Announcement: A4 Collision Detection

- Note: Block.overlap() is approximate:

```java
public static boolean overlaps(Block a, Vector2D u, Block b, Vector2D v) {
    Vector2D c1 = Vector2D.add(new Vector2D(a.position), u);
    Vector2D c2 = Vector2D.add(new Vector2D(b.position), v);
    return Vector2D.dist(c1, c2) < a.halfwidth + b.halfwidth;
}
```

Some GUI history

- `60s: Sutherland's "SketchPad"
- `60s: Xerox's "Star" workstation

---

**Graphical User Interface:** A type of user interface that allows users to interact with programs through manipulation of graphical elements as opposed to text-based interfaces.

---

Mouse tales…

Their mouse had a mean time between failure of ... a week ... it would jam up irreparably, or ... jam up on the table— ... It had a flimsy cord whose wires would break. Steve Jobs: "... Xerox says it can't be built for < $400, I want a $10 mouse that will never fail and can be mass produced, because it's going to be the primary interface of the computer ..." 

... Dean Hovey ... came back. "I've got some good and some bad news. Good news: we've got a new project with Apple. Bad news: I told Steve we'd design a mouse for 10 bucks." 

... year later ... we ... filed ... and were granted a patent, on the electro-mechanical-optical mouse of today; ... we ended up ... making the mouse as invisible to people as it is today.

Modern GUI Examples: “Katana”


Modern GUI Examples: “Houdini”


Modern GUI Examples: Drum machine

Many GUI tools & APIs

**APOLLO GUI SET**

Biggest Gui Set ever: 1000+ Elements

http://www.webdesignshock.com/wp-content/uploads/2012/05/Gui_Apollo_WDS_901.jpg

Many GUI tools & APIs: QT (cross platform)

http://upload.wikimedia.org/wikipedia/commons/f/f3/Qt_Designer_4_4_3.png

Many GUI tools & APIs: Swing (Java)


Example: Swing (Java)


GUI (Graphical User Interface)

- Provides a friendly interface between user and program
- Allows *event-driven* or *reactive* programming: The program reacts to events such as button clicks, mouse movement, keyboard input
- Often is *multi-threaded*: Different threads of execution can be going on simultaneously

We use Java's two packages for doing GUIs:

- **AWT** (*Abstract Window Toolkit*) —first one; very simple
- **Swing** —a newer one, which builds on AWT as much as possible

Two aspects to making a GUI:

1. Laying out components (buttons, text, etc.) in it.
2. Listening/responding to events

TODAY

Next Lecture
Class JFrame

A JFrame object is associated with a window on your monitor. Generally, a GUI is a JFrame object with various components placed in it.

Some methods in a JFrame object:
- hide()
- show()
- setVisible(boolean)
- getX()
- getY()
- getWidth()
- getHeight()
- setLocation(int, int)
- getTitle()
- setLocation(String)
- setLocation(int, int)
- getTitle()
- setTitle(String)
- getLocation()
- setLocation(int, int)
- setTitle(String)
- setLocation(String)
- setLocation(int, int)

Over 100 methods in a JFrame object!

Placing components in a JFrame

Layout manager: Instance controls placement of components.

JFrame layout manager default: BorderLayout.

BorderLayout layout manager: Can place 5 components:
- North
- South
- East
- West
- Center

Putting components in a JFrame

```java
import java.awt.*;
import javax.swing.*;

public class ComponentExample extends JFrame {
    public ComponentExample(String t) {
        super(t);
        Container cp = getContentPane();
        cp.add(new JButton("click me"), BorderLayout.EAST);
        cp.add(new JTextField("type here", 22), BorderLayout.NORTH);
        cp.add(new JCheckBox("I got up today"), BorderLayout.SOUTH);
        cp.add(new JLabel("label 2"), BorderLayout.WEST);
        cp.add(new JTextArea("type here", 4, 10), BorderLayout.CENTER);
        pack();
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }
}
```

Packs components to its contentPane

Basic Components

- Button
- Canvas
- Checkbox
- Choice
- Label
- List
- Scrollbar
- TextComponent
- TextField
- TextArea
- Container
  - JComponent
  - AbstractButton
  - JButton
  - JToggleButton
  - JCheckBox
  - JComboBox
  - JLabeledButton
  - JRadioButton
  - JCheckBox
  - JPanel
  - JPopupMenu
  - JScrollBar
  - JSlider
  - JTextComponent
  - JTextField
  - JTextArea

Pack components into its contentPane
Components that can contain other components

<table>
<thead>
<tr>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Box</td>
</tr>
<tr>
<td>Container</td>
</tr>
<tr>
<td>JComponent</td>
</tr>
<tr>
<td>JPanel</td>
</tr>
<tr>
<td>Applet</td>
</tr>
<tr>
<td>Window</td>
</tr>
<tr>
<td>Frame</td>
</tr>
<tr>
<td>JFrame</td>
</tr>
<tr>
<td>JWindow</td>
</tr>
</tbody>
</table>

- java.awt is the old GUI package.
- javax.swing is the new GUI package.
- When they wanted to use an old name, they put J in front of it.
- (e.g. Frame and JFrame)
- When constructing javax.swing, the attempt was made to rely on the old package as much as possible.
- So, JFrame is a subclass of Frame.
- But they couldn't do this with JPanel.

import java.awt.*
import javax.swing.*
/** Instance has labels in east/west, JPanel with four buttons in center. */
public class PanelDemo {
    JPanel p = new JPanel();
    public PanelDemo() {
        super("Panel demo");
        p.add(new JButton("0"));
        p.add(new JButton("1"));
        p.add(new JButton("2"));
        p.add(new JButton("3"));
        Container cp = getContentPane();
        cp.add(new JLabel("east"), BorderLayout.EAST);
        cp.add(new JLabel("west"), BorderLayout.WEST);
        cp.add(new JLabel(""), BorderLayout.CENTER);
        pack();
    }
}

Child Panel as a container

import java.awt.*
import javax.swing.*
/** Demo class Box. Comment on constructor says how frame is laid out. */
public class BoxDemo extends JFrame {
    /** Constructor: frame with title "Box demo", labels in east/west, blank label in south, horizontal Box with 4 buttons in center. */
    public BoxDemo() {
        super("Box demo");
        Box b = new Box(BoxLayout.X_AXIS);
        b.add(new JButton("0"));
        b.add(new JButton("1"));
        b.add(new JButton("2"));
        b.add(new JButton("3"));
        Container cp = getContentPane();
        cp.add(new JLabel("east"), BorderLayout.EAST);
        cp.add(new JLabel("west"), BorderLayout.WEST);
        cp.add(new JLabel(""), BorderLayout.SOUTH);
        cp.add(b, BorderLayout.CENTER);
        pack();
    }
}

Class Box: a container

Box layout manager default: BoxLayout.
BoxLayout layout manager: Place any number of components. They appear in the order added, taking only one row.

public class BoxDemo2 extends JFrame {
    /** Constructor: frame with title t and 3 columns with n, n+1, and n+2 buttons. */
    public BoxDemo2(String t, int n) {
        super(t);
        // Create Box b1 with n buttons.
        Box b1 = new Box(BoxLayout.Y_AXIS);
        for (int i = 0; i != n; i = i + 1)
            b1.add(new JButton("1 " + i));
        // Create Box b2 with n+1 buttons.
        Box b2 = ...
        // Create Box b3 with n+2 buttons.
        Box b3 = ...
        // Create horizontal box b containing b1, b2, b3
        Box b = new Box(BoxLayout.X_AXIS);
        b.add(b1);
        b.add(b2);
        b.add(b3);
        Container cp = getContentPane();
        cp.add(b, BorderLayout.CENTER);
        pack();
    }
}

Simulate BoxLayout Manager in a JFrame

To simulate using a BoxLayout manager for a JFrame, create a Box and place it as the sole component of the JFrame:

```java
JFrame jf = new JFrame("title");
Box b = new Box(BoxLayout.X_AXIS);
Add components to b;
jf.add(b, BorderLayout.CENTER);
```

1. Start developing a GUI by changing an already existing one. A lot of details. Hard to get all details right when one starts from scratch and has little idea about the Java GUI package.
2. Showed how to place components in a GUI. Next class: how to "listen" to things like button clicks in a GUI.

Ensure BoxLayout layout manager: Place any number of components. They appear in the order added, taking only one row.

Checkers Example
Checkers Example

mainBox

boardBox

row

Square ...
Square ...

infoBox

row

JButton

JButton

JButton

JLabel

JLabel

Layout Manager for Checkers game has to process a tree

pack(): Traverse the tree, determining the space required for each component

boardBox: vertical Box
row: horizontal Box
Square: Canvas or JPanel
infoBox: vertical Box

the GUI future…

Rethinking GUIs

Big Data

* In 2011 we will have generated more data than from the beginning of history – Peter H. Lawrentyev global pulse Summit

Rethinking GUIs

Rethinking GUIs

Graphics Performance

Human Performance