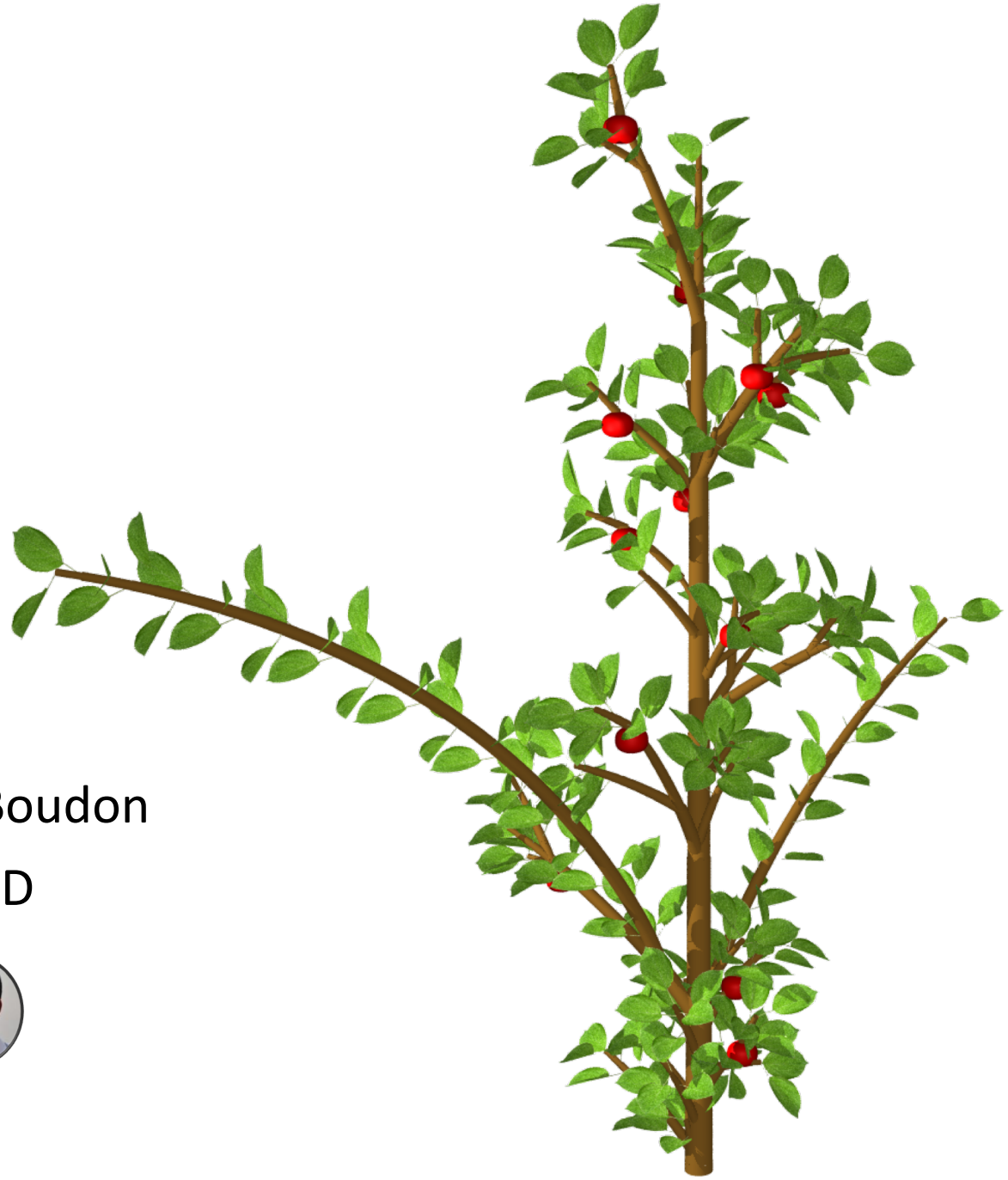
Modelling fruit tree architecture with L-systems



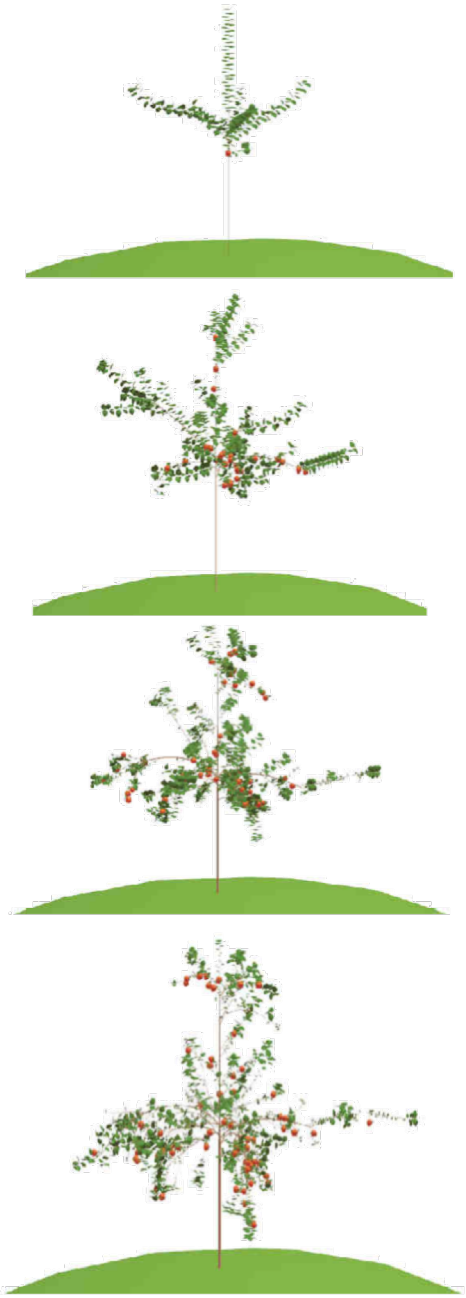
Frédéric Boudon

CIRAD

Motivation for a model

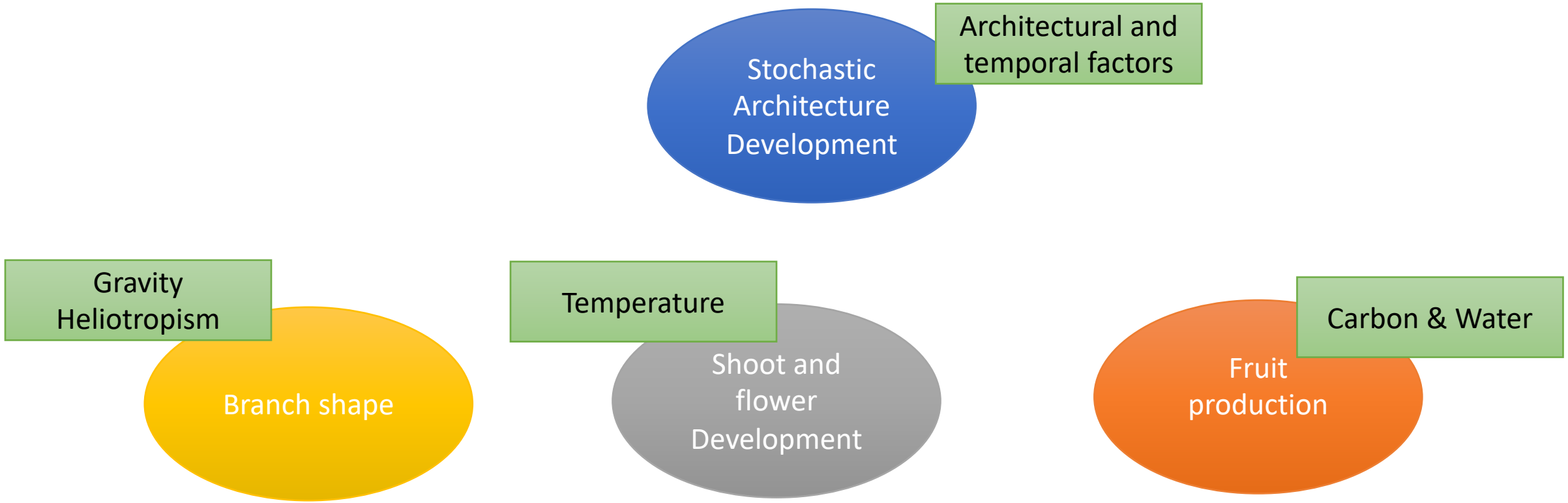
- To integrate an **integrative view**.

- Based on existing models of fruit trees
  - Apple tree (Costes et al. 2008),
  - Peach tree (Lopez et al., 2010).
  - Almond tree (De Jong et al., 2017).



A functional-structural model of fruit tree growth and development

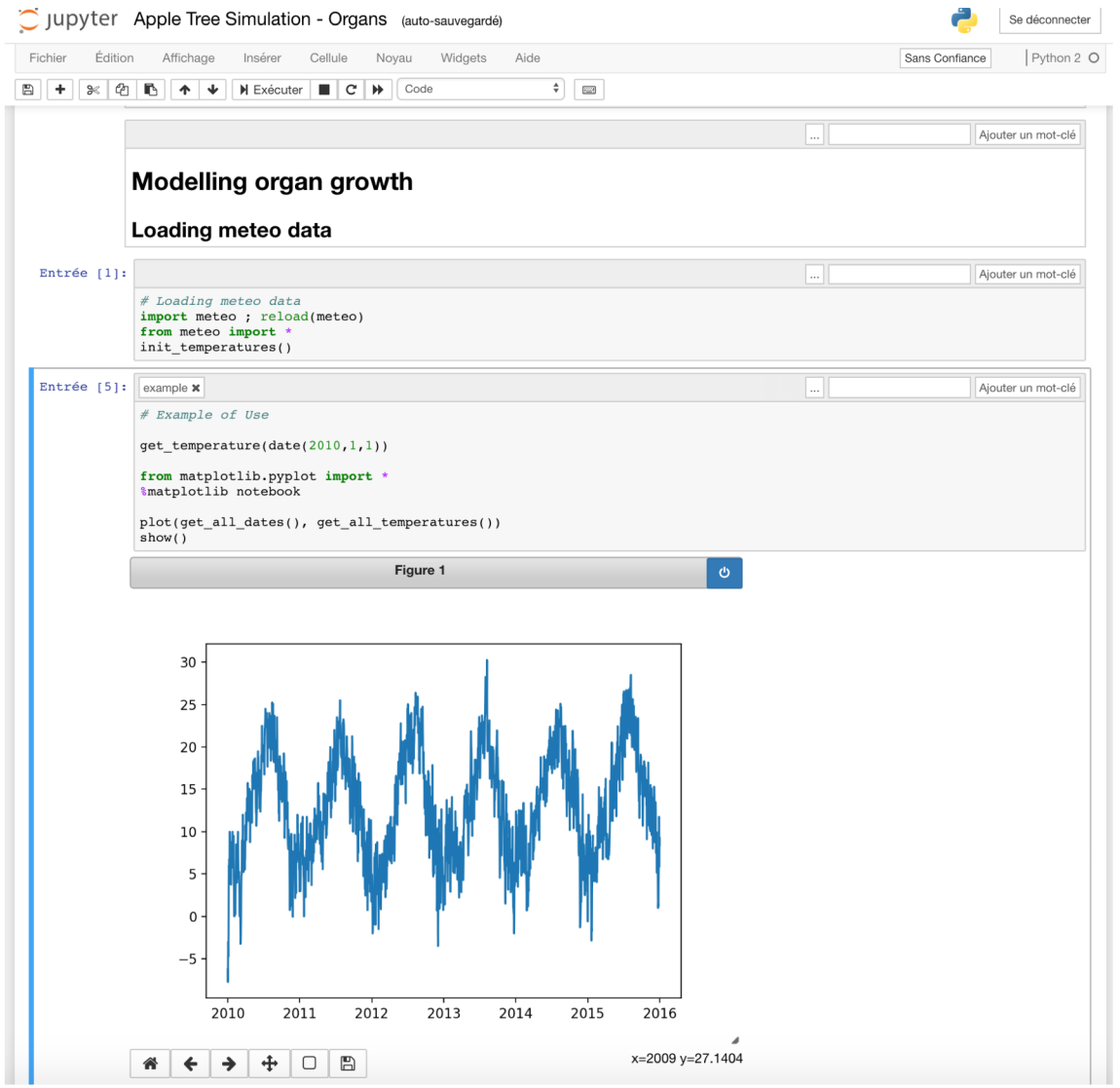
- Creation of an FSPM model that simulates development of architecture, phenology of organs, and production of fruits.



Exercices

- More details on …
  - ‘ArchiModelling.ipynb’
  - ‘Apple Tree Simulation - Organs.ipynb’
  - ‘Apple Tree Simulation - Architecture.ipynb’

  in the directory ‘archimodelling’



Modelling organ shape

Primitives

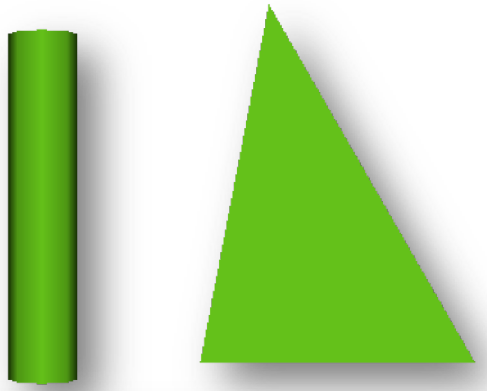
- Set of predefined shapes (Cylinder, Sphere, Quad, etc.)

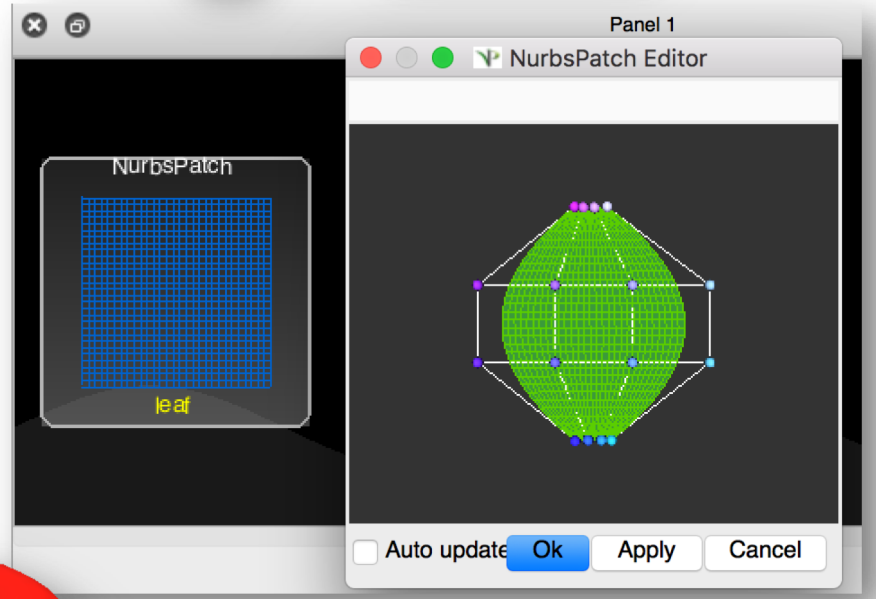
- Define polygon

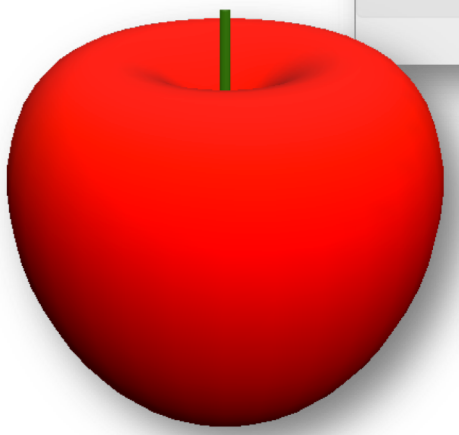
  `{ . f . +(120) f(1.5) . }(concave)`

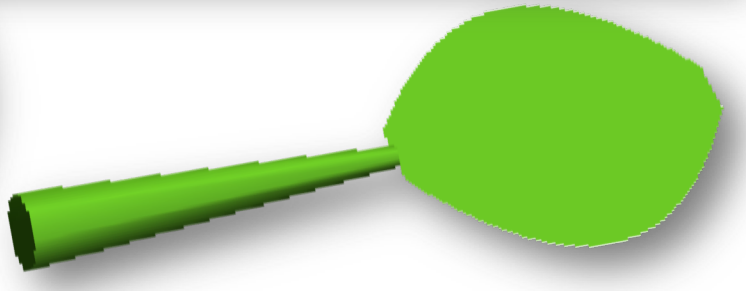
- Display specific shape
  - `@g(obj)`
    - Define graphically with editor
    - Define procedurally

      `@g(SOR([sectioncurve ]))`

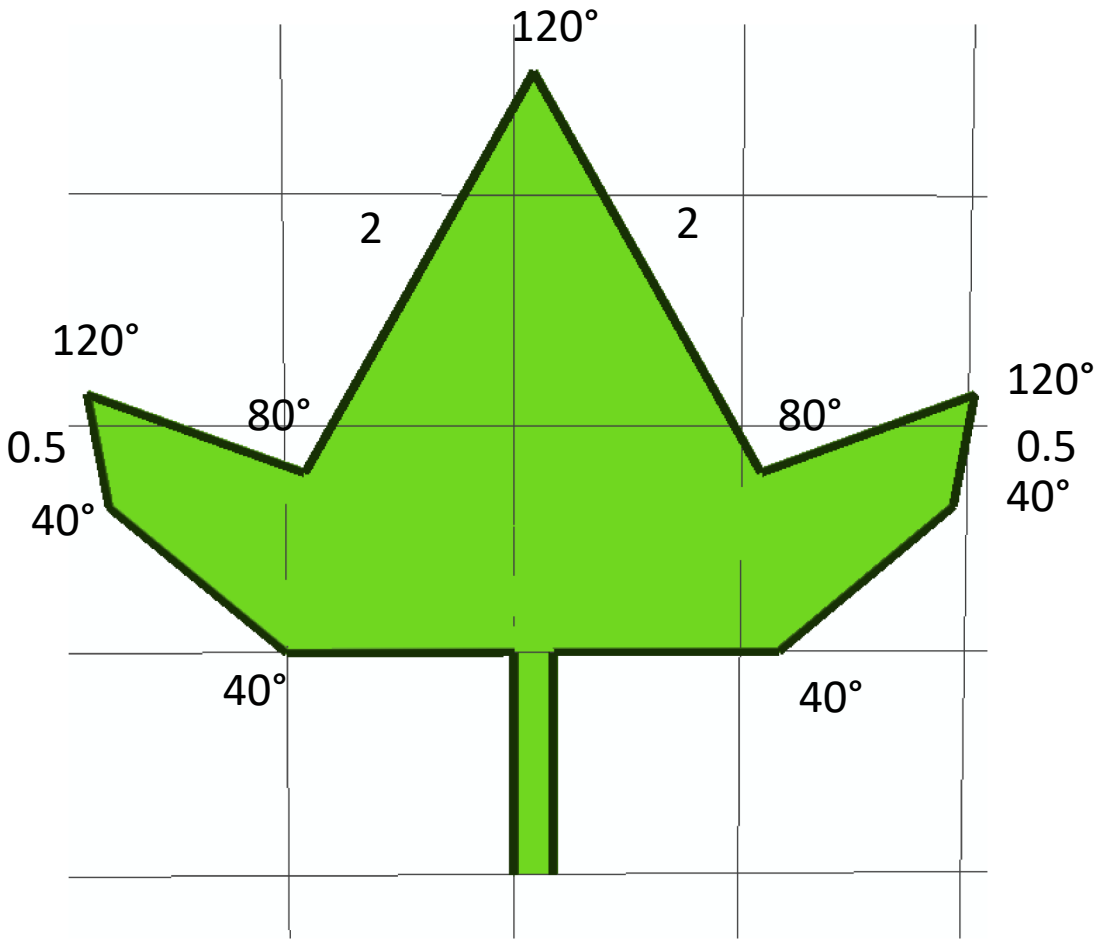








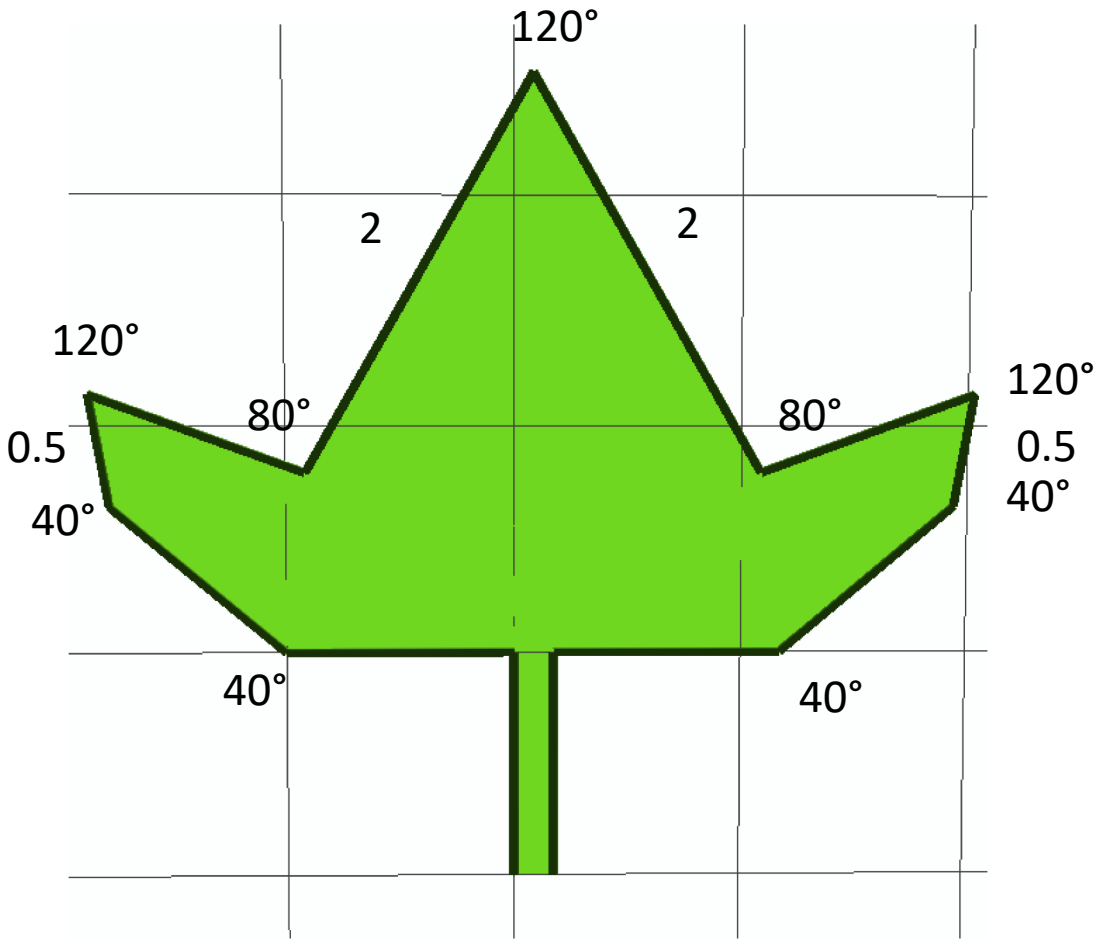
Exercise



Axiom: ,(2) _(0.02) { . F .... } (True)

F	Move forward, drawing a line
+(a)	Turn left *a* degrees
-(a)	Turn right *a* degrees

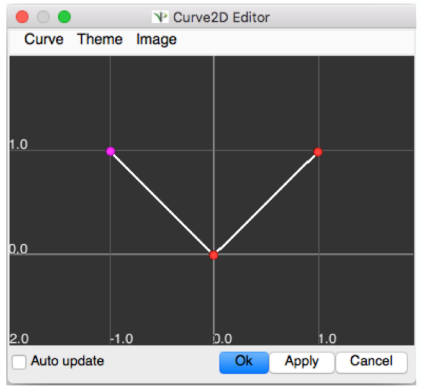
Exercise

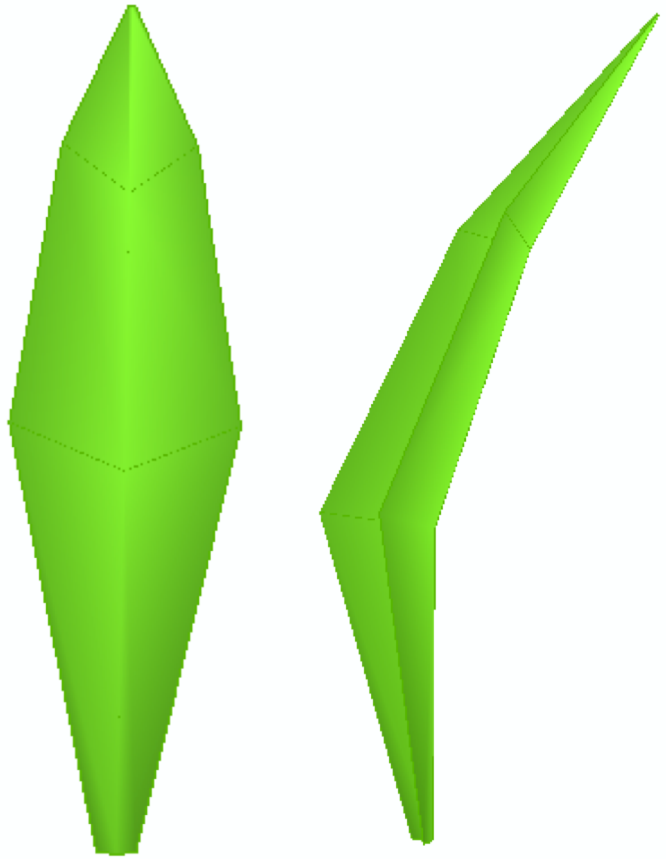


Axiom: ,(2) _(0.02) { . F +(90) F -(40) F -(40) F(0.5) -(120) F +(80) F(2) -(120) F(2) +(80) F -(120) F(0.5) -(40) F -(40) F(1) +(90) F } (True)

Sweep surfaces

- Definition of the section of turtle tracing
  - `SetContour(section)`
- Concatenation of sections
  - `@Gc`



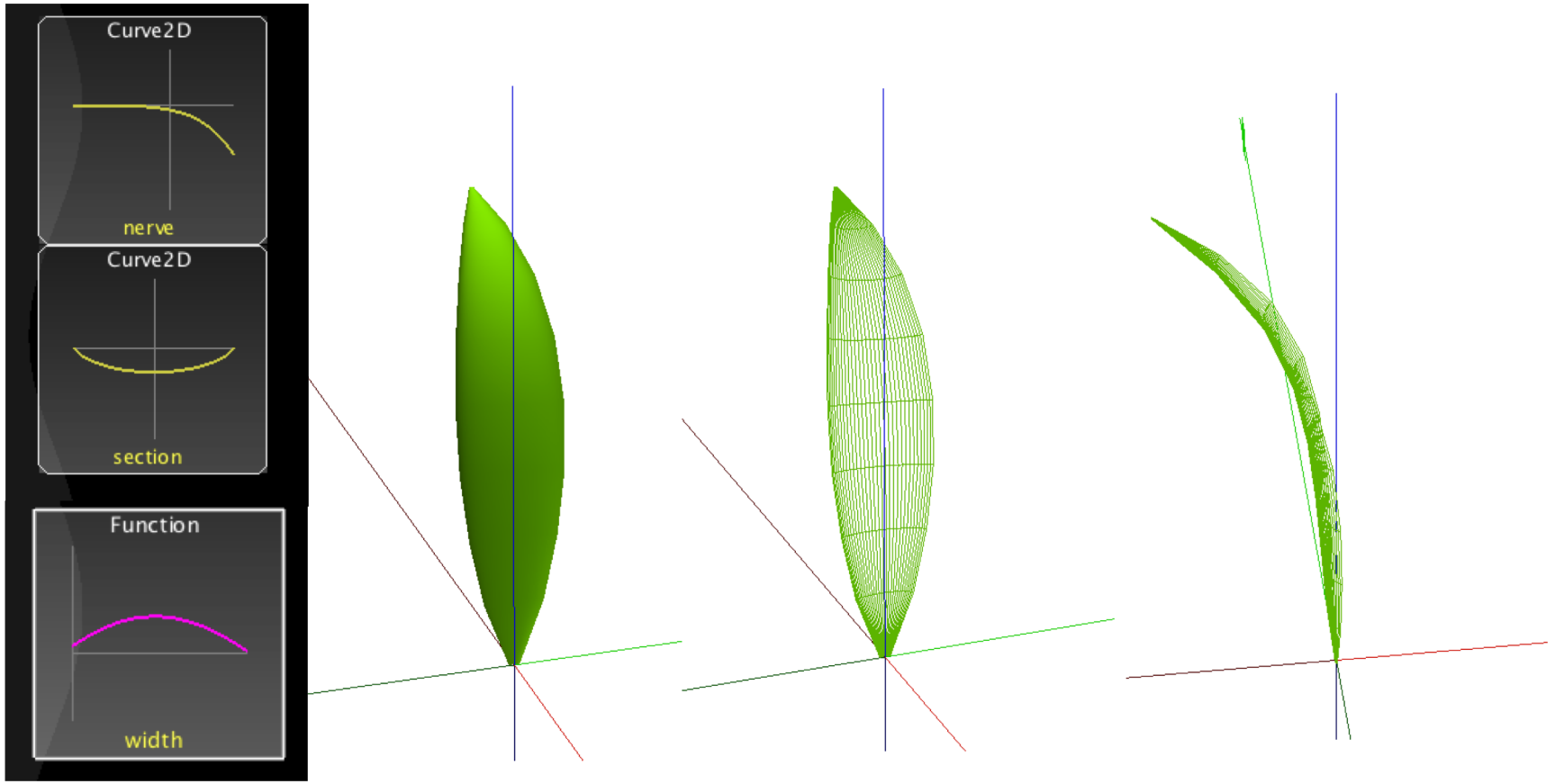


Axiom: `@Gc _(0.05) ,(2) SetContour(section) F(1,0.3) &(20) F(1,0.2) &(10) F(1,0.01)`

Sweep surfaces

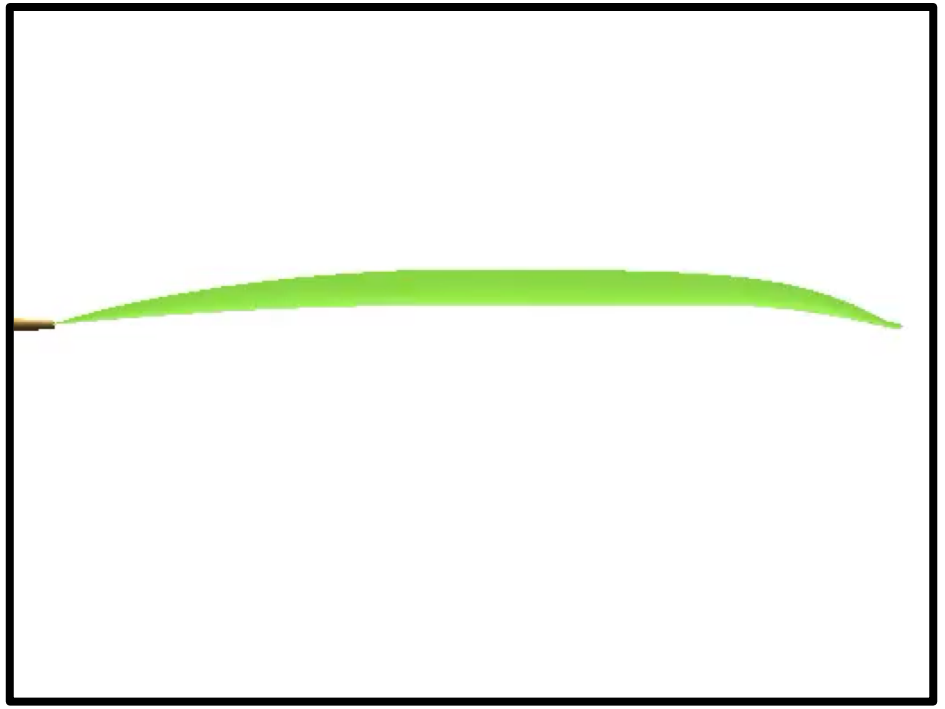
- Generalization with section and path of the tracing

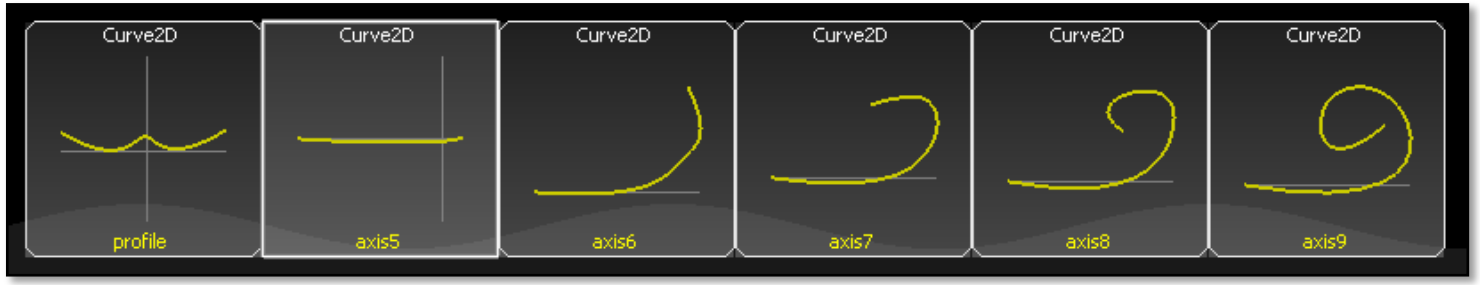
Sweep(nerve_curve, section_curve,length, segsize, width,width_func)



Interpolating profiles

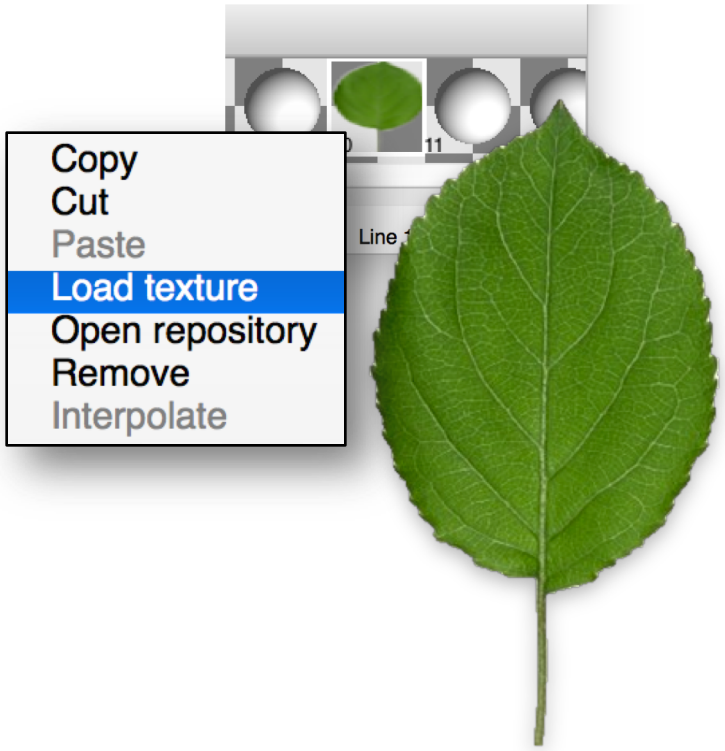
```
axisSet = [axis1, …, axisN]
times = [0, t1, …, 1.0]
interpol = ProfileInterpolation(axisSet,times,
                                degree =3)
Axiom: BG(0) Branch(length)
production:
BG(t) --> BG(t+dt)


interpretation:
BG(t) --> SetGuide(interpol(t),length)
endlsystem
```

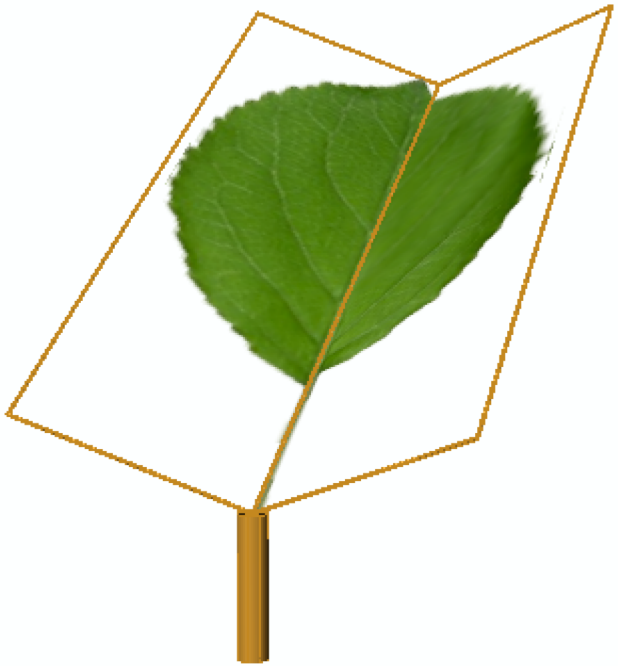




Texturing

- Applying a picture onto a polygon





Exercise : Creating an apple tree leaf

- Open model ‘organshape.lpy’ or the notebook
  - Use of ParameterSet to name attributes

```
Apex(p) --> Internode(ParameterSet(length = 2, width=0.1))
Internode(p) --> F(p.length)
```

- Creation of a model
  - In the graphical editor, create a patch, name it ‘leafshape’. Edit it to have the shape of a leaf. (In the noetbook it is made programmatically)
  - Complete the interpretation rule Leaf(p) so that it use your symbol with command @g

- Creation of a textured model
  - Import image ‘apple-leaf.png’ as texture in position 10.
  - Complete the interpretation rule Leaf(p)
    - Apply color 10
    - Set up the stretching of the texture on the leaf using TextureVScale

Modelling organ growth and development

Modelling growth

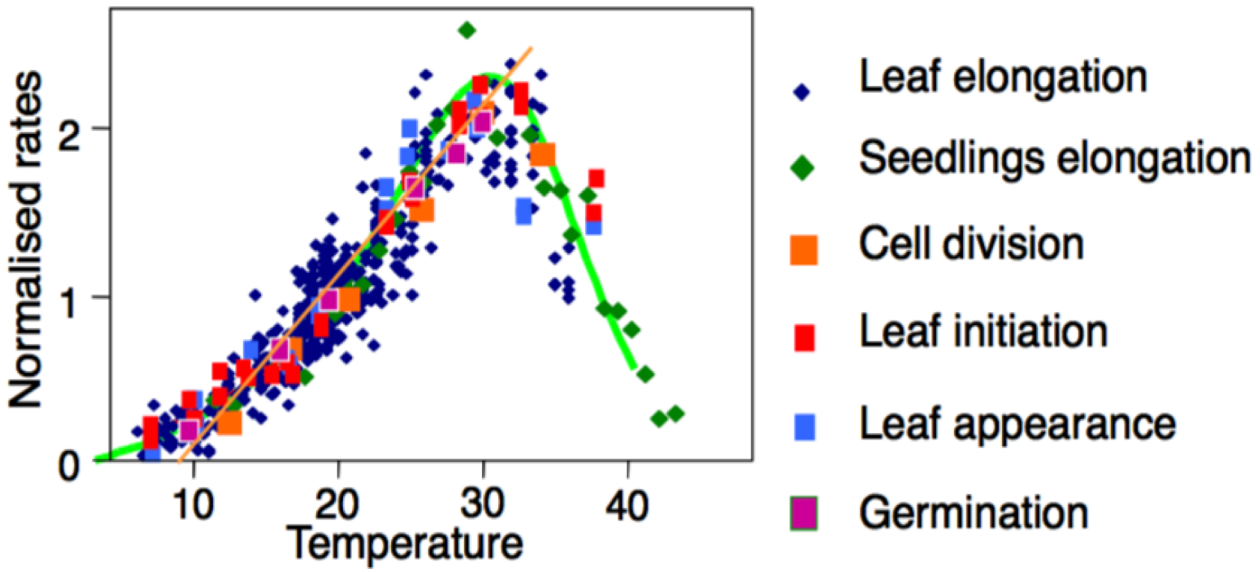
- Growth rate depends on temperature.

$$TTS = \sum_{i=1}^{n} (t_{mi} - \theta) \cdot \delta \qquad \delta = \begin{cases} 0, & t_{mi} \le \theta \\ 1, & t_{mi} > \theta \end{cases}$$

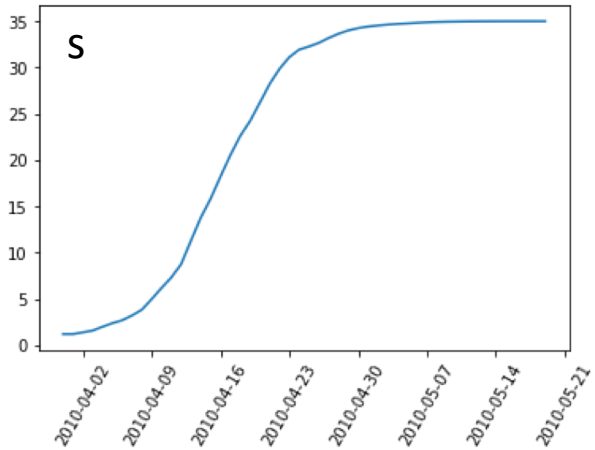
- Modelling growth with a logistic

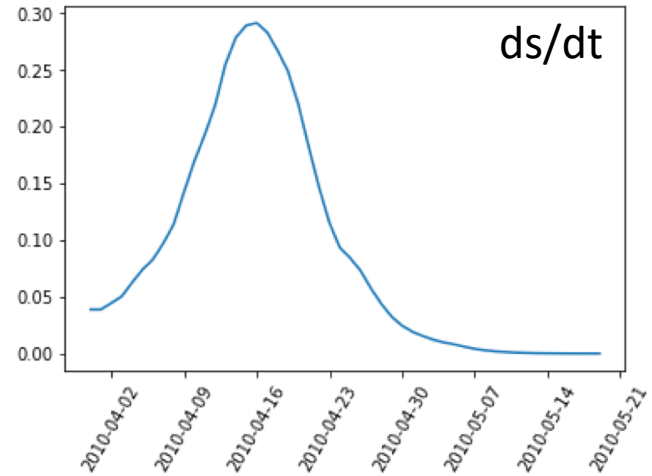
$$s(t) = \frac{A}{1+exp^{-\frac{t-t_{ip}}{b}}}$$

$$\frac{ds}{dt} = \frac{A.exp^{-\frac{t-t_{ip}}{b}}}{b.(1+exp^{-\frac{t-t_{ip}}{b}})^2}$$



Parent et al., J Exp Bot 2010.





Calendar and environmental information

- Each rewriting step correspond to a time step.
- Need to update global variables of the simulation such as
  - Date of the day
  - Current temperature, environmental conditions, etc.

Calendar and environmental information

- Each rewriting step correspond to a time step.
- Need to update global variables of the simulation such as
  - Date of the day
  - Current temperature, environmental conditions, etc.

```
cdate = None
ctemperature = None

def Start():
  global cdate, ctemperature
  cdate = firstday
  ctemperature = meteo.get_temperature(cdate)

def EndEach():
  global cdate, ctemperature
  cdate += timedelta(days=1)
  ctemperature = meteo.get_temperature(cdate)
```

More details on ‘Apple Tree Simulation - Organs.ipynb’

Exercices

- Complete the notebook and encode.
  - A thermal time model (`ThermalTime.get_effective_temperature`)

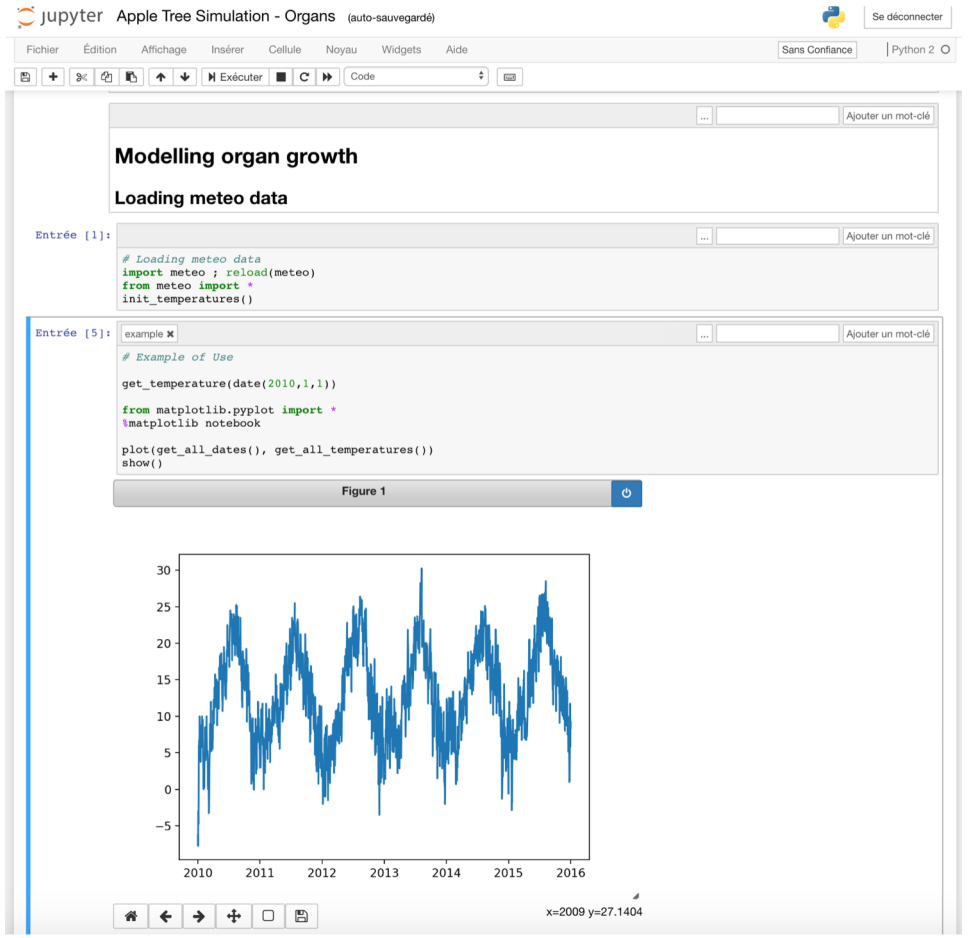
$$TTS = \sum_{i=1}^{n} (t_{mi} - \theta) \cdot \delta \qquad \delta = \begin{cases} 0, & t_{mi} \leq \theta \\ 1, & t_{mi} > \theta \end{cases}$$

  - a growth logistic function (`growth_logistic`)

$$s(t) = \frac{A}{1 + exp^{-\frac{t - t_{ip}}{b}}}$$

  - a growth rate function (`growth_rate`)

$$\frac{ds}{dt} = \frac{A.exp^{-\frac{t - t_{ip}}{b}}}{b.(1 + exp^{-\frac{t - t_{ip}}{b}})^2}$$



More details on ‘Apple Tree Simulation - Organs.ipynb’

Exercices

- Complete the notebook and encode.
  - A thermal time model

$$TTS = \sum_{i=1}^{n} (t_{mi} - \theta) \cdot \delta \qquad \delta = \begin{cases} 0, & t_{mi} \le \theta \\ 1, & t_{mi} > \theta \end{cases}$$

```
class ThermalTime:
  …
  def get_effective_temperature(self, cdate):
    ctemp = get_temperature(cdate)
    return max(0, ctemp-self.base_temperature)
```

  - a growth logistic function

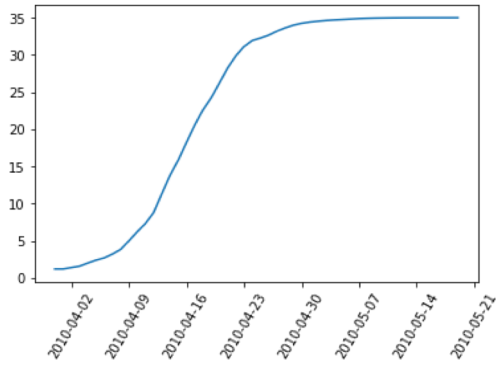
$$s(t) = \frac{A}{1+exp^{-\frac{t-t_{ip}}{b}}}$$

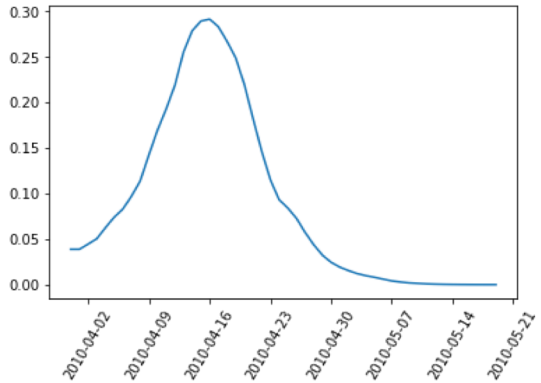
```
def growth_logistic(ttime, finalsize, tip, b):
  return finalsize / (1 + exp(-(ttime-tip)/b ))
```

  - a growth rate function

$$\frac{ds}{dt} = \frac{A.exp^{-\frac{t-t_{ip}}{b}}}{b.(1+exp^{-\frac{t-t_{ip}}{b}})^2}$$

```
def growth_rate(ttime, finalsize, tip, b):
  g = exp(-(ttime-tip)/b )
  return (finalsize * g)  / ( b * pow((1+g),2))
```





More details on ‘Apple Tree Simulation - Organs.ipynb’

Exercise : Modelling apple tree metamer development.

- Open the notebook *'Apple Tree Simulation - Organs.ipynb’* or the organdevelop.lpy model
- Complete the rule Internode and Leaf

```
Internode(p) :
  # Use mthermaltime to get the effective temperature of the day
  # and store it in variable ctime.
  cttime = mthermaltime.get_effective_temperature(cdate)
  # Compute length increment with growth_rate function.
  # Parameterize it with p.finallength, tip_internode and b_internode.
  # Integrate grow rate on ctime to have length increase.
  # Add it to p.length variable
  # p.length += ... to complete
  # Do the same on the width variable
  # p.width += ... to complete
  # Accumulate thermal time into variable p.ttime.
  p.ttime += cttime
  produce Internode(p)
```

More details on ‘Apple Tree Simulation - Organs.ipynb’

Exercise : Modelling apple tree metamer development.

- Internode rules

```
Internode(p) :
  cttime = mthermaltime.get_effective_temperature(cdate)

  p.length += growth_rate(p.ttime, p.finallength, tip_internode,
b_internode)*cttime

  p.width += growth_rate(p.ttime, p.finalwidth, tip_internode,
b_internode)*cttime
  p.ttime += cttime

  produce Internode(p)
```

More details on ‘Apple Tree Simulation - Organs.ipynb’

Exercise : Modelling apple tree metamer development.

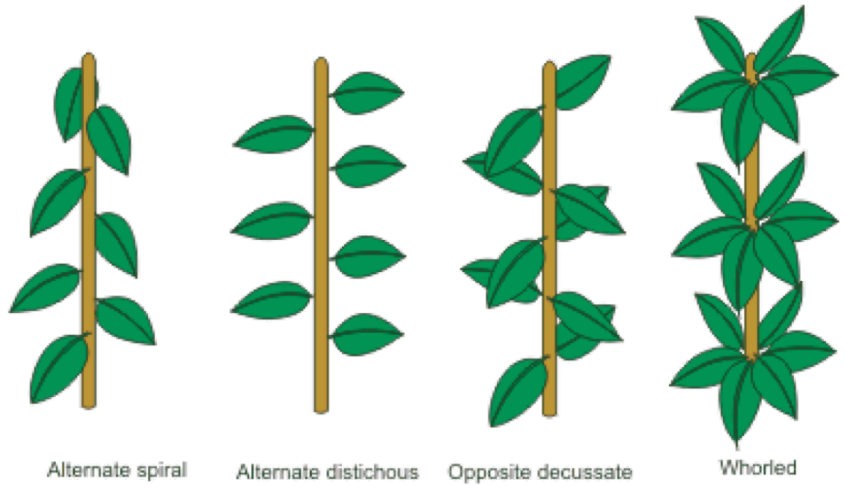
- Internode rules

```
Internode(p) :
  cttime = mthermaltime.get_effect

  p.length += growth_rate(p.ttime,
b_internode)*cttime

  p.width += growth_rate(p.ttime,
b_internode)*cttime
  p.ttime += cttime

  produce Internode(p)
```

Modelling Growth Unit development

Phyllotaxy

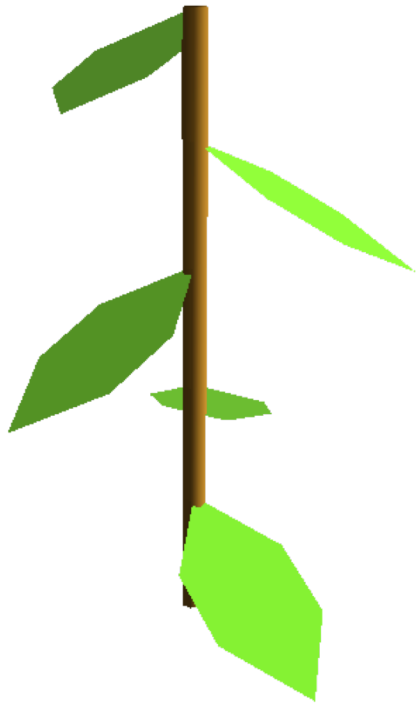
- Organisation of leaves along an axis



- Exercise: reproduce these organisations

Phyllotaxy

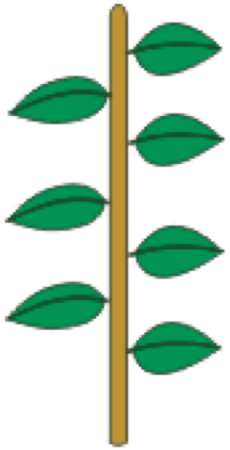
- Alternate phyllotaxy (spiral of 137.5°)

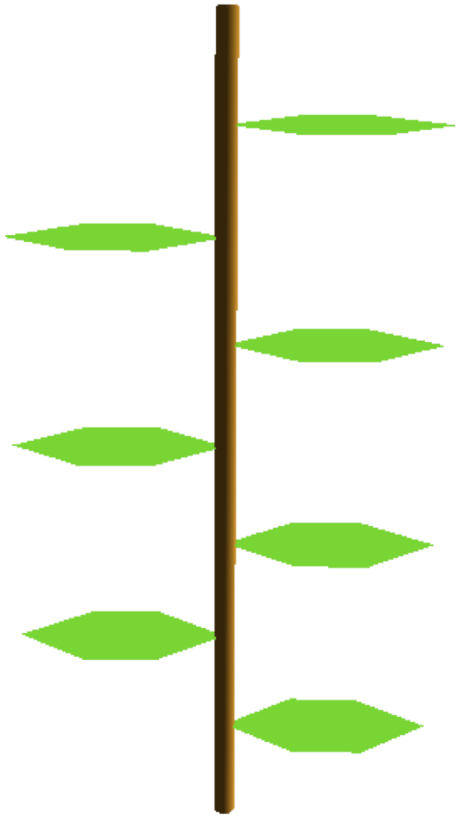


```
Axiom:
   for i in range(5):
     nproduce F(1) /(137.5) [ &(120) ,(2) ~l(2) ]
```

Phyllotaxy

- Alternate phyllotaxy



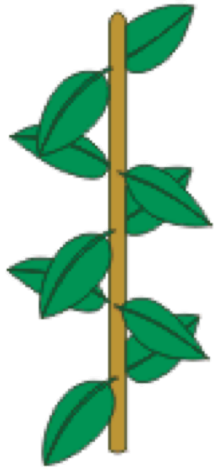


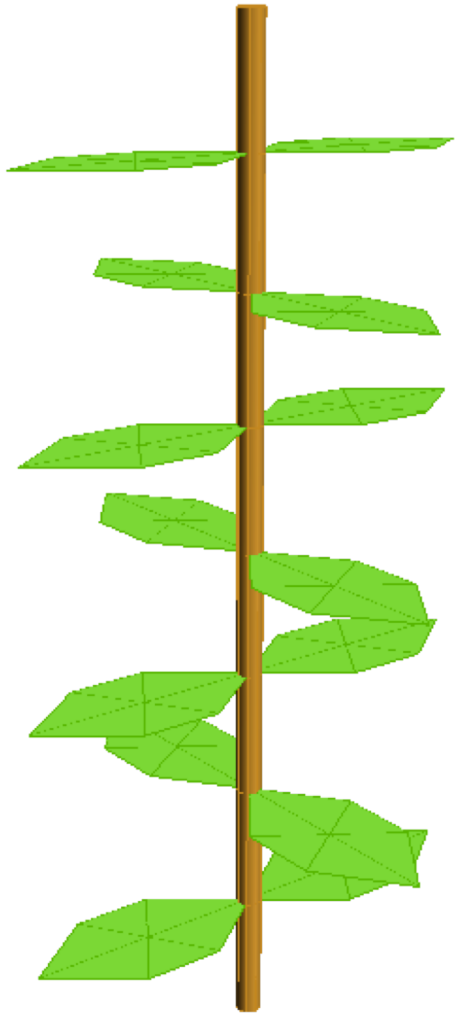
```
Axiom:
  for i in range(7):
    nproduce F(1) /(180) [ &(90) ,(2) ~l(2) ]
```

More details on ‘ArchiModelling.ipynb’

Phyllotaxy

- Opposite decussate phyllotaxy

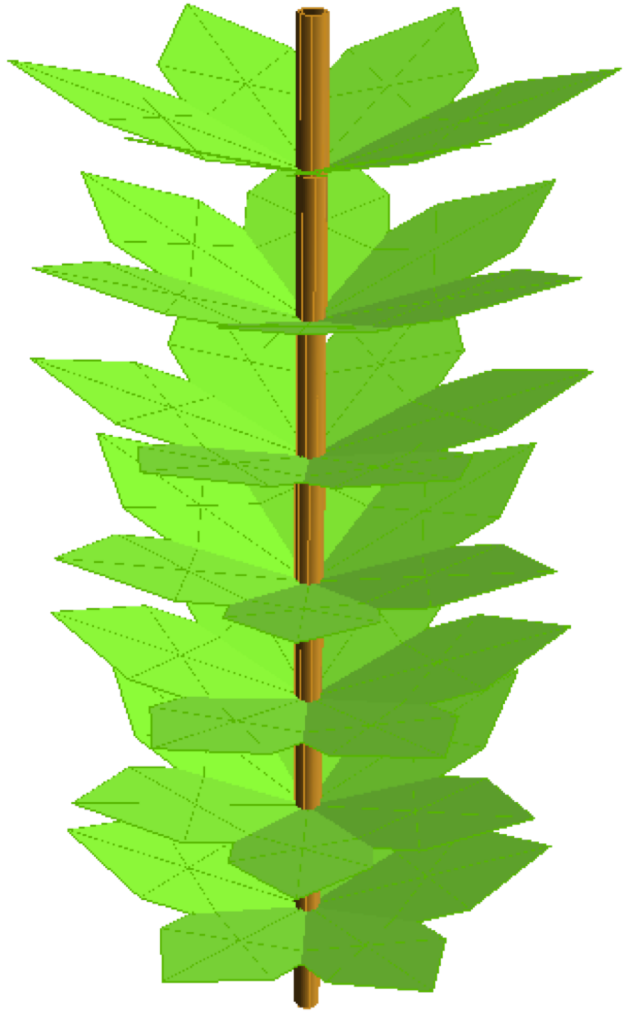




```
Axiom:
  for i in range(7):
    nproduce F(1) /(90) [ &(90) ,(2) ~l(2) ]
    nproduce [ /(180) &(90) ,(2) ~l(2) ]
```

Phyllotaxy

- Whorled phyllotaxy



```
Axiom:
   for i in range(7):
    nproduce F(1) /(30)
    for i in range(6):
       nproduce /(60)[&(70) ,(2) ~l(2) ]
```

More details on ‘Apple Tree Simulation - Organs.ipynb’

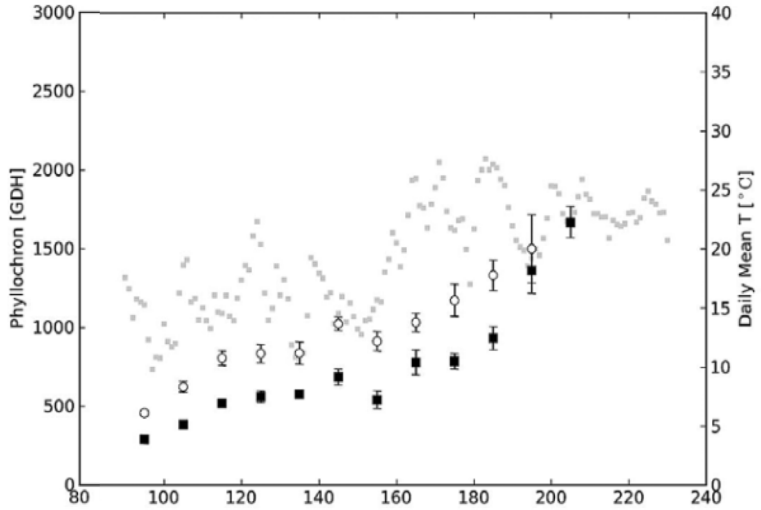
Exercise : Generating the development of an apple tree shoot

- Apple trees have alternate distichous phyllotaxy 2/5 meaning an angle of 144°

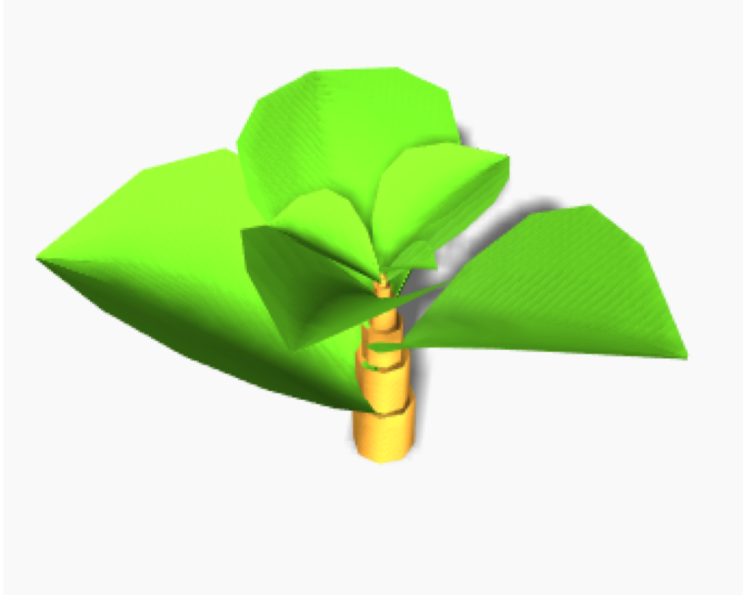
- Phyllochron:
  - Preformed metamers : burst with the shoot
  - Neoformed metamers : depend on the temperature

- Modelling Phyllochron with thermal time
  - Study the **MetamerProduction** structure defined in the notebook *'Apple Tree Simulation - Organs.ipynb’.*

  - The function *accumulate* is use to compute the sum of thermal time and return the number of metamer to produce



Phyllochron of peach tree, Davidson et al., 2017



More details on ‘Apple Tree Simulation - Organs.ipynb’

Exercise : Generating the development of an appletree axis

- Edit the notebook ‘Apple Tree Simulation - Organs.ipynb’ or open model axis.lpy
- Complete the rule for meristem.

```
Meristem(p):
  # Retrieve current nb of metamers produce by the meristem (p.metamer)
  # accumulate the temperature and retrieve the nb of new metamers to produce
  previousnbmetamer = p.nbmetamer
  nbnewmetamer = p.accumulate(cdate)
  for i in range(nbnewmetamer):
    # produce an Internode whose finallength is given by the internode_length function,
length and width and ttime set to 0
    # produce symbol for phyllotaxy (144 for the apple tree)
    # produce lateraly (with a branching angle of 60) a leaf whose finalarea is given by
function leaf_area, and area and width and ttime set to 0
    pass
  produce Meristem(p)
```

More details on ‘Apple Tree Simulation - Organs.ipynb’

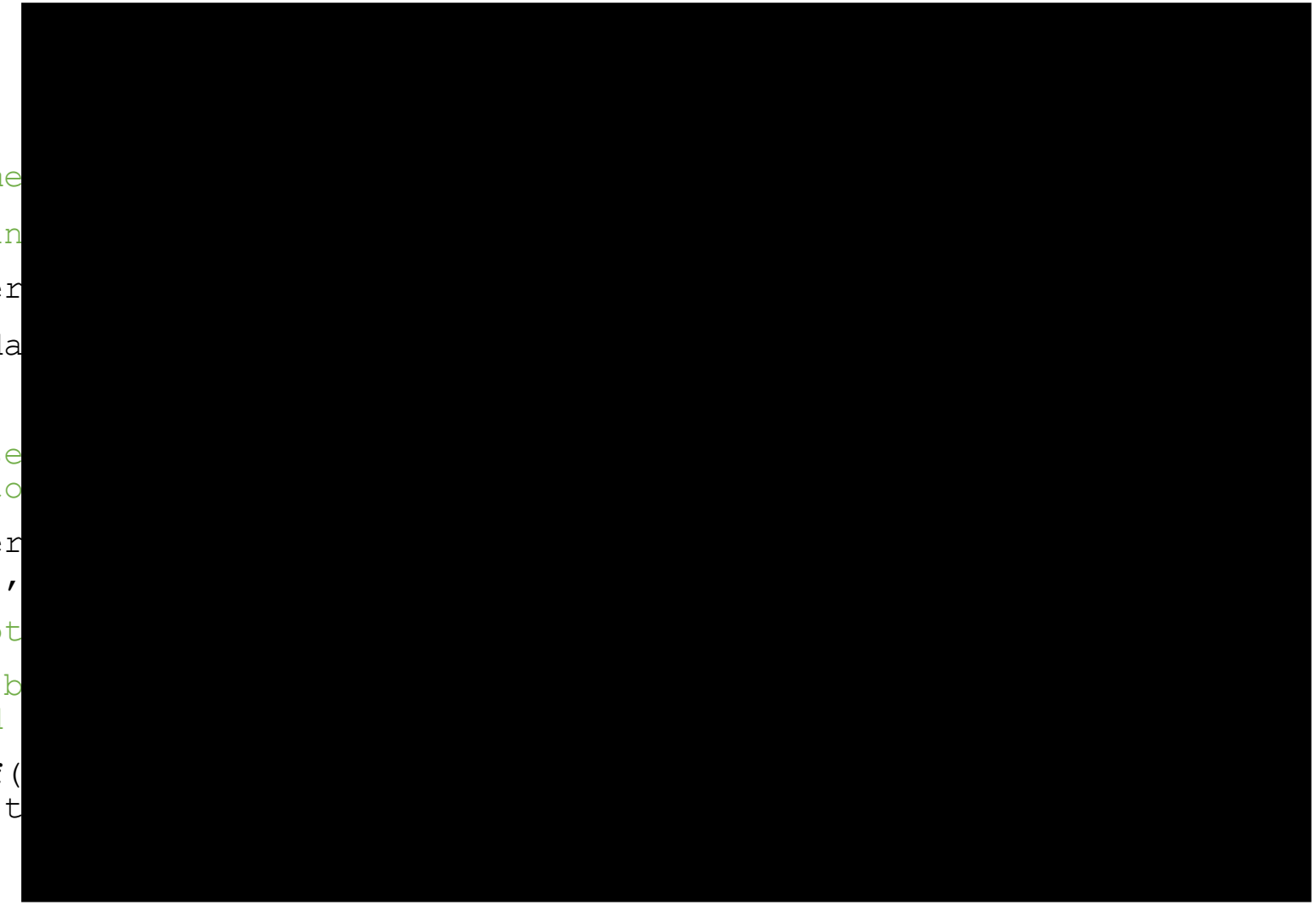
Exercise : Generating the development of an appletree axis

```
Meristem(p):
  # Retrieve current nb of metamers produce by the meristem (p.metamer)
  # accumulate the temperature and retrieve the nb of new metamers to produce
  previousnbmetamer = p.nbmetamer
  nbnewmetamer = p.accumulate(cdate)
  for i in range(nbnewmetamer):
     # produce an Internode whose finallength is given by the internode_length function,
length and width and ttime set to 0
     nproduce Internode(ParameterSet(length=0.01, finallength = internode_length(rank =
previousnbmetamer+i), width=0.01, ttime = 0))
     # produce symbol for phyllotaxy (144 for the apple tree)
     # produce lateraly (with a branching angle of 60) a leaf whose finalarea is given by
function leaf_area, and area and ttime set to 0
     nproduce /(144) [&(60) Leaf(ParameterSet(area=0, finalarea = leaf_area(rank =
previousnbmetamer+i), area = 0, ttime = 0)) ]
  produce Meristem(p)
```

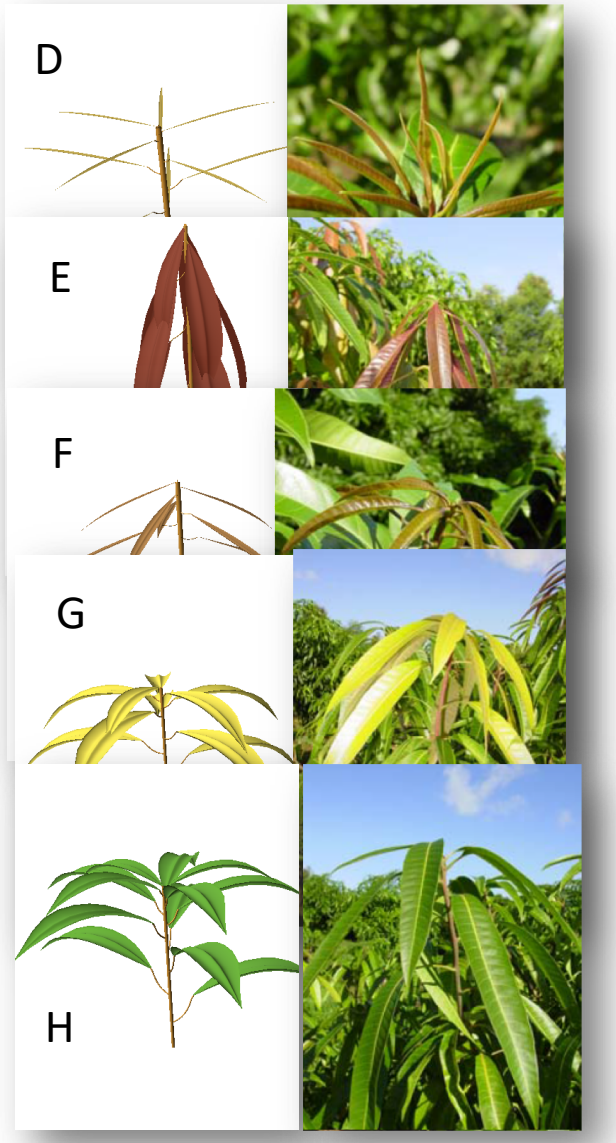
More details on ‘Apple Tree Simulation - Organs.ipynb’

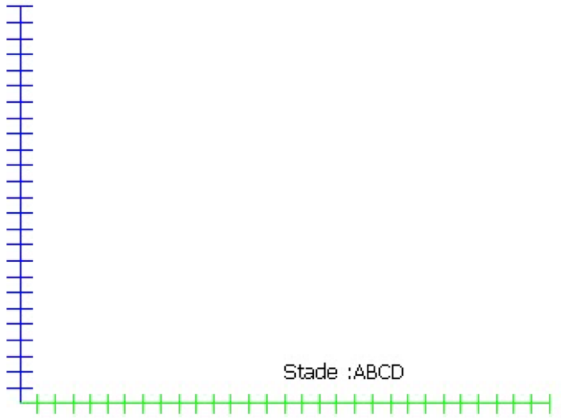
Exercise : Generating the development of an appletree axis

```
Meristem(p):
  # Retrieve current nb of metame
  # accumulate the temperature an
  previousnbmetamer = p.nbmetamer
  nbnewmetamer = p.accumulate(cda
  for i in range(nbnewmetamer):
    # produce an Internode whose
length and width and ttime set to
    nproduce Internode(Parameter
previousnbmetamer+i), width=0.01,
    # produce symbol for phyllot
    # produce lateraly (with a b
function leaf_area, and area and
    nproduce /(144) [&(60) Leaf(
previousnbmetamer+i), area = 0, t
  produce Meristem(p)
```



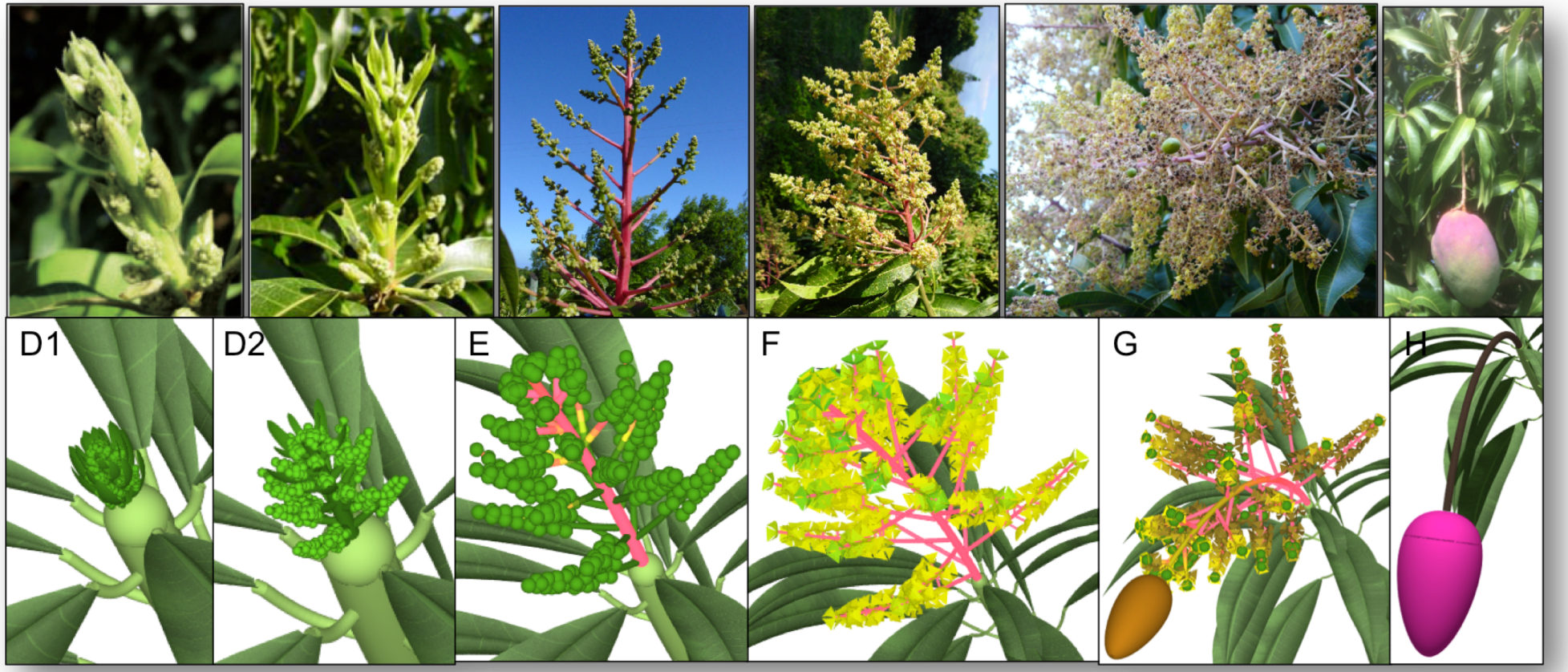
Mango Growth Unit development





2003-07-02 00:00:00

Mango Inflorescence development

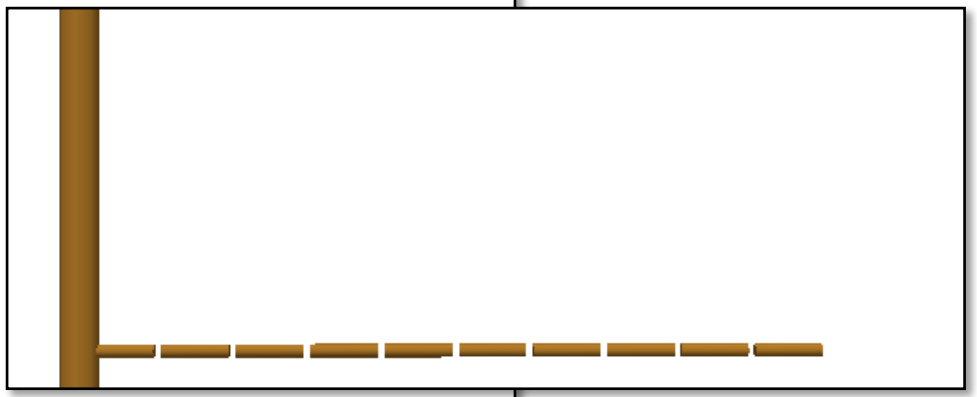


Modelling branch shape

Branch Geometry

- Geometrical embedding of a branch may be complex

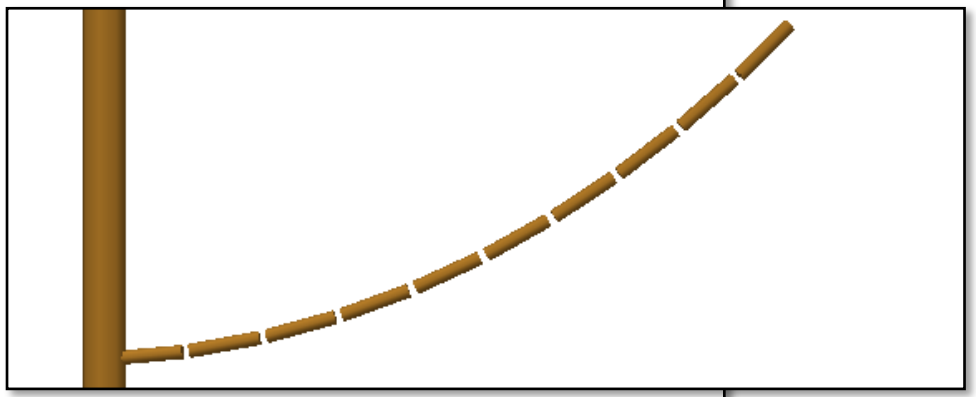
```
length = 10
Axiom: Branch(length)
derivation length: 1
production:



interpretation:
Branch(l) :
   for i in range(l):
       nproduce f(0.1) F(1)
endlsystem
```



Branch Geometry

- Use of tropism to change orientation of branch

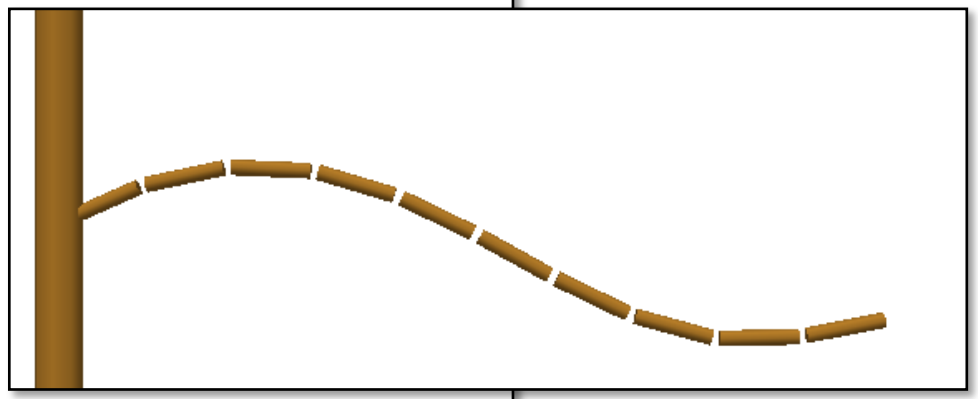
```
length = 10
Axiom: Elasticity(0.02) @Tp(0,0,1) Branch(length)
derivation length: 1
production:



interpretation:
Branch(l) :
   for i in range(l):
       nproduce f(0.1) F(1)
endlsystem
```

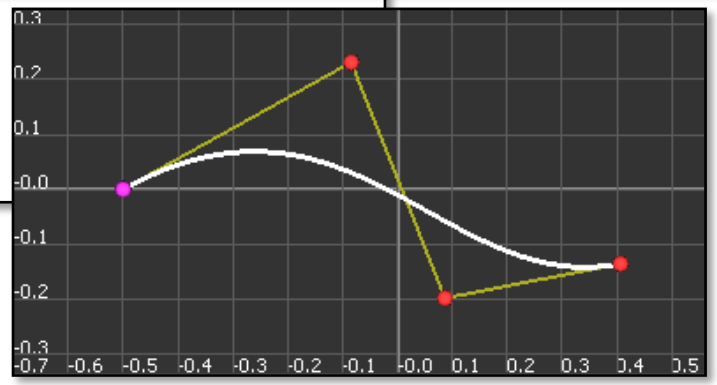


Branch Geometry

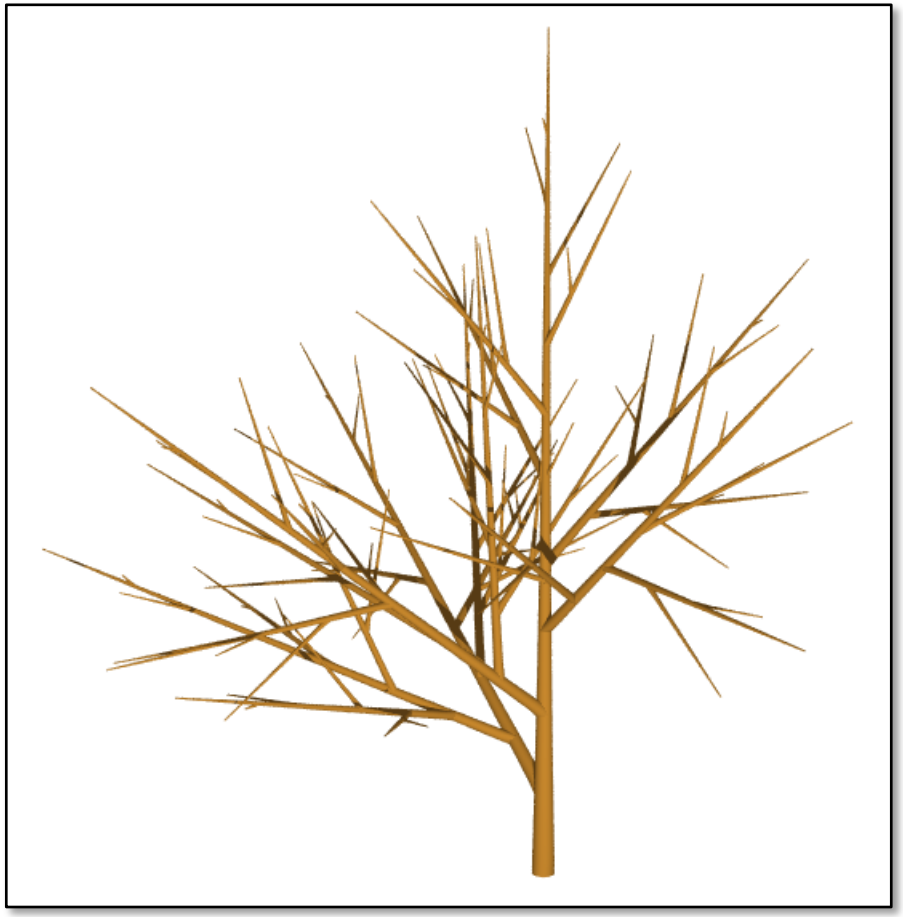
- Use of predefined geometrical embedding

```
length = 10
Axiom: SetGuide(path,length) Branch(length)
derivation length: 1
production:



interpretation:
Branch(l) :
   for i in range(l):
       nproduce f(0.1) F(1)
endlsystem
```



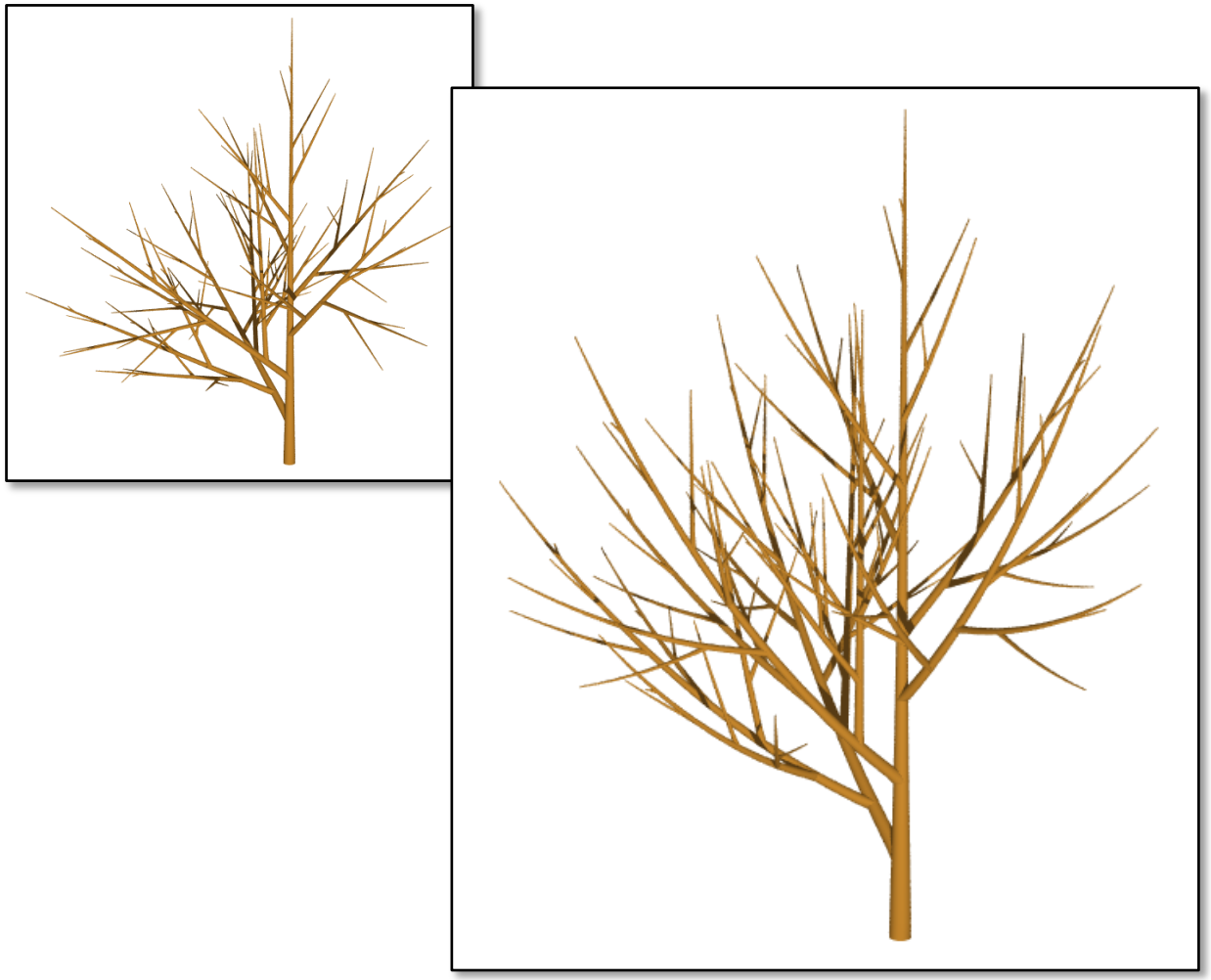


Example on a simple tree structure



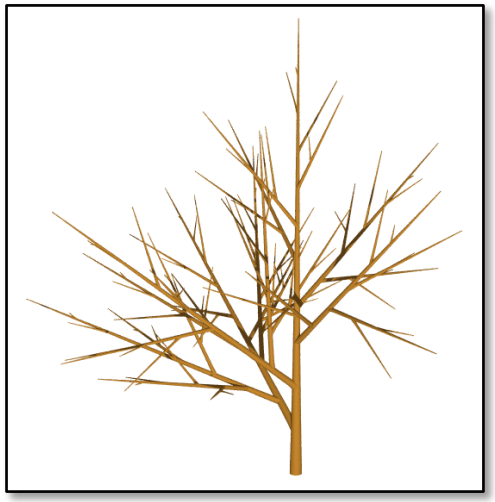
Simple recursive structure

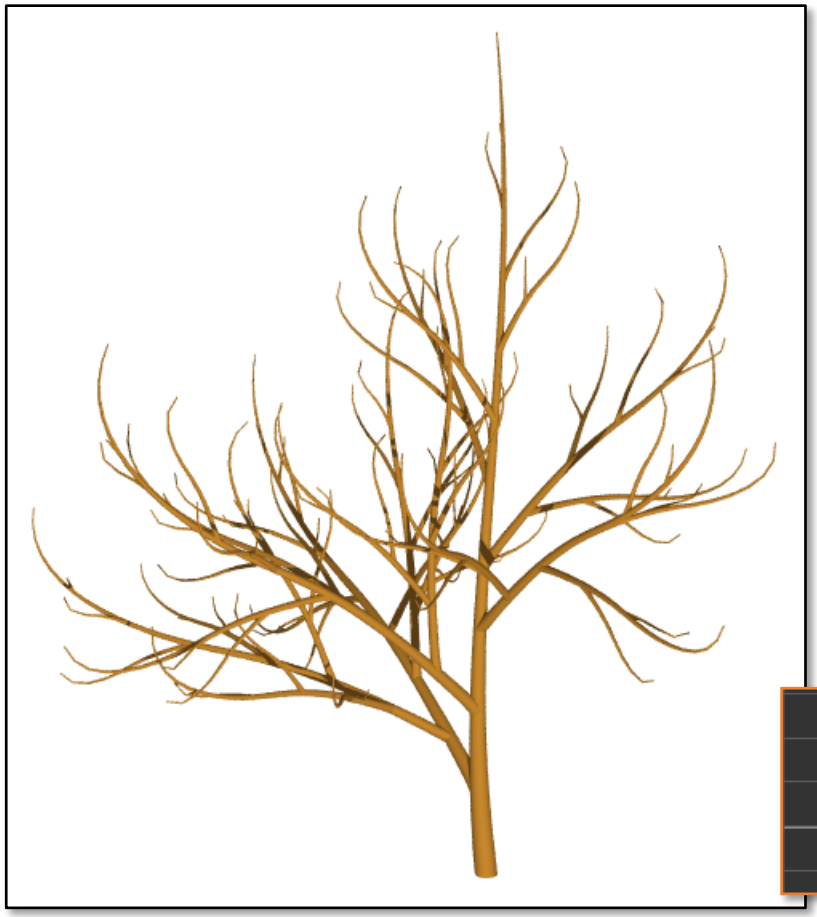
Example on a simple tree structure

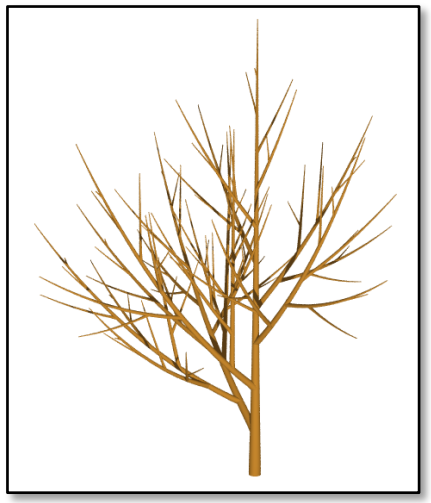


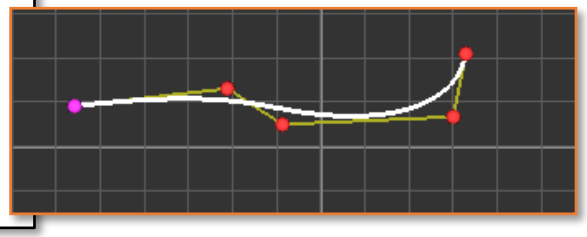
Using tropism

Example on a simple tree structure





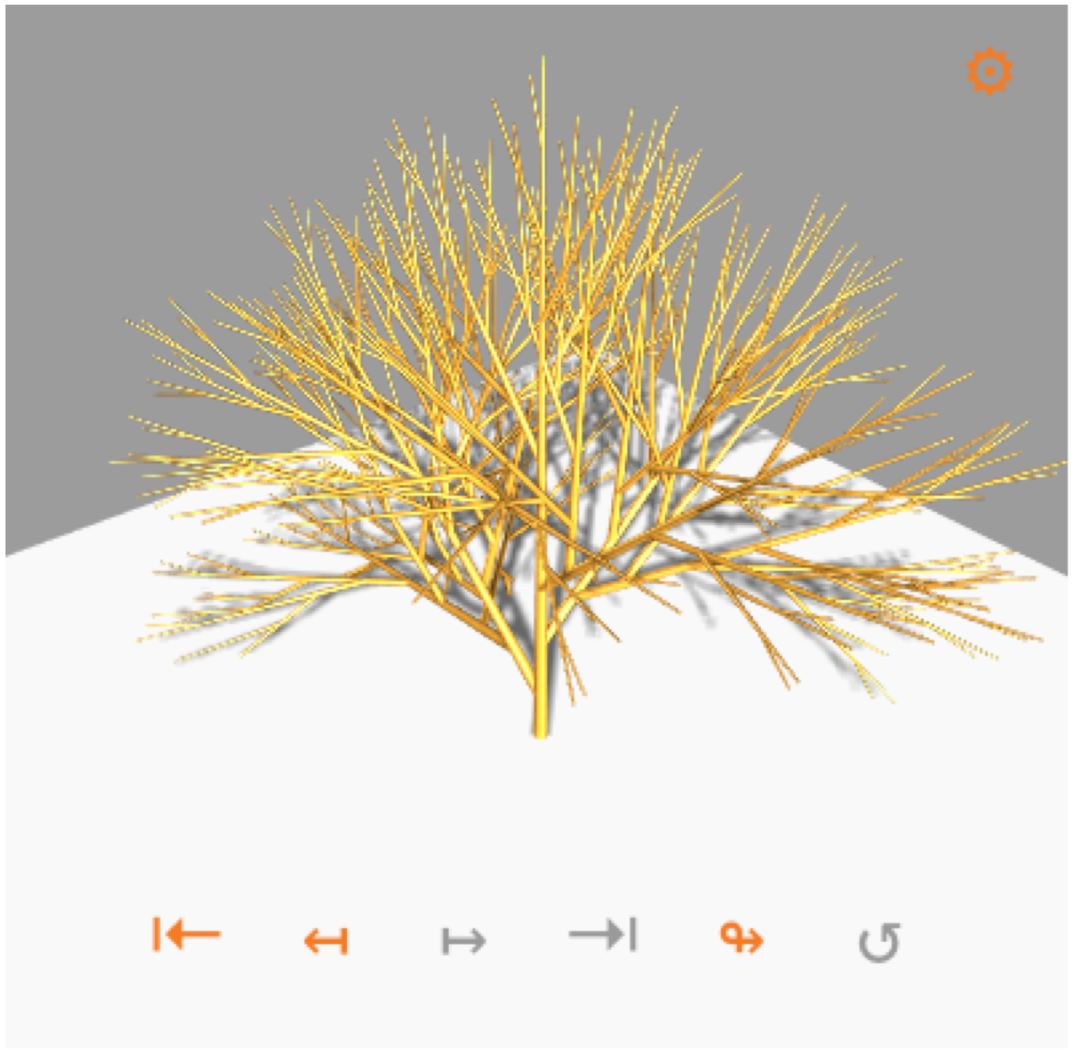




Using a predefined branch path

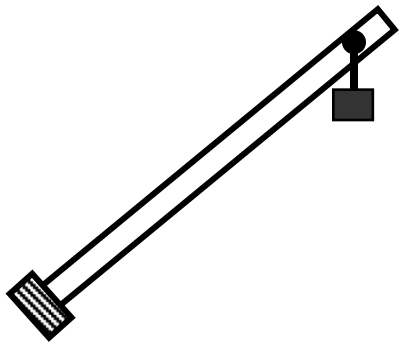
Exercise

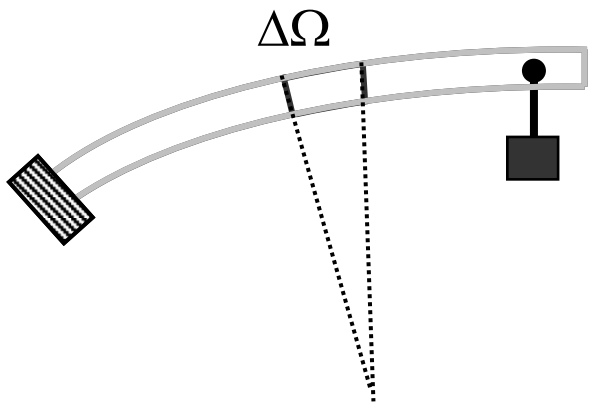
- Test on this simple recursive structure
  - To insert an helio-tropism into the axiom. Test different values of the elasticity. For this use modules **@Tp** and **@Ts**
  - To remove tropism and insert a guide at the beginning of each branch using command **SetGuide**. Edit the guide to achieve curved architecture.

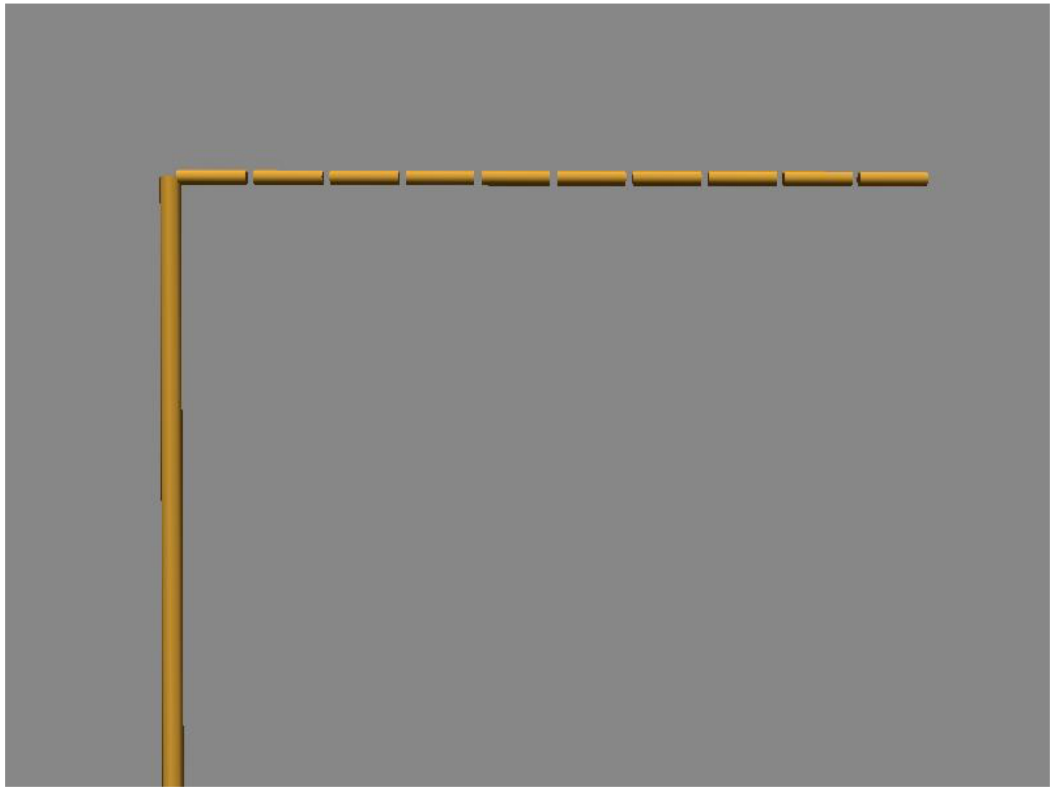


Biomechanical approach

- Consider shoot as a loaded beam







Require convergence

The total torque t is the sum of the torque due to gravity and the torque due to tropism

$$\tau = \tau_g + \tau_t$$

The rate of rotation Ω is

with $$\Omega_\tau = \frac{\tau}{R}$$

where R = E I
E : wood elasticity
I : section inertia

Modelling tree architecture

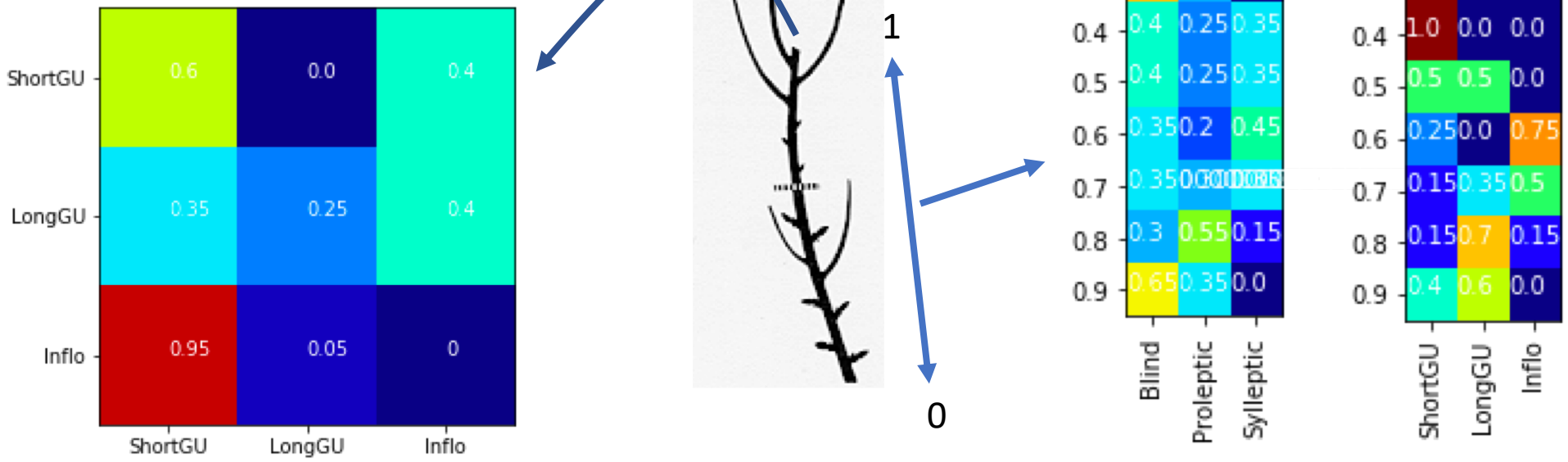
Simulation of apple tree architecture

- Modelling succession and branching of growth units in apple tree.
- Production is decomposed into fate and type.
- Different fates:
  - Proleptic
  - Sylleptic
  - Blind
- Different types :
  - Short GU,
  - Long GU
  - Flowering GU.

Simulation of apple tree architecture

- Probability of branching are defined according to normalized position along GU

- Succession:



Exercise :

More details on ‘Apple Tree Simulation - Architecture.ipynb’

- Edit notebook ‘Apple Tree Simulation – Architecture’ or open model ‘architecture.lpy’
- Change Probabilities of succession and branching

```
Meristem(p):
  ...
for i in range(nbnewmetamer):
  rank = previousnbmetamer+i
  normalizedrank = rank / p.mp.nbfinalmetamer
  nproduce Internode(…)
  nproduce /(144) [ &(60) Leaf(…) ]
  ramiffate, ramiftype = branching(p.gutype,
                                   normalizedrank)
  if ramiffate != Blind:
    nproduce [ … Meristem(init_meristem(ramiffate, ramiftype, … ) ]
  ...
else:
  produce Meristem(init_meristem(Proleptic, succession(p.gutype),
…))
```

Exercise :

More details on ‘Apple Tree Simulation - Architecture.ipynb’

- Edit notebook ‘Apple Tree Simulation – Architecture’ or open model ‘architecture.lpy’
- Add rule for flowering GU that always create a lateral short GU and a Flower in terminal position.

```
Meristem(p):
  ...
  if p.mp.nbmetamer < p.mp.nbfinalmetamer:
    # meristem need to still generate metamer
    nproduce Meristem(p)
  else:
    # succession
    if p.gutype == Inflo:
      # to be completed
      # generate a lateral sylleptic GU
      # generate a flower
    else:
      if p.gutype == TrunkGU: nproduce @Tp(0,0,-1) @Ts(0.005)
      produce Meristem(init_meristem(Proleptic, succession(p.gutype), p.init_date,
p.mp.nbfinalmetamer))
```

Exercise :

More details on ‘Apple Tree Simulation - Architecture.ipynb’

- Edit notebook ‘Apple Tree Simulation – Architecture’ or open model ‘architecture.lpy’

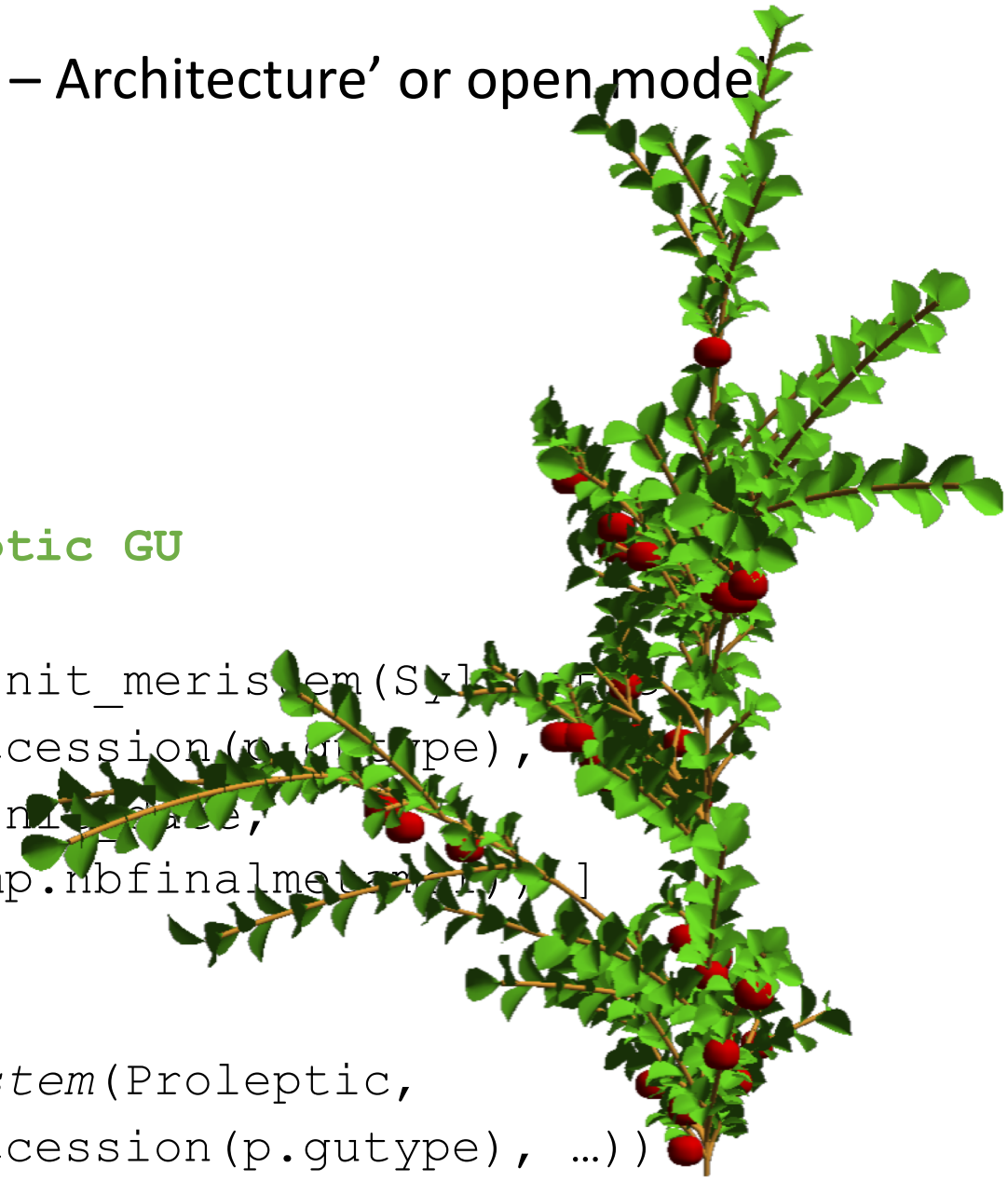
```
Meristem(p):
  ...
  # succession
    if p.gutype == Inflo:
      # generate a lateral sylleptic GU
      # generate a flower
      nproduce [ &(15) Meristem(init_meristem(Sylleptic,
                                succession(p.gutype),
                                p.init_date,
                                p.mp.nbfinalmetamer)) ]
      produce Flower(0)
    else:
      produce Meristem(init_meristem(Proleptic,
                                succession(p.gutype), …))
```

Exercise :

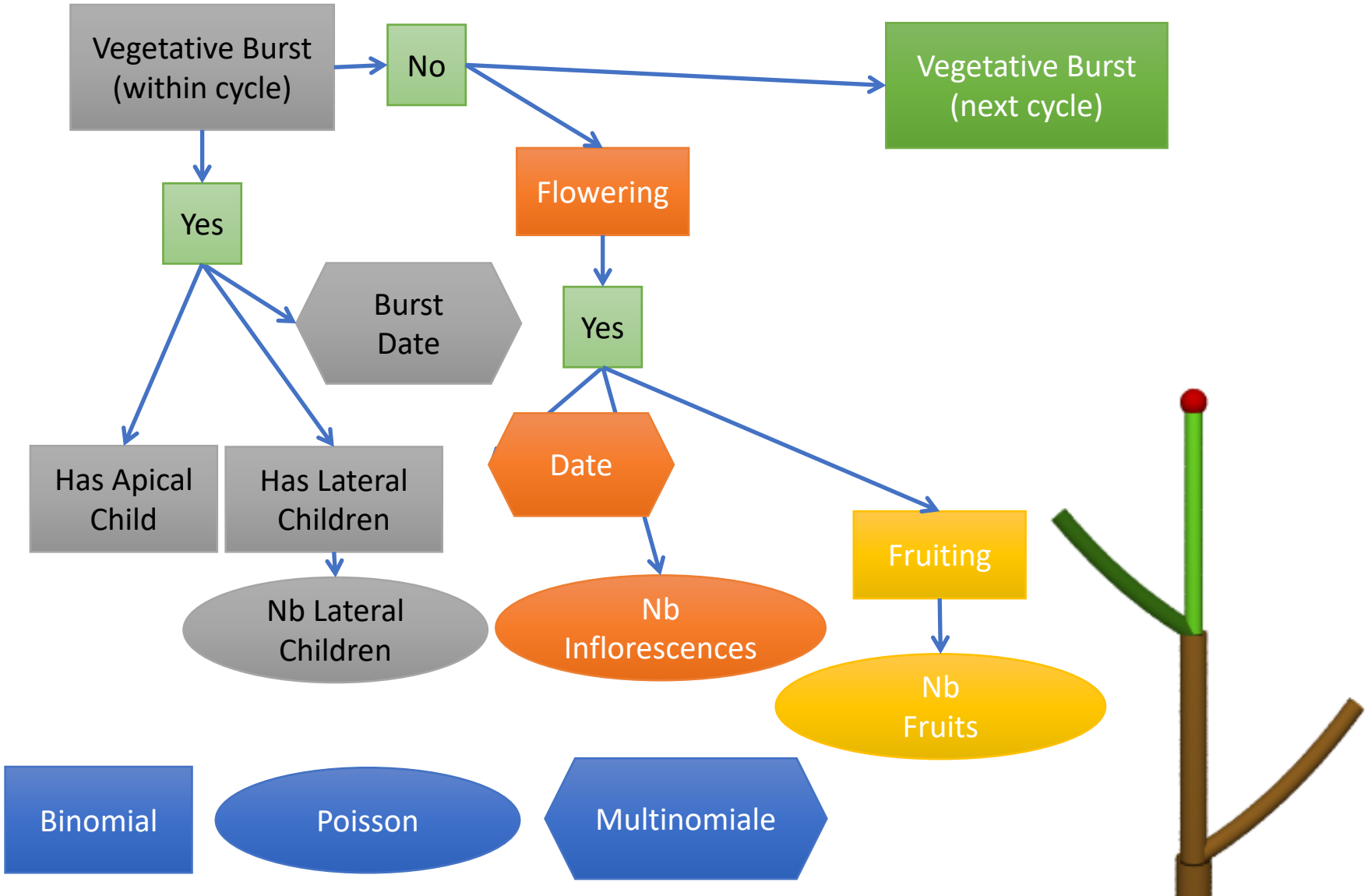
More details on ‘Apple Tree Simulation - Architecture.ipynb’

- Edit notebook ‘Apple Tree Simulation – Architecture’ or open model ‘architecture.lpy’

```
Meristem(p):
  ...
  # succession
    if p.gutype == Inflo:
      # generate a lateral sylleptic GU
      # generate a flower
      nproduce [ &(15) Meristem(init_meristem(Sy
                                succession(p.gutype),
                                p.in
                                p.mp.nbfinalmeta
      produce Flower(0)
    else:
      produce Meristem(init_meristem(Proleptic,
                                succession(p.gutype), …))
```



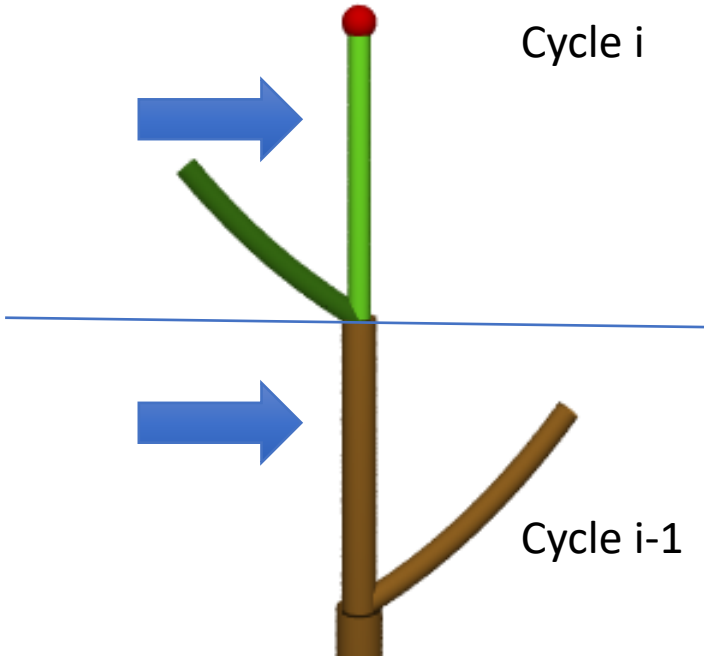
Simulation of mango architecture



Structural and temporal factors

- Mother Position : Apical / Lateral
- Mother Burst Date : Month of vegetative cycle
- Ancestor Position : Apical / Lateral
- Ancestor Fate : Vegetative / Flowering

(Dambreville et al., J. Ex.Bot., 2013)

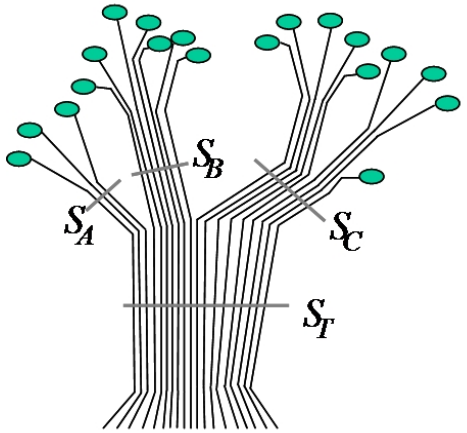


Visual result

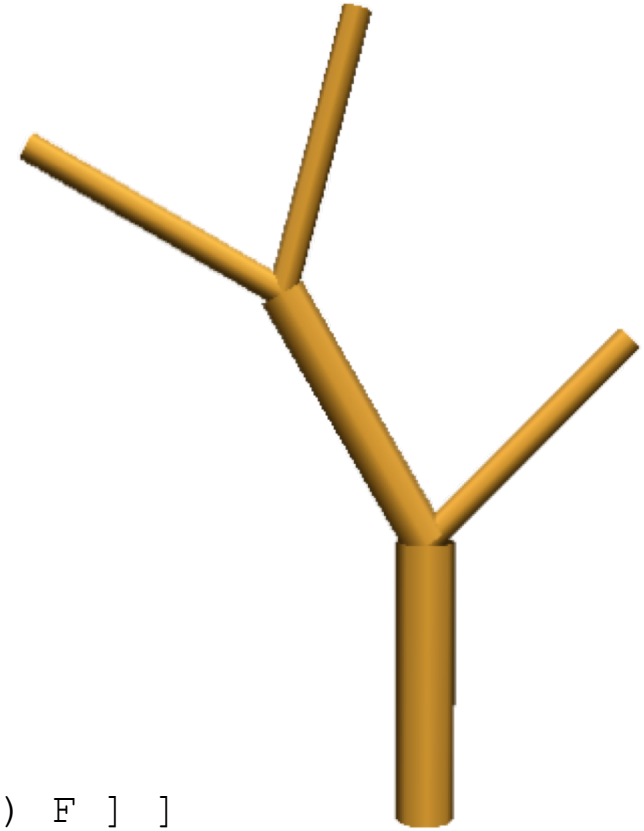
Modelling secondary growth

Diameter of branches

- Da Vinci formula



$$r_p^e = r_a^e + r_b^e$$



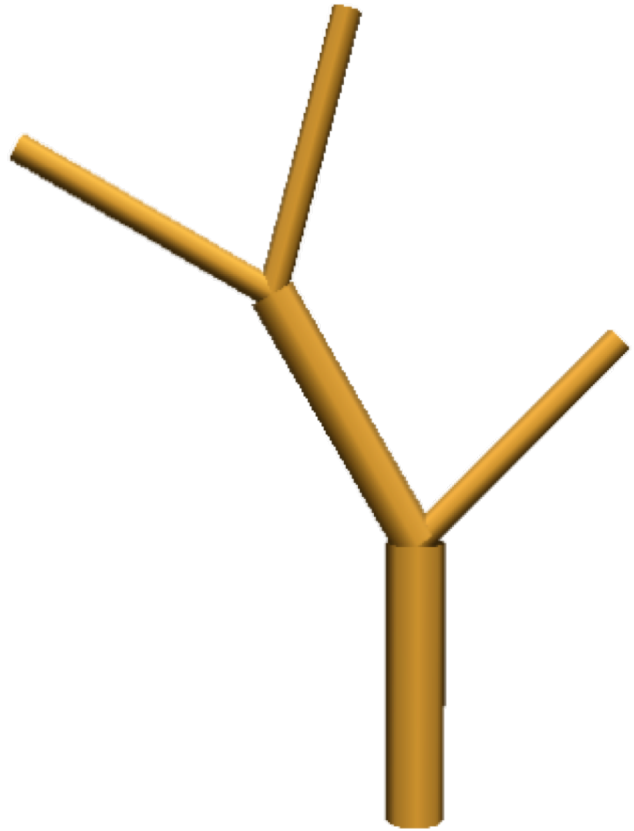
```
r1 = 0.05
e = 1.5
rp1 = pow(2 * pow(r1,e) + pow(r1,e), 1./e)
rp0 = pow(pow(rp1,e) + pow(r1,e), 1./e)

Axiom: _(rp0) F
   [ +(30) _(rp1) F [ +(30) _(r1) F ] [ -(45) _(r1) F ] ]
   [ -(45) _(r1) F ]
```

More details on ‘ArchiModelling.ipynb’

Diameter of branches

- Using contextual rules

```
r1 = 0.05
e = 1.5
backward()
def piperadius(ris):
        return pow(sum([pow(ri,e) for ri in ris]),1./e)

Axiom: _(r1) I(r1) [+(30)I(r1) [+(30)I(r1) ] [-(45)I(r1) ] ] [-
(45)I(r1) ]
production:
ignore: +-
I(r) >> [ I(r1) ] [ I(r2) ] --> I(piperadius([r1,r2]))

interpretation:
I(r) --> _(r) F
```



Test the pipe model on the apple tree

- Radius is estimated directly on the number of descendants.

$$r_i = r_0 . n_i^p$$

With n_i the number of descendants of internode *i*, r_0 the radius of the extremities and *p* the pipe exponent.

Exercice: Play with exponent and terminal radius on the model.

Modelling Mortality

Deletion of modules

- Rule of deletion should be explicit:

```
A(t) →  *
A(t) : produce
```

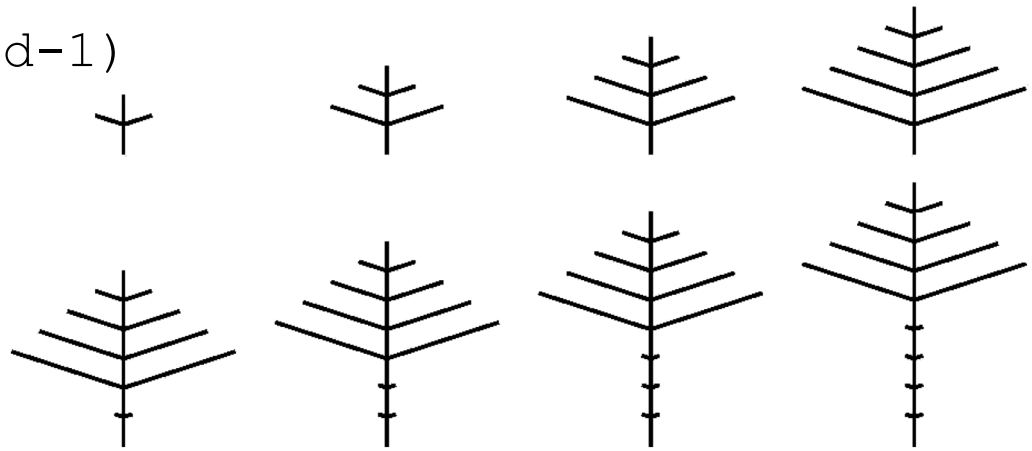
- For an entire branching system, one can use **%**

```
a[b%[cd]e[%f]]g[h[%i]j]k ==> a[b]g[h[]j]k
```

Deletion of modules

- Example

```
     Axiom : A
A --> F(1)[-X(3)B][+X(3)B]A
B --> F(1) B
X(0) --> F(0.3)%
X(d) --> X(d-1)
```



Basitonic structure with shedding

Exercise

- Into model on apple tree architecture, discuss group 3.

```
def StartEach():
  global cdate
  if cdate.month == 12:
    useGroup(3)
  else:
   useGroup(0)

group 3:
Leaf(p) --> %
Fruit --> %
```

Conclusion

- Example of a step by step build of a apple tree models
  - Dependent of various processes and environmental factors
    - See next classes for more details
  - Can be adapted to other fruit tress
    - Project assignment