ColorGame Apparatus Specification

Sam Martin, Andrey Kvochko

February 19, 2012

TD-19
LoCuS (Laboratory fOr CompUter Science) System is instructional software for teaching computer science theories through experiments (that is, laboratory exercises, as in other sciences like biology, chemistry, and physics). A locus (Latin for ”place”) is a ”collection of points which share a property” or ”the path through which a point moves to fulfill a given condition.” Our goal is for LoCuS to define a new place to emphasize the science of computation as well as a new path for computer science, in the form of labs.
1 Introduction

This apparatus is intended to be used for the Locality Lab. It illustrates the idea of storage hierarchy in computer systems: computer systems are equipped with fast (but limited) storage devices (like CPU caches, RAM) and slow (but large) storage devices like mechanical hard-drives. Depending on the order in which storage locations are accessed, the I/O performance that is achievable by the computer system varies.

This work was supported in part by NSF grant CNS-0938948.

2 Functionality

This apparatus should mimic a kindergarten paint-by-number coloring activity. It displays the drawing to be colored. The drawing would contain regions that need to be colored. Each region has a number corresponding to the color it should be painted in.

Unlike the regular kindergarten coloring activity, the apparatus would force the user to paint the regions in order (region 1 must be painted first, region 2 second and so on). The order of the regions will be passed into the apparatus. Additionally, two trays containing crayons are provided that are to be used to color the drawing: a small tray (holding only a few crayons, and initially empty) and a large tray (holding all the crayons). Each time a user uses a crayon, he/she will be charged $1. The user can only use the crayons that are in the small tray to color the drawing. However, the user could get crayons from the large tray into his small tray for a fee of $10. If no free slots are available in the small tray, the user will need to give up the least recently used crayon in the small tray in exchange for the new color from the large tray.

After the coloring activity is completed, the apparatus displays the total number of dollars required to complete the coloring. A larger dollar value means the order in which the regions were asked to be painted had low locality. And a small dollar value means the order of regions had high locality.

3 User Interface

The apparatus GUI looks something like Figure 1.1. A basic coloring apparatus would contain a GUI containing four different panes:

1. a pane for displaying the drawing to be colored,
2. pane for holding a small tray of crayons (initially empty). The number of crayons that this tray could hold will be passed in as a parameter (see XML Specification). Each crayon should have a label indicating its color (to accommodate people with color-blindness),
3. a pane for holding a large tray of crayons. This tray will contain crayons of all the colors that are required to complete the drawing. Again, each crayon should have a label indicating its color,
4. a pane to hold instructions and feedback for the user (lets call this feedback pane so we could refer to it later).
Figure 1: basic GUI for ColorGame Apparatus

Please paint region 5 in

Total charge: $64
4 XML Specification

The apparatus should accept parameters from locus in the form of an XML node object (passed in through `setOptions()` method. Please see Apparatus Developers Guide). Following is a sample XML node that the apparatus should accept:

```xml
<options>
  <imageFile>dinosaur.png</imageFile>
  <regionFile>dinosaur-low.cbynumber</regionFile>
  <crayonFile>defaultCrayons.txt</crayonFile>
  <smallTraySize>3</smallTraySize>
  <largeTraySize>12</largeTraySize>
  <disableReset>true</disableReset>
  <disableAutoPaint>true</disableAutoPaint>
</options>
```

The `options` node contains seven children:

- `imageFile(required)` specifies the file to read the image from.
- `regionFile(required)` specifies the file to read the drawing data from (a list of points to different image regions)
- `crayonFile(required)` specifies the file containing information about crayon images, their names and pictures.
- `smallTraySize(optional)` is the number of crayons in the small tray. By default value 3 is assumed.
- `largeTraySize(optional)` is the number of crayons in the large tray.
- `disableReset(optional)` disables “reset” button
- `disableAutoPaint(optional)` disables “autopaint” button.

5 Implementation Details

The standard Java2D API should suffice for all the graphics related functionality. We could use something like `java.awt.geom.GeneralPath` to draw/highlight regions, and use something like `java.awt.Graphics2D.fill(java.awt.Shape)` to paint a region.

The crayons should be draggable (from large-tray to small-tray and from small-tray to regions in the drawing). Java API provides drag-and-drop event listening functionality which could be used for this purpose.