

Capsis exercises

February 2019

François de Coligny, Nicolas Beudez

1. Create a new module called *Training* in Capsis

- Go to the Capsis installation directory (that is the `capsis4` directory). **All commands below must be typed from this installation directory.**
- In a terminal, adapt and type this command...

For Windows:
`ant create-module -Dname=training -Dprefix=Tra -Dauthor=F_de_Coligny -Dinstitute=INRA`

For Linux/MacOS:
`sh ant create-module -Dname=training -Dprefix=Tra -Dauthor=F_de_Coligny -Dinstitute=INRA`
- You can verify that the `etc/capsis.models` file contains an entry for the new module (to do it, simply open this file).
- Customize the `idcard.properties` file (type + description) located in `src/training` directory.
- Compile...
For Windows: `ant compile` For Linux/MacOS: `sh ant compile`
- Run Capsis...
For Windows: `capsis` For Linux/MacOS: `sh capsis.sh`
- In Capsis, go to the « Help » menu, then select « About Capsis » and select *Training* in the list. You can see informations about the *Training* module.
- Test the module under Capsis...
 - Project > New, then choose the « Training » module and click on the « Initialize » button
 - Load the input file: `capsis4/data/training/training.inv`
or load a file created during the Java exercises session: `trees.txt`

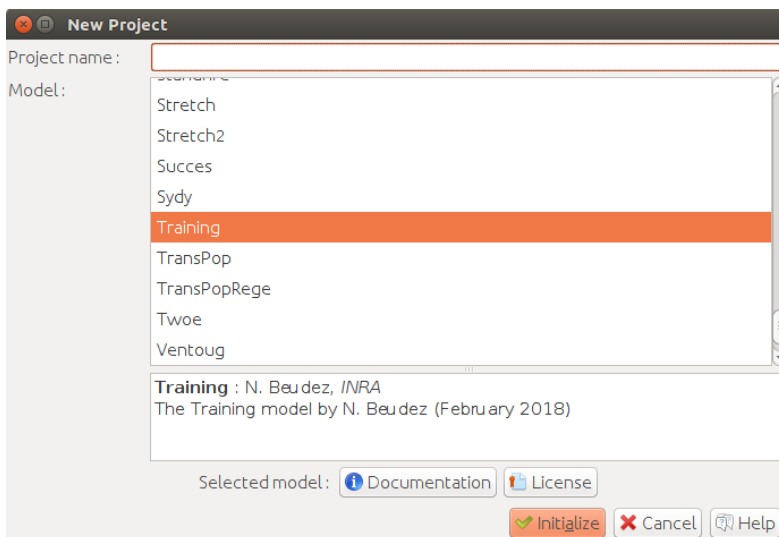


Illustration 1: project creation

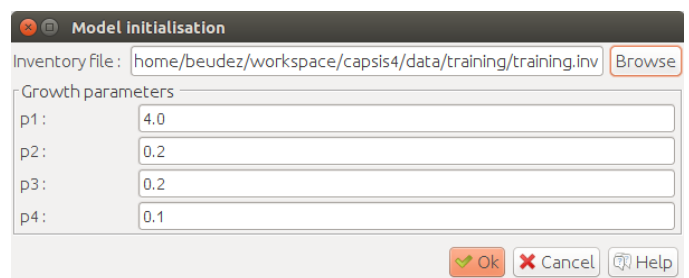


Illustration 2: model initialisation

- Click Ok → the initial scene is built.

- On the left panel, click on Scene Viewer > Simple viewer (double click)

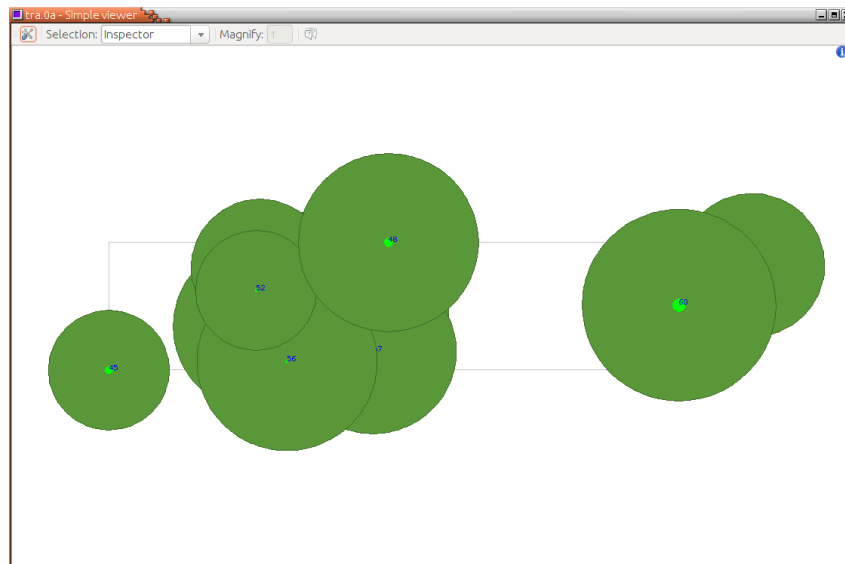


Illustration 3: simple viewer

2. Random regeneration

Add a regeneration method to add new trees each year, located randomly on the terrain.

Notes:

- new trees have $dbh = 3\text{ cm}$ and $height = 1.3\text{ m}$
- number of new trees: draw each year a random number between 0 and $regenerationMax$, where $regenerationMax$ is an integer chosen by the user at the beginning of the simulation
- choose new unique ids for the new trees

Helper:

- add $regenerationMax$ in TraInitialParameters class (located in `src/training/model` directory)
- add a regeneration method in TraModel class (located in `src/training/model` directory)
- draw random numbers with the java.util.Random class
- make a loop to create and add new trees
- you may pick ideas below...

```
import java.util.Random;
import jeeb.lib.util.Vertex3d;

public int regenerationMax;

private Random random;
random = new Random ();

// Call the regeneration method
regeneration (newScene);

int n = getSettings ().regenerationMax;
Vertex3d origin = newScene.getOrigin (); // m
double xSize = newScene.getXSize (); // m
double ySize = newScene.getYSize (); // m
int id = treeIdDispenser.next ();
double x = origin.x + random.nextDouble () * xSize;
TraTree t = new TraTree (id, newScene, age, height, dbh,
    crownBaseHeight, crownRadius, x, y, z);
newScene.addTree (t);
```

- create an initial dialog:

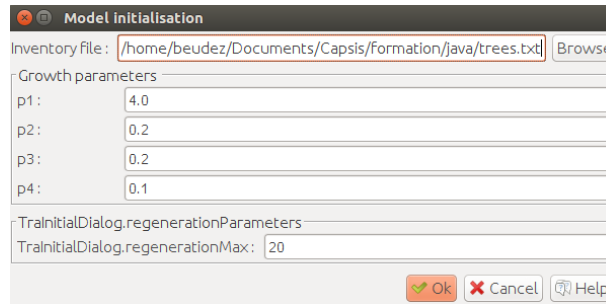


Illustration 4: initialisation of the model

- do an evolution of 20 years:

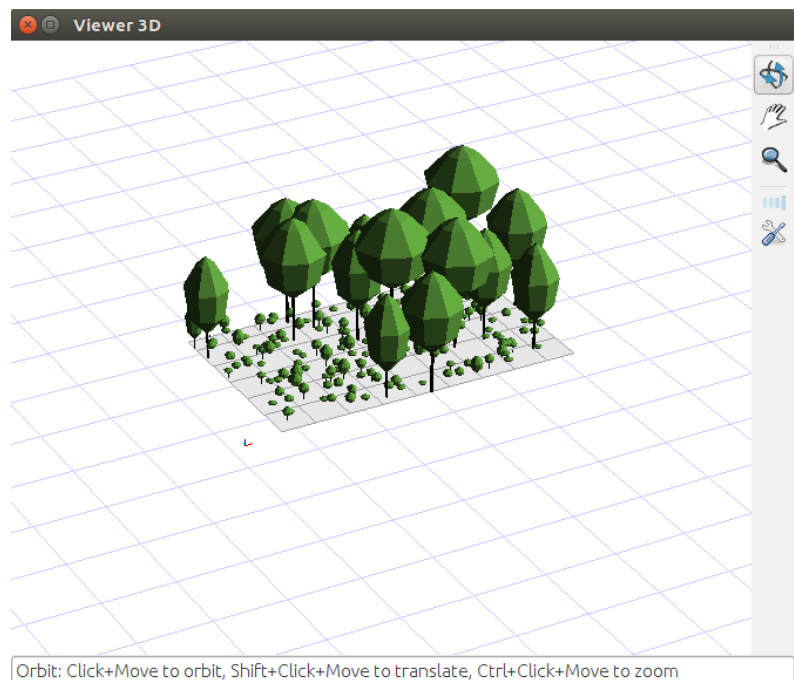


Illustration 5: after an evolution of 20 years

- add translations for the initial dialog:

```
# In training/TraLabels_en.properties
TraInitialDialog.regenerationParameters = Regeneration parameters
TraInitialDialog.regenerationMax = Maximum number of new trees each year

# In training/TraLabels_fr.properties
TraInitialDialog.regenerationParameters = Paramètres de régénération
TraInitialDialog.regenerationMax = Nombre maximum de nouveaux arbres chaque année
```

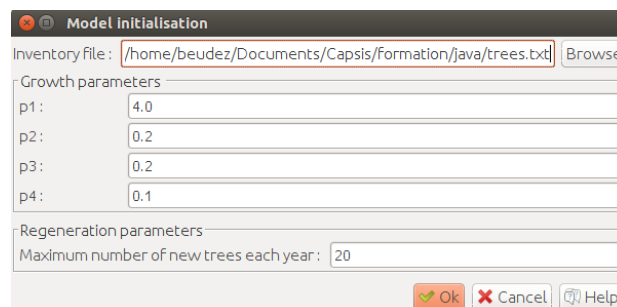


Illustration 6: initialisation of the model

3. Mortality

Add some code in the *Training* module to remove trees, according to a global survival probability of 98 % each year.

Helper:

- trees that are not added in newScene « die »
- you may pick ideas below...

```
double survivalProba = 0.98;  
double proba = random.nextDouble (); // [0,1[  
if (proba > survivalProba) ... // this tree is dead
```

- do an evolution of 15 years and look at the results after 10 years and 15 years:

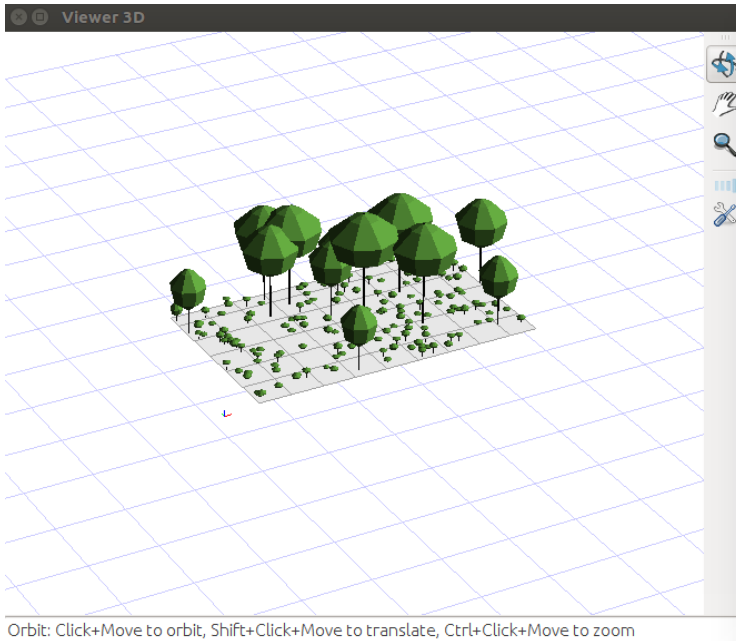


Illustration 7: after an evolution of 10 years

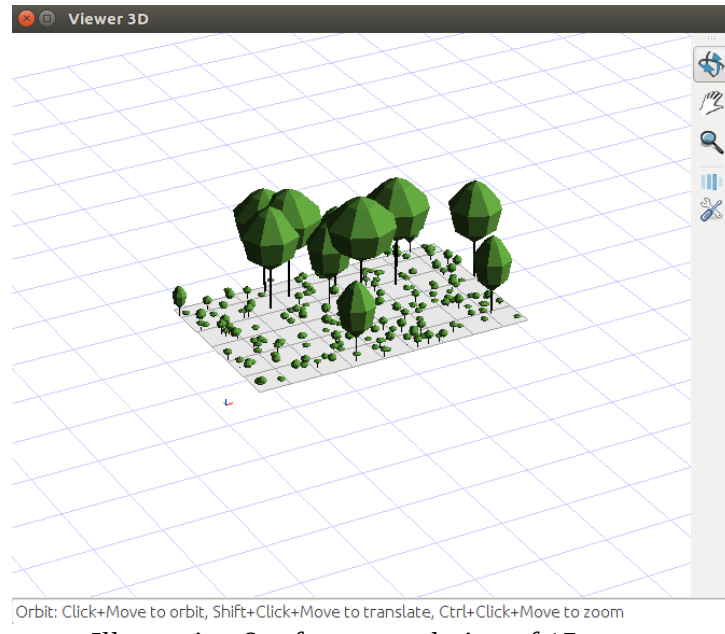


Illustration 8: after an evolution of 15 years

4. Add a geometrical plot made of square cells

Use the default proposals of Capsis to build a rectangular plot made of square cells on the ground.

Helper:

- the rectangular plot must be constructed at the end of the project initialisation
- you may have a look in these classes to check the initialisation process...

```
TraInitialParameters  
TraModel  
TraInventoryLoader
```

- expected result (here 15 cells):

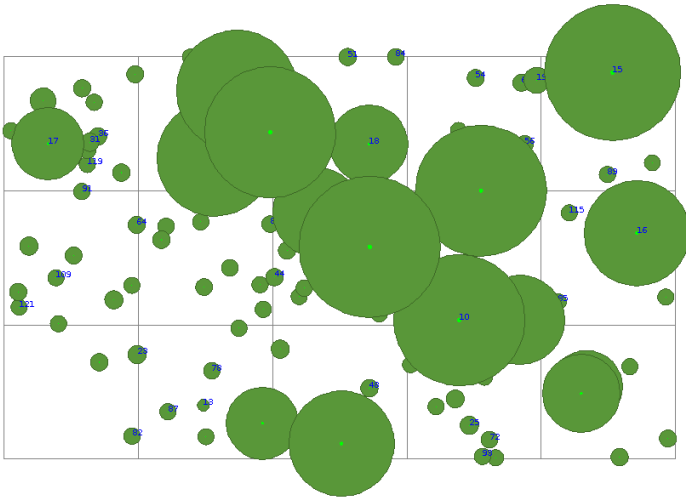


Illustration 9: 15 square cells on the ground

Property	Value
area	100.0
cells	
class	class capsis.defaulttype.plotofcells.SquareCell
elementLevel	true
empty	false
hashCode	50edd92f
holder	⇒ capsis.defaulttype.plotofcells.RectangularPlot (TraScene_10)
iGrid	0
id	1
immutable	⇒ capsis.defaulttype.plotofcells.SquareCell\$Immutable@a869b84
jGrid	0
level	1
mother	False
origin	⇒ (2.864, 19.738, 0)
plot	⇒ capsis.defaulttype.plotofcells.RectangularPlot (TraScene_10)
position	[0, 0]
serialVersionUID	1
shape	⇒ java.awt.geom.Rectangle2D\$Double[x=2.864452838897705,y=19.7378596149826
toString	SquareCell [0, 0]
treeLevel	true
treeNumber	13
trees	⇒ List - 13 items
vertices	⇒ List - 4 items
width	10.0
x	2.864452838897705
xCenter	7.864452838897705
y	19.7378596149826
yCenter	24.7378596149826
z	0.0
zCenter	0.0

Illustration 10: Cell[0,0] contains 13 trees at date 10

5. Make a graph: N / Time

Write adaptations to make the Capsis graph « N / Time » compatible with the *Training* module.

Notes:

- see the *capsis.extension.dataextractor.DETimeN* class
- the *matchWith ()* method tests compatibility
- the referent is a reference to the model class (here *TraModel*)
- what do you have to change to make this chart compatible ?

Helper:

- do not change anything in the chart class (generic)
- the changes must be done in the *Training* module classes
- look at the *TraMethodProvider...*

```
capsis.util.methodprovider.NProvider;
public class TraMethodProvider implements ...NProvider... {

/**
 * Number of trees.
 */
public double getN (GScene stand, Collection trees) {
    if (trees == null) {return -1;}
    return trees.size ();
}
}
```

- expected result:

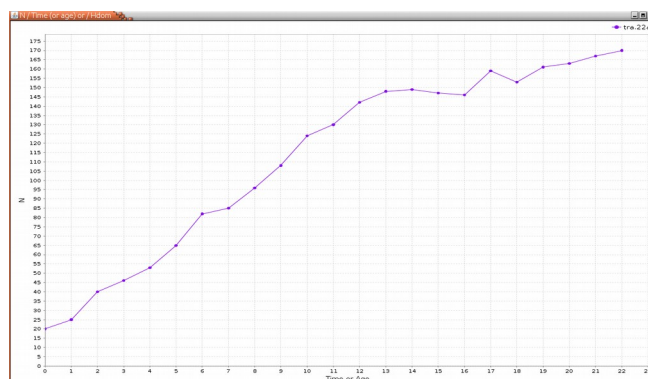


Illustration 11: number of trees over time

6. Script

Adapt the script example in *training/myscripts/* to load your own *treeFile*, set the *regenerationMax* parameter to 5 per year, and change the evolution stage to reach 100 years, performing an intervention every 25 years.

Helper:

- you may pick ideas below...

```
Linux: sh capsis.sh -p script training.myscripts.SimpleScript
Windows: capsis -p script training.myscripts.SimpleScript

import capsis.kernel.extensiontype.*;
import capsis.extension.intervener.*;

i.regenerationMax = 5;
s.init (i);

Step step = s.evolve (new TraEvolutionParameters (25));

// Intervention: cut tree between 5 and 15m height
Intervener thinner = new DHATHinner (DHATHinner.HEIGHT, 5, 15);
step = s.runIntervener (thinner, step);

date = step.getScene ().getDate ();
```

- expected result:

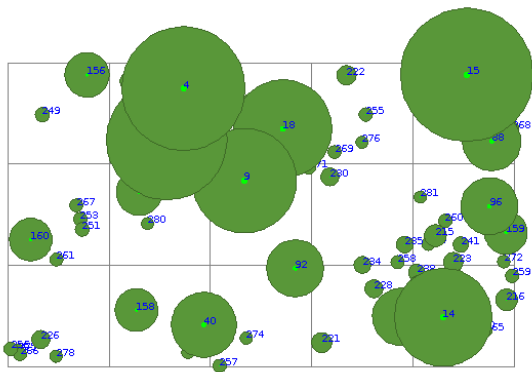


Illustration 12: the final scene at year 100 after the last cut

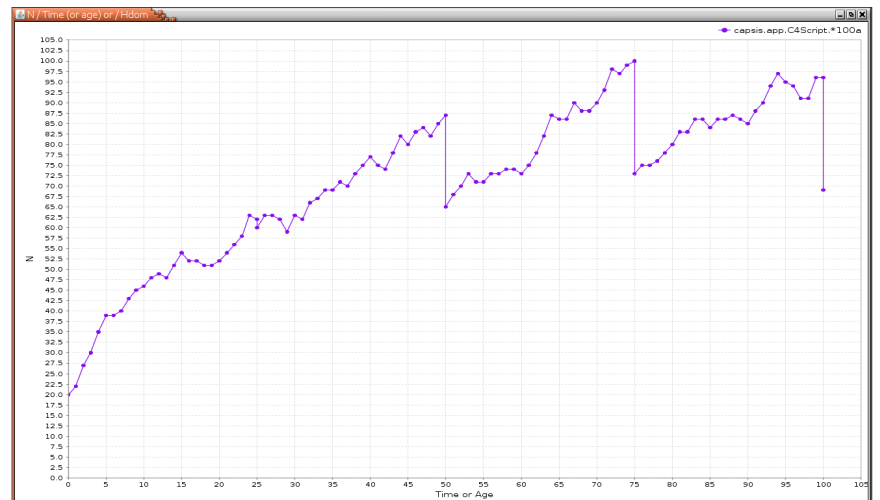


Illustration 13: number of trees over time

7. Regeneration around the mothers

Write an alternative regeneration method: only trees older than 20 years can regenerate, they give birth to at most *regenerationMax* trees and these new trees are located around their mother, at a maximum distance (m) of: $maxDistance = mother\ height / 2$.

Helper:

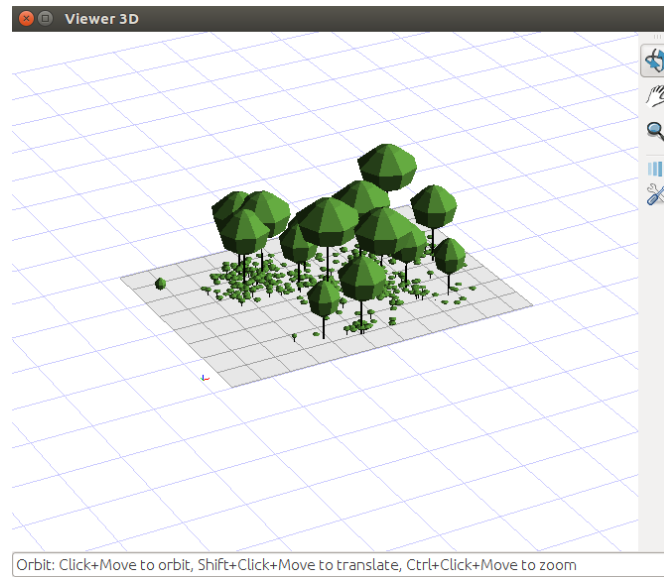
- you may pick ideas below...

```
List copy = new ArrayList (newScene.getTrees ());
for (Object o : copy) {
    TraTree mother = (TraTree) o;

    double maxDistance = mother.getHeight () / 2d;
    double x0 = mother.getX ();

    for (int i = 0; i < n; i++) {
        double distance = random.nextDouble () * maxDistance;
        double alpha = random.nextDouble () * 2 * Math.PI;
        double x = x0 + Math.cos (alpha) * distance;
        newScene.addTree (t) ;...
    }
}
```

- expected result:



*Illustration 14: scene at date 10 with
regenerationMax = 10*