Graphic Interface Programming II

Introduction (to Graphics)
Today

- Previous evaluation
- Administration
  - Registration
  - Groups (3-4 students/group)
  - CVS
- Diary
- Homepage
  - http://www.it.uu.se/edu/course/homepage/devgui2/vt12
- Intro to assignment
Previous courses

- Generally positive
  - high evaluation scores
  - good course setup

- Negatives
  - high workload
  - unclear grading system
Teachers

- Lars Oestreicher (1152)
  - general responsibility, lectures, seminars
- Simon Tschirner (1157)
  - assignments, CVS, seminars
- Mikael Laaksoharju (1157)
  - diary
Personal responsibility

- You are expected to do a lot of programming during this course

- You are expected to get the help you need if you have problems:
  - internet, books and lectures
  - fellow students
  - teachers
  in that order
Examination

- No written exam (smaller individual report)
- Active group work
  - Reported in diary
  - Group meetings (mandatory)
- Grading:
  - Overall assignment results
  - Individual diary
  - Individual activity – ability to show your work
  - Assignment presentation
  - Interview
Diary

- Diary MUST be used regularly
  - Records the time spent on the work in the course

- Diary is part of the individual EXAMINATION!
  - No, or sparse entries – lower individual grade
Goals

- Understanding of Graphics environments and Drawing mechanisms
- Using non-standard Swing-features
  - Advanced interfaces
  - Use of Graphics for components or
- Understanding of Design Patterns
- Good programming practices
  - Structuring
  - Critical thinking
  - Documentation
- Future GUI ideas
Groups

- You form the groups on your own
- Enter your group constellation in Studentportalen
- Only 4 students per group
Project news

- Each group will assign project leaders
- Everybody in the group has to be a project leader at some point
- Each project leader is responsible for one period of time
  - 1/3 or 1/4 of the course (depending on the number of participants in the group)
- The role of project leader will rotate with time
The Project Leader

- is responsible for the planning of the work over the week
- is responsible for a weekly (short) report
- is responsible for managing the group meetings
- will meet with the steering committee (a teacher) once a week (appr.)
- will perform a handover at the end of his or her term of service
Being a project leader…

…still means that you have to do programming work!
The project

- The project is a vehicle for learning!

- Important that you divide the tasks between you

- Very important that everybody understand the parts of the project they did not work with

- Not so important that the result is perfect (!!!)
Important

- Until Thursday 22/3:
  - Form groups
  - Make a schedule for project leaders
  - Make a project plan!

- Send email about groups to Lars and Simon as soon as you entered them in Studentportalen
  - Subject: devgui2
Project (Assignments)

- Reusing Code (not application)
  - Select an application from the previous course
  - Reuse parts (rewrite the code)
- Extend/rewrite application
  - Calendar Application with TODO-facility
  - Write your own Calendar
  - Do not use existing packages/libraries
    - Exception: to "calculate" the date and time.
- Add "extra" graphic features
Four stages

- Paced development

- All assignments are published from the beginning!

- Go through and use for the planning
Why???

- A popular question for some requirements
Why???

- A popular question for some requirements

- And the simple answer is
Why???

- A popular question for some requirements
- And the simple answer is

Because...!!!
Graphics

- A large part of the work will be with graphics

- Learn how to use the Graphic and Graphic2D classes

- Simple mechanism for drawing
  - Complex possibilities.
Advanced Graphics

- NOTE!
- Some advanced Graphics features are not available on all platforms
- One such feature is transparency
  - Some kinds may still work!

- It is necessary to test for this property when we implement more fancy features
From Simple to...

- Need to know more about the basic drawing mechanisms in Swing

- From the Basics
  - To the Skies
    - In small small steps
Begin with some simple GRAPHICS
Drawing

- Components are without shape and colour
  - (seen as objects in Java)
  - they have size(!)
  - one method is responsible for the looks

- `paintComponent(Graphics g)`
  - can be overridden
  - most components need to invoke `super.paintComponent()`
  - in order to draw the basic component graphics
Window drawing

- `paintComponent(Graphics g)` draws everything
  - Decorations
    - Borders
    - Title Bars
- `setUndecorated(true);`
  - By default Frames are decorated
  - If undecorated the titlebar disappears!
  - Moving the window becomes difficult…
Most of the time Swing uses 2D graphics
  • (even when we have a Graphics g component)

We (almost) never create a Graphics object
  • We get it in the call to paintComponent()

We manipulate the given Graphics object in the method
2D Rendering

1. Get a Graphics (Graphics2D) object
2. Set attributes on the Graphics object (*)
3. Draw graphics primitives with the Graphics object
4. Repeat 2 and 3 until satisfied

(*) might need some extra handling to turn out right
Example

```java
protected void paintComponent(Graphics g) {
    g.setColor(Color.RED);
    g.fillRect(0,0,width, height);
    g.setColor(Color.BLUE);
    g.fillOval(10,20,width/2, height/2);
}
```

- Items are drawn in execution order from the bottom up!
- Last item is “on top”!
  - The Rectangle will be seen as “behind” the Oval.
Graphics Cloning

- An important technique when we start working with animations.

- Changing the state of a Graphics object...
Graphics Cloning

- An important technique when we start working with animations.

- Changing the state of a Graphics object...

  CHANGES the state!
Graphics Cloning

- An important technique when we start working with animations.

- Changing the state of a Graphics object...

  CHANGES the state!

Sometimes you don’t want that!
Graphics Cloning

Protected void paintComponent(Graphics g) { 
    // Rotate the coordinate system
    g.rotate(45);

    // draw some content
    g.fillRoundRect(10,20,30,40);

    // and return without resetting g
}

// Destroys the general state of g
Graphics Cloning

Protected void paintComponent(Graphics g) {
    Graphics gTemp;
    gTemp = g.create();
    // Rotate the coordinate system
    gTemp.rotate(45);
    // draw some content
    gTemp.fillRoundRect(10, 20, 30, 40);
    // and return after disposing of gTemp
    gTemp.dispose();
}

// Keeps the general state of g
Mixing Components and Paint

- Difficult, they don’t form ”mixable” layers

- We will return to this later…
setOpaque()?

- JComponent.setOpaque( boolean ) is misnamed
- Does not do transparency!!!
- Collaborates with super.paintComponent

- It really should be called JComponent. wipeBackground( boolean )
Simple Transparent Button

class TransparentButton extends JButton {
    public TransparentButton(String text) {
        super(text);
        setOpaque(false);
    }

    public void paintComponent(Graphics g) {
        Graphics2D g2 = (Graphics2D) g.create();
        g2.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, 0.5f));
        super.paintComponent(g2);  // After changing!!!
        g2.dispose();
    }
}
class TransparentPopupMenu extends JPopupMenu {
    public TransparentPopupMenu() {
        super();
        setOpaque(false);  
    }
}

class TransparentMenuItem extends JMenuItem {
    public TransparentMenuItem(String text) {
        super(text);
        setOpaque(false);
    }
    public void paint(Graphics g) {
        Graphics2D g2 = (Graphics2D) g.create();
        g2.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, 0.5f));
        super.paint(g2);  // <---- OBS!
        g2.dispose();
    }
}
setComposite(Composite c)

- Specifies HOW:
  - the new pixels
  - are combined with
  - the existing pixels
  - on the graphics device (e.g. Graphics g)

- E.g. As more transparent colors
  - Don’t have to work on the colors directly!

- Composite is an interface!
Color

- Colors can also be transparent
- Color stores a color in 32 bits
  - 8 bits for each color
  - Bits $8 \times 3 = 24$; 8 bits missing:

- Alpha channels
  - Values 0 to 255 define transparency of the colour
Example

Color almostTransparentDarkOrange =
new Color( 0xff /* red */,
0x8c /* green */,
0x00 /* blue */,
0x1f /* alpha */ );

- #ff8c00 = DarkOrange as a colour code
Dynamic!

AlphaComposite.getInstance(AlphaComposite.SRC_OVER, 0.5f)

- Note that this can be changed during runtime
  - Since it is set in the paintComponent method
- New Colors can also be defined in paintComponent
  - But that creates a lot of instances (if we use loops and many colours)
Experiment!