组员：

苗新润 2183311162 计算机86

陈新妍 2196123751 机类902

蜘蛛纸牌游戏

一．操作规则

1：移牌

鼠标拖动移动一张或一组牌到令一张牌上面或空牌叠。每次移动的牌都只能放在一叠牌全部移除后的空白位置或者比它下面的一张牌大1点的牌上。

当只有一组牌全部为同一花色，方才可以拖动这一组牌。否则，只能移动这一张牌或同一花色的多张牌。

当移动形成同一花色由K到A顺序的一组牌时，这组牌会被自动移除整理至左下方，同时获得分数奖励。

2.发牌

当需要发牌时，单机界面右下角的未发牌叠，可执行发牌操作。该操作不扣分。执行发牌操作时，界面上方的牌叠位必须都有牌，否则不能执行发牌操作。

发牌时，将自动在每叠最上面发一张翻开状态的牌。

二．设计模式

所有继承Istate接口的都是状态模式的实例，每一个类也都用了单例模式 防止多个实例产生

三．代码源

using System;

using System.Collections.Generic;

using System.Linq;

using System.Drawing;

namespace Spyder

{

class Beginnig : IState

{

//Singleton模式

protected Beginnig() { }

private static Beginnig \_instance = null;

public static Beginnig Instance()

{

if (\_instance == null)

\_instance = new Beginnig();

return \_instance;

}

//绘图参数

private float buttonWidthToScreenWidth = 0.5F;

private float buttonHeightToScreenHeight = 0.2F;

private float buttonVerticalIntervalToScreenHeight = 0.3F;

private float buttonLeftUpHorizontalToOrigin = 0.25F;

private float buttonLeftUpVerticalToOrigin = 0.25F;

int width;

int height;

int buttonWidth;

int buttonHeight;

int buttonInterval;

int leftUpX;

int leftUpY;

//绘图工具

SolidBrush btnBrush = new SolidBrush(Color.Pink);

public void HandleMouseInput(int X, int Y)

{

Rectangle rectangle1 = new Rectangle(leftUpX, leftUpY, buttonWidth, buttonHeight);

Rectangle rectangle2 = new Rectangle(leftUpX, leftUpY + buttonInterval, buttonWidth, buttonHeight);

if (rectangle1.Contains(X, Y))

ChangeState(Gaming.Instance());

else if (rectangle2.Contains(X, Y))

ChangeState(HelpPage.Instance());

}

public void HandleKeyboardInput(char ch)

{

//什么也不做

}

public void Draw(Graphics graphics, int screentWidth, int screenHeight)

{

SetArgument(screentWidth, screenHeight);

graphics.FillRectangle(btnBrush, leftUpX, leftUpY, buttonWidth, buttonHeight);

graphics.FillRectangle(btnBrush, leftUpX, leftUpY + buttonInterval, buttonWidth, buttonHeight);

}

public void SetArgument(int screenWidth, int screenHeight)

{

buttonWidth = Convert.ToInt32(screenWidth \* buttonWidthToScreenWidth);

buttonHeight = Convert.ToInt32(screenHeight \* buttonHeightToScreenHeight);

buttonInterval = Convert.ToInt32(screenHeight \* buttonVerticalIntervalToScreenHeight);

leftUpX = Convert.ToInt32(screenWidth \* buttonLeftUpHorizontalToOrigin);

leftUpY = Convert.ToInt32(screenHeight \* buttonLeftUpVerticalToOrigin);

}

StateHolder \_stateHolder = null;

public void SetStateHolder(StateHolder holder) { \_stateHolder = holder; }

public void ChangeState(IState state)

{

\_stateHolder.ChangeState(state);

}

}

}

using System;

using System.Collections.Generic;

namespace Spyder

{

class Board

{

public const int Quantity\_Rows = 10;

private List<ICard>[] lists = new List<ICard>[Quantity\_Rows];

public List<ICard> GetCards(int row) { return lists[row]; }

public ICard GetCard(int row, int index) { return lists[row][index]; }

public int Num(int row)

{

if (row == 10)

return 0;

return lists[row].Count;

}

public int completeCards = 0;

public Board()

{

for (int i = 0; i < Quantity\_Rows; i++)

lists[i] = new List<ICard>();

}

public bool HavingVacancy()

{

foreach (var i in lists)

if (i.Count == 0)

return true;

return false;

}

public bool FlipCard(int row, int index)

{

if (lists[row][index].IsFliped == false && index == Num(row) - 1)

{

lists[row][index].Flip();

return true;

}

return false;

}

public bool Select(int row, int index)

{

if (!lists[row][index].IsFliped)

return false;

if (index == Num(row) - 1)

return true;

for (int i = index; i < Num(row) - 1; i++)

if (!lists[row][i + 1].IsConsistant(lists[row][i]))

return false;

return true;

}

public bool Move(int sourceRow, int headIndex, int targetRow)

{

if (sourceRow == targetRow)

return false;

if (lists[targetRow].Count == 0)

return true;

ICard card = lists[targetRow][Num(targetRow) - 1];

if (card.IsFliped && lists[sourceRow][headIndex].IsConsistant(card))

return true;

return false;

}

public void MoveCards(int sourceRow, int headIndex, int targetRow)

{

int sum = lists[sourceRow].Count - headIndex;

for (int i = headIndex; i < lists[sourceRow].Count; i++)

lists[targetRow].Add(lists[sourceRow][i]);

lists[sourceRow].RemoveRange(headIndex, sum);

}

public void RemoveCrads(int row)

{

int head = Num(row) - 13;

lists[row].RemoveRange(head, 13);

completeCards++;

}

public void AddCards(ICard[] cards)

{

for (int i = 0; i < cards.Length; i++)

lists[i].Add(cards[i]);

}

public void FlipTailCards()

{

for (int i = 0; i < Quantity\_Rows; i++)

if(Num(i) >= 0)

lists[i][Num(i) - 1].Flip();

}

public void FindAndRemoveCompleteCards()

{

for (int i = 0; i < Board.Quantity\_Rows; i++)

if (Num(i) >= 13)

{

int head = Num(i) - 13;

if (lists[i][head].key != Key.KeyK)

continue;

for (int j = head; j < Num(i) - 1; j++)

if (!lists[i][j + 1].IsConsistant(lists[i][j]))

continue;

RemoveCrads(i);

}

}

using System;

using System.Drawing;

namespace Spyder

{

enum Key{ Key1, Key2, Key3, Key4, Key5, Key6, Key7, Key8, Key9, Key10, KeyJ, KeyQ, KeyK, KeyA, KeySmall, KeyBig }

enum Type { Spade, Heart, Diamond, Club }

interface ICard

{

//返回是否被翻过来

bool IsFliped { get; }

//返回1， 2， 3， 。。。

Key key { get; }

//返回花色

Type type { get; }

//把牌翻过来

void Flip();

//判断给定的牌是否与自己构成连续

bool IsConsistant(ICard card);

void PaintBack(Graphics graphics, float X, float Y, float length, float height);

void Paint(Graphics graphics, Image image, float X, float Y);

void Paint(Graphics graphics, float X, float Y, float length, float height);

void Paint(Graphics graphics, SolidBrush brush, Pen pen, float X, float Y, float length, float height);

}

class Card : ICard

{

public Card(Key key, Type type) { this.key = key; this.type = type; }

public bool IsFliped { get; private set; }

public Key key { get; set; }

public Type type { get; set; }

public void Flip()

{

if (IsFliped == false)

IsFliped = true;

}

public bool IsConsistant(ICard card)

{

bool token = false;

switch(card.key)

{

case Key.KeyK:if (key == Key.KeyQ) token = true;break;

case Key.KeyQ: if (key == Key.KeyJ) token = true; break;

case Key.KeyJ: if (key == Key.Key10) token = true; break;

case Key.Key10: if (key == Key.Key9) token = true; break;

case Key.Key9: if (key == Key.Key8) token = true; break;

case Key.Key8: if (key == Key.Key7) token = true; break;

case Key.Key7: if (key == Key.Key6) token = true; break;

case Key.Key6: if (key == Key.Key5) token = true; break;

case Key.Key5: if (key == Key.Key4) token = true; break;

case Key.Key4: if (key == Key.Key3) token = true; break;

case Key.Key3: if (key == Key.Key2) token = true; break;

case Key.Key2: if (key == Key.Key1) token = true; break;

}

return token;

}

public void PaintBack(Graphics graphics, float X, float Y, float length, float height)

{

SolidBrush solid = new SolidBrush(Color.Blue);

Pen pen = new Pen(Color.Black, 2);

graphics.FillRectangle(solid, X, Y, length, height);

graphics.DrawRectangle(pen, X, Y, length, height);

}

public void Paint(Graphics graphics, float X, float Y, float length, float height)

{

if (IsFliped == false)

{

PaintBack(graphics, X, Y, length, height);

return;

}

Pen pen = new Pen(Color.Black);

Font font = new Font(FontFamily.GenericMonospace, 10);

SolidBrush brush = new SolidBrush(Color.Yellow);

graphics.FillRectangle(brush, X, Y, length, height);

graphics.DrawRectangle(pen, X, Y, length, height);

graphics.DrawString(key.ToString(), font, new SolidBrush(Color.Black), X, Y);

}

public void Paint(Graphics graphics, SolidBrush brush, Pen pen, float X, float Y, float length, float height)

{

if (IsFliped == false)

{

PaintBack(graphics, X, Y, length, height);

return;

}

Font font = new Font(FontFamily.GenericMonospace, 10);

graphics.FillRectangle(brush, X, Y, length, height);

graphics.DrawRectangle(pen, X, Y, length, height);

graphics.DrawString(key.ToString(), font, new SolidBrush(Color.Black), X, Y);

}

public void Paint(Graphics graphics, Image image, float X, float Y)

{

//什么也不做

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Spyder

{

public partial class Form1 : Form, StateHolder

{

public Form1()

{

InitializeComponent();

Beginnig.Instance().SetStateHolder(this);

Gaming.Instance().SetStateHolder(this);

}

//数据

private IState \_state = Beginnig.Instance();

public void ChangeState(IState state)

{

\_state = state;

}

private void Form1\_Paint(object sender, PaintEventArgs e)

{

\_state.Draw(e.Graphics, ClientSize.Width, ClientSize.Height);

}

private void Form1\_MouseClick(object sender, MouseEventArgs e)

{

\_state.HandleMouseInput(e.X, e.Y);

Invalidate();

}

private void Form1\_Resize(object sender, EventArgs e)

{

Invalidate();

}

}

}

namespace Spyder

{

partial class Form1

{

/// <summary>

/// 必需的设计器变量。

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// 清理所有正在使用的资源。

/// </summary>

/// <param name="disposing">如果应释放托管资源，为 true；否则为 false。</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows 窗体设计器生成的代码

/// <summary>

/// 设计器支持所需的方法 - 不要修改

/// 使用代码编辑器修改此方法的内容。

/// </summary>

private void InitializeComponent()

{

this.SuspendLayout();

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 15F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(800, 450);

this.DoubleBuffered = true;

this.Name = "Form1";

this.Text = "Form1";

this.Paint += new System.Windows.Forms.PaintEventHandler(this.Form1\_Paint);

this.MouseClick += new System.Windows.Forms.MouseEventHandler(this.Form1\_MouseClick);

this.Resize += new System.EventHandler(this.Form1\_Resize);

this.ResumeLayout(false);

}

#endregion

}

}

using System;

using System.Drawing;

namespace Spyder

{

class GameDrawing

{

private Board board;

private Shuffler shuffler = new Shuffler();

int SelectedRow = -1;

int SelectedIndex = -1;

public float cardBlank = 0.05F;

public float coverPart = 0.2F;

public float cardVerticalPart = 0.25F;

//牌堆参数

private float \_pilesX = 0.1F;

private float \_pilesY = 0.75F;

private float \_pilesInterval = 0.1F;

public float CoverPart(int row)

{

int num = board.Num(row);

if (num < 10)

return 0.2F;

else if (num < 20)

return 0.15F;

else

return 0.1F;

}

//洗牌器位置

public float shufferCoordinateX = 0.75F;

public float shufferCoordinateY = 0.75F;

private float shufflerInterval = 0.1F;

private SolidBrush backgroundBursh = new SolidBrush(Color.Green);

//绘制背景

public void PaintBackground(Graphics graphics, int CanvasX, int CanvasY, int CanvasCoordinateX, int CanvasCoordinateY)

{

graphics.FillRectangle(backgroundBursh, CanvasCoordinateX, CanvasCoordinateY, CanvasX, CanvasY);

}

//绘制发牌器

public void PaintShuffler(Graphics graphics, int CanvasX, int CanvasY, int CanvasCoordinateX, int CanvasCoordinateY)

{

ICard card = new Card(Key.Key1, Type.Club);

int leftUpX = Convert.ToInt32(CanvasX \* shufferCoordinateX);

int leftUpY = Convert.ToInt32(CanvasY \* shufferCoordinateY);

int width = Convert.ToInt32(CanvasX / 10 \* (1 - 2 \* cardBlank));

int height = Convert.ToInt32(CanvasY \* cardVerticalPart \* (1 - 2 \* cardBlank));

int interval = Convert.ToInt32(CanvasX / 10 \* shufflerInterval);

for (int i = 0; i < shuffler.Remains / 10 ; i++)

card.PaintBack(graphics, leftUpX + i \* interval, leftUpY, width, height);

}

//绘制已经完成的排堆

public void PaintPiles(Graphics graphics, int CanvasX, int CanvasY, int CanvasCoordinateX, int CanvasCoordinateY)

{

ICard card = new Card(Key.KeyK, Type.Club);

card.Flip();

int leftUpX = Convert.ToInt32(CanvasX \* \_pilesX);

int leftUpY = Convert.ToInt32(CanvasY \* \_pilesY);

int width = Convert.ToInt32(CanvasX / 10 \* (1 - 2 \* cardBlank));

int height = Convert.ToInt32(CanvasY \* cardVerticalPart \* (1 - 2 \* cardBlank));

int interval = Convert.ToInt32(CanvasX / 10 \* \_pilesInterval);

for (int i = 0; i < board.completeCards; i++)

card.Paint(graphics, leftUpX + i \* interval, leftUpY, width, height);

}

//绘制所有牌组

public void PaintCards(Graphics graphics, int CanvasX, int CanvasY, int CanvasCoordinateX, int CanvasCoordinateY)

{

//绘制参数

float intervalX = CanvasX \* 1.0F / Board.Quantity\_Rows;

float length = intervalX \* (1 - 2 \* cardBlank);

float intervalY = CanvasY \* cardVerticalPart;

float height = intervalY \* (1 - 2 \* cardBlank);

for (int i = 0; i < Board.Quantity\_Rows; i++)

for (int j = 0; j < board.Num(i); j++)

{

ICard card = board.GetCard(i, j);

card.Paint(graphics, CanvasCoordinateX + i \* intervalX + intervalX \* cardBlank, CanvasCoordinateY + j \* intervalY \* CoverPart(i) + intervalY \* cardBlank, length, height);

}

}

//绘制等待的牌组

public void PaintSlectedCards(Graphics graphics, int CanvasX, int CanvasY, int CanvasCoordinateX, int CanvasCoordinateY)

{

if (SelectedIndex == -1)

return;

SolidBrush selectedBrush = new SolidBrush(Color.Pink);

Pen pen = new Pen(Color.Black, 2);

int row = SelectedRow;

int index = SelectedIndex;

int num = board.Num(row);

//绘制参数

float intervalX = CanvasX \* 1.0F / Board.Quantity\_Rows;

float length = intervalX \* (1 - 2 \* cardBlank);

float intervalY = CanvasY \* cardVerticalPart;

float height = intervalY \* (1 - 2 \* cardBlank);

for (int i = index; i < num; i++)

board.GetCard(row, i).Paint(graphics, selectedBrush, pen, CanvasCoordinateX + row \* intervalX + intervalX \* cardBlank, CanvasCoordinateY + i \* intervalY \* CoverPart(row) + intervalY \* cardBlank, length, height);

}

public GameDrawing(Board board) { this.board = board; }

//判断是否有效点击，若有效则切换

public bool MouseClickInCards(int X, int Y, int CanvasX, int CanvasY, int CanvasCoordinateX, int CanvasCoordinateY)

{

//选择点到的牌

X -= CanvasCoordinateX;

Y -= CanvasCoordinateY;

int length = CanvasX / Board.Quantity\_Rows;

int height = Convert.ToInt32(CanvasY \* cardVerticalPart);

int row = X / length > 9 ? 9 : X / length;

int index;

if (board.Num(row) == 0 && SelectedIndex == -1)

return false;

if (Y > (board.Num(row) - 1) \* height \* CoverPart(row) && Y < (board.Num(row) - 1) \* height \* CoverPart(row) + height)

index = board.Num(row) - 1;

else

index = (Y - Convert.ToInt32(height \* cardBlank)) / Convert.ToInt32(height \* CoverPart(row));

if(index > board.Num(row))

{

SelectedIndex = SelectedRow = -1;

return true;

}

//选择牌

if (row == 10)

row--;

if (SelectedIndex == -1)

MouseToSetSelected(row, index);

//判断移动并移动

else if(!MouseToMove(row, index))

MouseToSetSelected(row, index);

return true;

}

//无选定牌操作

private void MouseToSetSelected(int row, int index)

{

if (board.FlipCard(row, index))

;

else if (board.Select(row, index))

{

SelectedIndex = index;

SelectedRow = row;

}

}

//判断移动并移动并移除完整牌组

private bool MouseToMove(int row, int index)

{

bool token = false;

if (board.Move(SelectedRow, SelectedIndex, row))

{

board.MoveCards(SelectedRow, SelectedIndex, row);

token = true;

board.FindAndRemoveCompleteCards();

}

SelectedIndex = SelectedRow = -1;

return token;

}

//判断是否点击发牌器

public bool MouseClickInShuffer(int X, int Y, int CanvasX, int CanvasY, int CanvasCoordinateX, int CanvasCoordinateY)

{

if (X > CanvasCoordinateX + CanvasX \* shufferCoordinateX && X < CanvasCoordinateX + CanvasX \* shufferCoordinateX + 50

&& Y > CanvasCoordinateY + CanvasY \* shufferCoordinateY && Y < CanvasCoordinateY + CanvasY \* shufferCoordinateY + 50)

return true;

return false;

}

//发牌

public void Shuffle(int num)

{

Card[] newCards = shuffler.SendCards(num);

board.AddCards(newCards);

SelectedRow = SelectedIndex = -1;

}

public void FlipTailCards() { board.FlipTailCards(); }

public bool Finished() { return board.completeCards == 8; }

}

}

using System;

using System.Drawing;

namespace Spyder

{

class Gaming : IState

{

GameDrawing gameDrawing;

//Singleton模式

protected Gaming()

{

gameDrawing = new GameDrawing(new Board());

gameDrawing.Shuffle(10);

gameDrawing.Shuffle(10);

gameDrawing.Shuffle(10);

gameDrawing.Shuffle(9);

}

private static Gaming \_instance = null;

public static Gaming Instance()

{

if (\_instance == null)

\_instance = new Gaming();

return \_instance;

}

//绘图参数

int width;

int height;

public void ChangeState(IState state)

{

}

public void Draw(Graphics graphics, int screentWidth, int screenHeight)

{

SetArgument(screentWidth, screenHeight);

gameDrawing.PaintBackground(graphics, screentWidth, screentWidth, 0, 0);

gameDrawing.PaintCards(graphics, screentWidth, screenHeight, 0, 0);

gameDrawing.PaintSlectedCards(graphics, screentWidth, screenHeight, 0, 0);

gameDrawing.PaintPiles(graphics, screentWidth, screenHeight, 0, 0); ;

gameDrawing.PaintShuffler(graphics, screentWidth, screenHeight, 0, 0);

gameDrawing.PaintSlectedCards(graphics, screentWidth, screenHeight, 0, 0);

}

public void HandleKeyboardInput(char ch)

{

//什么也不做

}

public void HandleMouseInput(int X, int Y)

{

if (gameDrawing.MouseClickInShuffer(X, Y, width, height, 0, 0))

{

gameDrawing.Shuffle(10);

gameDrawing.FlipTailCards();

}

else if (gameDrawing.MouseClickInCards(X, Y, width, height, 0, 0))

;

if (gameDrawing.Finished())

ChangeState(Triumph.Instance());

}

public void SetArgument(int screenWidth, int screenHeight)

{

width = screenWidth;

height = screenHeight;

}

StateHolder \_holder = null;

public void SetStateHolder(StateHolder holder)

{

\_holder = holder;

}

}

}

using System;

using System.Drawing;

namespace Spyder

{

class HelpPage : IState

{

//Singleton模式

protected HelpPage() { }

private static HelpPage \_instance = null;

public static HelpPage Instance()

{

if (\_instance == null)

\_instance = new HelpPage();

return \_instance;

}

public void ChangeState(IState state)

{

throw new NotImplementedException();

}

public void Draw(Graphics graphics, int screentWidth, int screenHeight)

{

throw new NotImplementedException();

}

public void HandleKeyboardInput(char ch)

{

throw new NotImplementedException();

}

public void HandleMouseInput(int X, int Y)

{

throw new NotImplementedException();

}

public void SetArgument(int screenWidth, int screenHeight)

{

throw new NotImplementedException();

}

public void SetStateHolder(StateHolder holder)

{

throw new NotImplementedException();

}

}

}

using System;

using System.Drawing;

namespace Spyder

{

public interface StateHolder

{

void ChangeState(IState state);

}

public interface IState

{

void HandleMouseInput(int X, int Y);

void HandleKeyboardInput(char ch);

void Draw(Graphics graphics, int screentWidth, int screenHeight);

void SetArgument(int screenWidth, int screenHeight);

void SetStateHolder(StateHolder holder);

void ChangeState(IState state);

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Spyder

{

static class Program

{

/// <summary>

/// 应用程序的主入口点。

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Spyder

{

class Shuffler

{

private List<Card> \_cards = new List<Card>();

public int Remains { get; private set; }

public Card[] cards = new Card[13 \* 8];

public Shuffler()

{

for (int i = 0; i < 13 \* 8; i++)

\_cards.Add(new Card(Key.Key1, Type.Club));

Remains = 13 \* 8;

for (int i = 0; i < 8; i++)

InitializeCrads(13 \* i, i % 4);

Card[] aux = new Card[13 \* 8];

Random random = new Random(DateTime.Now.Minute);

for (int i = 0; i < 13 \* 8; i++)

{

int index = random.Next() % \_cards.Count;

aux[i] = \_cards.ElementAt(index);

\_cards.RemoveAt(index);

}

aux.CopyTo(cards, 0);

}

private void InitializeCrads(int index, int typeIndex)

{

\_cards[index] = new Card(Key.Key1, Type.Club);

\_cards[index + 1] = new Card(Key.Key2, Type.Club);

\_cards[index + 2] = new Card(Key.Key3, Type.Club);

\_cards[index + 3] = new Card(Key.Key4, Type.Club);

\_cards[index + 4] = new Card(Key.Key5, Type.Club);

\_cards[index + 5] = new Card(Key.Key6, Type.Club);

\_cards[index + 6] = new Card(Key.Key7, Type.Club);

\_cards[index + 7] = new Card(Key.Key8, Type.Club);

\_cards[index + 8] = new Card(Key.Key9, Type.Club);

\_cards[index + 9] = new Card(Key.Key10, Type.Club);

\_cards[index + 10] = new Card(Key.KeyJ, Type.Club);

\_cards[index + 11] = new Card(Key.KeyQ, Type.Club);

\_cards[index + 12] = new Card(Key.KeyK, Type.Club);

Type type = Type.Spade;

switch (typeIndex % 4)

{

case 0: type = Type.Club; break;

case 1: type = Type.Diamond; break;

case 2: type = Type.Heart; break;

case 3: type = Type.Spade; break;

}

for (int i = index; i < index + 13; i++)

\_cards[i].type = type;

}

public Card[] SendCards(int num)

{

int numToSend = num > Remains ? Remains : num;

Card[] ret = new Card[numToSend];

for (int i = 0; i < numToSend; i++)

ret[i] = cards[Remains - i - 1];

Remains -= numToSend;

return ret;

}

}

}

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Spyder

{

class Triumph : IState

{

protected Triumph() { }

private static Triumph \_instance = null;

public static Triumph Instance()

{

if (\_instance == null)

\_instance = new Triumph();

return \_instance;

}

public void ChangeState(IState state)

{

\_holder.ChangeState(state);

}

public void Draw(Graphics graphics, int screentWidth, int screenHeight)

{

//绘制完成画面

}

public void HandleKeyboardInput(char ch)

{

//什么也不做

}

public void HandleMouseInput(int X, int Y)

{

}

public void SetArgument(int screenWidth, int screenHeight)

{

}

StateHolder \_holder = null;

public void SetStateHolder(StateHolder holder)

{

holder = \_holder;

}

}

}

四．运行结果