Games – A Basic Intro

Chapter 15 - Lecture Slides
The essentials

- Player control
  - 
- Game world
  - 
  - 
  - 
  - 
- Brains (aka. Artificial Intelligence)
  - 
    - Smart & unpredictable movements by bad guys
    - Storytelling

Design

- Games require forethought in design
- Modularize functionality into “objects” (classes)
  - Player control
  - Game world
    -
    -
    -
  - Brains
Player Control
Player Control

- Keyboard, arrow keys, joystick, mouse
- Based on event, change Game world
  - Sprite
  - Affects
    - Hit/collide/shoot/push/etc.
Player Control

- Collisions
  - Player with object
  - Player with wall
  - Player with enemy

- How detect collisions?
Player Control

- How detect collisions?
  - Bounding boxes

- What to do with a collision?
  - Move entity back to previous position?
  - Use physics to determine a bounce?
Game World

- Backdrop
- Walls
- Objects
  - 
  - 
- Entities
  - 
  - 
  - 
Backdrop

- Type
  - 
  - 
  - 

- Need to be redrawn when something moves
  - Object picked up
  - Entity moves
  - Something disappears
  - Etc.
Objects

- Consumables, pick-up-ables
- Maintain info
  - Info (gives life/health is food/weapon/etc)
  - Amount of [units] (amt of life/food/ strength of weapon)
Drawings

- Loading images for game
  - Use MediaTracker
  - Can do other things while images load
- MediaTracker

```java
import java.awt.*;

MediaTracker tracker;
tracker = new MediaTracker(this);
    // Load images and add to the tracker
Image img = getImage( getDocumentBase(), "picture.jpg" );
tracker.addImage(img, 1);    // specify a unique # for each image
    // Wait until all images have loaded
    // (or display progress indicator while waiting)
tracker.waitForAll( );
```

(c) 2008 E.S.Boese All Rights Reserved.
Timings for Redrawing

- Careful on your animation speeds
  -
  -

“Do not squander time, for that is the stuff life is made of.”
-- Benjamin Franklin
Redrawing

- 3 Ways to redraw the screen
  - Clear the screen, then redraw
    - Complicated to figure out
  - Draw new screen on an offscreen, then redraw full screen on top of current screen
Brains
Brains

- Physics
  - Ball bounces,
  - \((x,y)\) based on speed and forward jump,

- Intelligence
Example: Breakout
Example: Breakout

- Set of bricks to be cleared by the user
  - Keep track of locations of ones left
  - Remove block when ball collides with
  - Calculate smart ball-bounce using physics

- Ball
  - Keep track of location, speed, direction
  - Calculate smart ball-bounce when colliding with bricks/wall/paddle

- Paddle
  - Keep track of location, speed
  - Respond to user events (arrow keys, joystick)
Example: Breakout

- **Phase 0**
  - Create a ball
  - **Does not extend** `JApplet`! **Helper class**
    - Doesn’t extend anything at all!
    - Rely on paint method to draw it
import java.awt.Color; import java.awt.Graphics; class Ball
{
    int x, y, size, speed;
    int dirX, dirY;
    int appletWdt, appletHgt;
    public Ball( int _x, int _y, int _size, int _speed, int w, int h )
    {
        x = _x; y = _y; size = _size; speed = _speed; dirX = 1; dirY = 1;
        appletWdt = w; appletHgt = h;
    }
    public void paint( Graphics g )
    {
        g.setColor( Color.BLUE );
        g.fillOval( x, y, size, size );
    }
    public void move( )
    {
        x = x + speed * dirX;
        y = y + speed * dirY;
        if ( x < 0 )
            dirX = 1;
        else if ( x > appletWdt )
            dirX = -1;
        if ( y < 0 )
            dirY = 1;
        else if ( y > appletHgt )
            dirY = -1;
    }
}

(c) 2008 E.S.Boese All Rights Reserved.
import java.awt.Color;
import java.awt.Graphics;
class Ball {

    int x, y, size, speed;
    int dirX, dirY;
    int appletWdt, appletHgt;

    public Ball( int _x, int _y, int _size, int _speed, int w, int h ) {
        x = _x; y = _y; size = _size; speed = _speed; dirX = 1; dirY = 1;
        appletWdt = w; appletHgt = h;
    }

    public void paint( Graphics g ) {
        g.setColor( Color.BLUE );
        g.fillOval( x, y, size, size );
    }

    public void move( ) {
        x = x + speed * dirX;
        y = y + speed * dirY;
        if ( x < 0 )
            dirX = 1;
        else if ( x > appletWdt )
            dirX = -1;
        if ( y < 0 )
            dirY = 1;
        else if ( y > appletHgt )
            dirY = -1;
    }
}

Keep track of
• coordinates x,y
• size of the ball
• speed
• direction (in x, y)
• boundaries

(c) 2008 E.S.Boese All Rights Reserved.
import java.awt.Color; import java.awt.Graphics;
class Ball {
    int x, y, size, speed;
    int dirX, dirY;
    int appletWdt, appletHgt;
    public Ball( int _x, int _y, int _size, int _speed, int w, int h )
    {
        x = _x; y = _y; size = _size; speed = _speed; dirX = 1; dirY = 1;
        appletWdt = w; appletHgt = h;
    }
    public void paint( Graphics g )
    {
        g.setColor( Color.BLUE );
        g.fillOval( x, y, size, size );
    }
    public void move( )
    {
        x = x + speed * dirX;
        y = y + speed * dirY;
        if ( x < 0 )
            dirX = 1;
        else if ( x > appletWdt )
            dirX = -1;
        if ( y < 0 )
            dirY = 1;
        else if ( y > appletHgt )
            dirY = -1;
    }
}

Constructors
Called when another class calls "new Ball"
e.g.
Ball myball = new Ball( 0, 0, 10, 5, 100, 100 );
Constructors are used to initialize all your instance variables (e.g. x, y, size, speed, etc.)
import java.awt.Color;
import java.awt.Graphics;
class Ball {
    int x, y, size, speed;
    int dirX, dirY;
    int appletWdt, appletHgt;
    public Ball( int _x, int _y, int _size, int _speed, int w, int h )
    {
        x = _x; y = _y; size = _size; speed = _speed; dirX = 1; dirY = 1;
        appletWdt = w; appletHgt = h;
    }
    public void paint( Graphics g )
    {
        g.setColor( Color.BLUE );
        g.fillOval( x, y, size, size );
    }
    public void move( )
    {
        x = x + speed * dirX;
        y = y + speed * dirY;
        if ( x < 0 )
            dirX = 1;
        else if ( x > appletWdt )
            dirX = -1;
        if ( y < 0 )
            dirY = 1;
        else if ( y > appletHgt )
            dirY = -1;
    }
}

How could you change the class to get custom-colored balls?
import java.awt.Color;
import java.awt.Graphics;
class Ball {
    int x, y, size, speed;
    int dirX, dirY;
    int appletWdt, appletHgt;
    public Ball( int _x, int _y, int _size, int _speed, int w, int h )
    {
        x = _x; y = _y; size = _size; speed = _speed; dirX = 1; dirY = 1;
        appletWdt = w; appletHgt = h;
    }
    public void paint( Graphics g )
    {
        g.setColor( Color.BLUE );
        g.fillOval( x, y, size, size );
    }
    public void move( )
    {
        x = x + speed * dirX;
        y = y + speed * dirY;
        if ( x < 0 )
            dirX = 1;
        else if ( x > appletWdt )
            dirX = -1;
        if ( y < 0 )
            dirY = 1;
        else if ( y > appletHgt )
            dirY = -1;
    }
}

Other program that creates a Ball instance will call our move method during each time interval

Here we change the x,y coordinates based on the speed, and if we hit the boundaries, change direction
Example: Breakout

- Phase 1
  - Create the set of bricks
Example: Breakout

- Phase 1
  - Create the set of bricks
  - Keep track of the bricks in an `ArrayList`
  - Store `Rectangle` objects in the list. Each one represents the bounding area of a brick.
  - To display the bricks, go through the `ArrayList` and draw a filled rectangle for each Rectangle object in the list
Example: Breakout

- **ArrayList**
  - Create:
    ```java
    ArrayList <Rectangle> list;
    list = new ArrayList <Rectangle>( );
    ```
  - Add to the list
    ```java
    list.add( anyObject);
    e.g.:  list.add( new Rectangle( 10, 20, 100, 10 ) );
    ```
  - Go through the list (similar to an array)
    ```java
    for( int i=0; i<list.size( ); i++ )
        Rectangle r = list.get(i);
    ```
Example: Breakout

- Phase 1

```java
import java.awt.*;
import javax.swing.*;
import java.util.*;
public class BreakoutPart1 extends JApplet {
    ArrayList <Rectangle> blocks;  
    int width, height;  // dimensions of applet
    int numBlocks;    // number of blocks horizontally
    int numRows;     // number of rows - 1 easy, 5 harder
}
```
Example: Breakout

- Phase 1

```java
public void init() {
    numBlocks = 10;
    numRows = 3;
    blocks = new ArrayList<Rectangle>();
    width = getWidth();
    height = getHeight();
    buildBlocks();
}
```
Example: Breakout

Phase 1

```java
public void buildBlocks( )
{
    int sizeOfBlock = width / numBlocks;
    int heightOfBlock = 15;
    for( int rows=0; rows<width; rows += sizeOfBlock )
        for( int cols=0; cols<numRows*heightOfBlock;
            cols += heightOfBlock )
            {
                Rectangle r = new Rectangle( rows, 80+cols,
                                            sizeOfBlock-2, heightOfBlock-2 );
                blocks.add(r);
            }
}
```
Example: Breakout

Phase 1

```java
public void paint( Graphics g ) {
    g.clearRect(0,0,width,height); // clear screen
    g.setColor( Color.RED );
    // color remaining blocks
    for (int i=0; i<blocks.size(); i++) {
        Rectangle r = blocks.get(i);
        g.fillRect( r.x, r.y, r.width, r.height );
    }
    g.setColor( Color.BLACK );
}
```

(c) 2008 E.S.Boese All Rights Reserved.
Example: Breakout

- Phase 2
  - Create a bouncing ball
Example: Breakout

- Phase 2

```java
public class BreakoutPart2 extends JApplet implements Runnable {
    ArrayList blocks;
    int width, height; // dimensions of applet
    int numBlocks; // number of blocks horizontally
    int numRows; // number of rows - 1 each, 5 harder
    // Part II
    Thread thread;
    Ball ball;
```
Example: Breakout

Phase 2

```java
public void init() {
    numBlocks = 10;  numRows = 3;
    blocks = new ArrayList();
    width = getWidth();
    height = getHeight();
    buildBlocks();

    // PART II
    ball = new Ball( 50, 120, 15, 5, width, height );
    thread = new Thread(this);
    thread.start();
}
```
Example: Breakout

■ Phase 2

```java
public void run() {
    while (true) {
        ball.move();
        checkForCollision();
        repaint();
        try {
            Thread.sleep(15);
        } catch (Exception ex) {
        }
    }
}
```
Example: Breakout

- **Phase 2**

```java
public void checkForCollision() {
    Rectangle ballR = new Rectangle(ball.x, ball.y, ball.size, ball.size);
    for (int i = 0; i < blocks.size(); i++) {
        Rectangle r = blocks.get(i);
        if (r.intersects(ballR)) {
            blocks.remove(r);
            getGraphics().clearRect(r.x, r.y, r.width, r.height);
            ball.dirX = -1 * ball.dirX;
            ball.dirY = -1 * ball.dirY;
            return;
        }
    }
}
```
Example: Breakout

- Phase 2
- No change
  - buildBlocks
  - update
  - paint
Example: Breakout

- Phase 3
  - Paddle Action
public class BreakoutPart3 extends JApplet
    implements Runnable, KeyListener
{
    ArrayList<Rectangle> blocks;
    int width, height; // dimensions of applet
    int numBlocks; // number of blocks horizontally
    int numRows; // number of rows - 1 each, 5 harder
    // Part II
    Thread thread;
    Ball ball;
    // Part III
    Rectangle paddle;
    int speed;

    Bounding box for paddle – not a component we add to
    applet. Instead, we’ll just draw in
    place and check for
    collisions.
Example: Breakout

Phase 3

```java
public void init( )
{
    numBlocks = 10;       numRows = 3;
    blocks = new ArrayList<Rectangle>( );
    width = getWidth( ); height = getHeight( ); buildBlocks( );
    // Part III
    paddle = new Rectangle( 50, height-30, 50, 10 );
    addKeyListener( this );
    speed = 10;
    setFocusable(true);
    // PART II
    ball = new Ball( 50, 120, 15, 5, width, height );
    thread = new Thread(this);
    thread.start( );
}
```

Enable focus to be able to listen for key events
Example: Breakout

- **Phase 3**

```java
public void checkForCollision() {
    Rectangle ballR = new Rectangle(ball.x, ball.y, ball.size, ball.size);
    for(int i=0; i<blocks.size(); i++) {
        Rectangle r = blocks.get(i);
        if (r.intersects(ballR)) {
            blocks.remove(r);
            getGraphics().clearRect(r.x, r.y, r.width, r.height);
            ball.dirX = -1 * ball.dirX;
            ball.dirY = -1 * ball.dirY;
            return;
        }
    }
    // Part III
    if (ballR.intersects(paddle)) {
        ball.dirX = -1 * ball.dirX;
        ball.dirY = -1 * ball.dirY;
    }
}
```

(c) 2008 E.S.Boese All Rights Reserved.
public void keyTyped( KeyEvent ke ) { }
public void keyReleased( KeyEvent ke ) { }
public void keyPressed( KeyEvent ke ) {
    int code = ke.getKeyCode( );
    getGraphics( ).clearRect( paddle.x, paddle.y, paddle.width, paddle.height);
    if ( code == KeyEvent.VK_LEFT ) {
        int x = paddle.x;
        x -= speed; paddle.x = x;
    } else if ( code == KeyEvent.VK_RIGHT ) {
        int x = paddle.x;
        x += speed;
        x += speed;
        paddle.x = x;
    }
}
Example: Breakout

- Phase 3
- Unchanged
  - run
  - paint
  - update
  - buildBlocks
Example: Breakout

- Phase 4 (final version)
  - Remove flicker
    - Draw screen to a buffered image
    - Draw buffered image to screen
Example: Breakout

- Phase 4 (final version)

```java
public class Breakout extends JApplet
    implements Runnable,
   KeyListener
{
    ...
    // Part IV
    BufferedImage buffer;
```
Example: Breakout

- Phase 4 (final version)

```java
public void init() {
    ...
    // Part IV
    buffer = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
```
Example: Breakout

- Phase 4 (final version)

```java
public void paint( Graphics g ) {
    Graphics bg = buffer.getGraphics();
    bg.fillRect(0,0,width,height); //ball.paint( getGraphics( ) );
    ball.paint( bg );
    // g.setColor( Color.RED );
    for (int i=0; i<blocks.size(); i++ ) {
        Rectangle r = blocks.get(i);
        bg.fillRect( r.x, r.y, r.width, r.height );
    }
    bg.setColor( Color.BLACK );
    bg.fillRect( paddle.x, paddle.y, paddle.width, paddle.height );
    g.drawImage(buffer,0,0,this);
}
```

Don’t clear the screen – causes flicker.
Example: Breakout

- Phase 4 (final version)

```java
public void paint( Graphics g ) {
    Graphics bg = buffer.getGraphics();
    bg.fillRect(0,0,width,height);
    //ball.paint( getGraphics( ) );
    ball.paint( bg );
    // g.setColor( Color.WHITE );
    // g.fillRect(0,0,width,height);
    bg.setColor( Color.RED );
    for (int i=0; i<blocks.size(); i++) {
        Rectangle r = blocks.get(i);
        bg.fillRect( r.x, r.y, r.width, r.height );
    }
    bg.setColor( Color.BLACK );
    bg.fillRect( paddle.x, paddle.y, paddle.width, paddle.height );
    g.drawImage(buffer,0,0,this);
}
```

Call Ball’s paint method with the buffered image Graphics object instead of the applet’s Graphics object.
public void paint( Graphics g )
{
    Graphics bg = buffer.getGraphics( );
    bg.fillRect(0,0,width,height);
    ball.paint( bg );
    bg.setColor( Color.RED );
    for (int i=0; i<blocks.size( ); i++ )
    {
        Rectangle r = blocks.get(i);
        bg.fillRect( r.x, r.y, r.width, r.height );
    }
    bg.setColor( Color.BLACK );
    bg.fillRect( paddle.x, paddle.y, paddle.width, paddle.height );
    g.drawImage(buffer,0,0,this);
}
Example: Breakout

- Phase 4 (final version)

```java
public void paint( Graphics g ) {
    Graphics bg = buffer.getGraphics();
    bg.fillRect(0,0,width,height);
    ball.paint(bg);
    bg.setColor( Color.RED );
    for (int i=0; i<blocks.size(); i++)
    {
        Rectangle r = (Rectangle )blocks.get(i);
        bg.fillRect( r.x, r.y, r.width, r.height );
    }
    bg.setColor( Color.BLACK );
    bg.fillRect( paddle.x, paddle.y, paddle.width, paddle.height );
    g.drawImage(buffer,0,0,this);
}
```

Draw the buffered image to the applet screen

(c) 2008 E.S.Boese All Rights Reserved.
There are *many* ways to implement

- One big base image, only display a portion at a time. Use Polygon to determine intersections
- Individual images for each ‘room’
- Scrolling backdrop
- Break ‘room’ into small squares as an Object with pattern image that can be either entered or bounce off
Dungeon Games

Two ways to do the Graphics *(based on what we know)*

- Custom **JComponents** we create and define size/location
  - `.setLocation(x, y)`
  - `.setBounds(x, y, width, height)`
- Paint to an off-screen **BufferedImage** then redraw on screen
  - Our example
**Dungeon Games**

- **Example:** BufferedImage
- **Define the borders – region player can move**
  - Rectangle or Polygon
    - `rectangle.intersects( polygon )`

```java
ArrayList walls = new ArrayList();
public void buildWalls() {
    // create Rectangle objects and store in ArrayList walls
    int wallThickness = 10;
    walls.add( new Rectangle( 0, 0, width, wallThickness ) ); // top
    walls.add( new Rectangle( 0, 0, wallThickness, height ) ); // left
    walls.add( new Rectangle( 50, height-wallThickness-120, wallThickness, 70 ) ); // bottom wall structure
    walls.add( new Rectangle( 50, height-wallThickness-120, width-150, wallThickness ) ); // vert on left
    walls.add( new Rectangle( 50, height-wallThickness-120, width-150, wallThickness ) ); // horiz top
    walls.add( new Rectangle( 50, height-wallThickness-50, width-120, wallThickness ) ); // horiz bot
    walls.add( new Rectangle( 50, height-wallThickness-50, width-120, wallThickness ) ); // vert on rt
    walls.add( new Rectangle( 200, height-230, wallThickness, 130 ) ); // vert on rt
}
```

(c) 2008 E.S.Boese All Rights Reserved.
Dungeon Games

- Order of drawing graphics is important

```java
public void run() {
    paintWalls();
    screenBuf = createBufImage( width, height );
    Graphics gapplet = (Graphics2D )screenBuf.getGraphics( );
    while( true )
    {
        // first draw background on buffer
        gapplet.drawImage( backgroundBuf, 0,0, this );
        // then draw player on buffer
        player.paintComponent( gapplet );
        repaint( ); // draw buffer to applet
        try {
            Thread.sleep(10);
        } catch( Exception ex ) { stop( ); }
    }
}
```
Dungeon Games

- Add Items (coins/weapon/bananas)

```java
public class Thing {
    Image image;
    int x, y;
    boolean visible = false;

    public Thing(Image i) {
        image = i;
        x = 0;
        y = 0;
    }

    public void setVisible(boolean b) {
        visible = b;
    }

    public boolean isVisible() {
        return visible;
    }

    public Rectangle getDimensions() {
        return new Rectangle(x, y, image.getWidth(null), image.getHeight(null));
    }

    public void paintComponent(Graphics g) {
        if (visible)
            g.drawImage(image, x, y, null);
    }
}
```

(c) 2008 E.S.Boese All Rights Reserved.
Summary

- Player Control
- Game world
- Brains