

# Introduction to Jam's Video Game Package

# The Plan

- ❖ **Scope of Video Game Package**
- ❖ **Basic Design of the Video Game Package**
- ❖ **Steps to making a game**
- ❖ **How the Pong was made**

# Scope of the Video Game Package

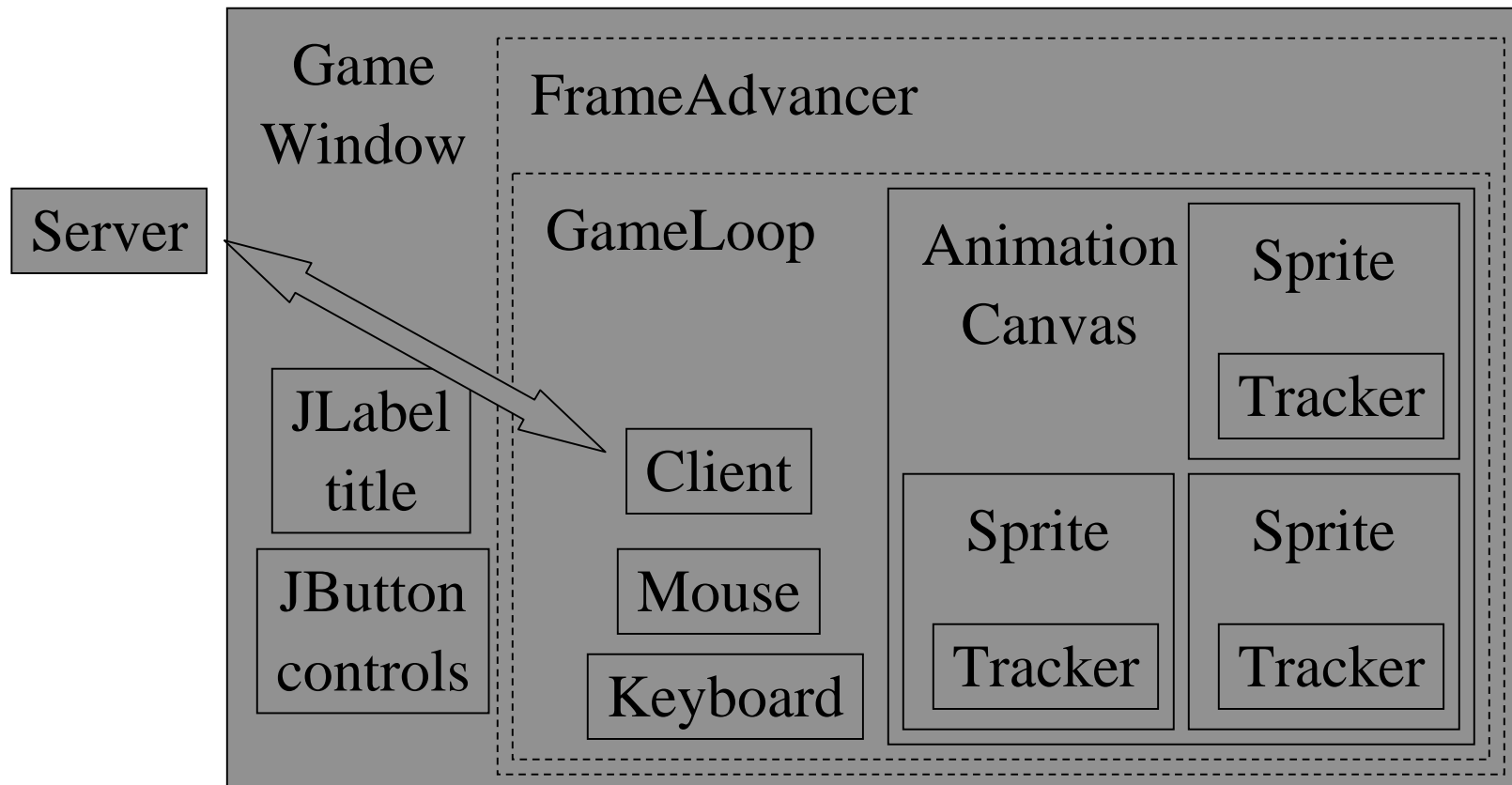
## ❖ Goals of Jam's Video Game Package

- ❑ Simple to use
- ❑ Reasonably fast
- ❑ Designed to be examined and modified for academic purposes in computer science courses

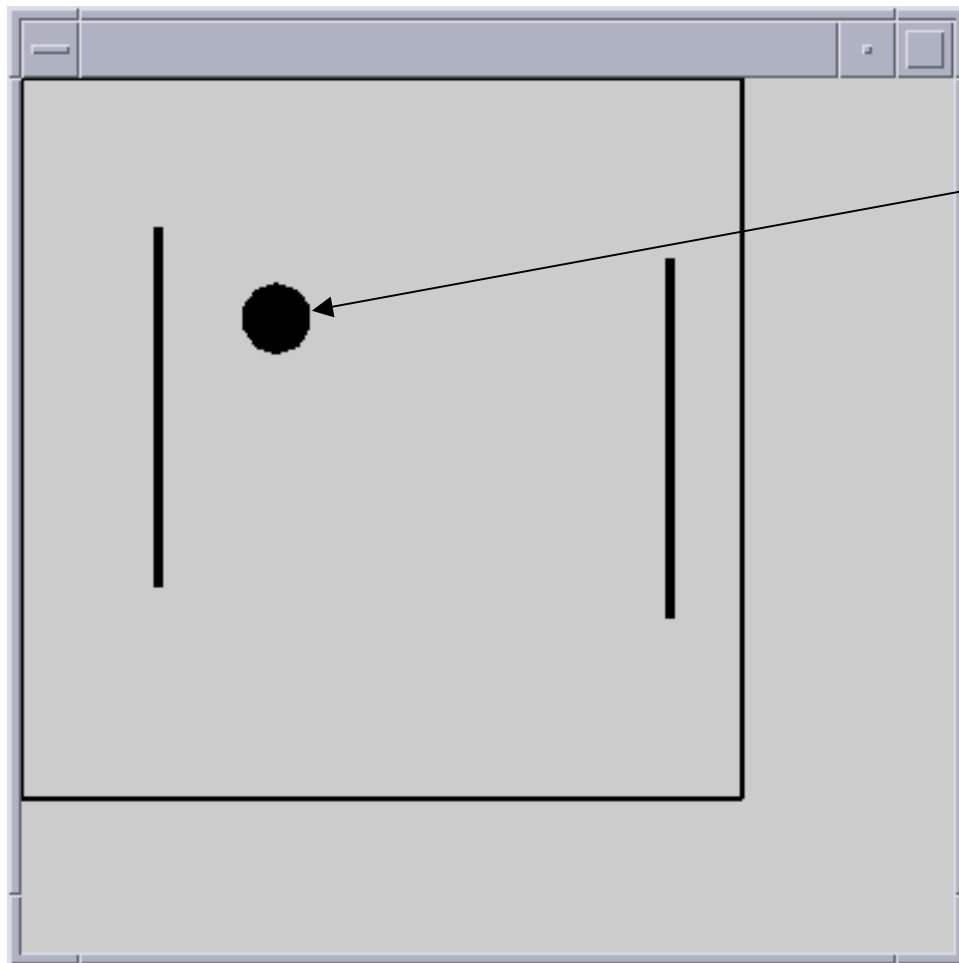
## ❖ What Jam's package is not designed for

- ❑ High speed animation/user input
- ❑ Scripting games
- ❑ 3D

# Basic Design of the Video Game Package



# Basic Design of the Video Game Package



Tracker

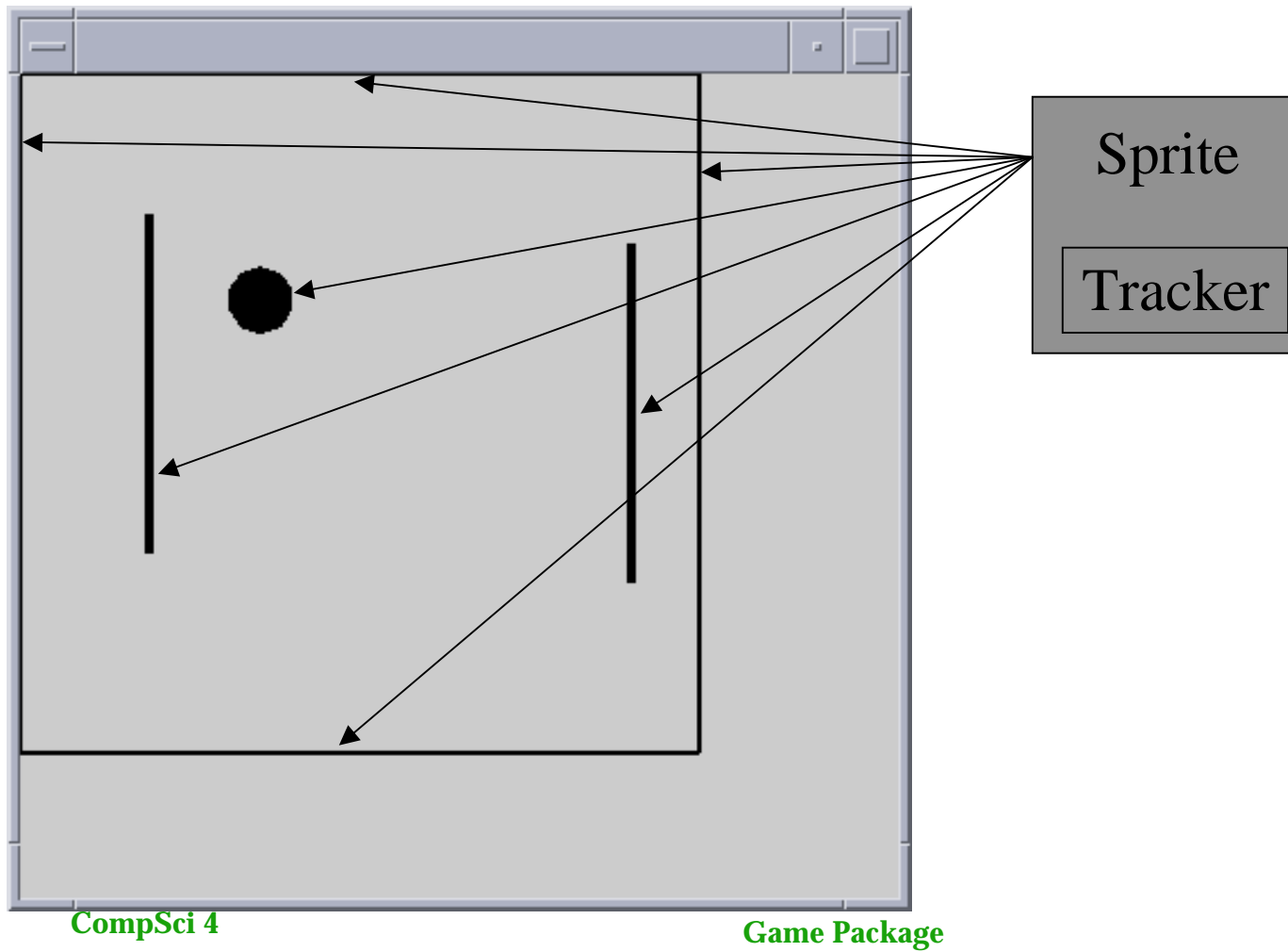
The Tracker is what  
Makes the BallSprite move

# Basic Design of the Video Game Package

Tracker

- ❖ **Point2D.Double getLocation()**
- ❖ **double getScaleFactor()**
- ❖ **double getRotationAddition()**
- ❖ **void advanceTime(double time)**

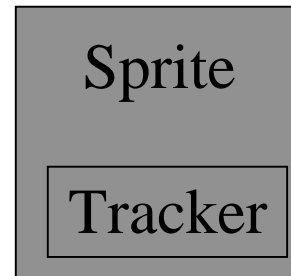
# Basic Design of the Video Game Package



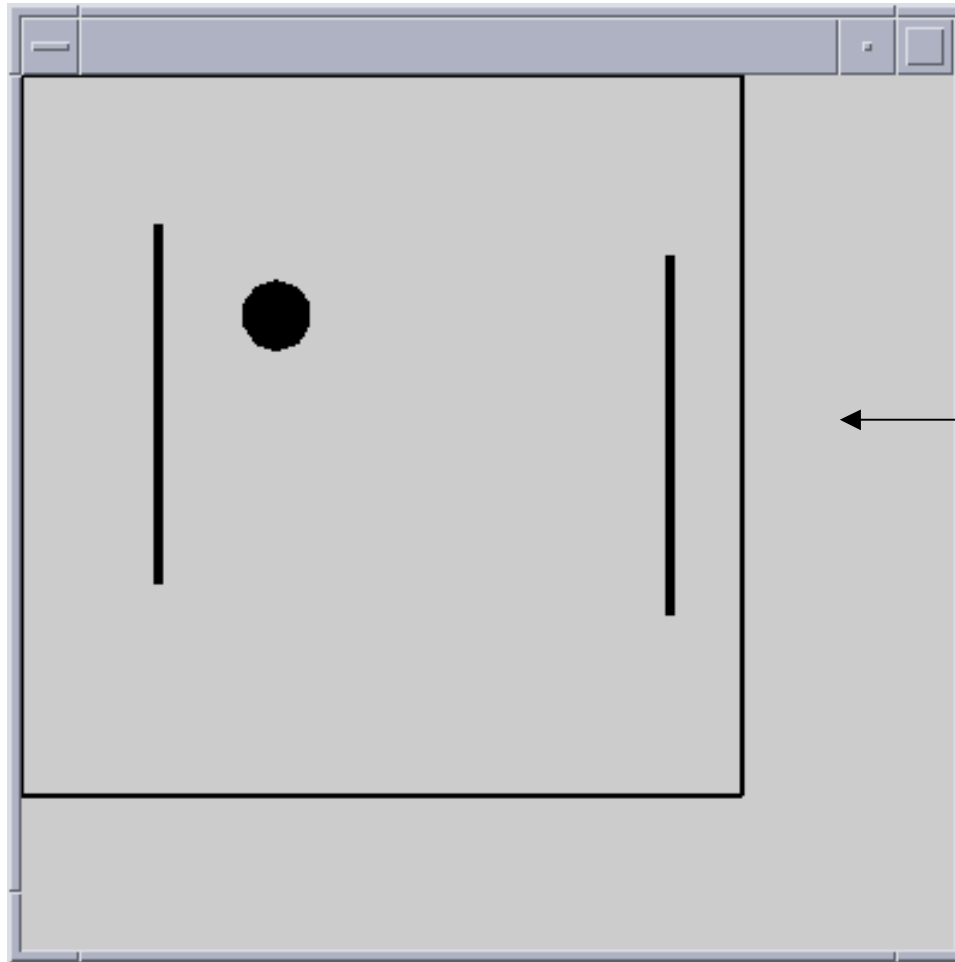
# Basic Design of the Video Game Package

**Has instance variables, and mutator and accessor methods for:**

- ❖ **Shape**
- ❖ **Location**
- ❖ **Size**
- ❖ **Rotation**
- ❖ **Color**

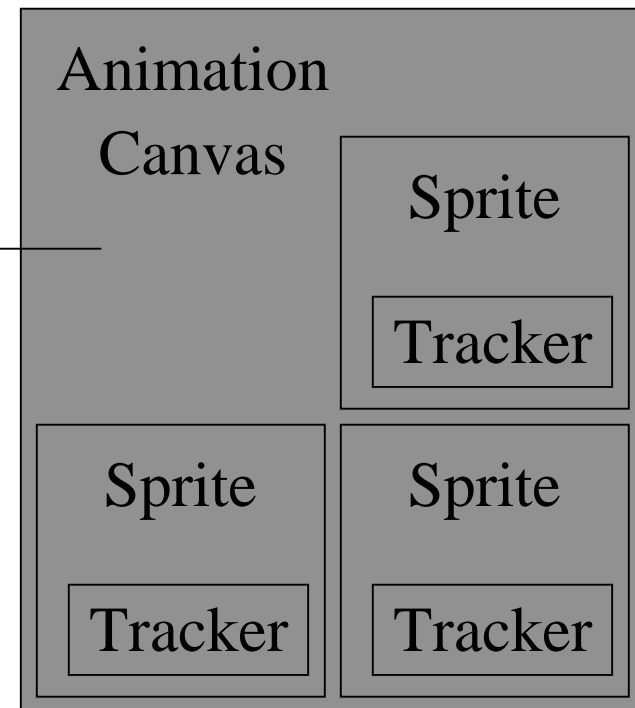


# Basic Design of the Video Game Package



CompSci 4

Game Package

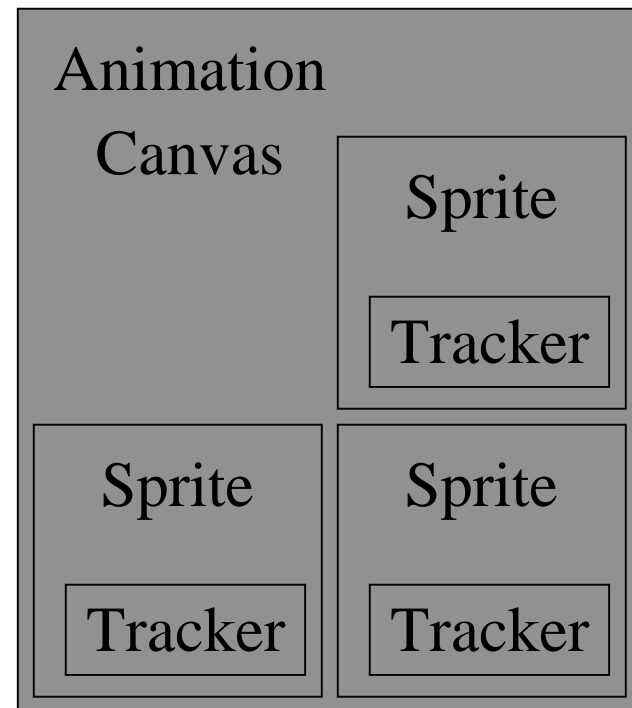


4.9

# Basic Design of the Video Game Package

**AnimationCanvas  
is a JPanel**

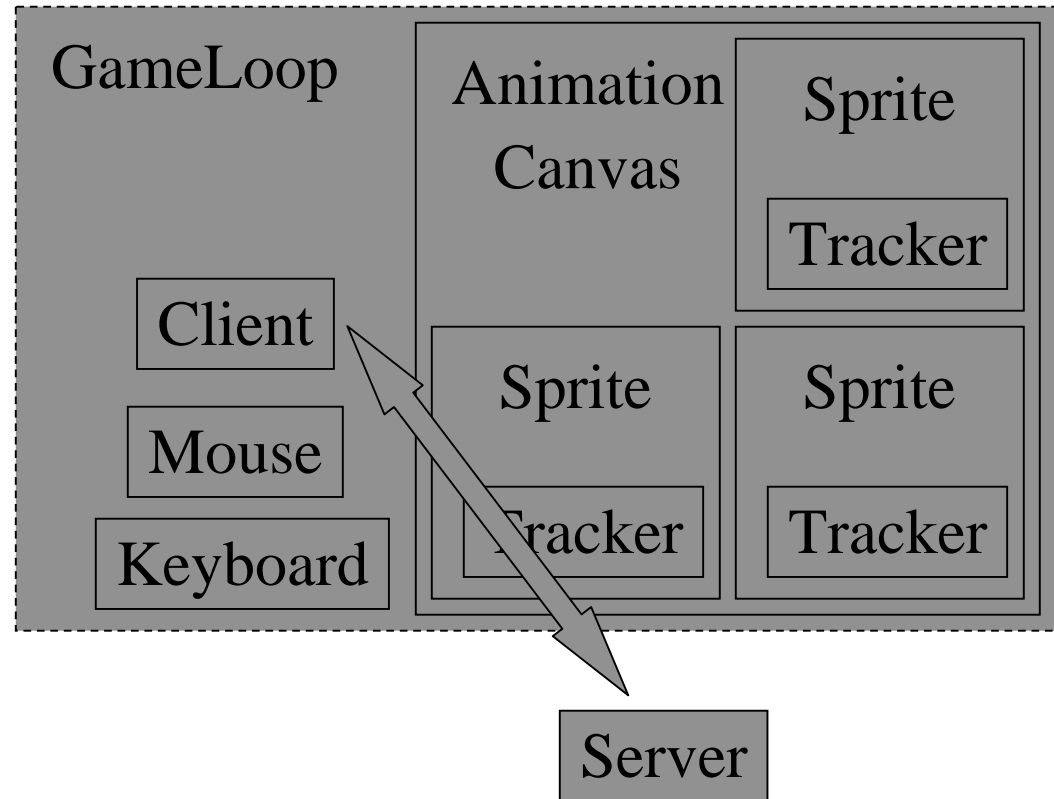
- ❖ **with a collection of Sprites**
- ❖ **that paints itself by painting all of its Sprites**



# Basic Design of the Video Game Package

**GameLoop adds:**

- ❖ **Client – Server Communications**
- ❖ **interaction via**
  - ❑ **Keyboard**
  - ❑ **Mouse**

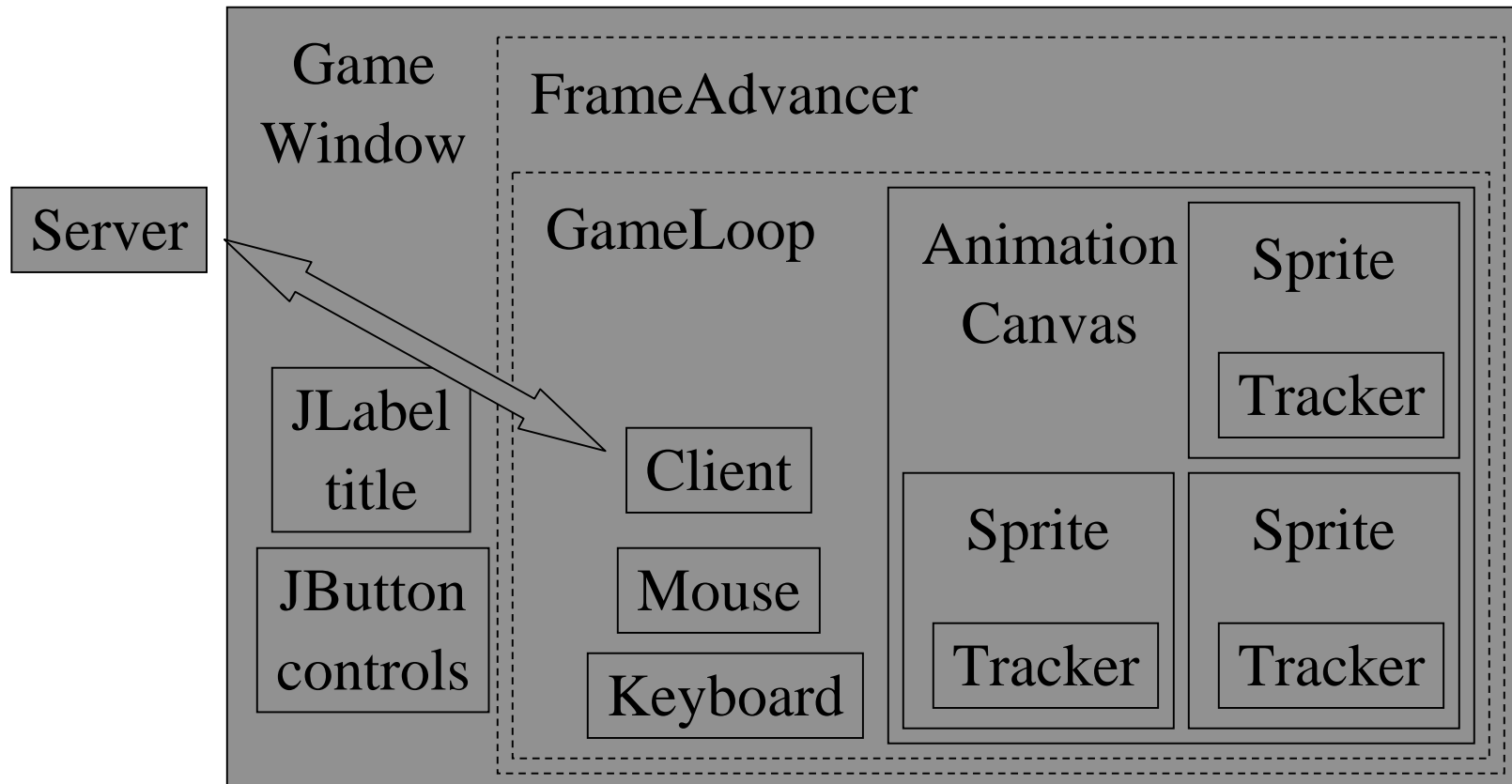


# Basic Design of the Video Game Package

**GameWindow adds:**

- ❖ **Controls for starting, pausing, and muting**
- ❖ **Ability to run as an applet for single player games**
- ❖ **Ability to run as an application for multiplayer games**

# Basic Design of the Video Game Package



# Steps to Making a Game

1. **Make packages for the parts of your game.**
2. **Organize your program's resources within the packages.**
3. **Write the program's classes.**

# Steps to Making a Game

## 1. Make the packages for the parts of your game.

- a. `tipgame.game.nameofyourgame`  
put your game logic classes here
- b. `tipgame.game.nameofyourgame.sprite`  
put all of your custom made sprites here
- c. `tipgame.game.nameofyourgame.tracker`  
put all of your custom made trackers here
- d. `tipgame.game.nameofyourgame.html`  
put your help file here
- e. `tipgame.game.nameofyourgame.audio`  
put all of your audio files here
- f. `tipgame.game.nameofyourgame.images`  
put your image files (jpg, gif, etc.) here

Be sure to replace *nameofyourgame* with the actual name of your game

# Steps to Making a Game

## 2. Organize your program's resources:

- a. Make a help screen in HTML and place the file in the package `tipgame.game.nameofyourgame.html`
- b. Copy all images needed in your game to the package `tipgame.game.nameofyourgame.images`
- c. Copy all audio needed in your game to the package `tipgame.game.nameofyourgame.audio`

# Steps to Making a Game

## 3. Write the program's classes:

- a. In the package `tipgame.game.nameofyourgame` write a class to extend `GameLoop`.
- b. (Optional) Implement/copy necessary `Sprite` extensions and place them in `tipgame.game.nameofyourgame.sprite`
- c. (Optional) Implement/copy necessary `Tracker` extensions and place them in `tipgame.game.nameofyourgame.tracker`

# Steps to Making a Game

## 1. Make the packages

- a. `tipgame.game.nameofyourgame` – put your game logic classes here
- b. `tipgame.game.nameofyourgame.sprite` – put all of your custom made sprites here
- c. `tipgame.game.nameofyourgame.tracker` – put all of your custom made trackers here
- d. `tipgame.game.nameofyourgame.html` – put your help file here
- e. `tipgame.game.nameofyourgame.audio` – put all of your audio files here
- f. `tipgame.game.nameofyourgame.images` – put your image files (jpg, gif, etc.) here

Be sure to replace *nameofyourgame* with the actual name of your game

## 2. Organize your program's resources:

- a. Make a help screen in HTML and place the file in the package `tipgame.game.nameofyourgame.html`
- b. Copy all images needed in your game to the package `tipgame.game.nameofyourgame.images`
- c. Copy all audio needed in your game to the package `tipgame.game.nameofyourgame.audio`

## 3. Write the program's classes:

- a. In the package `tipgame.game.nameofyourgame` write a class to extend `GameLoop`.
- b. (Optional) Implement/copy necessary `Sprite` extensions and place them in `tipgame.game.nameofyourgame.sprite`
- c. (Optional) Implement/copy necessary `Tracker` extensions and place them in `tipgame.game.nameofyourgame.tracker`

# How Pong Was Made

## 1. Made the packages:

- a. `tipgame.game.pong`
- b. `tipgame.game.pong.sprite`
- c. `tipgame.game.sprite.tracker`
- d. `tipgame.game.sprite.html`
- e. `tipgame.game.sprite.audio`
- f. `tipgame.game.sprite.images`

# How Pong Was Made

## 2. Organized the program's resources:

- a. Made a help screen `PongHelp.html` and placed it in the package `tipgame.game.pong.html`
- b. Copied the images `jam.JPG` and `mike.JPG` into the package `tipgame.game.pong.images`
- c. Copied `DingLower.wav` into the package `tipgame.game.pong.audio`

# How Pong Was Made

## 3. Wrote the program's classes:

- a. In the package `tipgame.game.pong` wrote `PongLoop` which extends `GameLoop`.
- b. Wrote `YourSprite` which extends `Sprite`. Placed this class in `tipgame.game.pong.sprite`
- c. Wrote `ProjectileTracker` which extends `Tracker`. Placed this class in `tipgame.game.pong.tracker`

# How Pong was Made

**Some of the more complex parts of Pong that we'll talk about later in more detail:**

- ❖ **Loops**
- ❖ **Conditionals**
- ❖ **Event handling**
- ❖ **Inheritance & Interfaces**
- ❖ **Collections**
- ❖ **Collision detection**

# How Pong was Made

**What you will need to know shortly:**

- ❖ **Basic classes and which package they belong in**
- ❖ **Basic structure of the gaming package**
- ❖ **Basic steps to making the game**
- ❖ **General idea about how `PongLoop.java` works**
- ❖ **Enough familiarity with the code to make minor modifications to `YourSprite` and `PongLoop`**

# How Pong was Made

What you *don't* need to understand yet:

- ❖ All of the classes in the gaming package
- ❖ All of the code in the basic classes
- ❖ How to make your own classes like PongLoop from scratch