

Laboration 3

Drawing program

Introduction

The purpose of the exercise is that you will learn how window-oriented programs can be written and how objects interact to solve a complicated problem. You will get an occasion to use design patterns and to document your solution using UML diagrams.

Preparations

Put your files in a separate directory and make sure that it can be read only by those in your lab group. If you want the teaching assistants to be able to read your files, give read permissions to the AFS group `oopk_e00:assistenter`.

Load the course module before starting work:

```
module load oopk_e00
```

Copy the make file from the course directory:

```
cp $OOPKHOME/Makefile .
```

You need to edit this file before compiling.

In this exercise, you will use the graphics package Grain. It is documented in a manual page that is printed in the course material. You can also view the manual page on-line by typing `man grain`. If you use the Makefile from the course directory, your code will be linked with the Grain library.

Look at the design patterns Composite och Observer, which are both relevant for this programming exercise. You should draw UML diagrams that describe your program. Therefore, review the lecture notes about UML and the UML Quick reference that can be found in the course material.

The problem

You are to write a simple drawing program. When the program is started, a window is to be displayed on the screen in which the user can draw simple geometric objects like rectangles and ellipses. For the implementation, you *have to* use the graphics package Grain. Your program should have the following properties:

- *One should be able to create objects of different types and arbitrary size by using the mouse.*
The program initially only needs to be able to create rectangles and ellipses, but it should be easy to add new types of objects, e.g., general polygons, with minimal changes to the program code. The user determines what to draw by selecting an appropriate drawing tool.
- *It should be possible to move the objects around on the drawing surface using the mouse.*
Each object has a position on the drawing surface. The user should be able to drag objects around the drawing surface by pressing the right mouse button when the marker is at an object's position.
- *Scrolling of the drawing surface should be possible.*
The user should be able to decide what part of the drawing surface that is visible in the window by dragging the drawing surface (the mouse button is pressed down then the marker is over the background, i.e., not at an object). The effect is approximately the same as if dragging all objects at once. However, the scrolling should only affect the view that is currently selected (see below about multiple views).
- *Grouping of objects should be possible.*
It should be possible to select a number of objects and to create a group of these objects. The group should behave like it was a single object. If, for example, the user drags some object in the group, all of the objects in the group should move together. A group may contain other groups, and it should be possible to ungroup objects that are grouped together.
- *It should be possible to delete objects.*
It should be possible to delete one or several objects. If a group (see above) has been selected, all objects in the group should be deleted.
- *An object may be visible in several views (windows) at the same time.*
If a drawing is large, no window can show all of the drawing at once. Many drawing programs there-

fore allows for creating several *views* of the drawing. Each view is shown in its own window. All views show the same objects, but they can be scrolled individually. This gives the user the opportunity of viewing parts of the drawing that are far apart at the same time. All changes of the drawing surface should immediately be visible in all views that are affected (where the area of change is visible). For example, if the user drags an object, this should be shown in all views in which the object is visible. If the user deletes an object, it should immediately disappear from all views. However, if the user merely scrolls a view, no other views should be affected by this operation. The user should be able to have arbitrary many views open at the same time. It should be possible to close views, and when the last view is closed, the program should terminate.

It should thus be possible to use the mouse to move object, scroll views, and to select objects for grouping and deletion. Apart from these constraints, you can make the user interface as simple as possible. You may, for example, let the user select the drawing tool or operation using key presses (e.g., r=draw rectangles, g=group, s=split, etc.)

For the operations group, split, and delete, the user must be able to select several objects before the operation is committed as follows. The program should keep a list (or similar) *s* of selected objects. *s* may be empty. The objects in *s* are marked on screen for example by drawing them using extra thick lines. If the user clicks the object *x* while holding down the `Shift` key when no drawing tool is selected, *x* should be added to *s*. If the object is already in the list, it should be removed (toggle select). If `Shift` was not pressed, *s* should be flushed before *x* is added (so that *x* becomes the sole marked element). If the user clicks the background, *s* should also be flushed (deselect). The operation *group* should create a group of all objects in *s*. The operation *split* should split the first level of all groups in *s* and ignore other objects. The operation *delete* should remove all objects in *s*.

Draw UML diagrams to document the classes and their relations and interaction. You may draw the diagrams by hand, but they should be readable and follow the UML syntax. There should be both class diagrams and interaction diagrams showing how the drawing program, the geometrical objects, and the views interact when the user drags an object, deletes an object, and creates a group. The diagrams should explain how the contents of the windows is updated.

Hints

Do not try to complete all of the exercise at once. Start with some simple test program to acquaint yourself with the Grain package. Especially make sure that you understand when and how events are reported by Grain and when you may draw in a window.

Then continue with a simplified version of the drawing program, for example, without multiple views and without groups. Think about the interfaces of the classes (rectangle, ellipse, etc.) used by the main program, and how to represent the drawing tools.

Then add grouping. How can you make a group behave like a single geometrical object, say a rectangle? See the design pattern Composite.

Add the rest of the functionality. Spend some time thinking about which objects are responsible for what in the program, especially, what objects are responsible for deleting what other objects, and about where to place information (object positions, the drawing surface's position relative to the view, the list of selected objects, the list of all objects, the list of views, etc.) Check that memory management is satisfactory.

You can simplify debugging by defining a simple file format for a drawing. This saves the work of redrawing whatever is needed to make a bug appear each time you restart the program. File handling capabilities are, however, not required to pass the examination.