Code Sample

---

1. Snake Game

Snake.java:

```
package com.xhan.SnakeGame;


import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.util.Hashtable;
import java.util.Map;

import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.UIManager;



/*
 * @(#) Snake.java   version 1.0     01/2015
 *
 * Copyright by Xu Han. All rights reserved.
 *
 * Use is subject to license terms.
 *
 * This is the first version of the tiny java GUI game SnakeGame.
 *
 * It is a training project completed totally by Xu Han.
 *
 * If you find bugs, please contact the developer by the email address
 *
 * xhan@wpi.edu
 *
 * */




public class Snake {

        //background array
```

```
	private static int[] B;

	//background two dimensional sizes
	final static int PP_weight=30;
	final static int PP_height=30;


	final private static int Blank=1;


	public static void main(String[] args){
		try {                                    //use Nimbus style

			UIManager.setLookAndFeel
			("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");
			} catch (Exception e) {
			e.printStackTrace();
			}

		B = new int[PP_weight*PP_height];
		for (int i=0; i<PP_weight*PP_height; i++){
			B[i]=Blank;
		}

	    Snakegui snakegui = new Snakegui(B);

		 //GUI
		snakegui.gui();

		Playsnake playsnake = new Playsnake();

		//play
	    playsnake.play(snakegui, B);

	}

}

class Snakegui extends JFrame{                  //class for GUI


	/**
	 * this class creates the Graphic User Interface of Snake Game
	 */

	 PlayPanel PlayPanel;
	JLabel message;

	private static final long serialVersionUID = 1L;

	// direction parameters and size parameters
	final private int East=3;
	final private int South=6;
	final private int West=9;
	final private int North=12;
	final private int Frame_weight=401;
	final private int Frame_height=531;
	final private int PP_weight = Snake.PP_weight;
	final private int PP_height = Snake.PP_height;
	final private int MainPanel_height=80;
	final private int Message_height=30;

	final private int Blank=1;
	final private int Food=8;
	 static boolean start=false;
```

```
        int Input=East;


        //Grid Background position
        static int[] B;

        //constructor
        Snakegui(int[] BB){

              B = BB;

        }


        // method to create GUI

       public void gui(){

               //Control Panel
              JPanel main = new JPanel();

              main.setBackground(Color.white);
              main.setPreferredSize(new Dimension(Frame_weight,MainPanel_height));
              main.setBorder(BorderFactory.createTitledBorder("Control"));

              //Buttons on main panel
              JButton Start = new JButton("Start");

              JButton About = new JButton("About");

              main.add(Start);

              main.add(About);


              //action listener for button start
              Start.addActionListener(new ActionListener(){
                     @Override
                     public void actionPerformed(ActionEvent event){
                            for (int i=10; i<PP_weight*PP_height; i++){
                                //set blank
                                B[i]=Blank;
                            }

                            //put food
                            B[Playsnake.food(B)]=Food;
                            Snakegui.start = true;
                            message.setText("Game start!");
                     }
              });

              //action listener for button about
              About.addActionListener(new ActionListener(){
                     @Override
                     public void actionPerformed(ActionEvent event){
                            JOptionPane.showMessageDialog(null,"Snake Game v1.1 beta
\nDesigned by Shawn Han.\n"
                                          + "If you find any bug or want to give me
advices, please contact me, thank you!\n"
                                          + "Email: shawnxhan@outlook.com\n"
                                          + "Enjoy this game!", "Snake Game
v1.1",JOptionPane.PLAIN_MESSAGE);
                            message.setText("About");
                     }
```

```
        });

        //Panel for playing
        PlayPanel = new PlayPanel();

        /** to control the moving direction of snake, we need to add listeners to
         *
         * all the views which are possible to get focus on. This can avoid being
         *
         * able to control the snake
         */
        keylistener keylistener = new keylistener();
        this.addKeyListener(keylistener);
        Start.addKeyListener(keylistener);
        About.addKeyListener(keylistener);
        PlayPanel.addKeyListener(keylistener);
        main.addKeyListener(keylistener);

        // Message panel to display control information
        JPanel MessagePanel = new JPanel();
        MessagePanel.setBackground(Color.white);
        MessagePanel.setPreferredSize(new
        Dimension(Frame_weight,Message_height));
        message = new JLabel();
        MessagePanel.add(message);

        // Layout set up for the GUI
        this.getContentPane().add(main,BorderLayout.NORTH);
        this.getContentPane().add(PlayPanel,BorderLayout.CENTER);
        this.getContentPane().add(MessagePanel,BorderLayout.SOUTH);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(Frame_weight,Frame_height);
        this.setResizable(false);
        this.setVisible(true);
    }
    //update UI
    public void Print(){
        PlayPanel.repaint();
    }
    // provide game over information
```

```
	public void GameOver(int score){

		JOptionPane.showMessageDialog(null,"Game over! Score is "+score, "Game Over!",JOptionPane.PLAIN_MESSAGE);
		message.setText("Game Over!");
	}


	private class PlayPanel extends JPanel{

		/**
		 *  Panel to paint background and snake. it can always update
		 *
		 *  the movement of snake unless game over.
		 */

		private static final long serialVersionUID = 1L;
		private String ImagePath="/image/background.png";


		@Override
		public void paintComponent(Graphics gs){

			Graphics2D g = (Graphics2D) gs;
			super.paintComponent(g);

			Image image=Toolkit.getDefaultToolkit().getImage(getClass().getResource(ImagePath));
			g.drawImage(image,0,0,400,420,this);


			for(int i=0; i<PP_height ; i++){
				for(int j=0; j<PP_weight; j++){

					if(B[i*PP_weight+j]==1){		//space without snake and food


					}

					if((B[i*PP_weight+j]==0)|(B[i*PP_weight+j]==8)){	//space with snake and food
						g.setColor(Color.BLACK);
						g.fillRect(12*j+20, 12*i+20, 10, 10);
					}
				}
			}

		}
	}



	private class keylistener implements KeyListener{

	    /**
	     * the keyboard listener to get the control of player
	     *
	     * so that the player is able to control the snake and play
	     *
	     */

		String ainput;
```

```
            public void keyPressed(KeyEvent event){

            Map <String,Integer> input = new Hashtable<String, Integer>();

                   input.put("W", North);

                   input.put("D", East);

                   input.put("S", South);

                   input.put("A", West);

            ainput=String.format(KeyEvent.getKeyText(event.getKeyCode()));

                   if(ainput.equals("W")|
                   ainput.equals("A")|
                   ainput.equals("S")|
                   ainput.equals("D")
                   ){

              Input=input.get(ainput);

            }

            }
            public void keyReleased(KeyEvent event){

            }
            public void keyTyped(KeyEvent event){

            }
        }
}
```

PlaySnake.java:

```
package com.xhan.SnakeGame;

import com.xhan.SnakeGame.Snake;
import com.xhan.SnakeGame.Snakegui;


public class Playsnake {

        /**
         * This is the back end control class whose duty is to complete the
         *
         * correct movement of snake based on the action get from the keyboard
         *
         * It can correctly judge if the game is over.
         *
         * Another functions include put food randomly and speed up the sanke
         *
         * as the time consumes
         *
         */

       //Direction, size, and control parameters

       final private int East=3;
       final private int South=6;
       final private int West=9;
       final private int North=12;
       final private int Snake_Length=10;
       final private int Snake_Speed=150;
       final private int PP_weight = Snake.PP_weight;
       final private int PP_height = Snake.PP_height;

       final private static int Blank=1;
       final private int Snake_Body=0;
       final private int Food=8;
       final private int Speed_up=30;
       final private int Speed_up_round=5;
       final private int Score_ratio=10;

       boolean over=false;


       public void play(Snakegui gui,int[] B){

              int[] s;
              int oldd=East;
              int length = Snake_Length;
              int eat=Snake_Length;
              int speed=Snake_Speed;
              int round=0;

              s = new int[PP_weight*PP_height];

             while(length<900){

              length = Snake_Length;
```

```
        eat=Snake_Length;

        //set for restart
        for (int i=0; i<Snake_Length; i++){
                    s[i]=length-1-i;
              }

           while(!over){

                //start
                if(Snakegui.start){

                      gui.Print();

                      int newd=oldd;

                      //move in a specific speed

                      try{Thread.sleep(speed);

                      }

                      catch(Exception ex){}

                      //get action input
                      newd=gui.Input;

                      //turn direction
                      oldd=turn(oldd,newd);

                      //keep snake moving
                      length=follow(oldd,s,length,B);

                      //set new food if snake became longer
                      if(length>eat){

                            B[food(B)]=Food;
                            eat=length;
                            round++;
                            if((round%Speed_up_round==0)&&(speed>-1)){
                                  speed-=(round%Speed_up_round+1)*Speed_up;
                            }
                      }

                     // update movement
                     gui.Print();
                }
                try{Thread.sleep(50);
                }
                catch(Exception ex){}
        }
        over=false;
        Snakegui.start=false;
        gui.GameOver(round*Score_ratio);
    }
}

//set food in the background
public static int food(int[]B){
        boolean put=false;
        int p=0;

        while(!put){
          //random location
                p=(int)(Math.random()*900);
```

```
                if(B[p]==Blank){
                    //not in the location of snake
                    put=true;
                }
        }
        return p;
}

//turn direction method

private int turn (int oldd, int newd){
        if(Math.abs(oldd-newd)==6){
            //make sure direction is valid
                    newd=oldd;
        }

    switch(newd){
    //turn
                case East: return East;

                case South: return South;

                case West: return West;

                default: return North;
        }

}

//move the snake
private int follow (int dir, int[] s, int length, int[] b){

        int former;    //head
        int end;       //end

        former=s[0];
        end=s[length-1];

        //turn head based on specific direction
        //we need also to distinguish the edge of the map

        switch(dir){
                case 3: {
                                if(((s[0]+1)%PP_weight==0)&&
                                 (former%PP_weight==PP_weight-1))

                                        s[0]-=PP_weight-1;

                                else s[0]++;
                                break;
                        }

                case 9: {
                                if(((s[0]-1)%PP_weight==PP_weight-1)&&
                                (former%PP_weight==0))

                                        s[0]+=PP_weight-1;

                                else s[0]--;
                                break;
                        }
                case 6: {
                                if((s[0]+PP_weight)>PP_weight*PP_height-1)
                                s[0]-=PP_weight*(PP_height-1);
```

```
                                else s[0]+=PP_weight;
                                break;
                        }

                case 12:{
                                if((s[0]-PP_weight)<0)

                                        s[0]+=PP_weight*(PP_height-1);

                                else s[0]-=PP_weight;
                                break;
                        }

            }

            //eat itself then over

            if(b[s[0]]==0) over=true;


            //eat food then becomes longer
            if(b[s[0]]==8) {
              length++;
              end=s[length-1];
        }

            //move the body
            for (int i=1; i<length; i++){
                    int temp;
                    temp=s[i];
                    s[i]=former;
                    former=temp;
            }

            for(int i=0;i<length;i++){
                    b[s[i]]=Snake_Body;
            }

            b[end]=Blank;

            return length;      //update length
        }
}
```

Android Application: MoodNote

MainActivity

```
package com.example.slipui;

import java.io.IOException;

import org.apache.http.client.ClientProtocolException;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.ActionBar;
import android.app.Activity;
import android.content.ClipData;
import android.content.Context;
import android.content.Intent;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Point;
import android.graphics.drawable.ColorDrawable;
import android.graphics.drawable.Drawable;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.DragEvent;
import android.view.GestureDetector;
import android.view.GestureDetector.OnGestureListener;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.DragShadowBuilder;
import android.view.View.OnLongClickListener;
import android.view.ViewGroup.LayoutParams;
import android.view.animation.AccelerateInterpolator;
import android.view.animation.AlphaAnimation;
import android.view.animation.Animation;
import android.view.animation.AnimationSet;
import android.view.animation.AnimationUtils;
import android.view.animation.LinearInterpolator;
import android.view.animation.ScaleAnimation;
import android.view.animation.TranslateAnimation;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.RelativeLayout;
import android.widget.SearchView;
import android.widget.TextView;


public class MainActivity extends Activity {
	final static private String IMAGEVIEW_TAG = "icon bitmap";
	final static private String B_LIKE_TAG= "like button";
	final static private String B_OPTIONS_TAG= "option button";
	final static private String TEXTVIEW_TAG= "display";
	private static float Display_x;
	private static float Display_y;
	private static boolean get_drag_location=false;


	private TextView Display1;
```

```
	private TextView Display2;
	private Button Button_like;

	private Button Button_last;


	private DataBase database;
	private ActionBar actionBar;


	@Override
	protected void onCreate(Bundle savedInstanceState) {

		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);


		showActionBar();



		ImageButton Button_attach = (ImageButton)findViewById(R.id.attach);
	    Button_attach.setOnClickListener(new View.OnClickListener() {

			@Override
			public void onClick(View v) {
				// TODO Auto-generated method stub
				Intent open_menu = new
Intent("com.example.slipui.ATTACHACTIVITY");
				startActivity(open_menu);

			}
		});




		Display1 = (TextView) findViewById(R.id.display1);

		Display2 = (TextView) findViewById(R.id.display2);


		database = new DataBase();


		Button_like=(Button) findViewById(R.id.like);
		Button_like.setTag(B_LIKE_TAG);

		Button_like.setOnTouchListener(new View.OnTouchListener(){

			@Override
			public boolean onTouch(View v, MotionEvent event) {
				// TODO Auto-generated method stub
				if(event.getAction()==MotionEvent.ACTION_DOWN){

					database.like();




					AnimationSet animationT = new AnimationSet(true);
```

```
                                TranslateAnimation translate = new
TranslateAnimation(
                                        Animation.RELATIVE_TO_SELF,0.0f,
                                        Animation.RELATIVE_TO_SELF,0.0f,
                                        Animation.RELATIVE_TO_SELF,0.0f,
                                        Animation.RELATIVE_TO_SELF,0.5f);

                                AlphaAnimation alpha = new AlphaAnimation(0.8f,0.5f);

                                alpha.setDuration(100);
                                alpha.setFillAfter(true);
                                translate.setDuration(200);
                                translate.setInterpolator(new
AccelerateInterpolator());
                                animationT.addAnimation(translate);
                                animationT.addAnimation(alpha);


                                Display1.startAnimation(animationT);
                                Display2.setText(database.peek());
                                Display1.setText(database.next());

                                AnimationSet animation = new AnimationSet(true);

                                ScaleAnimation scale = new ScaleAnimation(
                                        1.0f,0.85f,1.0f,0.85f,
                                        Animation.RELATIVE_TO_SELF,0.5f,
                                        Animation.RELATIVE_TO_SELF,0.5f);

                                scale.setDuration(80);

                                animation.addAnimation(scale);

                                animation.setFillAfter(true);
                                v.startAnimation(animation);



                        }
                        if(event.getAction()==MotionEvent.ACTION_UP){


                                AnimationSet animation = new AnimationSet(true);
                                ScaleAnimation scale = new ScaleAnimation(
                                        0.8f,1.1f,0.8f,1.1f,
                                        Animation.RELATIVE_TO_SELF,0.5f,
                                        Animation.RELATIVE_TO_SELF,0.5f);

                                scale.setDuration(100);

                                animation.addAnimation(scale);
                                v.startAnimation(animation);

                        }

                        return true;
                }


        });



        Button_last=(Button) findViewById(R.id.last);
        Button_last.setOnTouchListener(new View.OnTouchListener(){
```

```
		@Override
		public boolean onTouch(View v, MotionEvent event) {
			// TODO Auto-generated method stub
			if(event.getAction()==MotionEvent.ACTION_DOWN){
				Display1.setText(database.last());
				Display1.append("\nNow like is "+database.getLike());

				AnimationSet animation = new AnimationSet(true);

				ScaleAnimation scale = new ScaleAnimation(
						1.0f,0.85f,1.0f,0.85f,
						Animation.RELATIVE_TO_SELF,0.5f,
						Animation.RELATIVE_TO_SELF,0.5f);

				scale.setDuration(80);

				animation.addAnimation(scale);

				animation.setFillAfter(true);
				v.startAnimation(animation);
			}
			if(event.getAction()==MotionEvent.ACTION_UP){


				AnimationSet animation = new AnimationSet(true);
				ScaleAnimation scale = new ScaleAnimation(
						0.8f,1.1f,0.8f,1.1f,
						Animation.RELATIVE_TO_SELF,0.5f,
						Animation.RELATIVE_TO_SELF,0.5f);

				scale.setDuration(100);

				animation.addAnimation(scale);
				v.startAnimation(animation);

			}
			return true;
		}


	});

	Display1.setOnTouchListener(new View.OnTouchListener() {

		@Override
		public boolean onTouch(View v, MotionEvent event) {
			// TODO Auto-generated method stub

			if(event.getAction()==MotionEvent.ACTION_DOWN){
				ClipData data = ClipData.newPlainText("", "");
				View.DragShadowBuilder shadowBuilder= new
View.DragShadowBuilder(v);


				v.startDrag(data, shadowBuilder, v, 0);

				get_drag_location=false;
				return true;
				}


			return true;
		}
```

```
		});


		Display1.setOnDragListener(new View.OnDragListener(){
			@Override
			public boolean onDrag(View v, DragEvent event) {
				// TODO Auto-generated method stub

				switch(event.getAction()){
				case DragEvent.ACTION_DRAG_STARTED:
					Display1.setText(database.peek());
					Display1.append("\nNow like is "+database.getLike());

					break;
				case DragEvent.ACTION_DRAG_ENTERED:

					break;
				case DragEvent.ACTION_DRAG_LOCATION:
					Display_x =  event.getX();
					Display_y =  event.getY();
					get_drag_location=true;
					break;
				case DragEvent.ACTION_DRAG_EXITED:
					break;

				case DragEvent.ACTION_DROP:
					Display_x =  event.getX();
					Display_y =  event.getY();
					get_drag_location=true;
					break;
				case DragEvent.ACTION_DRAG_ENDED:

							if(Display_y>220.0f&&get_drag_location){

								database.like();
								Display1.setText(database.next());
							}else{
//							if(Display_x<250.0f&&Display_x>50.0f){
								if(get_drag_location==true){
								Display1.setText(database.current());
								get_drag_location=false;
								}else{

Display1.setText(database.next());
								}
							}
							Display1.append("\nNow like is
"+database.getLike());



					break;
				default:
					break;
				}

				return true;
			}
		});
	}
```

```
	private void showActionBar(){



			actionBar=getActionBar();
			actionBar.show();
			LayoutInflater inflater = (LayoutInflater)
this.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
			View v = inflater.inflate(R.layout.custom_actionbar,null);

		actionBar.setDisplayShowCustomEnabled(true);

		actionBar.setCustomView(v);

		ImageButton Button_options = (ImageButton)findViewById(R.id.options);
		Button_options.setOnClickListener(new View.OnClickListener() {

				@Override
				public void onClick(View v) {
					// TODO Auto-generated method stub
					Intent open_menu = new
Intent("com.example.slipui.MENUACTIVITY");
					startActivity(open_menu);

				}
			});
	}



}
```