Universität Stuttgart

Institut für Visualisierung
und Interaktive Systeme

Assignment 5

Information Visualization & Visual Analytics (WS 2019/20)

Due: Monday, 25.11.2019, 11:59 AM **Presentation:** Wednesday, 27.11.2019

Please solve the assignment in **groups of up to three (3) students**. Choose one student who submits your solution in ILIAS as PDF (for theoretical submissions) or ZIP (for practical submissions Impl) as group submission. Make sure that you create your team before uploading the solution. **Important:** The submitted files must follow the naming scheme `yourlastname1_yourlastname2_yourlastname3` with respective file extension, of course.

Task 1 Trees *[Points: 10]*

(a) *(2 points)* In typical file explorers of GUI-based operating systems the directory structure is often visualized as indented tree. Name and describe at least two advantages of using this type of tree visualization for browsing files and folders.

(b) *(2 points)* In addition to the plain structure of the file system, you now want to get an overview of the disk usage per folder. How would you visually incorporate the size of each directory into the indented tree?

(c) *(6 points)* Imagine you have tracked all your expenses over a year with an app that collected the following attributes for each item: the amount, a short title, a general category (e.g. 'leisure'), and a more specific category (e.g. 'cinema'). Draft (sketch) how you would visualize these expenses in a hierarchical way to better understand how you spent your money. You can assume that each specific category is associated with just one general category. How scalable is your proposal regarding the number of expenses? How compact is your representation relative to the size of the data you present? How well can you compare different expenses and categories to each other?

Task 2 Impl Tree-Maps with Slice & Dice *[Points: 10]*

A tree-map can be a powerful visualization for hierarchical data. The readability however highly depends on the layouting algorithm used. Your task is to implement a tree map layout for a given data structure. You will find the project skeleton on ILIAS.

The data structure of this assignment is as follows:

DataProvider

getTree(): Tree – returns the tree that has to be drawn as tree-map

Tree

getRoot(): TreeNode – returns the root node of the tree

TreeNode

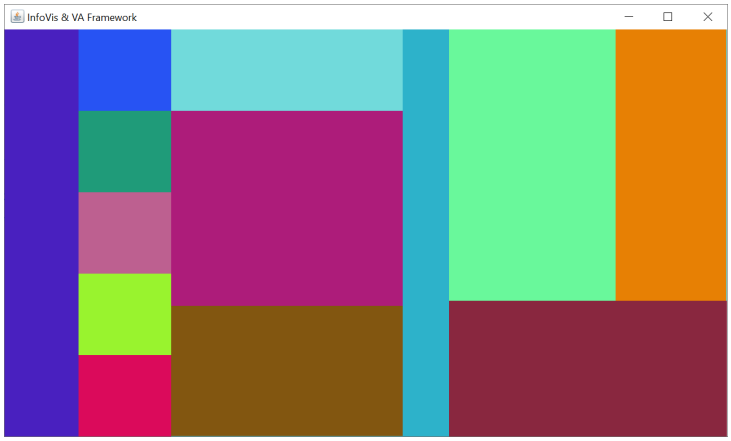
getWeight(): int – returns the total weight of this branch including all leafs.
getContent(): String – returns the caption of this node
getChildren(): List<TreeNode> – returns the node's children.

The following simple code snippet shows how to handle the tree data. The method `getTree()` in `DataProvider` returns an instance of the tree. To access the root node of that tree, call the method `getRoot()` on the returned instance:

```
1  public List<AbstractGeometry> mapData() {
2    Tree tree = new DataProvider().getTree();
3    for(TreeNode node: tree.getRoot().getChildren()) {
4      System.out.println(node.getContent() +
5          " [total weight: " + node.getWeight() +
6          " - no children: " + node.getChildren().size() + "]");
7    }
8
9    Rectangle rect = new Rectangle(0, 0,
10       InfoVisFramework.defaultScreenWidth,
11       InfoVisFramework.defaultScreenHeight);
12   rect.setColor(Color.WHITE);
13   LinkedList<AbstractGeometry> list = new LinkedList<AbstractGeometry>();
14   list.add(rect);
15   return list;
16 }
```

(a) *(8 points)* Draw the corresponding tree-map of the given structure using the Slice & Dice algorithm[1]. Choose the size of the boxes with respect to the sum of the weights of the leaf nodes (rough approximation is okay). Use a distinct (e.g., random) color for each rectangle.

(b) *(2 points)* The algorithm blindly alternates between horizontal and vertical subdivision depending on the level depth. Modify your code to choose the direction of the subdivision based on the longest side of the current parent rectangle, i.e., if the width is bigger than the height then split horizontally, otherwise vertically. Your result should look similar to this:



(c) *(+2 points)* **Bonus (optional)**
Implement a boxed tree-map to better visualize the hierarchy of the tree structure, i.e., introduce padding on each subdivision.

[1] Shneiderman, Ben. "Tree visualization with tree-maps: 2-d space-filling approach." ACM Transactions on graphics (TOG) 11.1 (1992): 92-99