ISSN: 1813-4890

Design and Implementation of the Hit Pig Head Game

Yunzheng Ding

School of Information Engineering, Jingdezhen Ceramic university, Jingdezhen 333403, P.R.China

Corresponding author: jciddyz@163.com

Abstract

The computer gaming industry has a development history of over 20 years worldwide. After more than 20 years of rapid development, the computer gaming industry has become one of the entertainment industries on par with film, music, and others, with annual sales exceeding Hollywood's annual revenue. The emergence of the Internet provides another fascinating carrier and tool for computer games, and injects fresh blood into the further development of the entire industry. Compared with online games, PC single player games are also popular among the public because with just a simple personal computer, users can enjoy the fun of a large number of games. However, to have a home TV console on the market, such as the PS series and Microsoft Xbox series, users also need to spend an additional amount of money. Undoubtedly, except for those enthusiastic hardcore gamers, few users want to pay this extra fee, and it is only for the sake of the game. With the improvement of personal computer performance, single player computer games have shown enormous potential that the home video game market does not have.

Keywords

J2SE, java applet, Java Games.

1. Introduction

Hit Pig Head Game is a single player computer game that is accompanied by visual sounds during the game, and features level levels with smooth graphics. When entering the game start screen, users can choose to start the game and enter the game screen, or choose to exit the game to leave; When the user enters the game screen, the game officially begins, and pig heads will randomly appear in the game screen. At this time, the user only needs to move the mouse and click on the pig head. Clicking on a pig head will add 10 points. When the score reaches 100 points within 15 seconds, a congratulatory dialog box will pop up. Click the "OK" button to enter the second level, otherwise a failure dialog box will pop up. Click the "OK" button to return to the game start screen; When the score reaches 200 points within 20 seconds, a congratulatory dialog box will also pop up. Click the "OK" button to enter the third level, otherwise a failure dialog box will pop up. Click the "OK" button to return to the game start screen; When the score reaches 300 points within 25 seconds, a success dialog box will pop up, indicating that the user has successfully completed all levels. Clicking the "OK" button will return to the game start screen. Of course, while playing the game, you can also click the "Exit Game" button to leave the game, or click the "About" button to view the game rules.

2. The requirements for the game

Launch the game start screen, accompanied by background music, click the "Start Game" button to start the game;

ISSN: 1813-4890

After entering the game screen, the timer starts counting. Use the mouse to click on the randomly appearing pig heads on the screen, click on one to get 10 points. When the specified score is reached within the specified time, a "Success" dialog box will pop up.Click the "OK" button to enter the next level; Otherwise, a "failure" dialog box will pop up, and the game can only be restarted;

During the game, when clicking the "End Game" button, a "Exit Game" dialog box will appear. Clicking the "Cancel" button will take you back to the current game, and clicking the "OK" button will take you back to the game start screen;

During the game, you can click the "End Game" button at any time to leave the game;

From the start of the game screen, you can click the "About" button at any time to view the game introduction and copyright ownership information;

When the required score is reached within the specified time, a "Congratulations" dialog box will pop up, indicating successful completion of the level. Click the "OK" button to proceed to the next level;

Enter the game screen or start screen with the mouse, play background music, and replace the mouse with a hammer; The mouse leaves the game screen and the music ends. The hammer is replaced with a mouse. When the hammer hits the pig's head, there is a sound of the hammer striking; When the hammer is dragged, there is also another sound accompanying it.

3. Overall design

This game has one Java source file, one HTML file, and seven image files.

The Java source file contains an abstract class SuperSprite, a class LovelyBig, a class Hammer, a main class HitPigHead, a game start screen class StartScreen, and a window close class CloseDialog.

Below are brief introductions of their functions:

The abstract class SuperSprite is used to define some methods that will be used soon so that the subclass LovePig can inherit them;

Class LovePig is a subclass that inherits from its parent class SuperSprite, which defines the states of appearance and disappearance of pig heads, the conditions for drawing pig heads, and hitting pig heads;

Hammer class is used to define a hammer and its initial position;

The main class HitPigHead is the core part of the entire program, which includes loading game images, pig heads during game execution, changes in hammer and sound, and game level settings;

Class StartScreen is the screen at the beginning of the game, which is the prelude to starting the game. It will only enter the game screen when the user clicks the "Start Game" button;

The CloseDialog class handles the event of closing a window. When the user clicks the "Exit Game" button, a dialog box will appear to close the window, and the user can choose to end or continue the game.

4. Detailed design

4.1. Main class HitPigHead

4.1.1. Member variable

Table 1 Member variables

Member variable describe	Variable type	names

ISSN: 1813-4890

Image	Image	frame,pig,hammer1,hamme2,bkImage, OffScreen,PigHead1, PigHead2;
Thread	Thread	newThread;
Cursor	Cursor	Hammer1,Hammer2,currentImage;
Audio	AudioClip	A1,A2,A3 ;
Panel	Panel	Status, Control;
Label	Label	Time, Score;
Button	Button	start, end;
Time	GregorianCalendar	time;

Image is used to define the images used in game graphics; Thread is used to define a new thread; Cursor is used to define custom cursor; AudioClip is used to define the accompanying music in games; Panel is used to define two new panels, Status and Control; Label is used to define time and score labels; Button is used to define the start and end buttons; GregorianCalendar is used to define time.

4.1.2. Methods

Table 2 Methods

names	Function
endgame()	game over
mouseExited()	Mouse leaves the game screen
mouseClicked()	Press and release the mouse button
mouseEntered()	Mouse enters the game screen
mousePressed()	Mouse button pressed
mouseReleased()	Mouse button released
mouseMoved()	Mouse movement
mouseDragged()	Mouse draged

4.1.3. Code analysis

The main class is the main part of the game's functional implementation, which means the core part of the program is here.

```
//First, load the corresponding class:
import java.awt.*;
import java.awt.event.*;/* In addition to Runnable, MouseListener, MouseMotionListener, and ActionListener have also been implemented to handle events*/
/*Then inherit from MouseListener, MouseMotionListener, and others in the class declaration ActionListener interface:*/
public class HitPigHead extends Applet implements
Runnable, MouseListener, MouseMotionListener, ActionListener
//Next, register event handling methods in the initialization section of the class:
public void init ()
{addMouseListener (this);//Register event handling methods
addMouseMotionListener (this); // Registration event handling method
//Finally, add handling functions for these three types of events in the code:
```

ISSN: 1813-4890

//Include in MouseListener event:
Public void mouseExited (MouseEvent e)//Mouse leaves Component
Public void mouseClicked (MouseEvent e)//Release the mouse button after being pressed
Public void mouseEntered (MouseEvent e)//Mouse enters Component
Public void mousePressed (MouseEvent e)//Mouse button pressed
Public void mouseReleased (MouseEvent e)//Release the mouse button
//The MouseMotionListener event includes handling functions:
Public void mouseMoved (MouseEvent e)//When the mouse moves
Public void mouseDragged (MouseEvent e)//When dragging the mouse
//Of course, regardless of whether it is used in your program or not, it must be declared.
//The ActionListener event includes handling functions:
Public void actionPerformed (ActionEvent e)//Event Handling}

4.2. Start Screen Class StartScreen

4.2.1. Member variable

Table 3 Member variable

Member variable describe	Variabletype	names
Image	Image	Normal,bkImage,Hit,CurrentImage;
character	Thread	F1，F2;
String	String	ChineseTitle,EnglishTitle;

Image is used to define the pig image, background image, hammer image, and current pig image; Thread is used to define the font of the font that appears in the game screen; String is used to define the string that appears in the game screen.

4.2.2. Methods

Table 4 Methods

names	function
StartScreen()	Constructor function
UpdateStatus()	Update animation status
paintScreen(Graphics g)	Draw animation

4.2.3. 4.2.3. code analysis

This type is the starting screen for the game when the user chooses to play, which is a prerequisite for the user to be able to play the game. The main function of this class is to load images into the mini program and set the font of the font to appear in the screen.
//First, load the corresponding class:
The detailed code is as follows:
Class StartScreen//Start Screen class
{
int width,height,StringWidth,StringHeight,Ascent,Descent,X,Y;
int ImageLeftBound, ImageRightBound, ImageX, ImageY, ImageWidth,
ImageHeight, VX;
Font F1, F2, F3;
Image Normal, bkImage, Hit, currentImage;
String ChineseTitle, EnglishTitle, PressEnter;
HitPigHead Game;

ISSN: 1813-4890

```
FontMetrics FM;
public StartScreen (int AppletWidth, int AppletHeight, HitPigHead Game,
Image normal, Image hit, Image bk)
{
Public void updateStatus ()//Function for updating animation status
{
ImageX = ImageX + VX;            // Specify the new location of the image
If (ImageX<=ImageLeftBound)//If it touches the left boundary
{
currentImage = Hit;       // Specify the current image as pig image 2
ImageX = ImageLeftBound;  // Set a new position for the image
VX = -VX;             // Change the direction of image movement
}
If (ImageX>=ImageRightBound)//If encountering the right boundary
{
currentImage = Normal;       // Specify the current image as pig image 1
ImageX = ImageRightBound;    // Set a new position for the image
VX = -VX;                 // Change the direction of image movement
}
}
```

Figure 1 Game start screen

4.3. Close Window Class CloseDialog

4.3.1. Member variable

Table 5 Member variables

Member variable describe	Variable type	names
Panel	Panel	P1,P2;
Button	Button	B1,B2;

The variable Panel is used to define panels P1 and P2, enabling the loading of tag time and scores into panel P1, and the loading of buttons "Start Game" and "End Game" into panel P2; The variable Button is used to define buttons B1 and B2.

ISSN: 1813-4890

4.3.2. Methods

Table 6 Methods

names	function
CloseDialog() actionPerformed(ActionEvent e)	Constructor function event processing

4.3.3. Code analysis

This type is the event handling for users who click the "close" button:

```
//First, load the corresponding class:
import java.awt.*;
import java.awt.event.*; // In order to handle events, in addition to Dialogue, we also implemented
class CloseDialog extends Dialog implements ActionListener
//Next, register event handling methods in the initialization section of the class:
public CloseDialog (HitPigHead Game, Frame owner)  {
B1.addActionListener(this);  // Registration event handling method
B2.addActionListener (this);
//Finally, add a handling function for this event in the code:
public void ActionPerformed (ActionEvent e)
}
```

The detailed code is as follows:

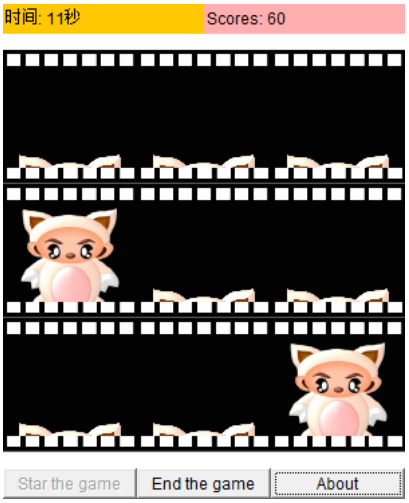
```
class CloseDialog extends Dialog implements ActionListener
//Close Window Class
{
Panel      P1,P2;  // Define two panels P1 and P2
Button     B1,B2;  // Define two buttons B1 and B2
HitPigHead Game;
Thread     newThread;
Toolkit tk=this.getToolkit ();
Dimension  screenSize=tk.getScreenSize ();
int frameHeight=100, frameWidth=100;}
```

4.4. Game Screen

When selecting the "Stick Pig Head" program to start, enter the game start interface and click the "Start Game" button on the game start screen to enter the game screen. At this point, the timer starts counting. As shown in Figure 2:

At this point, click on the randomly appearing pig head on the screen with the mouse. Clicking on a pig head will earn 10 points, and the points will accumulate accordingly. As shown in Figure 3.

ISSN: 1813-4890



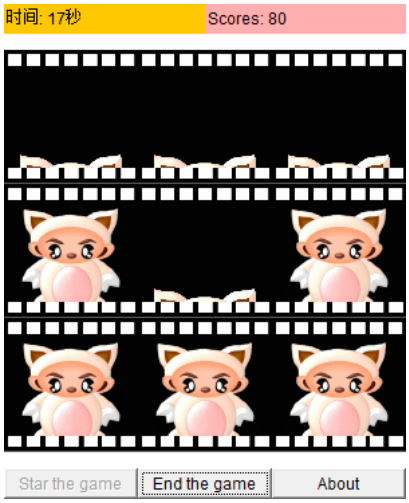


Figure 2 Timer timing screen Figure 3 Score accumulation screen

4.5. Viewing Information

After starting the game, you can click the "About" button at any time to view the game introduction and copyright ownership information. The code is as follows:

JOptionPane.showmessageDialog (this, "This is a simple single player game, enter

After entering the game screen, click on the randomly appearing pig head with the mouse and hit the pig head\The more n "+" there are, the higher the score. \Regarding JOptionPane.PLAIN_MESSAGE;The running result is shown in Figure 4:

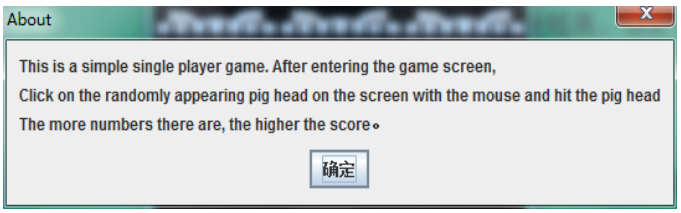


Figure 4 View information screen

4.6. Mouse Processing Events

The interaction with users is the main function of Java, which is also the reason why Java is attractive. Users can communicate with Java Applet programs through a mouse. Regarding the handling of mouse events:

//First, load the corresponding class:

import java.awt.*;

import java.awt.event.*; // In addition to Runnable, MouseListener and MouseMotionListener have also been implemented to handle events.

//Then inherit from the MouseListener and MouseMotionListener interfaces in the class declaration:

public class Mouse extends Applet implements Runnable, MouseListener, Mouse Motion Listener

//Next, register event handling methods in the initialization section of the class:

public void init ()

{addMouseListener (this);//Register event handling methods

ISSN: 1813-4890

addMouseMotionListener (this);
//Finally, add handling functions for these two types of events separately in the code:
//Include in MouseListener event:
Public void mouseExited (MouseEvent e)//Mouse leaves Component
Public void mouseClicked (MouseEvent e)//Release the mouse button after being pressed
Public void mouseEntered (MouseEvent e)//Mouse enters Component
Public void mousePressed (MouseEvent e)//Mouse button pressed
Public void mouseReleased (MouseEvent e)//Release the mouse button
//The MouseMotionListener event includes handling functions:
Public void mouseMoved (MouseEvent e)//When the mouse moves
Public void mouseDragged (MouseEvent e)//When dragging the mouse
//Of course, regardless of whether it is used in your program or not, it must be declared
The running result is shown in Figure 5:

Figure 5 Mouse screen

4.7. Loading Sound

Java supports playback of sound files through a class of Applets, and can play a sound only once or repeatedly as a loop.

The simplest way to obtain and play sound is through the play() method in the class Applet. Like the getImage() method, the play() method uses two formats:

Play() with one parameter - a URL object - it can load and play audio clips stored at that URL.

Play() with two parameters - a basic URL and a folder path name - can load and play that audio file. The first parameter can usually be a call to getDocumentBase() or getCodeBase().

After being called, the play() method will retrieve and play the given sound as quickly as possible. If the sound file cannot be found, the indication of the relevant problem information that can be received is that there is no sound. It will not display any error messages. To repeatedly play a sound, start and stop it, or play it as a loop, it must be placed into an AudioClip object using the getAudioClip() method of the mini application. AudioClip is part of the java. applet package, so it must be imported into the program for use.

The method getAudioClip() uses one or two parameters in the same style as the method play(). The first participant The number is a URL parameter that indicates the sound file, while the

ISSN: 1813-4890

second parameter is a reference to the folder path. The following statement loads a sound file into the clip object:

AudioClip clip=getAudioClip (getCodeBase (),"audio.au");

The method getAudioClip() can only be called in mini applications. In the application, sound files can be loaded by using newAudioClip(), which is java. awt A class method of the Applet class. Here is the scenario where the previous example is rewritten for use in an application:

AudioClip clip=Applet.newAudioClip ("audio.au");

The detailed code is as follows:

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public abstract class shengyin extends Applet
implements MouseListener, MouseMotionListener
{
int     AppletWidth, AppletHeight, drawX, drawY;
Image       hammer1, hammer2;
Cursor      Hammer1, Hammer2;
Toolkit     TK;
MediaTracker MT;
Image    OffScreen;
Graphics drawOffScreen;
AudioClip    A1, A2, A3;
public void init ()
{
//A1 is the sound when the hammer strikes, A2 is the sound when the hammer moves,
//A3 is the sound of the hammer entering the Applet
A1       = getAudioClip (getDocumentBase (),"Audio/audio1.au");
A2       = getAudioClip (getDocumentBase (),"Audio/audio2.au");
A3       = getAudioClip (getDocumentBase (),"Audio/audio3.au");
addMouseListener(this);          // Registration event handling method
addMouseMotionListener (this);
TK       = getToolkit();         // Get Toolkit
//Get custom cursor image
hammer1 = getImage (getDocumentBase (),"Images/hammer1.gif");
hammer2 = getImage (getDocumentBase (),"Images/hammer2.gif");
MT       = new MediaTracker (this);
MT.addImage (hammer1, 0);
MT.addImage (hammer2, 0);
AppletWidth   = getSize().width;   // Obtain the height of the Applet
AppletHeight  = getSize().height;  // Obtain the width of the Applet
OffScreen     = createImage (AppletWidth, AppletHeight);
drawOffScreen = OffScreen.getGraphics ();
```

ISSN: 1813-4890

4.8. Image loading

Due to the asynchronous transmission of image files, we can use interfaces to process information during the image transmission process; Of course, when necessary, we can use a MediaTracker class to track the transmission of images. The game uses a media tracker to load images. A media tracker is a MediaTracker type object specifically designed for tracking image loading. This class is defined in the Java.awt package, although currently it is only used to manage the loading of images, it can also be extended in the future to track the loading of other types of media. There is only one MediaTracker constructor that requires passing a reference to the component as an argument - the component object is an object of the image being loaded.

The purpose of using a media tracker is that users may face issues when attempting to display images from slower network connections (such as 14.4 kilobytes per second modems). When the user starts drawing an image, the image may not have been fully acquired yet. Users can use an auxiliary class called MediaTracker to determine if the image is ready for display.

The principle of using a media tracker is that after calling the getImage() method to obtain a reference to an Image object, it can pass the image reference to the tracker by calling its addImage() method. This makes the MediaTracker object responsible for loading images. The addImage() method has two independent variables: a reference to the tracked image and an identifier associated with the int type image.

The following statement can create a media tracker in the mini program:

MediaTracker MT=new MediaTracker(this);// This refers to the mini program

Next, attempt to obtain the image that the user wishes to display:

Image myImage=getImage ("Img/hammer1.gif");

Now users are telling MediaTracker to stare at the image with their eyes. When a user adds an image to

When using MediaTracker, users can also provide a numerical ID:

MT.addImage (Image image, int id)

The ID value can be used for multi image display, and when users want to know if the entire set of images is ready, they can use a single ID to check it.

Once the user initiates image tracking, they can load the image and wait for it to be ready by using the waitForID method.

MT.waitForID (int id)

This statement waits for all images with ID number id.

MT.waitForID (int id, long ms)

This statement waits for all images with ID number id, up to a maximum of ms milliseconds.

Users can also use the waitForAll method to wait for all images:

MT.waitForAll ()

Like using the waitForID method, users can provide the maximum number of milliseconds to wait:

MT.waitForAll (long ms)

The detailed code is as follows:

```
pulic void init ()
{
    Image frame, pig, hammer1, hammer2, OffScreen, PigHead1, bkImage, PigHead2;
    MediaTracker MT;
```

ISSN: 1813-4890

```
MT      = new MediaTracker (this);
pig     = getImage (getDocumentBase (),"Img/pig.gif");
frame   = getImage (getDocumentBase (),"Img/frame.gif");
hammer1 = getImage (getDocumentBase (),"Img/hammer1.gif");
hammer2 = getImage (getDocumentBase (),"Img/hammer2.gif");
PigHead1 = getImage (getDocumentBase (),"Img/pighead1.gif");
PigHead2 = getImage (getDocumentBase (),"Img/pighead2.gif");
bkImage      = getImage (getDocumentBase (),"Img/back.jpg");
MT.addImage (pig, 0);
MT.addImage (frame, 0);
MT.addImage (hammer1, 0);
MT.addImage (hammer2, 0);
MT.addImage (PigHead1, 0);
MT.addImage (PigHead2, 0);
MT.addImage (bkImage, 0);
}
```

4.9. Color Settings

To create a beautiful Java Applet program, color settings are essential, including background and foreground colors. Java provides a variety of color processing functions. Here we only introduce the simplest color settings. This program sets the background to blue, the foreground to green, and fills a rectangle with the foreground color:

```
import java.awt.*;
import java.applet.*;
public class SetColor extends Applet
{ public void paint (Graphics g)
{setBackground (Color. blue);//Set the background color to blue
setForeground(Color.green);// Set the foreground color to green
```

Additionally, the seColor() function in the Graphics class can be used to change the brush color.

```
g.setColor(Color.red); // Set the string color to red
 g.setColor(Color.white); // Set the foreground color to white
```

4.10. Custom cursor

Due to the fact that in the game screen, the hammer moves with the cursor, and both the cursor and hammer are visible at the same time, using a custom cursor can replace the cursor with a hammer, making the game screen more beautiful and elegant.

The cursor is a resource, ultimately an image. So we can use the following definition:

```
Image   hammer1, hammer2;
Cursor   Hammer1, Hammer2;
```

ISSN: 1813-4890

```
Toolkit  TK;
//First obtain the Toolkit, then obtain the image:
//Get Toolkit
TK = getToolkit();
//Get custom cursor image:
hammer1 = getImage (getDocumentBase (),"Images/hammer1.gif");
hammer2 = getImage (getDocumentBase (),"Images/hammer2.gif");
//Obtain a cursor with a registration point:
Hammer1 = TK.createCustomCursor (hammer1, new Point (0, 0),"hammer1");
Hammer2 = TK.createCustomCursor (hammer2, new Point (0, 0),"hammer2");
//Finally, use the following code to set the cursor:
setCursor (Hammer1);
setCursor (Hammer2);
```

The running result is shown in Figure 6:

Because the mouse has been replaced by a hammer at this point, the hammer is no longer visible in the picture.

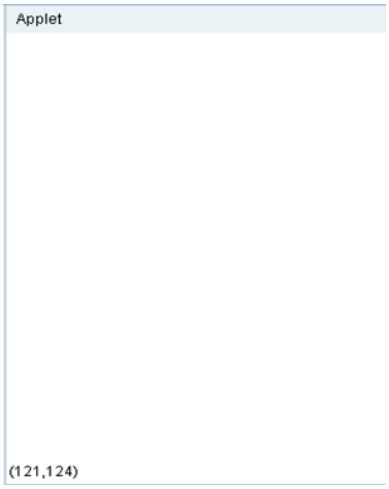


Figure 6 Cursor screen

5. Conclusion

The software has the following characteristics in specific implementation:

The entire software is designed using an object-oriented approach, which can effectively achieve software reuse and improve software development efficiency. Java is a pure object-oriented development language, where all Java code is encapsulated in classes and software reuse is achieved through class inheritance.

Flexible application, which can be used in the form of an application in the command window or in a web environment.

Make full use of Java's multi-threaded feature to reasonably partition the functions of the application, and improve the performance and scalability of the application.

The user interface is user-friendly, and during operation, users can easily click on the pig head by moving the mouse.

Added multimedia support, inserted sound prompts during tapping, and also added background music.

ISSN: 1813-4890

References:

[1] Yang Hongbo ,Wang Zhishun, "J2SE evolution history", Programmers,2005(07),P50-52

[2] Chen Limin ,Twinings nine," JAVA graphical interface development exploration", Journal of Southwest University for Nationalities (Natural Science Edition), 2006(02),P405-409

[3] Li rui-ge," Computer software development of Java programming language and applications", Programmers, 2022(06), P15-16.

[4] LIU Xiao-zheng, "Analysis of Java GUI programming tool set", SCIENCE & TECHNOLOGY INFORMATION, 2012(35), P596-597.

[5] Hua Weizhong, Zhao Chunyun," An in-depth look at Java threads", Computer and Information Technology. 1997(02), P29-30+33.

[6] Song Weiwei, Chen Shuzhen, Sun Xiao 'an, " Multithreading and dual buffering in the Java language",Electronic Computers and External Equipment,1998(06),P30-31.