O'REILLY

OSCON™
Open Source Convention

Advanced Effects in Java Desktop Applications

Kirill Grouchnikov, Senior Software Engineer, Amdocs
kirillcool@yahoo.com
http://www.pushing-pixels.org

OSCON 2007

Agenda

- Swing pipeline
- Hooking into the pipeline
  - RepaintManager
  - Playing with opacity
  - Glass pane
  - Layering in UI delegates
- Rainbow demo
- Q&A

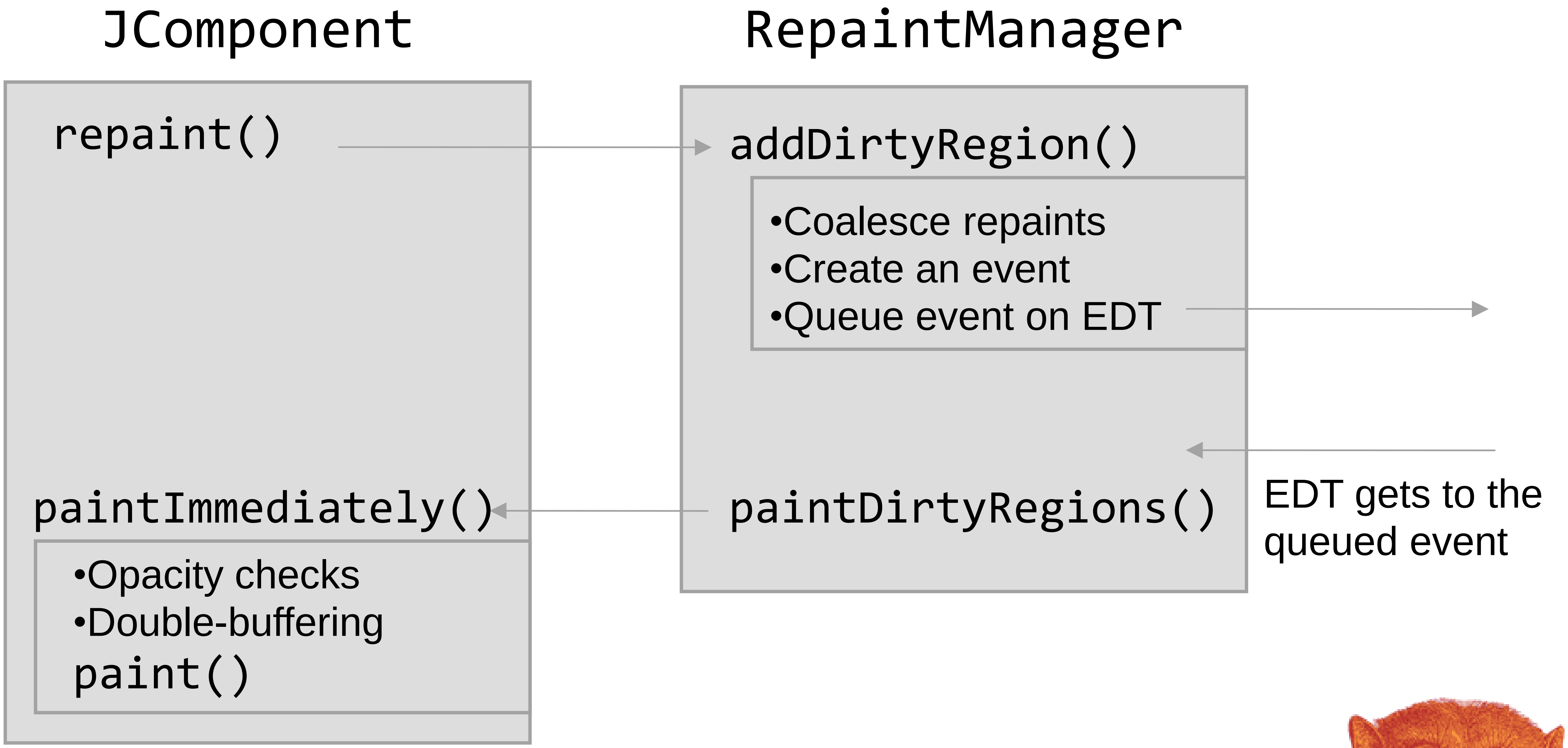
Swing basics

- UI toolkit for Java applications
- What is a lightweight component?
  - Very flexible
  - Provides a lot of hooks for custom behavior
  - Not trivial to implement
- Heavyweight counterparts – AWT and SWT

Swing painting pipeline

- Three major “participants”
    - JComponent
    - RepaintManager
    - ComponentUI
- Provide various hooks to customize behavior
- Vary in flexibility, robustness and ease of use

Swing painting pipeline – part I



JComponent

repaint()

paintImmediately()

- Opacity checks
- Double-buffering

paint()

RepaintManager

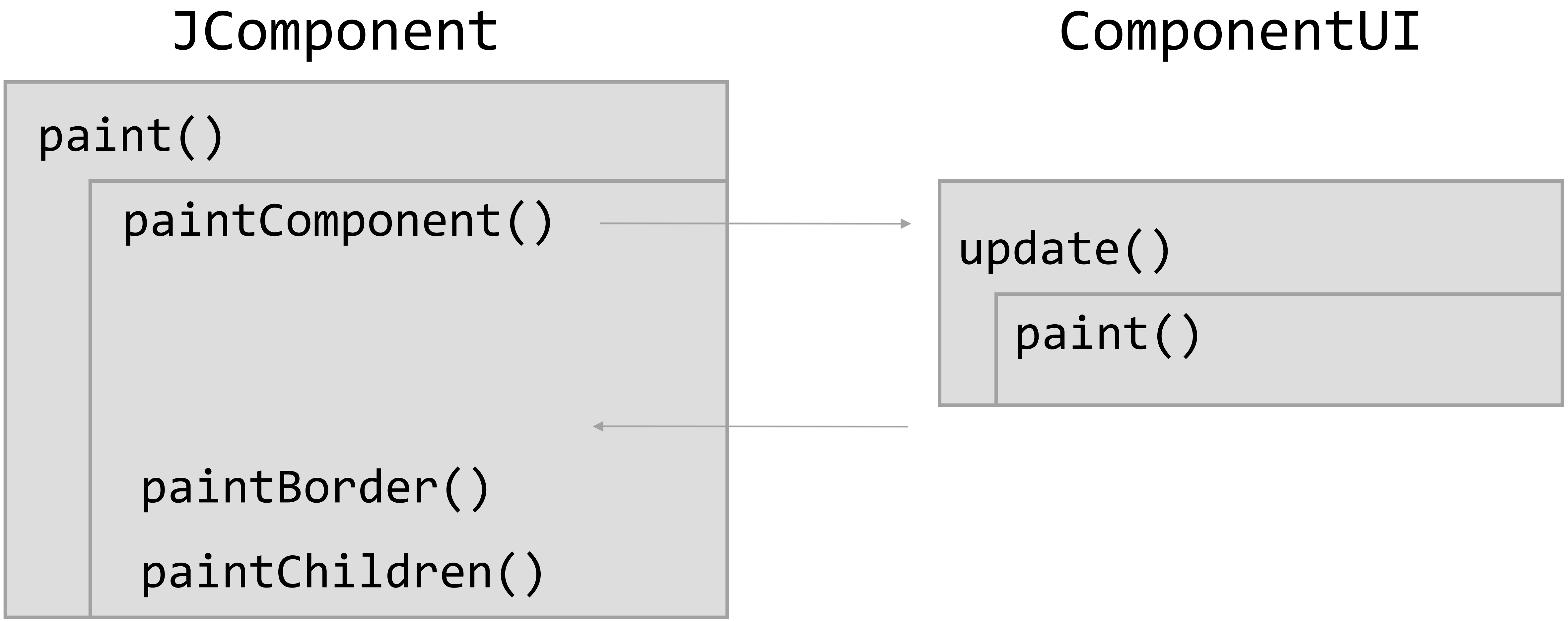
addDirtyRegion()

- Coalesce repaints
- Create an event
- Queue event on EDT

paintDirtyRegions()

EDT gets to the queued event

Swing painting pipeline – part II



Swing pipeline hooks

- JComponent
  - Override paint or paintComponent
  - Or even repaint or paintImmediately
- RepaintManager
  - Install a custom implementation (singleton)
- ComponentUI
  - Provide custom painting for a specific component class

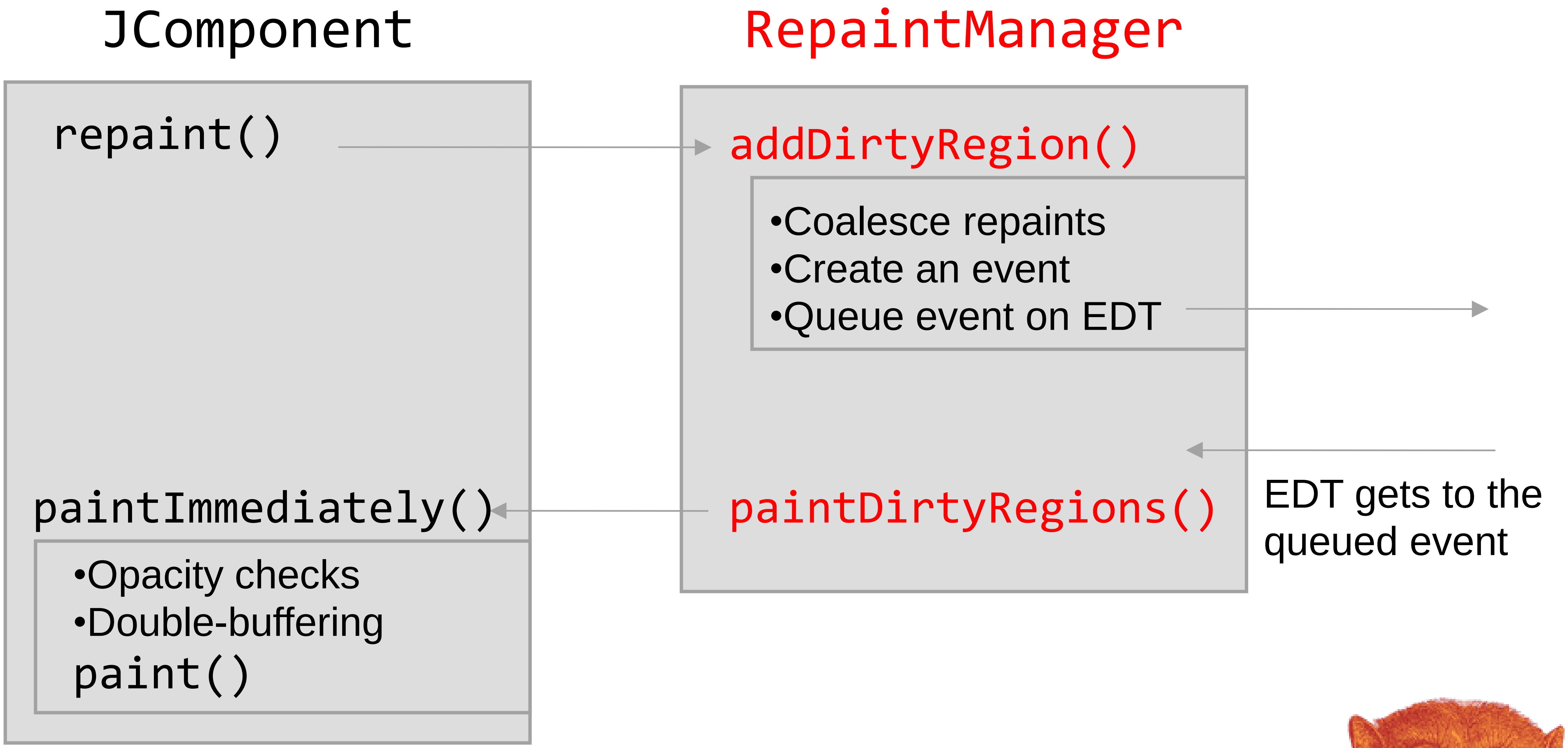
What we can achieve?

- Translucency
- Non-rectangular components
- Layering
- Image filtering
- Animation

Agenda

- Swing pipeline
- Hooking into the pipeline
    - RepaintManager
    - Playing with opacity
    - Glass pane
    - Layering in UI delegates
- Rainbow demo
- Q&A

Swing painting pipeline hooks



JComponent

repait()

paintImmediately()

- Opacity checks
- Double-buffering

paint()

RepaintManager

addDirtyRegion()

- Coalesce repaints
- Create an event
- Queue event on EDT

paintDirtyRegions()

EDT gets to the queued event

RepaintManager example

- SwingX project
- JXPanel that provides translucency
    - setAlpha(float)
    - using RepaintManagerX – see code

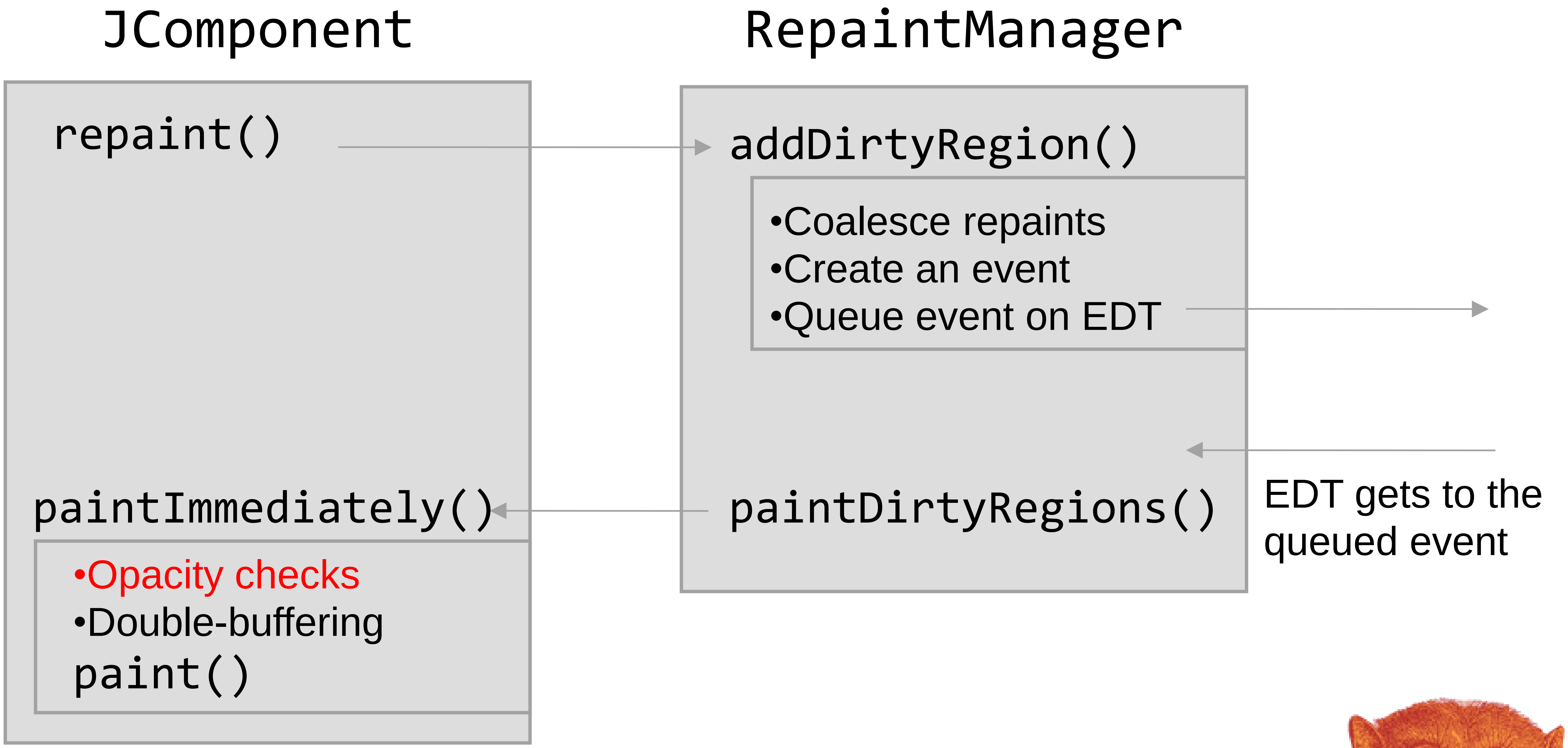
There can be only one (singleton)

```
class JXPanel {
  public void setAlpha(float alpha) {
    if (alpha > 0f && alpha < 1f) {
      ...
      RepaintManager.setCurrentManager(
         new RepaintManagerX());
     }
   }
```

Agenda

- Swing pipeline
- Hooking into the pipeline
  - RepaintManager
  - Playing with opacity
  - Glass pane
  - Layering in UI delegates
- Rainbow demo
- Q&A

Swing painting pipeline hooks



JComponent

repaint()

paintImmediately()

- Opacity checks
- Double-buffering

paint()

RepaintManager

addDirtyRegion()

- Coalesce repaints
- Create an event
- Queue event on EDT

paintDirtyRegions()

EDT gets to the queued event

Opacity basics - setOpaque

- `setOpaque(false)` == “draw stuff behind me”
    - Useful for translucent or non-rectangular components
- `setOpaque(true)` == “I’ll handle it”
    - During repainting of an opaque component Swing does not repaint any components behind

Transition effects using opacity

- UIs changes are immediate
  - Showing / hiding a control
  - Moving a control to new location
  - Tab switch
- Solution – use transitions (cross fades, fly-in / out)
- Making controls non-opaque to enable the transition effects

DEMO

Transition layout demo

Transition layout manager

```
TransitionLayoutManager.getInstance().
      track(myTabbedPane, true);

TransitionLayoutManager.getInstance().
      track(myPanel, true);
```

- Play with opacity (set to false during animation cycle)
- Set translucency (for fades)
- Custom layout manager (for sliding effects)

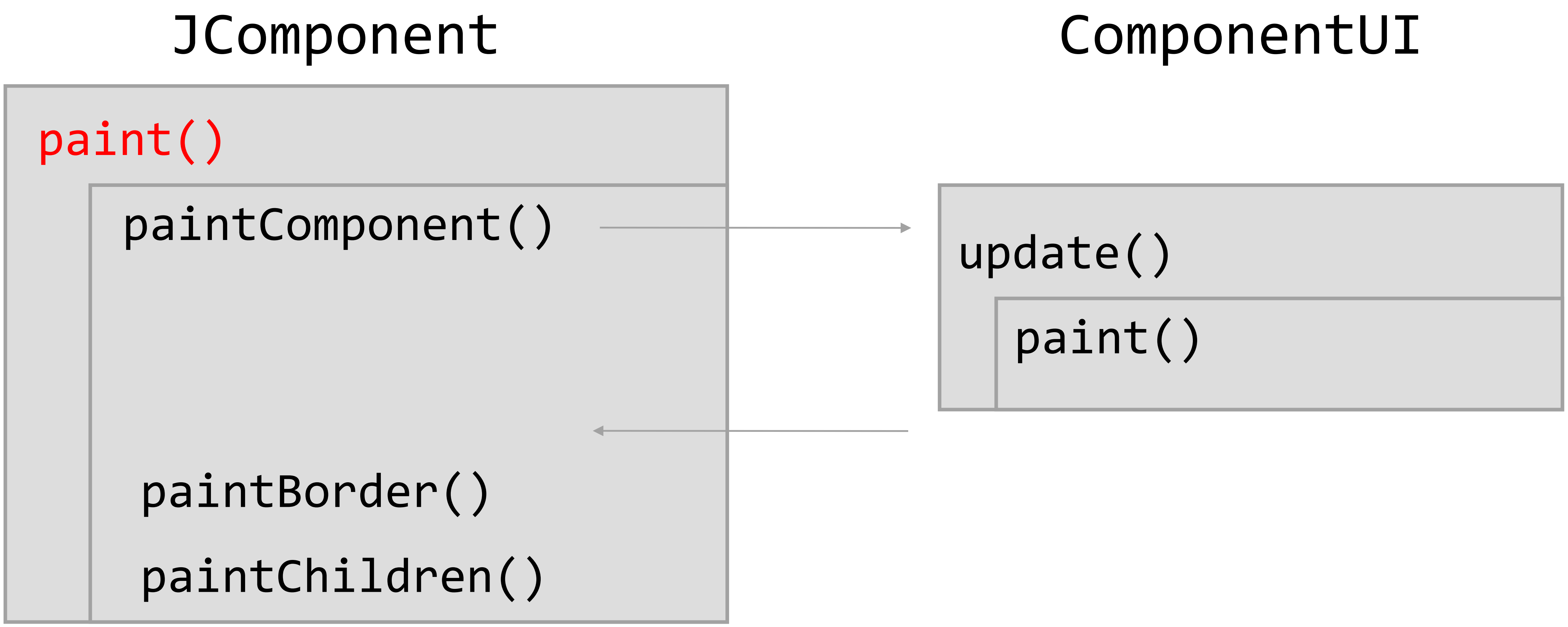
Transition scenarios

- Remains visible and has the same bounds
- Remains visible and has different bounds
- Becomes invisible
- Added or becomes visible
- Remains invisible

Agenda

- Swing pipeline
- Hooking into the pipeline
    - RepaintManager
    - Playing with opacity
    - Glass pane
    - Layering in UI delegates
- Rainbow demo
- Q&A

Swing painting pipeline hooks



JComponent

paint()

paintComponent()

paintBorder()

paintChildren()

ComponentUI

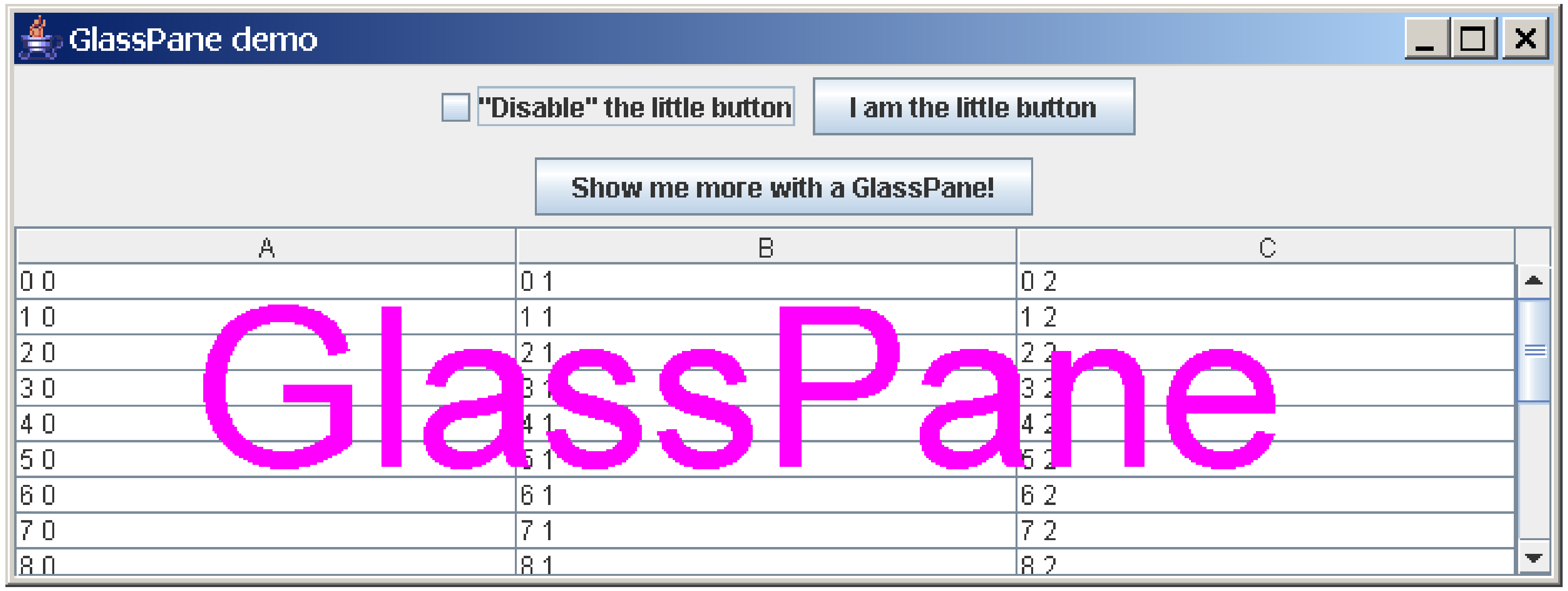
update()

paint()

Glass pane basics

- Painting over all the components

```
frame.setGlassPane(new CustomGlassPanel());
frame.getGlassPane().setVisible(true);
```



A	B	C
0 0	0 1	0 2
1 0	1 1	1 2
2 0	2 1	2 2
3 0	3 1	3 2
4 0	4 1	4 2
5 0	5 1	5 2
6 0	6 1	6 2
7 0	7 1	7 2
8 0	8 1	8 2

Glass pane

- Pros
  - Does not affect component's state
- Cons
  - Global resource (for a frame)
  - Everything is repainted (performance)

JXLayer overview

- It is a component wrapper like JScrollPane
    - You have access to the wrapped component's state
- It does not use glassPane from the frame
    - It has its own a transparent panel on the top
- JXLayer.paint() delegates all painting to the painter
    - A flexible way to modify component's appearance

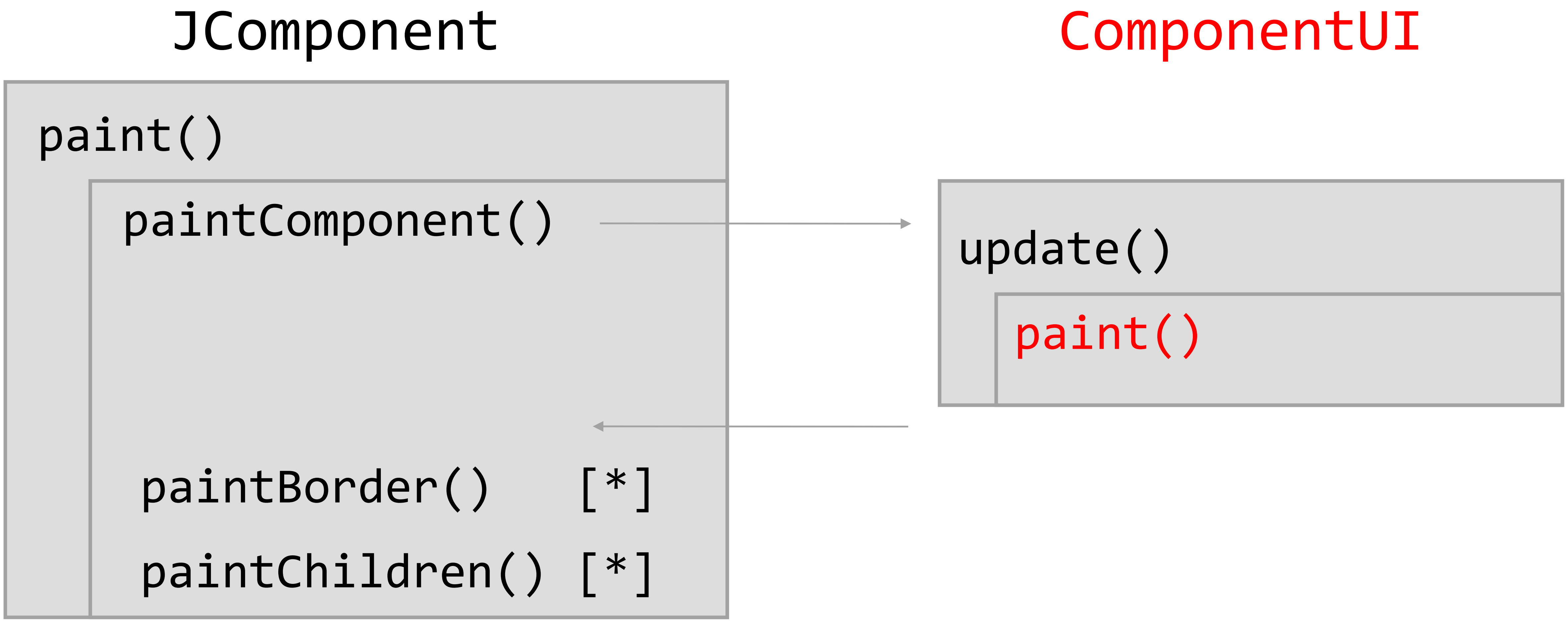
JXLayer overview

- Painters API
- Image filtering
- Translucency
  - PainterModel.setAlpha(float)
- Non-rectangular components
  - MouseEvents filtering

Agenda

- Swing pipeline
- Hooking into the pipeline
    - RepaintManager
    - Playing with opacity
    - Glass pane
    - Layering in UI delegates
- Rainbow demo
- Q&A

Swing painting pipeline hooks



JComponent

ComponentUI

paint()

paintComponent()

update()

paint()

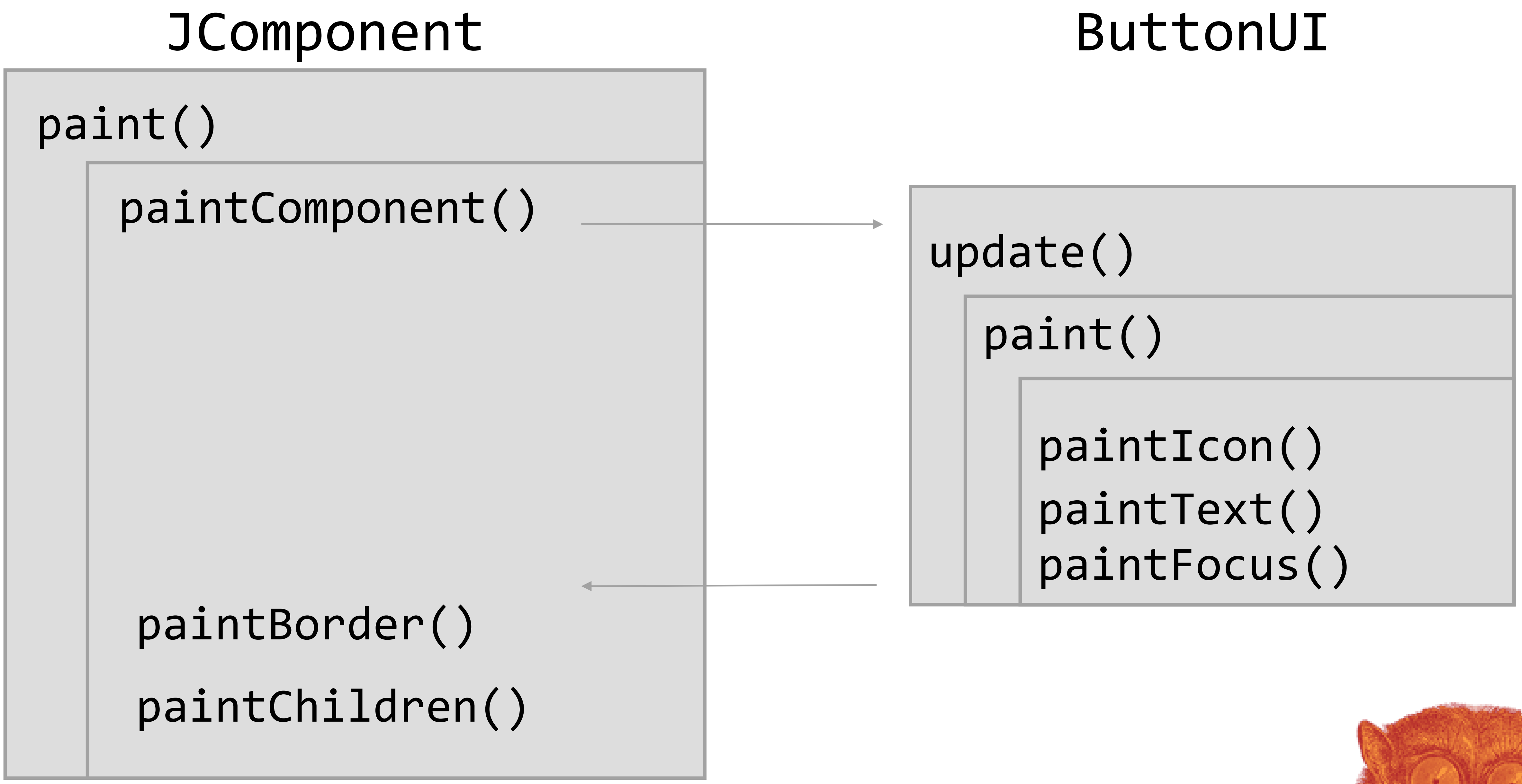
paintBorder()   [*]

paintChildren() [*]

UI delegates basics

- UI delegates – classes responsible for painting Swing components.
  - JPanel – PanelUI delegate [*]
  - JButton – ButtonUI delegate [*]
  - ... (41 different UI delegates)
- Provide flexible control over painting different visual layers of Swing components

UI delegate flow



JComponent

paint()

paintComponent()

paintBorder()

paintChildren()

ButtonUI

update()

paint()

paintIcon()

paintText()

paintFocus()

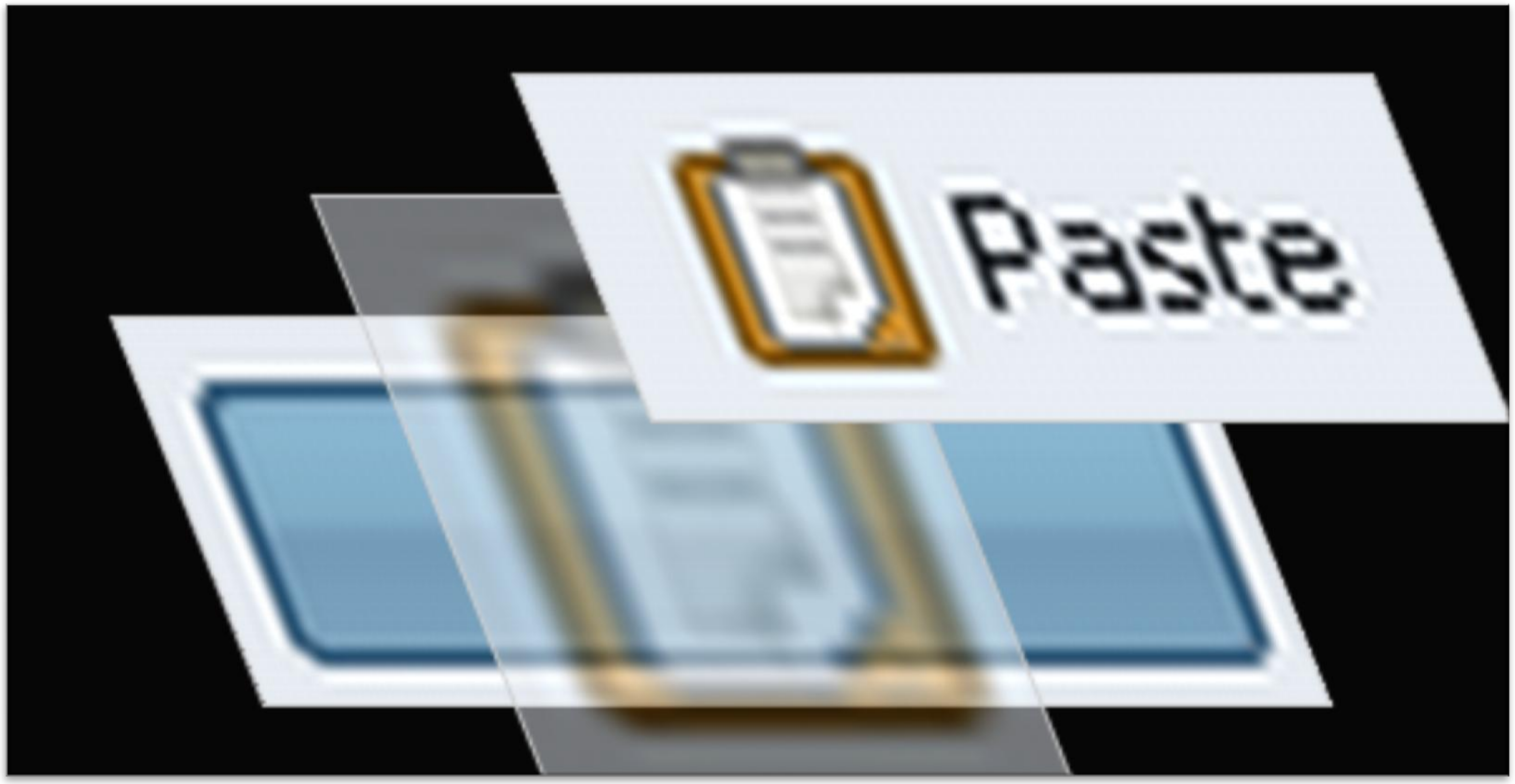
Alternatives

- Repaint manager and glass pane - much higher level
- UI delegate can
  - Add drop shadow to the button text
  - And get all the rest from the core implementation
- Opens the field to a wide array of effects
  - Ghost images / springs
  - Ripples
  - ...

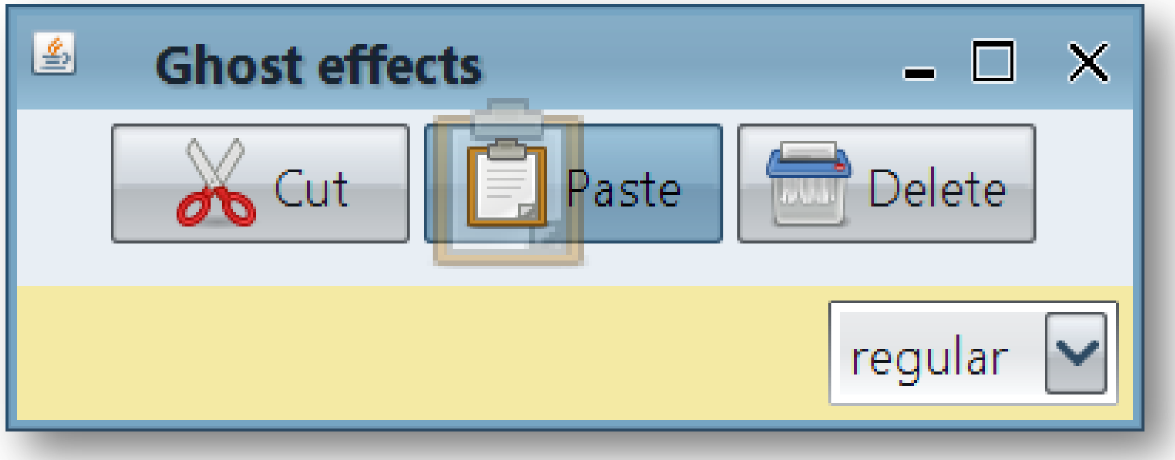
DEMO

- Ghost effects

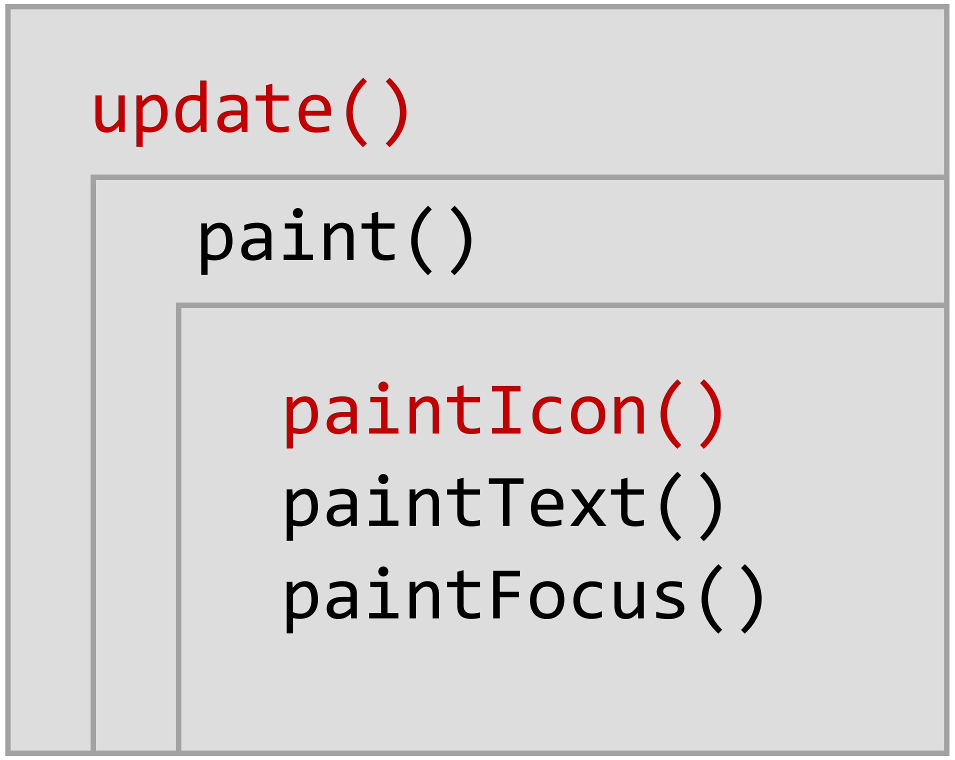
Ghost effects sequence



Ghost effects implementation



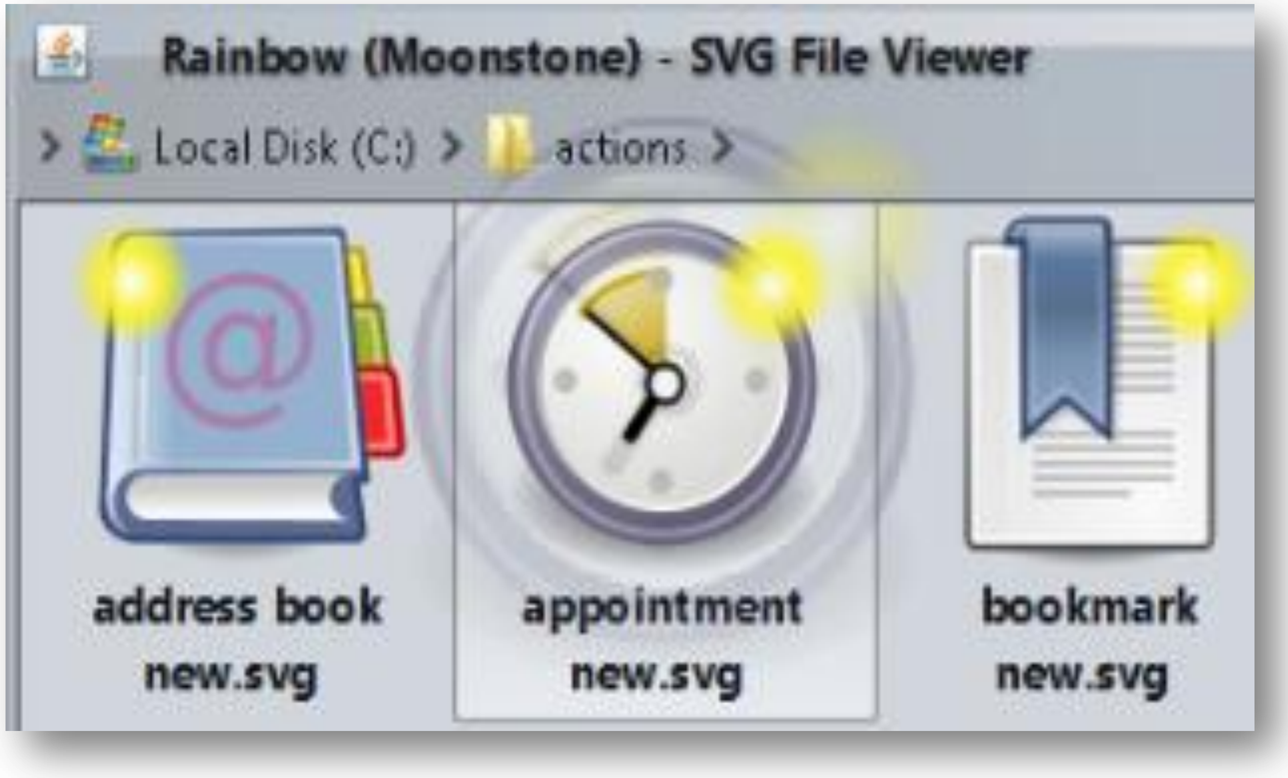
- Custom painting code in:
  - `ButtonUI.paintIcon()` or
  - `ButtonUI.update()`



```
update()
  paint()
    paintIcon()
    paintText()
    paintFocus()
```

Ghost effects eye candy

Icon ghosting over multiple components



Ghost effects

- Pros
  - Minimal changes in the application code.
  - No need for custom painting code
  - Available under multiple look and feels (use bytecode injection)
- Cons
  - Custom paintComponent implementations

Agenda

- Swing pipeline
- Hooking into the pipeline
  - RepaintManager
  - Playing with opacity
  - Glass pane
  - Layering in UI delegates
- Rainbow demo
- Q&A

DEMO

Rainbow demo

https://rainbow.dev.java.net

Sources + WebStart link

Links

- JXLayer project https://swinghelper.dev.java.net/
- Laf-Widget project http://laf-widget.dev.java.net
- SwingX project http://swingx.dev.java.net/

- Old blog http://weblogs.java.net/blog/kirillcool/
- New blog http://www.pushing-pixels.org

Q&A

Kirill Grouchnikov

kirillcool@yahoo.com