Flickering and Double Buffering

Yung-Hsiang Lu
yunglu@purdue.edu
Flickering
No Flickering
Flickering

- Goal: continuously iterate through the three images
- Approach: draw background first then draw an image
- Problem: occasionally, only the background is shown on the screen without any of the three images
- Result: the character sometimes disappears
public void draw(Graphics g) {
    // draw background
    g.drawImage(bgImage, 0, 0, null);
    // draw image
    g.drawImage(anim.getImage(), 0, 0, null);
}

sent to screen
Possible Solutions

• erase and draw difference

⇒ find the differences between each pair of images is not trivial. Interleaving is still a problem.

• make draw atomic (i.e. cannot interleave)

⇒ When the screen refreshes and drawing is incomplete, the screen will appear black.
Double Buffering + Page Flipping

- Double buffering
  - buffer 1: working buffer for drawing, incomplete, invisible to user
  - buffer 2: completed, shown on screen
  - switching between the two buffers for frame updates

- Page flipping
  - the pixels are not copied
  - instead, a pointer switching between the two buffers
  - flipping between monitor refresh
java.awt.image

Class BufferStrategy

java.lang.Object
   | java.awt.image.BufferStrategy

Direct Known Subclasses:

public abstract class BufferStrategy
extends Object

The BufferStrategy class represents the mechanism with which to organize complex memory on a particular Canvas or
Window. Hardware and software limitations determine whether and how a particular buffer strategy can be implemented. These
limitations are detectible through the capabilities of the GraphicsConfiguration used when creating the Canvas or Window.

It is worth noting that the terms buffer and surface are meant to be synonymous: an area of contiguous memory, either in video
device memory or in system memory.
Double buffering:

```
************  ************
*            * --<>--    *
[To display] <----- * Front B * Show * Back B * <---- Rendering
*            * <-----    *
************  ************
```

Triple buffering:

```
[To display]  ************  ************  ************
*            *  --------+--------+------*    *
<----- * Front B * Show * Mid. B. * Back B * <---- Rendering
*            *  <-----+------+----*    *
************  ************  ************
```

Here is an example of how buffer strategies can be created and used:
window.createBufferStrategy(2);
BufferStrategy strategy = window.getBufferStrategy();
Graphics g = strategy.getDrawGraphics();
draw(g);
g.dispose();
strategy.show();
Screen Manager with BufferStrategy
Automatic Document Generation using Javadoc
public class ScreenManager {

    private GraphicsDevice device;

    /**
     * Creates a new ScreenManager object.
     */
    public ScreenManager() {
        GraphicsEnvironment environment =
                GraphicsEnvironment.getLocalGraphicsEnvironment(
                device = environment.getDefaultScreenDevice();
    }

    /**
     * Returns a list of compatible display modes for the
     */
    public List<Config> getDisplayModes() {
        return new ArrayList<Config>();
    }

    // Other methods...
import java.awt.*;
import java.awt.image.BufferStrategy;
import java.awt.image.BuffersedImage;
import java.lang.reflect.InvocationTargetException;
import javax.swing.JFrame;

/**
The ScreenManager class manages initializing and displaying
full screen graphics mode.

Finished building chap02 - 2D Graphics and Animation (javadoc)
```java
import javax.swing.JFrame;
import java.awt.image.BufferStrategy;
import java.awt.image.BufferedImage;
import java.lang.reflect.InvocationTargetException;

public class ScreenManager {

    private GraphicsDevice device;

    public ScreenManager() {
        GraphicsEnvironment environment = GraphicsEnvironment.getLocalGraphicsEnvironment();
        device = environment.getDefaultScreenDevice();
    }

    public List<DisplayMode> getDisplayModes() {
        return ...
    }
}

/**
The ScreenManager class manages initializing and displaying full screen graphic modes.
*/
```
```java
import java.awt.*;
import javax.swing.ImageIcon;

public class AnimationTest2 {
    public static void main(String args[]) {
        AnimationTest2 test = new AnimationTest2();
        test.run();
    }

    private static final DisplayMode POSSIBLE_MODES[] = {
        new DisplayMode(800, 600, 32, 0),
        new DisplayMode(800, 600, 24, 0),
        DisplayMode(800, 600, 16, 0),
        DisplayMode(640, 480, 32, 0),
        DisplayMode(640, 480, 24, 0),
        DisplayMode(640, 480, 16, 0)
    }

    static final long DEMO_TIME = 50000;

    ScreenManager screen;
    Image bgImage;
    Animation anim;

    void loadImages() {
        // Load images
        Image image = loadImage("images/background.jpg");
        Image layer1 = loadImage("images/layer1.png");
    }
```
Class ScreenManager

java.lang.Object

ScreenManager

public class ScreenManager extends Object

The ScreenManager class manages initializing and displaying full screen graphics modes.

Field Summary

private GraphicsDevice device

Constructor Summary

ScreenManager()
ScreenManager()

Creates a new ScreenManager object

## Method Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BufferedImage</td>
<td>createCompatibleImage(v, h, int transparency)</td>
<td>Creates an image compatible with the current display.</td>
</tr>
<tr>
<td>boolean</td>
<td>displayModesMatch(DisplayMode mode1, DisplayMode mode2)</td>
<td>Determines if two display modes &quot;match.&quot;</td>
</tr>
<tr>
<td>DisplayMode</td>
<td>findFirstCompatibleMode(DisplayMode[] modes)</td>
<td>Returns the first compatible mode in a list of modes.</td>
</tr>
<tr>
<td>DisplayMode[]</td>
<td>getCompatibleDisplayModes()</td>
<td>Returns a list of compatible display modes for the default device on the system.</td>
</tr>
<tr>
<td>DisplayMode</td>
<td>getCurrentDisplayMode()</td>
<td>Returns the current display mode.</td>
</tr>
<tr>
<td>JFrame</td>
<td>getFullScreenWindow()</td>
<td>Returns the window currently used in full screen mode.</td>
</tr>
<tr>
<td>Graphics2D</td>
<td>getGraphics()</td>
<td>Gets the graphics context for the display.</td>
</tr>
<tr>
<td>int</td>
<td>getHeight()</td>
<td>Returns the height of the window currently used in full screen mode.</td>
</tr>
</tbody>
</table>
import java.awt.*;
import java.awt.image.BufferStrategy;
import java.awt.image.BufferedImage;
import java.lang.reflect.InvocationTargetException;
import javax.swing.JFrame;

/**
 * The ScreenManager class manages initializing and displaying
 * full screen graphics modes.
 */

public class ScreenManager {

    private GraphicsDevice device;

    /**
     * Creates a new ScreenManager object.
     */
    public ScreenManager() {
        GraphicsEnvironment environment =
            GraphicsEnvironment.getLocalGraphicsEnvironment();
        device = environment.getDefaultScreenDevice();
    }

    /**
     * Returns a list of compatible display modes for the
     * default device on the system.
     */

```java
/*
 * public DisplayMode[] getCompatibleDisplayModes() {
 *     return device.getDisplayModes();
 * }
 *
 * /**
 * Returns the first compatible mode in a list of modes.
 * Returns null if no modes are compatible.
 * */
 * public DisplayMode findFirstCompatibleMode(
 *     DisplayMode modes[])
 * {
 *     DisplayMode goodModes[] = device.getDisplayModes();
 *     for (int i = 0; i < modes.length; i++) {
 *         for (int j = 0; j < goodModes.length; j++) {
 *             if (displayModesMatch(modes[i], goodModes[j])) {
 *                 return modes[i];
 *             }
 *         }
 *     }
 *     return null;
 * }
 *
 * /**
 * Returns the current display mode.
 * */
 * public DisplayMode getCurrentDisplayMode() {
 *     return device.getDisplayMode();
 * }
 */
```
/**
 * Determines if two display modes "match". Two display
 * modes match if they have the same resolution, bit depth,
 * and refresh rate. The bit depth is ignored if one of the
 * modes has a bit depth of DisplayMode.BIT_DEPTH_MULTI.
 * Likewise, the refresh rate is ignored if one of the
 * modes has a refresh rate of
 * DisplayMode.REFRESH_RATE_UNKNOWN.
 */

public boolean displayModesMatch(DisplayMode mode1,
       DisplayMode mode2) {
    if (mode1.getWidth() != mode2.getWidth() ||
        mode1.getHeight() != mode2.getHeight()) {
        return false;
    }

    if (mode1.getBitDepth() != DisplayMode.BIT_DEPTH_MULTI &&
        mode2.getBitDepth() != DisplayMode.BIT_DEPTH_MULTI &&
        mode1.getBitDepth() != mode2.getBitDepth()) {
        return false;
    }

    if (mode1.getRefreshRate() !=
        DisplayMode.REFRESH_RATE_UNKNOWN &&
        mode2.getRefreshRate() !=
        DisplayMode.REFRESH_RATE_UNKNOWN &&
        mode1.getRefreshRate() != mode2.getRefreshRate()) {
        return false;
    }

    return true;
}
return true;

/**
 * Enters full screen mode and changes the display mode.
 * If the specified display mode is null or not compatible
 * with this device, or if the display mode cannot be
 * changed on this system, the current display mode is used.
 */

public void setFullScreen(DisplayMode displayMode) {
    final JFrame frame = new JFrame();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setUndecorated(true);
    frame.setIgnoreRepaint(true);
    frame.setResizable(false);

    device.setFullScreenWindow(frame);

    if (displayMode != null &&
        device.isDisplayChangeSupported()) {
        try {
            device.setDisplayMode(displayMode);
        } catch (IllegalArgumentException ex) {
        }
    // fix for mac os x
    frame.setSize(displayMode.getWidth(),

YHL
ScreenManager.java

```java
    // avoid potential deadlock in 1.4.1_02
    try {
        EventQueue.invokeLater(new Runnable() {

            public void run() {
                frame.createBufferStrategy(2);
            }
        });
    } catch (InterruptedException ex) {
        // ignore
    } catch (InvocationTargetException ex) {
        // ignore
    }

    /**
     * Gets the graphics context for the display. The ScreenManager uses double
     * buffering, so applications must call update() to show any graphics drawn.
     * <p>
     * The application must dispose of the graphics object.
     */
    public Graphics2D getGraphics() {
        Window window = device.getFullScreenWindow();
        if (window != null) {
            BufferStrategy strategy = window.getBufferStrategy();
            return (Graphics2D) strategy.getDrawGraphics();
        } else {
```
else {
    return null;
}

/**
 * Updates the display.
 */
public void update() {
    Window window = device.getFullScreenWindow();
    if (window != null) {
        BufferStrategy strategy = window.getBufferStrategy();
        if (!strategy.contentsLost()) {
            strategy.show();
        }
    }

    // Sync the display on some systems.
    // (on Linux, this fixes event queue problems)
    Toolkit.getDefaultToolkit().sync();
}

/**
 * Returns the window currently used in full screen mode.
 * Returns null if the device is not in full screen mode.
 */
public JFrame getFullScreenWindow() {
    return (JFrame) device.getFullScreenWindow();
}
/**
 * Returns the width of the window currently used in full screen mode. Returns 0 if the device is not in full screen mode.
 */

public int getWidth() {
    Window window = device.getFullScreenWindow();
    if (window != null) {
        return window.getWidth();
    } else {
        return 0;
    }
}

/**
 * Returns the height of the window currently used in full screen mode. Returns 0 if the device is not in full screen mode.
 */

public int getHeight() {
    Window window = device.getFullScreenWindow();
    if (window != null) {
        return window.getHeight();
    } else {
        return 0;
    }
}
public void restoreScreen() {
    Window window = device.getFullScreenWindow();
    if (window != null) {
        window.dispose();
    }
    device.setFullScreenWindow(null);
}

/**
 * Creates an image compatible with the current display.
 */
public BufferedImage createCompatibleImage(int w, int h, int transapancy) {
    Window window = device.getFullScreenWindow();
    if (window != null) {
        GraphicsConfiguration gc = window.getGraphicsConfiguration();
        return gc.createCompatibleImage(w, h, transapancy);
    }
    return null;
}
Images may be stored in volatile memory for performance reasons.
AnimationTest2
Using ScreenManager
Using ScreenManager

POSSIBLE_MODES: an array of different DisplayMode

create ScreenManager

findFirstCompatibleMode(POSSIBLE_MODES)

setFullScreen(displayMode)

animationLoop

getGraphics
draw
dispose
update
import java.awt.*;
import javax.swing.ImageIcon;

public class AnimationTest2 {

    public static void main(String args[]) {
        AnimationTest2 test = new AnimationTest2();
        test.run();
    }

    private static final DisplayMode POSSIBLE_MODES[] = {
        new DisplayMode(800, 600, 32, 0),
        new DisplayMode(800, 600, 24, 0),
        new DisplayMode(800, 600, 16, 0),
        new DisplayMode(640, 480, 32, 0),
        new DisplayMode(640, 480, 24, 0),
        new DisplayMode(640, 480, 16, 0)
    };

    private static final long DEMO_TIME = 50000;
    private ScreenManager screen;
    private Image bgImage;
    private Animation anim;

    public void loadImages() {
        // load images
        bgImage = loadImage("images/background.jpg");
        Image player1 = loadImage("images/player1.png");
        Image player2 = loadImage("images/player2.png");
        Image player3 = loadImage("images/player3.png");
// create animation
anim = new Animation();
anim.addFrame(player1, 250);
anim.addFrame(player2, 150);
anim.addFrame(player1, 150);
anim.addFrame(player2, 150);
anim.addFrame(player3, 200);
anim.addFrame(player2, 150);
}

private Image loadImage(String fileName) {
    return new ImageIcon(fileName).getImage();
}

public void run() {
    screen = new ScreenManager();
    try {
        DisplayMode displayMode = 
            screen.findFirstCompatibleMode(POSSIBLE_MODES);
        screen.setFullScreen(displayMode);
        loadImages();
        animationLoop();
    } finally {
        screen.restoreScreen();
    }
}

public void animationLoop() {

}
public void animationLoop() {
    long startTime = System.currentTimeMillis();
    long currTime = startTime;

    while (currTime - startTime < DEMO_TIME) {
        long elapsedTime =
            System.currentTimeMillis() - currTime;
        currTime += elapsedTime;

        // update animation
        anim.update(elapsedTime);

        // draw and update screen
        Graphics2D g = screen.getGraphics();
        draw(g);
        g.dispose();
        screen.update();

        // take a nap
        try {
            Thread.sleep(20);
        } catch (InterruptedException ex) {
        }
    }
}

public void draw(Graphics g) {
// update animation
anim.update(elapsedTime);

// draw and update screen
Graphics2D g = screen.getGraphics();
draw(g);
g.dispose();
screen.update();

// take a nap
try {
    Thread.sleep(20);
} catch (InterruptedException ex) {
}

public void draw(Graphics g) {
    // draw background
    g.drawImage(bgImage, 0, 0, null);

    // draw image
    g.drawImage(anim.getImage(), 0, 0, null);
}
ECE 462
Object-Oriented Programming
using C++ and Java

Moving Image: Sprite

Yung-Hsiang Lu
yunglu@purdue.edu
Sprite
import java.awt.Image;

public class Sprite {

    private Animation anim;
    // position (pixels)
    private float x;
    private float y;
    // velocity (pixels per millisecond)
    private float dx;
    private float dy;

    /**
     * Creates a new Sprite object with the specified Animation.
     */
    public Sprite(Animation anim) {
        this.anim = anim;
    }

    /**
     * Updates this Sprite's Animation and its position based on the velocity.
     */
    public void update(long elapsedTime) {
        x += dx * elapsedTime;
        y += dy * elapsedTime;
        anim.update(elapsedTime);
    }
}
/**
 * Gets this Sprite's current x position.
 */
public float getX() {
    return x;
}

/**
 * Gets this Sprite's current y position.
 */
public float getY() {
    return y;
}

/**
 * Sets this Sprite's current x position.
 */
public void setX(float x) {
    this.x = x;
}

/**
 * Sets this Sprite's current y position.
 */
public void setY(float y) {
    this.y = y;
}
/**
 * Gets this Sprite's width, based on the size of the current image.
 */

public int getWidth() {
    return anim.getImage[].getWidth(null);
}

/**
 * Gets this Sprite's height, based on the size of the current image.
 */

public int getHeight() {
    return anim.getImage[].getHeight(null);
}

/**
 * Gets the horizontal velocity of this Sprite in pixels per millisecond.
 */

public float getVelocityX() {
    return dx;
}

/**
 * Gets the vertical velocity of this Sprite in pixels per millisecond.
 */
```java
public float getVelocityY() {
    return dy;
}

/**
 * Sets the horizontal velocity of this Sprite in pixels per millisecond.
 */
public void setVelocityX(float dx) {
    this.dx = dx;
}

/**
 * Sets the vertical velocity of this Sprite in pixels per millisecond.
 */
public void setVelocityY(float dy) {
    this.dy = dy;
}

/**
 * Gets this Sprite's current image.
 */
public Image getImage() {
    return anim.getImage();
}
```
SpriteTest1
import java.awt.*;

public class SpriteTest1 {

    public static void main(String args[]) {
        SpriteTest1 test = new SpriteTest1();
        test.run();
    }

    private static final DisplayMode POSSIBLE_MODES[] = {
        new DisplayMode(800, 600, 32, 0),
        new DisplayMode(800, 600, 24, 0),
        new DisplayMode(800, 600, 16, 0),
        new DisplayMode(640, 480, 32, 0),
        new DisplayMode(640, 480, 24, 0),
        new DisplayMode(640, 480, 16, 0)
    };

    private static final long DEMO_TIME = 10000;

    private ScreenManager screen;
    private Image bgImage;
    private Sprite sprite;

    public void loadImages() {
        // load images
        bgImage = loadImage("images/background.jpg");
        Image player1 = loadImage("images/player1.png");
// create sprite
Animation anim = new Animation();
anim.addFrame(player1, 250);
anim.addFrame(player2, 150);
anim.addFrame(player1, 150);
anim.addFrame(player2, 150);
anim.addFrame(player3, 200);
anim.addFrame(player2, 150);
sprite = new Sprite(anim);

// start the sprite off moving down and to the right
sprite.setVelocityX(0.2f);
sprite.setVelocityY(0.2f);

private Image loadImage(String fileName) {
    return new ImageIcon(fileName).getImage();
}

public void run() {
    screen = new ScreenManager();
    try {
        DisplayMode displayMode =
            screen.findFirstCompatibleMode(POSSIBLE_MODES);
        screen.setFullScreen(displayMode);
    }
```java
screen.setFullScreen(displayMode);
loadImages();
animationLoop();
}

finally {
    screen.restoreScreen();
}

public void animationLoop() {
    long startTime = System.currentTimeMillis();
    long currTime = startTime;

    while (currTime - startTime < DEMO_TIME) {
        long elapsedTime =
            System.currentTimeMillis() - currTime;
        currTime += elapsedTime;

        // update the sprites
        update(elapsedTime);

        // draw and update the screen
        Graphics2D g = screen.getGraphics();
        draw(g);
        g.dispose();
        screen.update();
    }
}
```
```java
// take a nap
try {
    Thread.sleep(20);
} catch (InterruptedException ex) {
}

public void update(long elapsedTime) {
    // check sprite bounds
    if (sprite.getX() < 0) {
        sprite.setVelocityX(Math.abs(sprite.getVelocityX()));
    }
    else if (sprite.getX() + sprite.getWidth() >=
             screen.getWidth()) {
        sprite.setVelocityX(-Math.abs(sprite.getVelocityX()));
    }
    if (sprite.getY() < 0) {
        sprite.setVelocityY(Math.abs(sprite.getVelocityY()));
    }
    else if (sprite.getY() + sprite.getHeight() >=
             screen.getHeight()) {
        sprite.setVelocityY(-Math.abs(sprite.getVelocityY()));
    }
}
```
```java
if (sprite.getY() < 0) {
    sprite.setVelocityY(Math.abs(sprite.getVelocityY()));
}
else if (sprite.getY() + sprite.getHeight() >=
    screen.getHeight())
{
    sprite.setVelocityY(-Math.abs(sprite.getVelocityY()));
}

// update sprite
sprite.update(elapsedTime);

public void draw(Graphics g) {
    // draw background
    g.drawImage(bgImage, 0, 0, null);

    // draw sprite
    g.drawImage(sprite.getImage(),
        Math.round(sprite.getX()),
        Math.round(sprite.getY()),
        null);
}
```
ECE 462
Object-Oriented Programming
using C++ and Java

Unified Modeling Language

Yung-Hsiang Lu
yunglu@purdue.edu
Unified Modeling Language UML

• **why to model?**
  – abstraction, ignore details, or multiple levels of details
  – identify participating objects
  – communicate with people
  – use tools to generate code, check correctness ...

• **why UML?**
  – language independent
  – platform independent
  – international standard
  – expressive (state, sequence, time, interface ...)
  – tool rich (UML → code, code → UML ... )
Why Model With UML?

Contributed by Kris Richards and Cindy Castillo, maintained by Cindy Castillo

Last Updated: June 2007

Abstract

The purpose of the Unified Modeling Language (hereinafter referred to as UML) is to provide a language-independent and platform independent modeling notation. UML tools are versatile as the UML is foundational. This article serves as a primer for the basic concepts of UML while providing an understanding of the purposes of modeling. It is not meant to be a how-to manual, but links are provided, where appropriate, to take you to well-executed tutorials to illustrate the steps on how to use the UML features offered in the NetBeans...
NetBeans UML® Project

The UML Modeling project provides UML modeling features to the NetBeans IDE. UML modeling allows analysts and designers to design applications using a standard modeling language. Developers are then able to generate source code from the UML model and update the model from changes made in their source code.

What's New

The UML team is in active development for NetBeans 6.1.next. See the UML developer wiki for the latest information.

For UML Users

If you are an UML user and want to learn more about the UML modules and download full releases for Windows, Linux, or Solaris, please visit the product page.

Bug reporting and Feature Tracking

username:

password:

Login

Forgot your password?
UML Example in Netbeans
1. Choose Project
2. ...

Categories:
- Java
- Web
- Enterprise
- Mobility
- UML
- Ruby
- C/C++
- NetBeans Modules
- Samples

Description:
Choose Project

Categories:
- NetBeans Modules
  - Samples
    - Java
    - Web
  - Web Services
  - Enterprise
  - Mobility
  - UML
  - Identity Blueprints
  - Ruby
  - C/C++
  - NetBeans Modules

Projects:
- UML Bank. App Sample

Description:
Sample application that shows how to reverse engineer the source code of an existing Java application into a UML Modeling project.
### New Project

#### Steps
1. Choose Project
2. Name and Location

#### Name and Location

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Project Name</td>
<td>UMLBankAppSample</td>
</tr>
<tr>
<td>Project Location</td>
<td>C:\ylugu\java</td>
</tr>
<tr>
<td>Java Project Folder</td>
<td>C:\ylugu\java\UMLBankAppSample</td>
</tr>
<tr>
<td>UML Project Name</td>
<td>UMLBankAppSample-Model</td>
</tr>
<tr>
<td>UML Project Folder</td>
<td>C:\ylugu\java\UMLBankAppSample-Model</td>
</tr>
</tbody>
</table>

[Image of the New Project window with filled-in fields]
UML Modeling Module: Bank App Sample

This document is a quick overview for the UML Bank App sample. The projects included in the sample are provided as a means of introducing to the Reverse Engineering feature of the NetBeans IDE. The sample Java project has a default name of UMLBankAppSample and the model has the default name of UMLBankAppSample-Model. The sample Java project was engineered to generate the sample UML Modeling project and its elements.

To open the sample UML Modeling project:

1) In the IDE's Projects window, navigate to the sample UML project. If you used the default project name values, the UML Modeling project will be UMLBankAppSample-Model.

2) Expand the UMLBankAppSample-Model > Model > bankpack node. The elements displayed under this node were generated using the source files you can find under the UMLBankAppSample > Source node.

3) Expand the UMLBankAppSample-Model > bankpack > BankAccount node. Notice the attributes, operations, and relationship nodes that were automatically generated.
What Does a Model Say?

- relationships among objects
  - which objects participate in an activity
  - actions and interfaces
  - transitions of attributes
- sequence of actions
- specialization, composition, ownership
- quantities

In addition, one critical value of a model is to help a developer think before doing.
Modeling a Bank

• objects:
  – people: customers, tellers, bank managers, loan evaluators, ATM maintainers ...
  – data: accounts
  – properties: branch offices, ATM machines, furniture ...
• actions: deposit, withdraw, apply for loans, approve or reject applications, collect money from ATM ...
• relationships among objects
  – customer can deposit, withdraw cash, or talk to teller
  – customer cannot talk to ATM maintainer
  – ATM maintainer cannot approve or reject loan applications
  – ATM can accept deposit; desk cannot ...
customer

deposit

withdraw
reject loan application

loan evaluator

reject loan application
Model States and Transitions

• Many objects' behaviors depend on the values (i.e. state) of the objects' attributes.
  – age $\Rightarrow$ vote
  – account balance $\Rightarrow$ withdraw
  – available credit $\Rightarrow$ purchase

• Objects' behaviors often follow strict orders based on the transitions of the attributes.
  – vending machine must accept payment before returning changes
  – customer must open an account before withdrawing money
  – a user must enter the password before checking email
  – a bank customer must insert the ATM card and enter PIN before deposit
YHL UML 15

NetBeans IDE 6.0

Projects

UMLProject1

Model

Diagrams

Imported El...
Sequence of Actions

- State diagrams do not express the objects involved.
- For example, the previous diagram does not specify the necessity of a customer and an ATM machine.
- In fact, three objects are involved
  - customer
  - ATM
  - account (and the balance)
Class Diagram
**Class Diagram**

**BankAccount**
- **Attributes**
  - private int balance
  - private String name
- **Operations**
  - public BankAccount(String name)

**CD**
- **Attributes**
  - private float rate
  - private Date maturityDate
- **Operations**
  - public CD(float r, Date d)
Quantity
Start a UML Project in Netbeans
NetBeans IDE 6.1 Features

» Overview

» Base IDE
  Editor
  Version Control & Developer Collaboration

» Java SE
  Swing GUI Builder
  Profiler
  Debugger

» Web & Java EE
  Ajax

UML
Steps
1. Choose Project
2. ...

Choose Project
Categories:
- Java
- Web
- Enterprise
- Mobility
- IMI
- Ruby
- C/C++
- NetBeans Modules
- Samples

Projects:
- Platform-Independent Model
- Java-Platform Model
- Reverse Engineered Java-Platform Model

Description:
Creates a platform-dependent modeling project in the system with empty containers for your model and elements. As you create and populate your model, **Java modeling (business) rules** are applied to the model.