Spatial Simulation

Winter Semester 2023 / 24, MSc Applied Geoinformatics

Obtaining results from simulations

Obtaining and visualising results

Displaying global variables in charts

```
display WaterVolume background: #white {
        chart "Chart Title" type: series {
                data "My global variable" value: my_variable color: #blue ;
        }
}
```

Displaying species / grid variables in charts

```
display WaterVolume background: #white {
        chart "The total volume of water in the watershed" type: series {
                data “water" value: sum(myCA collect each.water_level) color: #blue ;
        }
}
```

Any container: a grid, a species, a list, etc.
Here: myCA

A statement that loops through all entities in the given container

A pseudovariable that represents each of the entities in the container that is collected. Here: each cell in myCA

Types of charts: scatter, series, pie, histogram

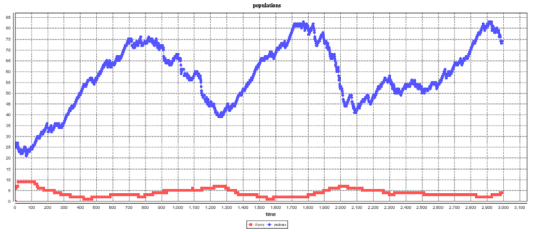
For example a scatterplot:

```
experiment mySimulation type: gui {

   output {
      display  my_scatterplot {
         chart "my chart" type:scatter {
         data “scatter" value: myCA collect [each.water_elev, each.water_level] color:#black;
         }
      }
   }
}
```

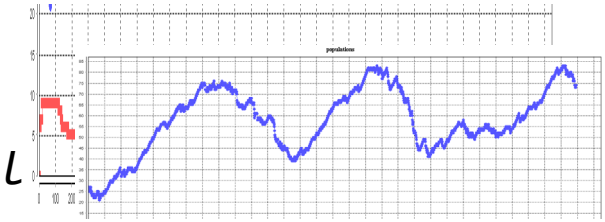
More info about chart types: https://gama-platform.github.io/wiki/DefiningCharts

Chart displays



Either:

```
display TimeSeries background: #white {
        chart "Water flow" type: series {
                data "Some variable" value: my_variable color: #blue ;
                data "Another variable" value: my_2_variable color: #red ;
        }
}
```



Or:

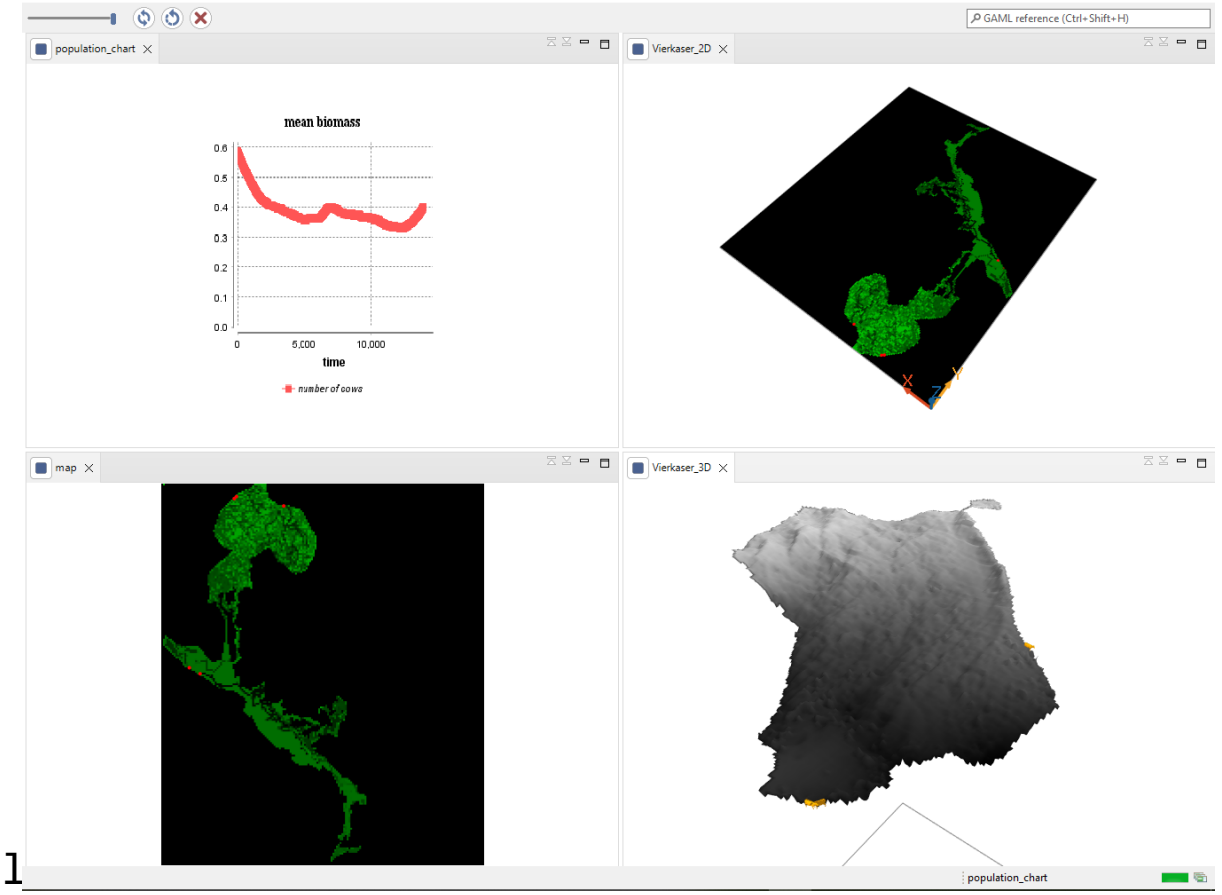
```
display WaterVolume background: #white {
        chart “The total volume of water" type: series {
                data “Some variable" value: my_variable color: #bl
        }
}
display MinElevation  {
        chart “Minimum Elevation" type: series background: #white {
                data "Another variable" value: my_2_variable color: #red ;
        }
}
```

Layouts

```
//Simulation
experiment xxx type:gui {
  output {
    layout #split;
    display myGraph type: opengl {..}
    display myMap type: 2d {..}
    display myMap2 type: opengl {..}
    display myDEM type: opengl {..}
  }
}
```

Possible layout facets:

#stack, #split, #horizontal or #vertical



3D outputs / camera

```
//3d aspect of the cow species
aspect sphere3D{
  pair<float,point> r0 <- -90::{1,0,0};
  pair<float,point> pitch <- 0 ::{1,0,0};
  pair<float,point> roll <- 0::{0,1,0};
  pair<float,point> yaw <- heading::{0,0,1};
  draw obj_file("../includes/cow.obj", rotation_composition(r0,pitch,roll,yaw)) at:{location.x, location.y, dem[location] + 2} size: 20
color: #orange;
}



display Vierkaser_2D type:opengl {
  //2d Viz
  camera 'default' location: {-176.0,-289.0,1330.0} target: {334.0,387.0,0.0};
  species cows aspect:spehere3D;
  mesh biomass color:#green scale:0;
}
```

Saving data to files

Supported files types:

- shp
- asc
- geotiff
- text
- csv

```
save my_variable to: "../results/results.csv" type: "csv" rewrite:false header:true;

save cows to: “cows-out.shp" type: "shp" with: [name::“cowname", location::“xy"] crs:
"EPSG:4326";

save urbanCells to: “city.asc" type: "asc" crs: "EPSG:4326";
```

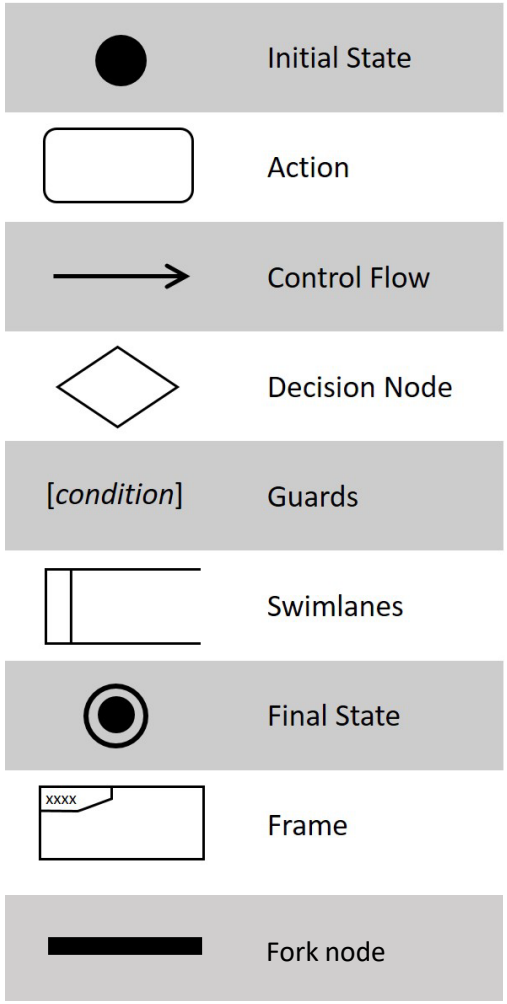
Saving data to files

```
experiment saveData type: gui {
  reflex saveMyData {
    save my_var to: "../results/singlerun.csv" type: "csv" rewrite:false header:true;
  }
}
```

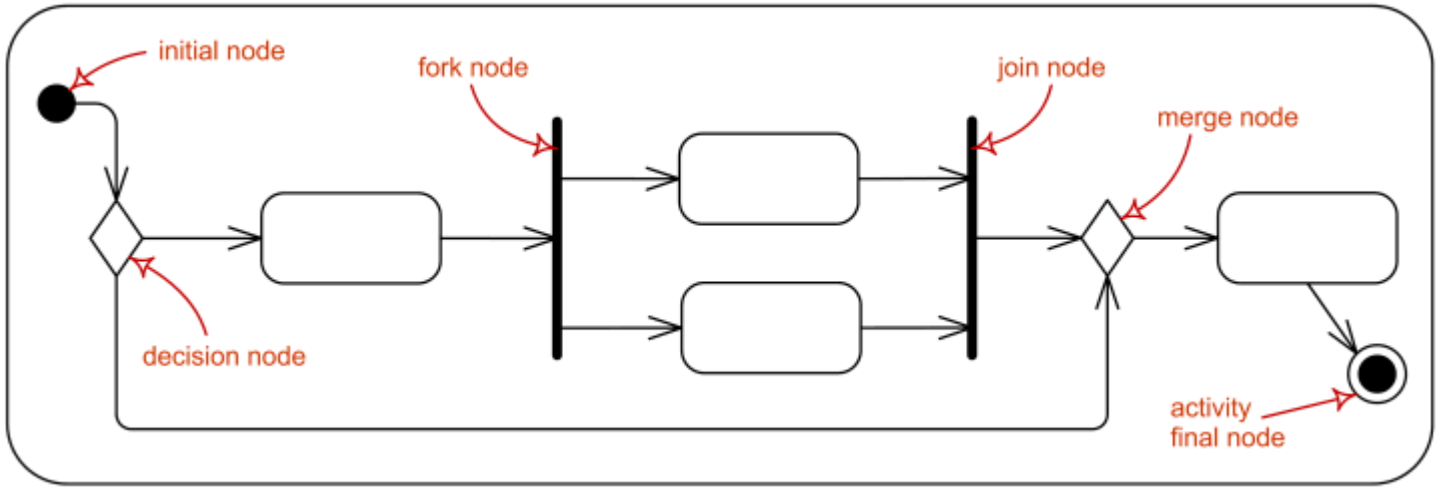
Conceptual model design

with UML

Conceptual model



UML Activity diagram



A nice and visual reference of the UML standard is here:
https://www.uml-diagrams.org/activity-diagrams-reference.html

UML Activity diagram: an example