Exercises for

Chapter 4: Visualization Pipeline

1 EXERCISE 1

The visualization pipeline offers an intuitive architectural model to design complex data-processing and/or data-visualization applications by combining lower-level functionality in a so-called *dataflow graph*. Think of two visualization applications related to your own professional or daily experience. Describe these applications in terms of a dataflow graph. For each graph node, explain what the functionality of the respective node is, and also the kinds of datasets it reads and writes. Try to be as specific as possible.

2 EXERCISE 2

The visualization pipeline is, often, implemented in software as a set of data-processing modules that are connected in a directed graph (the dataflow graph). Here, each graph node is such a module; and each (directed) edge is the connection of a module's output to another module's input. Can you imagine such a graph which would contain loops (cycles)? If so, sketch a conceptual visualization application represented by such a graph, and explain why a loop would be useful. If not, explain which problems would occur if loops were present in the dataflow graph.

3 EXERCISE 3

The dataflow model used for constructing visualization applications is often supported by so-called *visual builders*, where users can interactively construct a visualization application

by placing modules on a canvas and connecting their inputs and outputs to form a dataflow graph. Describe another application domain (apart from data visualization) where you know that, or imagine that, this kind of visual programming would be an effective paradigm. For that domain, give a few examples of modules by describing their functionality, inputs, and outputs.

4 EXERCISE 4

Visual application builders (VABs) offer an alternative to classical textual programming (CTP) constructing dataflow applications such as those present in visualization contexts. However, there are also contexts in which VABs are less effective and/or efficient to use than CTP. Consider the VAB examples illustrated in Chapter 4. Based on this information and/or your concrete experience with a VAB

- Enumerate four advantages of VAB vs CTP
- Enumerate four advantages of CTP vs VAB
- Present a possible system design which would combine the advantages of VAB and CTP while limiting their separate disadvantages.

Hints: First, consider the tasks that a programmer or end-user would like to accomplish by using both application-building paradigms.

5 EXERCISE 5

Visualization techniques and tools can be classified using the five-element model of Marcus *et al.* (task, audience, target, medium, and representation) described in Section 4.3, Chapter 4. Give two examples of visualization applications of your choice, and explain, for each example, which are the five elements of the above model.

6 EXERCISE 6

Visual programming is a useful tool for quickly prototyping relatively simple visualization applications. However, building a large and complex application whose dataflow graph consists of hundreds of modules, and where each module has many inputs and outputs, can be challenging in terms of the manual effort required to find the right modules, place them at good

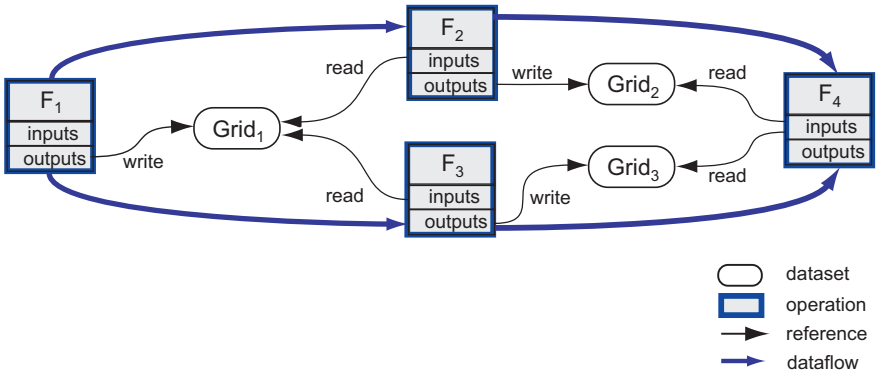
positions on the canvas, and connect the right inputs and outputs. If you were the designer of the next-generation visual programming tool, propose *three functions* you would add to a visual-programming tool in order to speed up the building process.

Hints: Think where the bottlenecks are for a beginner user in terms of the operations needed to construct the right dataflow graph. Think also about the repetitive actions an advanced user needs to do.

7 EXERCISE 7

Consider the conceptual data visualization pipeline. Here, data read from an input source is transformed by various filters, next it is mapped to geometric primitives, which are finally rendered on the screen. Consider now that the user is interested to select any visible element in the final image, *e.g.* a polygon or vertex, and ask the visualization system "From which raw data elements has this element come? And via which operations?"

Your visualization system is implemented based on the operator-dataset model outlined in Figure 4.5 (also shown below). That is, the pipeline consists of a sequence of computational functions or operators that read, respectively write, dataset object. How would you implement the above 'back tracing' functionality in such a system?



Visualization pipeline as a directed graph of datasets and operators (see Chapter 4).

Hints: Start from the end towards the beginning. Any visible element that the user can select is, in essence, a geometric primitive coming from the last dataset-object that the visualization pipeline produces and feeds to the rendering operator. Think of how you can augment the dataset representation with back-tracing information that encodes, at a low level (cells and vertices) both origins of these data elements and the operations they were generated by.

End of Exercises for
Chapter 4: Visualization Pipeline