Programming Structures and Abstractions (CS 2334)
Lab 6: Graphical User Interface Basics

October 20, 2010

Due: Friday, October 22, 2010, 11:29am

Group members (same as for your project):

Introduction

Graphical user interfaces are a primary means by which many of our programs interact with a human user. In order for these interfaces to be intuitive and easy to use, the information must be presented in a consistent and logical manner (a topic that you will explore in great detail in your Human-Computer Interaction course). The Java graphics facilities provide a large set of classes that can make the design, implementation and tuning of these interfaces a straightforward process. In addition, these facilities are a clear example of how object oriented programming can make very complicated designs tractable enough to implement and debug.

Objectives

By the end of this lab exercise, you should be able to:

1. create and manipulate basic graphical user interface components, including JPanel, JLabel, GridLayout and TitledBorder

2. display data in a graphical user interface, and
3. use information entered into the GUI to affect the state internal data structures.

Problem Context

In this project, we will be creating a graphical user interface (GUI) to interface with the Finch. In particular, this interface will display the current light, obstacle, temperature and accelerometer values. In addition, the user will be able to control the color of the nose LEDs (light emitting diodes) using a set of slider bars.

The sample code provides a basic skeleton for creating a window that can display the light and accelerometer sensor values and accept input from slider bars. Specifically, you will:

- analyze the structure of the existing code base,
- use the slider bar data to change the state of the LEDs, and
- add display elements for the following sensors: temperature and obstacle,

Milestones

Milestone 1: Analyze Existing Program

Download lab6.zip from the class website. Compile and run the lab6Driver program that has been provided. Briefly explain what happens in the GUI when you move the Finch around:

The **FinchDisplay()** constructor is responsible for creating an instance of the display class. This class extends the **JFrame** class, and therefore inherits a lot of functionality. Because JFrame is a **container**, it provides an **add()** method that allows one to add graphical child objects to the frame (it adds in the "has-a" sense). Other classes, such as **JPanel**, are also containers. In addition, the **FinchDisplay()** constructor makes a variety of calls to create and attach a set of **LayoutManagers()**.

Given the **FinchDisplay()** constructor, draw on engineering paper the FinchDisplay instance and all of its children and other relations. Do this recursively (so the children's' children, etc.). Make sure that the FinchDisplay instances are included in your UML instance diagram.

Notes:

1. you only have to show the child instances that are created in the FinchDisplay() constructor.

2. although we are using the "parent" and "child" terminology here, it refers to a "has-a" relationship from parent to child, and not to an inheritance relationship.

Milestone 2: Use the Slider Bars to Change LED State

The FinchDisplay class includes a private property called **LED**. This array of integers stores the LED brightness values that are "intended" by the user (the values are specified by the indicated sliders). Given the implementation of the constructor, briefly explain your intuition about how LED[1] is changed (you should not have to look at Chapter 15 to answer this):

The FinchDisplay class provides an accessor method to this private property, **getLED(),** which is used by the driver class. However, nothing is currently done by the driver with this information. Add the necessary code to the driver that will take the value obtained from getLED() and change the state of the Finch nose.

Note: get this working before you move on to the next milestone.

Milestone 3: Add Other Sensors

Add the necessary code so that the state of the following sensors is also shown in the GUI:

- obstacle, and
- temperature.

Following the implementation of the light and accelerometer sensors, add the following components for each sensor:

- (FinchDisplay) a separate JPanel with an appropriate TitledBorder and GridLayout that contains JLabels for each of the values to be displayed,
- (FinchDisplay) a mutator method that allows the driver class to update the values to be displayed by the interface, and
- (driver) within the main loop, read each sensor from the Finch and a write the value to the display window.

Milestone 4: Alter the Interface Layout

Alter the layout of the different panels in the graphics window. Specifically, alter the arrangement of the different components of the window so that:

- the accelerometers occur in the middle of the window,
- the light sensors are placed at the top of the window,
- the obstacle sensors are shown on the left hand side of the window, and
- the temperature sensor is shown on the right hand side of the window.

What to Hand In

All materials are due: Friday, October 22, 2010, 11:29am

Hand in the following:

- a hard copy of your instance diagram,
- a copy of this handout containing the provided answers, and

- an electronic copy of your modified code in the form of lab6.zip (to D2L)

NOTE: ONLY HAND IN ONE COPY PER GROUP.

In addition to handing in a copy of the code, you must do a short demonstration of your working code for the TA or the instructor. Ideally you will do this before the end of the lab period. Otherwise, please make an appointment before the deadline. All group members should be in attendance during the demonstration.