCH_ANGLE) Bud ]
      b
   ; ...
]
```

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Monopodial branching and branching order (with XL)

```
module Bud extends Sphere(0.1)
   { { setColor(0x336600); } }

module Internode extends Cylinder(1, 0.05);

const float BRANCH_ANGLE = 40;

protected void init()
[
   Axiom ==> Bud;
]

public void grow()
[  b:Bud ==>
      Internode
      [ RL(BRANCH_ANGLE) Bud ]
      [ RL(-BRANCH_ANGLE) Bud ]
      b
   ; ...
]
```



GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Monopodial branching and branching order (with XL)

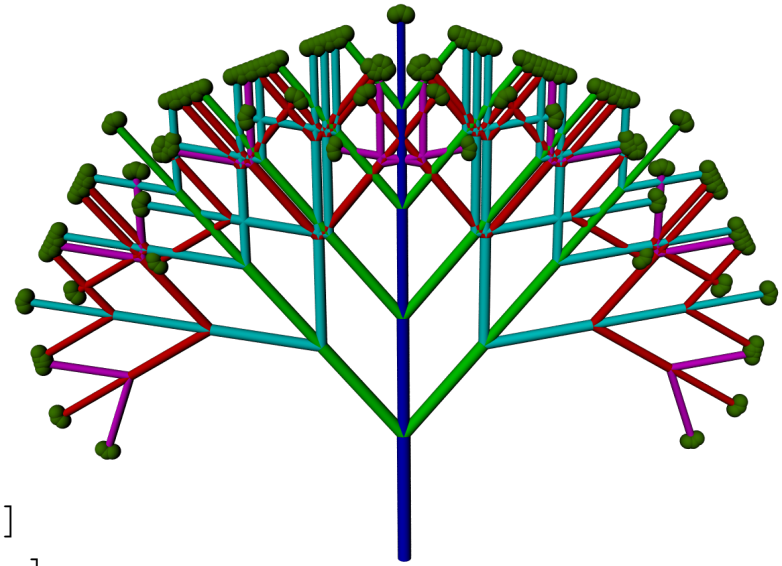
```
module Bud(int order) extends Sphere(0.1)
   { { setColor(0x336600); } }

module Internode(int order) extends Cylinder(1, 0.05);

const float BRANCH_ANGLE = 40;

protected void init()
[
   Axiom ==> Bud(1);
]

public void grow()
[  b:Bud(order) ==>
                Internode(order)
      [ RL(BRANCH_ANGLE) Bud(order+1) ]
      [ RL(-BRANCH_ANGLE) Bud(order+1) ]
      b
   ; ...
]
```

Monopodial branching and branching order (with XL)

```
module Bud(int order) extends Sphere(0.1)
   { { setColor(0x336600); } }

module Internode(int order) extends Cylinder(1, 0.05);

const float BRANCH_ANGLE = 40;

protected void init()
[
   Axiom ==> Bud(1);
]

public void grow()
[  b:Bud(order) ==>
      P(order) Internode(order)
      [ RL(BRANCH_ANGLE) Bud(order+1) ]
      [ RL(-BRANCH_ANGLE) Bud(order+1) ]
      b
   ; ...
]
```

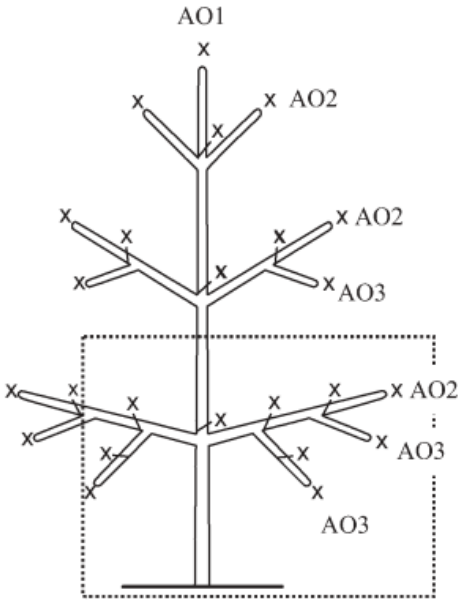
Monopodial branching and branching order (with XL)

```
module Bud(int order) extends Sphere(0.1)
   { { setColor(0x336600); } }

module Internode(int order) extends Cylinder(1, 0.05);

const float BRANCH_ANGLE = 40;

protected void init()
[
   Axiom ==> Bud(1);
]

public void grow()
[  b:Bud(order) ==>
      P(order) Internode(order)
      [ RL(BRANCH_ANGLE) Bud(order+1) ]
      [ RL(-BRANCH_ANGLE) Bud(order+1) ]
      b
   ; ...
]
```



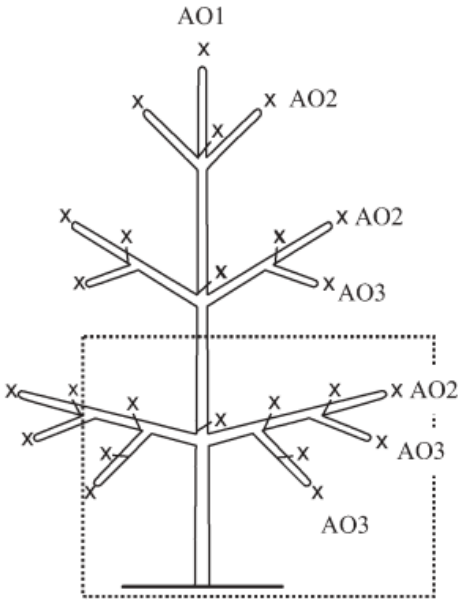
GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

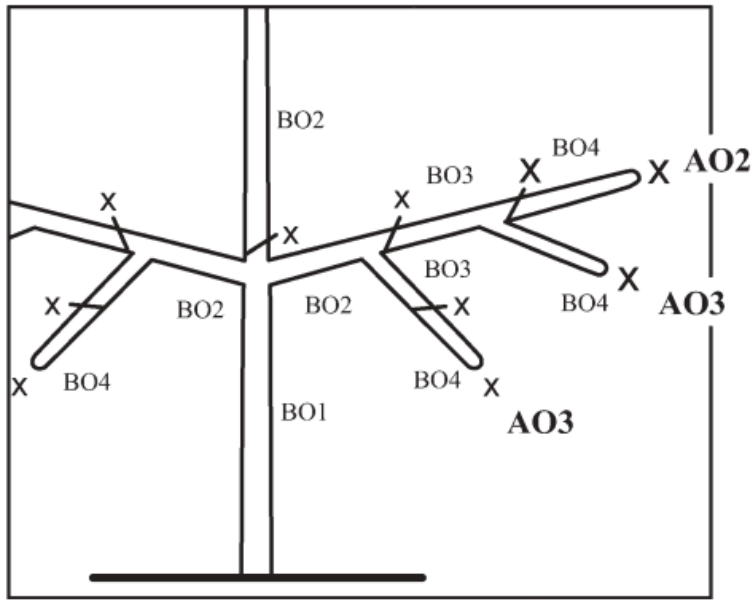
Sympodial branching and (apparent) branching order



GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Sympodial branching and (apparent) branching order





GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Sympodial branching and branching order (with XL)

```
public void grow()
[
   b:Bud(order) ==>
      P(order) Internode(order)
      [ RL(BRANCH_ANGLE) Bud(order+1) ]
      [ RL(-BRANCH_ANGLE) Bud(order+1) ]

   ;
]
```

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

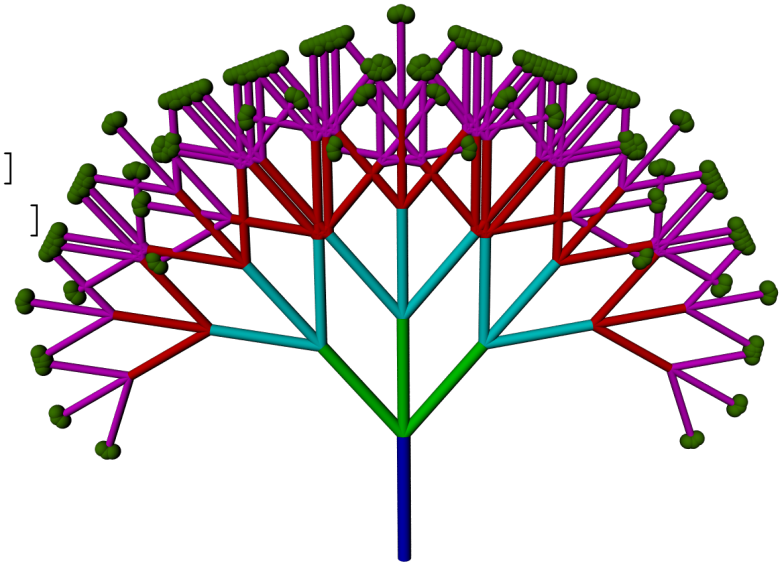
Sympodial branching and branching order (with XL)

```
public void grow()
[
   b:Bud(order) ==>
      P(order) Internode(order)
      [ RL(BRANCH_ANGLE) Bud(order+1) ]
      [ RL(-BRANCH_ANGLE) Bud(order+1) ]
      Bud(order+1)
   ;
]
```

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

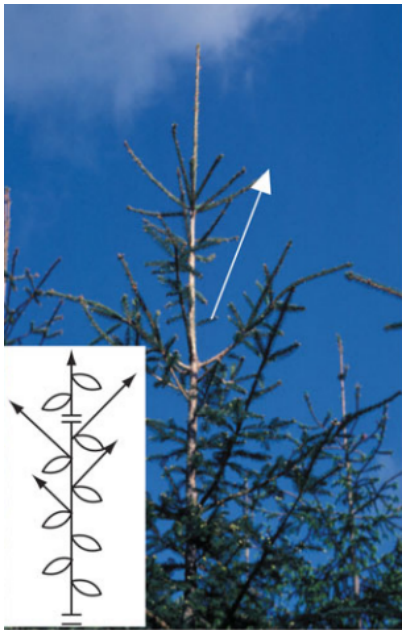
Sympodial branching and branching order (with XL)

```
public void grow()
[
   b:Bud(order) ==>
      P(order) Internode(order)
      [ RL(BRANCH_ANGLE) Bud(order+1) ]
      [ RL(-BRANCH_ANGLE) Bud(order+1) ]
      Bud(order+1)
   ;
]
```



GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Acrotonic branching



Acrotonic branching (with XL)

```
module Bud(int order) extends Sphere(0.1)
   { { setColor(0x336600); setTransform(0, 0, 0.1); } }
module Internode(int order) extends Cylinder(1, 0.05)
   { { setColor(0x663300); } }
const int BRANCH_ANGLE = 40;

protected void init()
[  Axiom ==>
      Bud(1)
   ;
]
public void grow()
[  b:Bud(order ==>
      Internode(order)

         [ RL(BRANCH_ANGLE) Internode(order+1) ]
         [ RL(-BRANCH_ANGLE) Internode(order+1) ]

      b
   ; ...
]
```

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

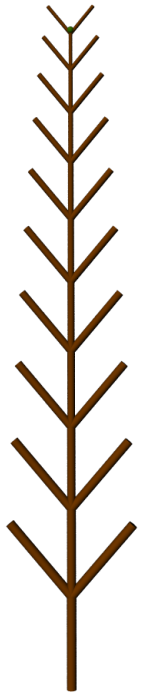
Acrotonic branching (with XL)

```
module Bud(int order) extends Sphere(0.1)
   { { setColor(0x336600); setTransform(0, 0, 0.1); } }
module Internode(int order) extends Cylinder(1, 0.05)
   { { setColor(0x663300); } }
const int BRANCH_ANGLE = 40;

protected void init()
[  Axiom ==>
      Bud(1)
   ;
]
public void grow()
[  b:Bud(order ==>
      Internode(order)
      if (order == 1) (
         [ RL(BRANCH_ANGLE) Internode(order+1) ]
         [ RL(-BRANCH_ANGLE) Internode(order+1) ]
      )
      b
   ; ...
]
```

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Acrotonic branching (with XL)

```
module Bud(int order) extends Sphere(0.1)
   { { setColor(0x336600); setTransform(0, 0, 0.1); } }
module Internode(int order) extends Cylinder(1, 0.05)
   { { setColor(0x663300); } }
const int BRANCH_ANGLE = 40;

protected void init()
[  Axiom ==>
      Bud(1)
   ;
]
public void grow()
[  b:Bud(order ==>
      Internode(order)
      if (order == 1) (
         [ RL(BRANCH_ANGLE) Internode(order+1) ]
         [ RL(-BRANCH_ANGLE) Internode(order+1) ]
      )
      b
   ; ...
]
```



GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Acrotonic branching (with XL)

```
module Bud(int order, float relPos) extends Sphere(0.1)
   { { setColor(0x336600); setTransform(0, 0, 0.1); } }
module Internode(int order, super.length) extends
   Cylinder(length, 0.05) { { setColor(0x663300); } }
const int BRANCH_ANGLE = 40;

protected void init()
[  Axiom ==>
      Bud(1, 0)
   ;
]
public void grow()
[  b:Bud(order, relPos), (relPos < 1) ==>
      Internode(order, 1)
      if (order == 1) (
         [ RL(BRANCH_ANGLE) Internode(order+1,                    ) ]
         [ RL(-BRANCH_ANGLE) Internode(order+1,                    ) ]
      )
      b { b[relPos] :+= 0.1; }
   ; ...
]
```

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

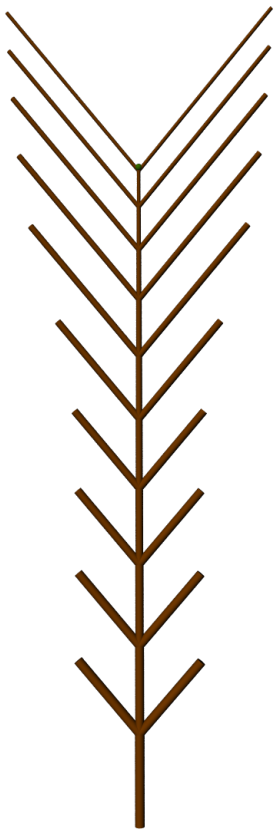
Acrotonic branching (with XL)

```
module Bud(int order, float relPos) extends Sphere(0.1)
   { { setColor(0x336600); setTransform(0, 0, 0.1); } }
module Internode(int order, super.length) extends
   Cylinder(length, 0.05) { { setColor(0x663300); } }
const int BRANCH_ANGLE = 40;
const FunctionRef acrotony = function("acrotony");
protected void init()
[  Axiom ==>
      Bud(1, 0)
   ;
]
public void grow()
[  b:Bud(order, relPos), (relPos < 1) ==>
      Internode(order, 1)
      if (order == 1) (
         [ RL(BRANCH_ANGLE) Internode(order+1, acrotony[relPos]) ]
         [ RL(-BRANCH_ANGLE) Internode(order+1, acrotony[relPos]) ]
      )
      b { b[relPos] :+= 0.1; }
   ; ...
]
```

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Acrotonic branching (with XL)

```
module Bud(int order, float relPos) extends Sphere(0.1)
   { { setColor(0x336600); setTransform(0, 0, 0.1); } }
module Internode(int order, super.length) extends
   Cylinder(length, 0.05) { { setColor(0x663300); } }
const int BRANCH_ANGLE = 40;
const FunctionRef acrotony = function("acrotony");
protected void init()
[  Axiom ==>
      Bud(1, 0)
   ;
]
public void grow()
[  b:Bud(order, relPos), (relPos < 1) ==>
      Internode(order, 1)
      if (order == 1) (
         [ RL(BRANCH_ANGLE) Internode(order+1, acrotony[relPos]) ]
         [ RL(-BRANCH_ANGLE) Internode(order+1, acrotony[relPos]) ]
      )
      b { b[relPos] :+= 0.1; }
   ; ...
]
```



GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Mesotonic branching

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

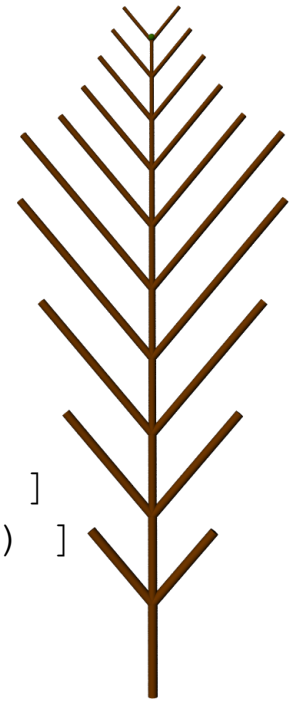
Mesotonic branching (with XL)

```
const FunctionRef mesotony = function("mesotony");

...
[ RL(BRANCH_ANGLE) Internode(order+1, mesotony[relPos]) ]
[ RL(-BRANCH_ANGLE) Internode(order+1, mesotony[relPos]) ]
...
```

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Mesotonic branching (with XL)

```
const FunctionRef mesotony = function("mesotony");

...
[ RL(BRANCH_ANGLE) Internode(order+1, mesotony[relPos]) ]
[ RL(-BRANCH_ANGLE) Internode(order+1, mesotony[relPos]) ]
...
```



GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Basitonic branching

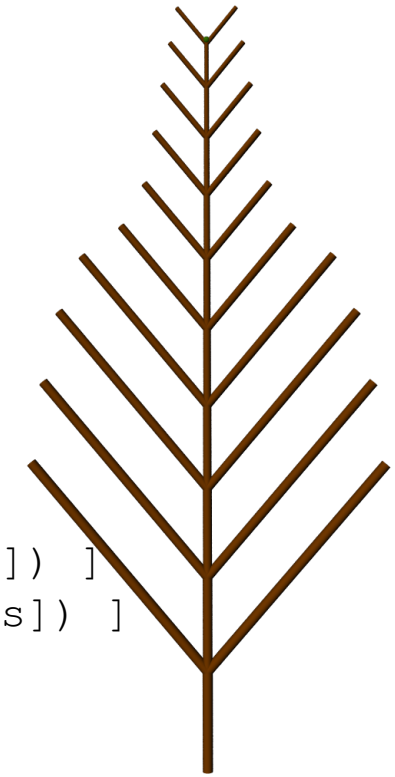
Basitonic branching (with XL)

```
const FunctionRef basitony = function("basitony");

...
[ RL(BRANCH_ANGLE) Internode(order+1, basitony[relPos]) ]
[ RL(-BRANCH_ANGLE) Internode(order+1, basitony[relPos]) ]
...
```

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Basitonic branching (with XL)

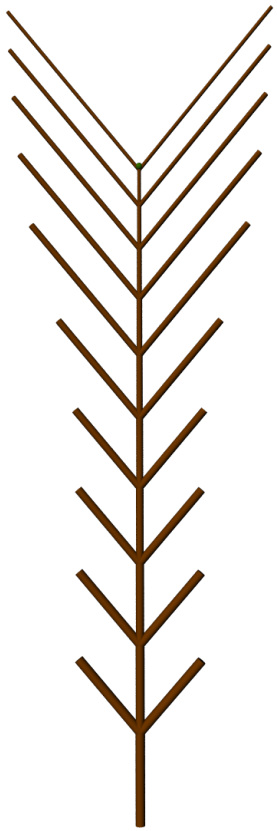
```
const FunctionRef basitony = function("basitony");

...
[ RL(BRANCH_ANGLE) Internode(order+1, basitony[relPos]) ]
[ RL(-BRANCH_ANGLE) Internode(order+1, basitony[relPos]) ]
...
```



GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

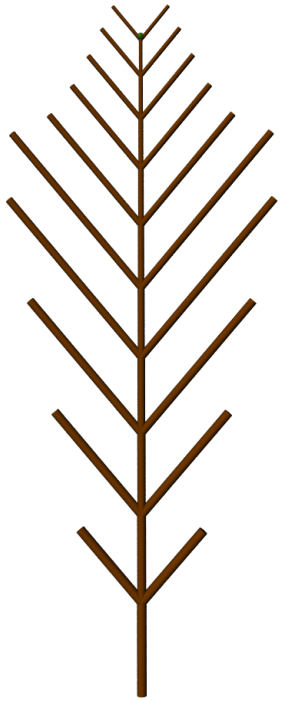
acrotony.func

range: 0.0 1.0
points: 5
0.0 1
0.3 1.5
0.4 2
0.5 5
1.0 6



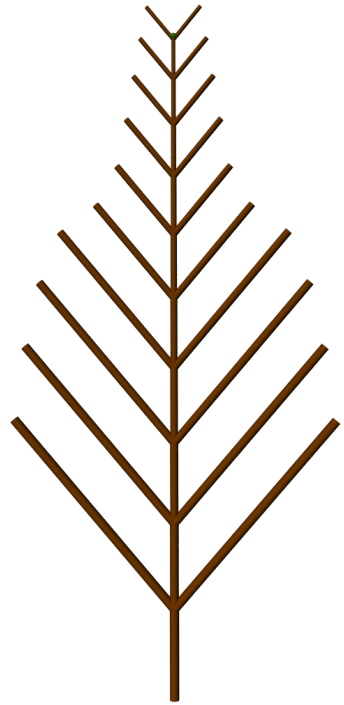
mesotony.func

range: 0.0 1.0
points: 5
0.0 1
0.3 5
0.4 6
0.5 2
1.0 1



basitony.func

range: 0.0 1.0
points: 5
0.0 6
0.3 5
0.4 2
0.5 1.5
1.0 1



GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

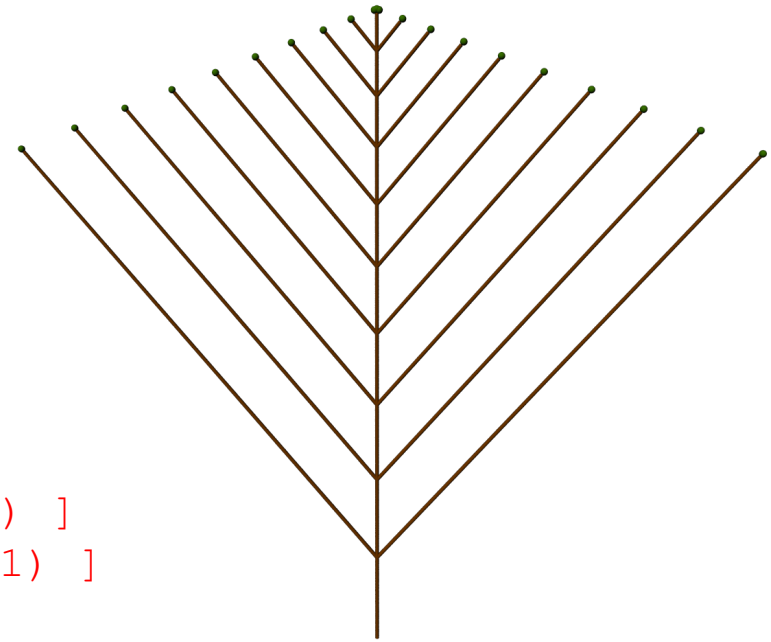
Orthotropy

Orthotropy (with XL)

```
module Bud(int order) extends Sphere(0.1)
   { { setColor(0x336600); setTransform(0, 0, 0.1); } }
module Internode(int order, super.length)
   extends Cylinder(length, 0.05) { { setColor(0x663300); } }
const float BRANCH_ANGLE = 40;
protected void init ()
[  Axiom ==>
      Bud(1)
   ;
]
public void grow()
[  b:Bud(order) ==>
      Internode(order, 1)
      if (order == 1) (
         [ RL(BRANCH_ANGLE) Bud(order+1) ]
         [ RL(-BRANCH_ANGLE) Bud(order+1) ]
      )

        b
   ; ...
]
```

Orthotropy (with XL)

```
module Bud(int order) extends Sphere(0.1)
   { { setColor(0x336600); setTransform(0, 0, 0.1); } }
module Internode(int order, super.length)
   extends Cylinder(length, 0.05) { { setColor(0x663300); } }
const float BRANCH_ANGLE = 40;
protected void init ()
[  Axiom ==>
      Bud(1)
   ;
]
public void grow()
[  b:Bud(order) ==>
      Internode(order, 1)
      if (order == 1) (
         [ RL(BRANCH_ANGLE) Bud(order+1) ]
         [ RL(-BRANCH_ANGLE) Bud(order+1) ]
      )

        b
   ; ...
]
```



GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

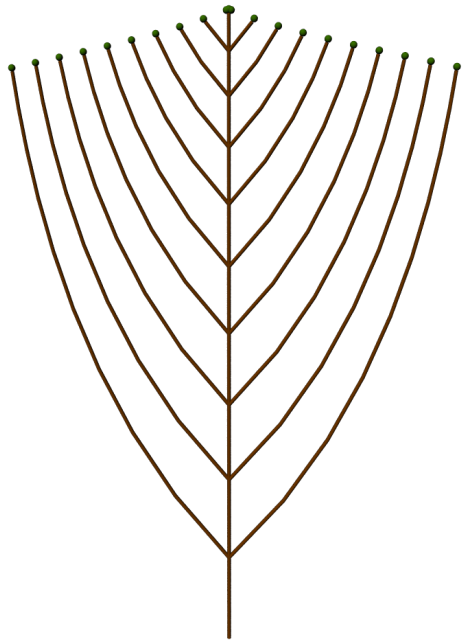
Orthotropy (with XL)

```
module Bud(int order) extends Sphere(0.1)
   { { setColor(0x336600); setTransform(0, 0, 0.1); } }
module Internode(int order, super.length)
   extends Cylinder(length, 0.05) { { setColor(0x663300); } }
const float BRANCH_ANGLE = 40;
protected void init ()
[  Axiom ==>
      Bud(1)
   ;
]
public void grow()
[  b:Bud(order) ==>
      Internode(order, 1)
      if (order == 1) (
         [ RL(BRANCH_ANGLE) Bud(order+1) ]
         [ RL(-BRANCH_ANGLE) Bud(order+1) ]
      ) else (
         RD(new Vector3d(0, 0, 1), 0.2)
      ) b
   ; ...
]
```

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Orthotropy (with XL)

```
module Bud(int order) extends Sphere(0.1)
   { { setColor(0x336600); setTransform(0, 0, 0.1); } }
module Internode(int order, super.length)
   extends Cylinder(length, 0.05) { { setColor(0x663300); } }
const float BRANCH_ANGLE = 40;
protected void init ()
[  Axiom ==>
      Bud(1)
   ;
]
public void grow()
[  b:Bud(order) ==>
      Internode(order, 1)
      if (order == 1) (
         [ RL(BRANCH_ANGLE) Bud(order+1) ]
         [ RL(-BRANCH_ANGLE) Bud(order+1) ]
      ) else (
         RD(new Vector3d(0, 0, 1), 0.2)
      ) b
   ; ...
]
```



GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Plagiotropy

Plagiotropy (with XL)

```
protected void init()
[  Axiom ==>

      Bud(1)
   ;
]
public void grow()
[  b:Bud(order) ==>
      Internode(order, 1)
      if (order == 1) (
         [ RL(BRANCH_ANGLE) Bud(order+1) ]
         [ RL(-BRANCH_ANGLE) Bud(order+1) ]
      )




        b
   ; ...
]
```

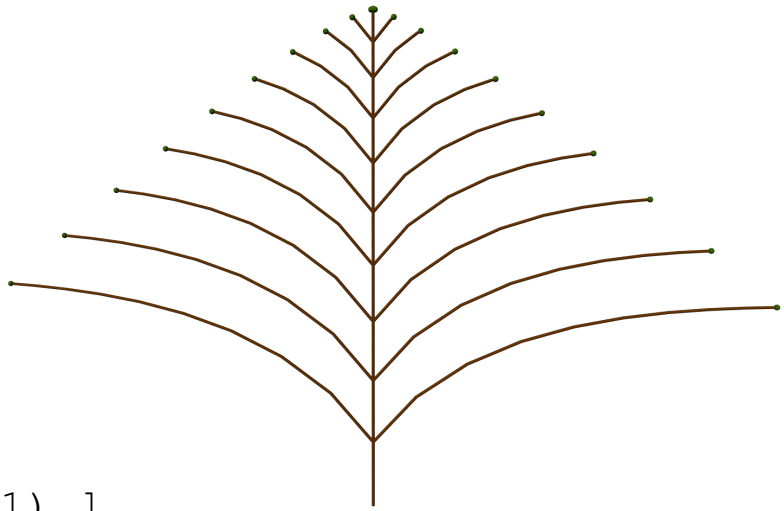
Plagiotropy (with XL)

```
module Root; const Root root = new Root();
protected void init()
[  Axiom ==>
      root
      Bud(1)
   ;
]
public void grow()
[  b:Bud(order) ==>
      Internode(order, 1)
      if (order == 1) (
         [ RL(BRANCH_ANGLE) Bud(order+1) ]
         [ RL(-BRANCH_ANGLE) Bud(order+1) ]
      ) else (
      {  Vector3d d = b - root;
         d.z = 0;
         d.normalize();
      }  RD(d, 0.3)
      ) b
   ; ...
]
```

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Plagiotropy (with XL)

```
module Root; const Root root = new Root();
protected void init()
[  Axiom ==>
      root
      Bud(1)
   ;
]
public void grow()
[  b:Bud(order) ==>
      Internode(order, 1)
      if (order == 1) (
         [ RL(BRANCH_ANGLE) Bud(order+1) ]
         [ RL(-BRANCH_ANGLE) Bud(order+1) ]
      ) else (
      {  Vector3d d = b - root;
         d.z = 0;
         d.normalize();
      }  RD(d, 0.3)
      ) b
   ; ...
]
```



GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

- Sympodial branching with the random choose of a lateral bud that continues growing in the direction of its mother axis
- 3 meristem states
- Bud probabilities

- Adding some features of FSPM (as shown in the presentation by Gerhard)
- ...

Thank you for your attention.

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN