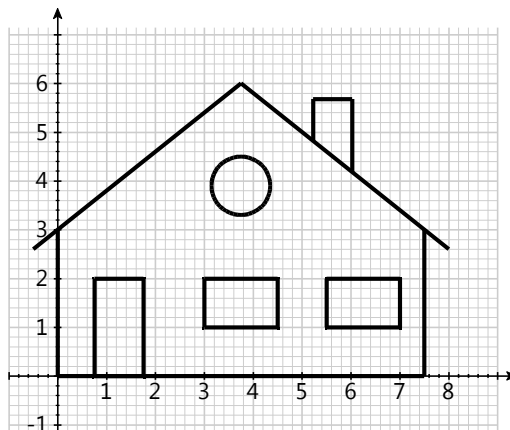


Exercise Sheet 3

House



Preparation

A graphic is to be programmed in this exercise. To do this, first create the following two classes *HouseTVG* and *House*. Start the *House* class. This starts the Physolator and loads the physical system *House* into the Physolator together with the graphic component *HouseTVG*. An empty graphic with a coordinate system appears.

```
import de.physolator.usr.tvg.TVG;

public class HouseTVG extends TVG {

    public HouseTVG() {
        geometry.setUserArea(-1, 9, -1, 7);
        geometry.setRim(30, 30, 30, 30);
        scalesStyle.visible = true;
    }

    public void paint() {
        style.useUCS = true;
        style.strokeWidth = 3;
        // place for your drawing commands
    }
}
```

```
import de.physolator.usr.*;

public class House extends PhysicalSystem {

    public void initGraphicsComponents(GraphicsComponents g) {
        g.addTVG(new HouseTVG());
    }

    public static void main (String args[]) {
        start();
    }
}
```

Explanations

The class *HouseTVG* is a graphic component. The class *House* is a physical system that only serves to load the class *HouseTVG* into the Physolator and display it in the Physolator.


Basic settings for the graphic are made in the constructor of the class *HouseTVG*. The command *geometry.setUserArea(-1,9, -1,7)* specifies that a coordinate system is to be used that extends from -1 to 9 in the x-direction and from -1 to 7 in the y-direction. The command *geometry.setRim (30,30,30,30,30)* specifies that there shall be a border of 30 pixels around the graphic in all four directions. The assignment *scalesStyle.visible=true;* specifies that the coordinate axes shall be displayed and the subsequent assignments specify that grid lines shall be displayed as well.

In this exercises you shall place your drawing commands inside the *paint()*-method of the *HouseTVG* class at the marked position. The two assignments at the beginning of the method determine how the subsequent commands are to work. The assignment *style.useUCS=true;* specifies that the following drawing commands shall use the user coordinate system defined in the constructor using *setUserArea* rather than the pixel coordinate system. The assignment *style.strokeWidth=3;* specifies that the following commands shall use a line width of 3 pixels. The default value for the line width would be 1. Lines with a width of one pixel are harder to see against the background.

Exercise 1

The house in the picture above shall be drawn on the screen. To do this, insert *drawLine* and *drawCircle* commands at the marked point in the paint method.

<code>drawLine(x1,y1,x2,y2);</code>	draws a line from (x1,y1) to (x2,y2)
<code>drawCircle(x,y,r);</code>	draws a circle with a center located at (x,y) and Radius r

Hint: After you have made added your program code to class *HouseTVG*, you can apply the changes by first saving your changes in your editor and then pressing the reload button  in your Physolator framework.

Exercise 2

Now several houses are to be inserted in different places in the drawing. To do this, add a method *drawHouse(dx, dy)* to the class *HouseTVG*. This method is intended to draw a house that has been moved by (dx, dy) in comparison to the original image. The house should therefore be positioned so that the lower left corner of the house is in position (dx, dy).

```
private void drawHouse(double dx, double dy) {
    // place for your drawing commands
}
```

To make enough drawing space available for several houses, change the command `geometry.setUserArea(-1, 9, -1, 7);` to `geometry.setUserArea(-1, 28, -1, 15);`

Exercise 3

The next step is to draw a larger number of houses arranged in a grid: 8 houses horizontally and 6 houses vertically. A total of 48 houses. The grid spacing, i. e. the distance from the lower left corner of a house to the lower left corner of the neighboring house, shall be 9.

Enlarge the drawing area appropriately by adjusting the `geometry.setUserArea(...)` parameter values.

Hint: Use one or more for loops to draw the houses.


Exercise 4

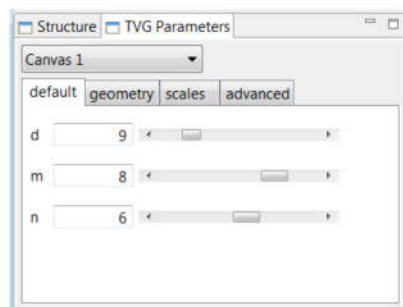
The number of houses in horizontal direction is now marked with m , the number of houses in vertical direction with n and the grid spacing with d . In the previous exercise these variables have had the following values: $m=8$, $n=6$ and $d=9$. To do this, insert the following program code with variable declarations of m , n and d above the paint method.

```
@Parameter
@Slider(min = 7, max = 20, step = 0.1, width = 200)
public double d = 9;

@Parameter
@Slider(min = 1, max = 10, width = 200)
public int m = 8;

@Parameter
@Slider(min = 1, max = 10, width = 200)
public int n = 6;
```

Now inside the `paint` method replace the numbers 8, 6 and 9 with the variables m , n and d , respectively. After reloading the graphic has, move the mouse to the graphic and press the icon . The following dialog appears in which you can change the three variables m , n and d . As soon as the values in this dialog box are changed, the graphic shall change accordingly.



Explanations: The annotations `@Parameter` and `@Slider` are always located before a variable declaration. The Annotation `@Parameter` causes the variable to be displayed in the TVG parameter dialog. The variable's numerical value is displayed in a text field and can be changed by the user. If you want to be able to change the value with a slider, you need an `@Slider`-Annotation in addition to the `@Parameter` annotation. The `@Slider` annotation has four parameters: `min`, `max`, `step` and `width`. The values `min` and `max` define the lower limit and the upper limit, i. e. the values that are reached when the slider is pushed to the left or right. The parameter `step` stands for the basic increment. It describes how much the value shall change when you press the arrow keys at the left and right end of the slider. The width parameter specifies how wide the slider should be on the screen. This specification is made in pixels.