

# 贪吃蛇称霸世界项目概要设计说明书

作者: 202056304 李瑞彬

202056330 王欣宇

202056314 袁金宇

## 文档变更记录

日期	版本号	修订内容	修订	审核
2022.5.11	V1.0	填写引言、项目概述、游戏策划、项目进度安排。	王欣宇 袁金宇 李瑞彬	李瑞彬
2022.5.15	V1.1	填写引言、项目概述、项目用例描述。	王欣宇 袁金宇 李瑞彬	李瑞彬
2022.6.16	V1.2	填写引言、项目概述、项目设计思想, 系统总体设计, 系统接口设计, 系统数据结构设计, 系统运行设计, 系统出错设计, 附录	李瑞彬 王欣宇 袁金宇	李瑞彬

## 1. 引言

本说明书对贪吃蛇游戏的游戏玩法方面进行了要求和说明。软件开发小组的产品实现人员阅读和参考本说明进行代码的编写和测试。本说明书的预期玩家为软件开发小组的产品实现人员。

## 2. 项目概述

项目开发背景: "贪吃蛇" 游戏是一个经典的游戏, 它因操作简单、娱乐性强而广受欢迎, 本文基于 C# 编程技术, 开发了一个操作简单、界面美观、功能较为齐全的“贪吃蛇”游戏。贪吃蛇的设计简单, 实用和娱乐性高, 其玩法早已人尽皆知。伴随着科技的发展, 贪吃蛇的版本越来越多, 玩法也不尽相同。

项目开发意义: 贪吃蛇游戏适合各同学联合开展活动, 因为它比拓展训练更具趣味及凝聚力, 能更好的把校园文化渗透到每个同学, 增进同学与同学之间的亲密感, 从中学会团队协作。另一方面它具有强身健体的作用, 对于埋头学习或者游戏的学生来说更具吸引力。它的竞技性、趣味性及观赏性带给同学们另一种

新奇，容易使整个活动形成兴奋、热烈、互动的气氛，应变能力能够进一步的提高。

目标：完成贪吃蛇游戏的完整功能设计。

1. 游戏开始时，蛇身长度固定，位置可以不固定；
2. 蛇头撞到自身或围栏游戏结束，从最初级别重新开始，撞到食物蛇身长度增加一个单位；
3. 蛇可以不断捕食食物；
4. 根据难度选择，用户可自行选择属于自己的节奏；
5. 可以用键盘控制蛇的运动方向，以及暂停和退出游戏，无控制时蛇有默认运动方向，并按此方向运动；
6. 显示当前游戏级别、得分情况及帮助。

适用范围：针对所有手机用户（如学生、老师、职员、工人等）。

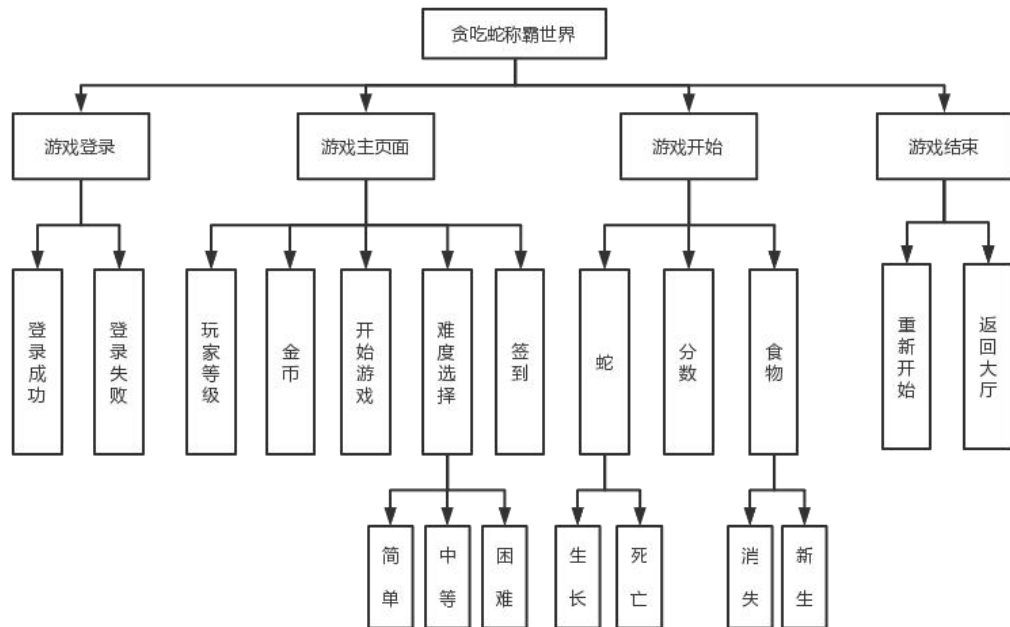
作用：许多新版的贪吃蛇可玩性很高，不仅增加了游戏难度，而且还能和朋友一起实时游戏，逐渐成为了人们消磨时间、放松心情的必玩小游戏。

### 3. 项目设计思想

程序关键在于调用另一个窗体来完成登录，同时在主页面设置 combobox 组件来完成难度选择，在游戏里面通过画圆形来表示蛇的身体。用一个小圆形块表示蛇的一节身体，身体每长一节，增加一个矩形块。移动时必须从蛇头开始，所以蛇不可以向相反的方向移动，如果不按任意键，蛇自行在当前方向右前移，但按下有效方向键后，蛇头朝该方向移动，一步移动一节身体，所以按下有效方向键后，先确定蛇头的位置，而后蛇的身体随蛇头移动，图形的实现是从蛇头新位置开始画出蛇，这时，由于未清屏的原因，原来的蛇的位置和新的位置差一个单位，所以看起来蛇多一节身体，所以将蛇的最后节用背景色覆盖。食物的出现与消失也是画矩形块和覆盖矩形块。为了便于理解，定义两个结构体：食物与蛇，同时还定义了一个随意的墙。然后利用图形驱动，制作美观的游戏界面，通过随机函数产生随机的食物，控制游戏过程食物的出现。定义键盘操作控制游戏过程蛇的移动方向，画出边界，并判断游戏是否结束，统计游戏过程中蛇吃的食物数量，计算并输出游戏成绩。

## 4. 系统总体设计

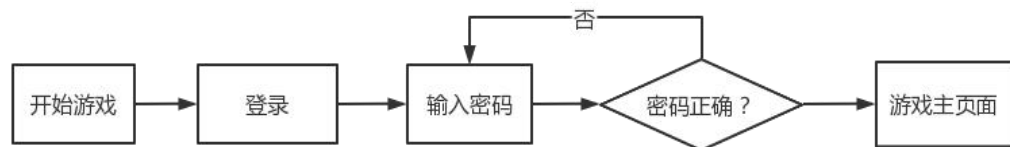
### 4.1 系统架构设计

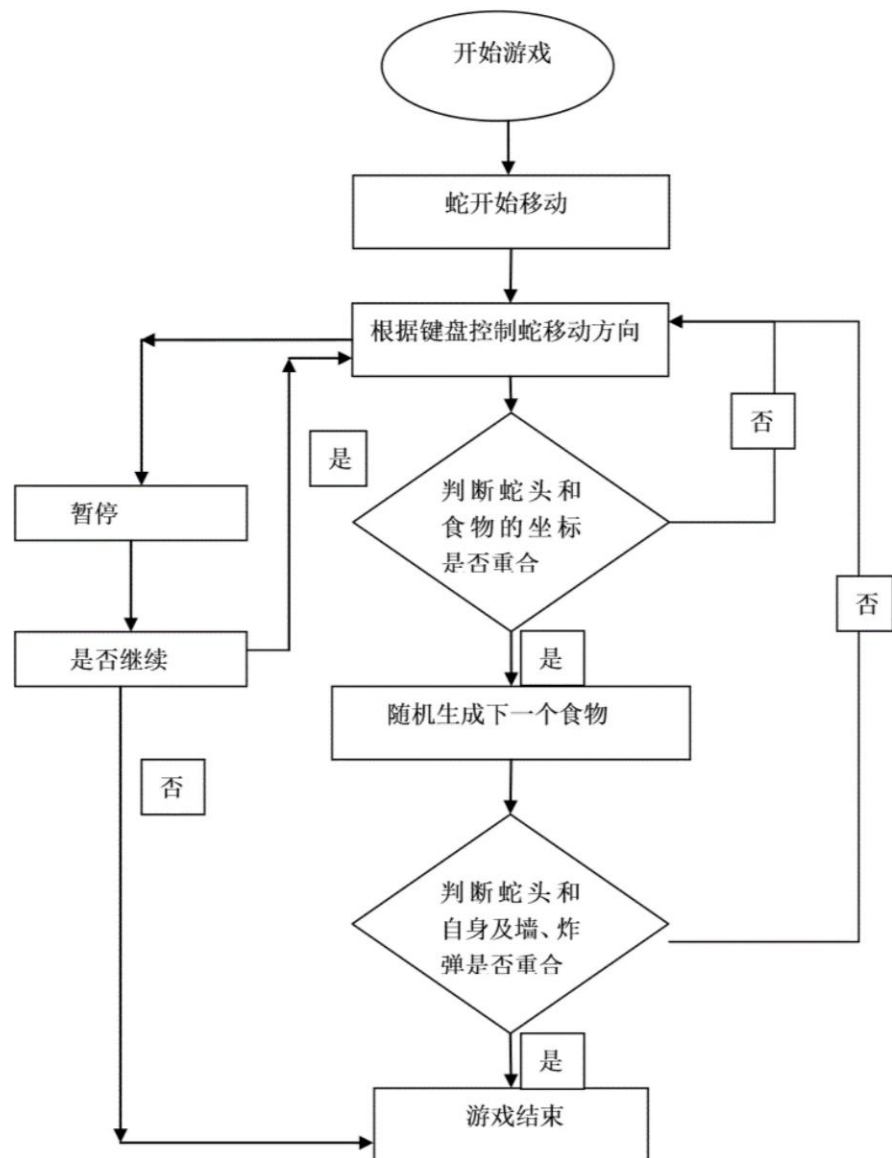


### 4.2 系统运行环境

- 运行环境: Windows10
- 编程语言: C#
- 使用工具: Visual Studio 2019

### 4.3 系统结构

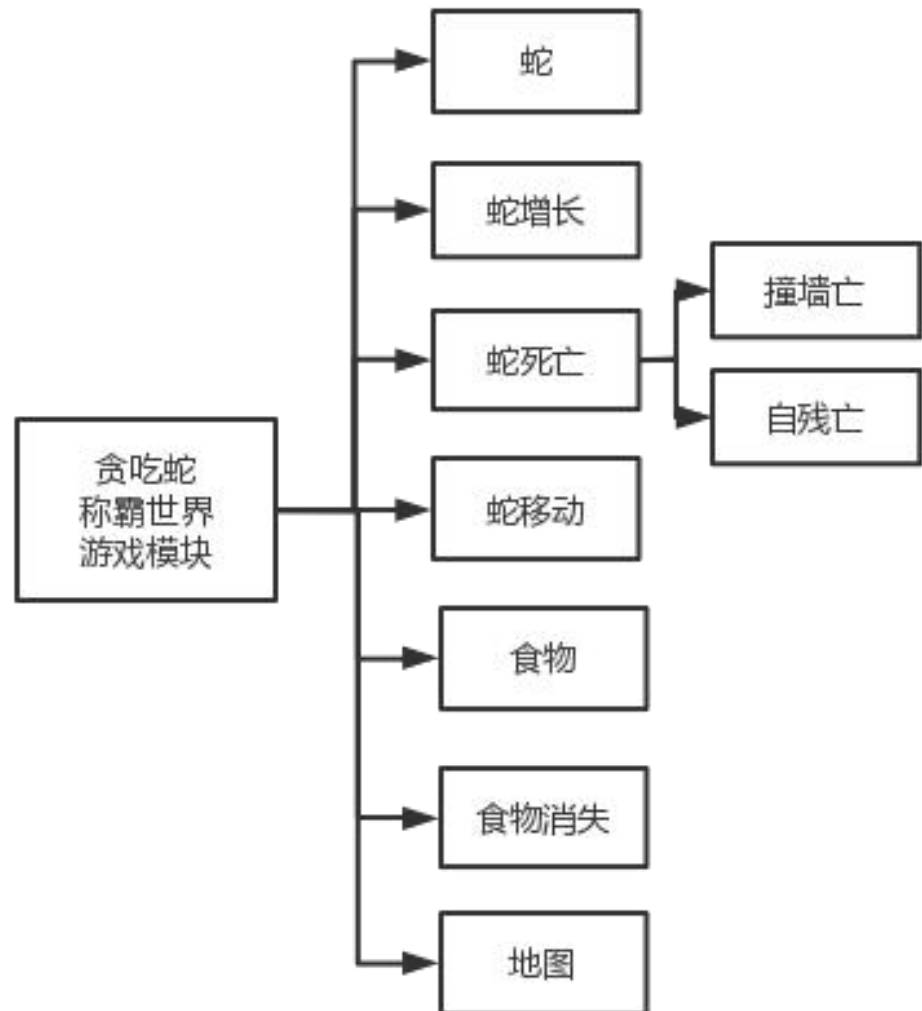




#### 4.4 尚未解决的问题

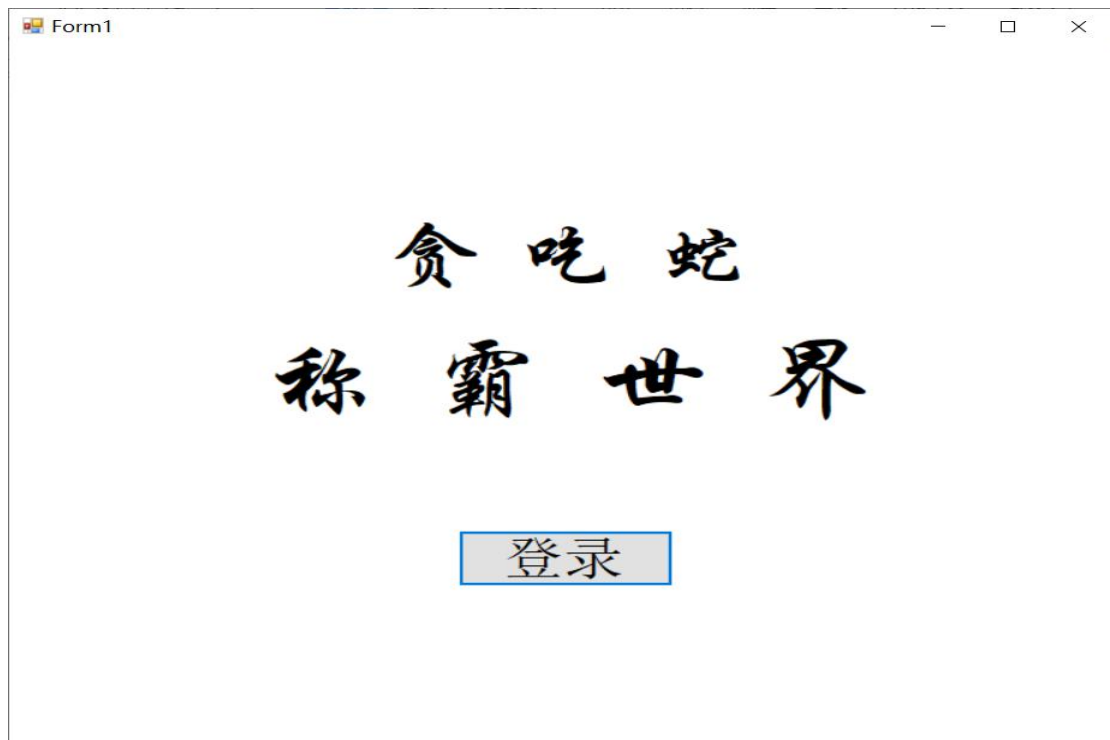
1. 登录时只能验证，未实现注册功能
2. 游戏结束不会增加经验和金币，未实现其增加功能
3. 新食物有时出现在动物身体内
4. 蛇头只能和蛇身保持一致的颜色，无法设置蛇头的颜色
5. 按键过快导致蛇莫名其妙的死亡

## 5. 系统接口设计

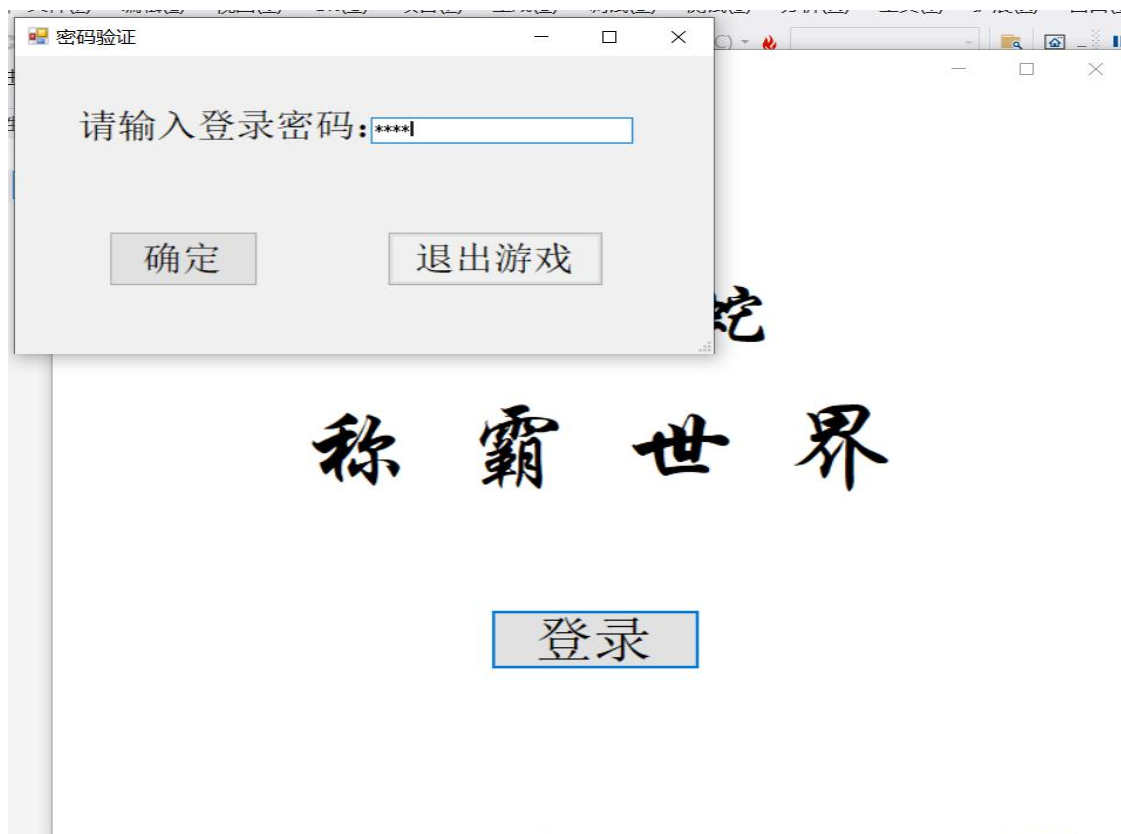


## 6. 系统运行设计

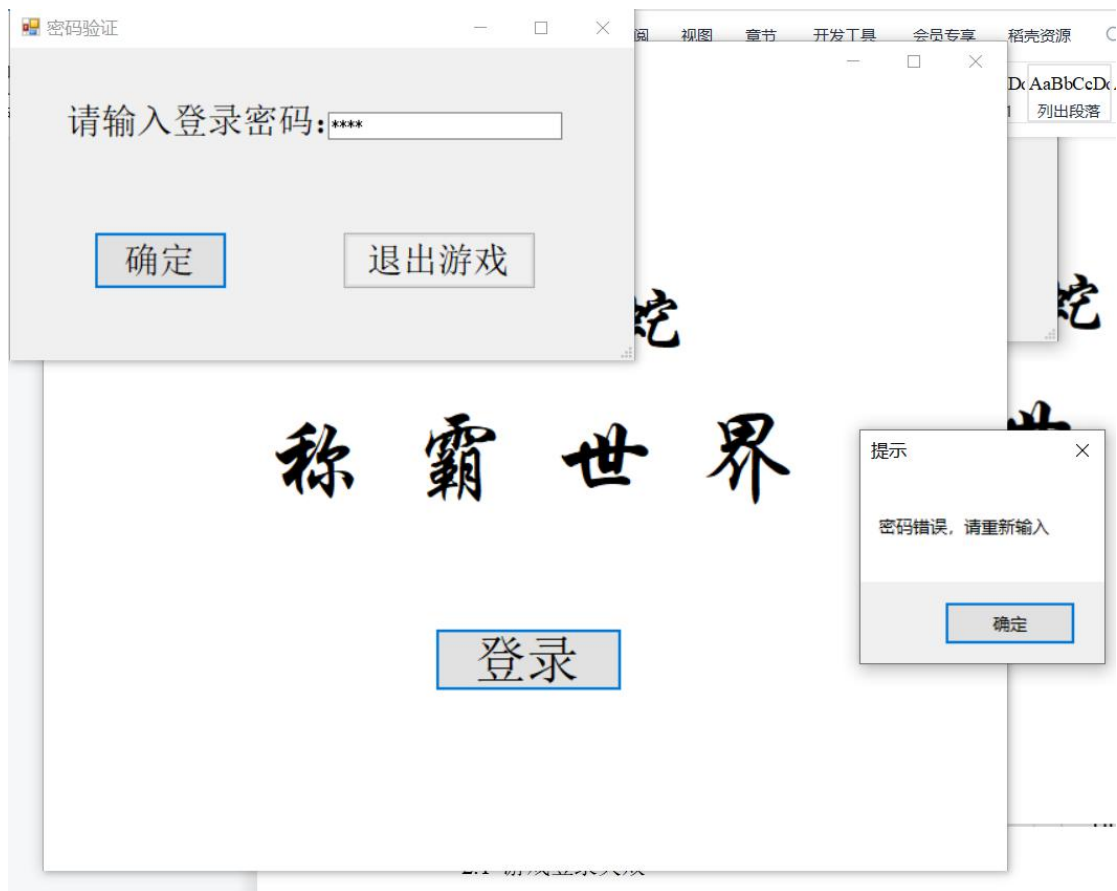
### 1. 游戏起始页面



### 2. 游戏登录页面



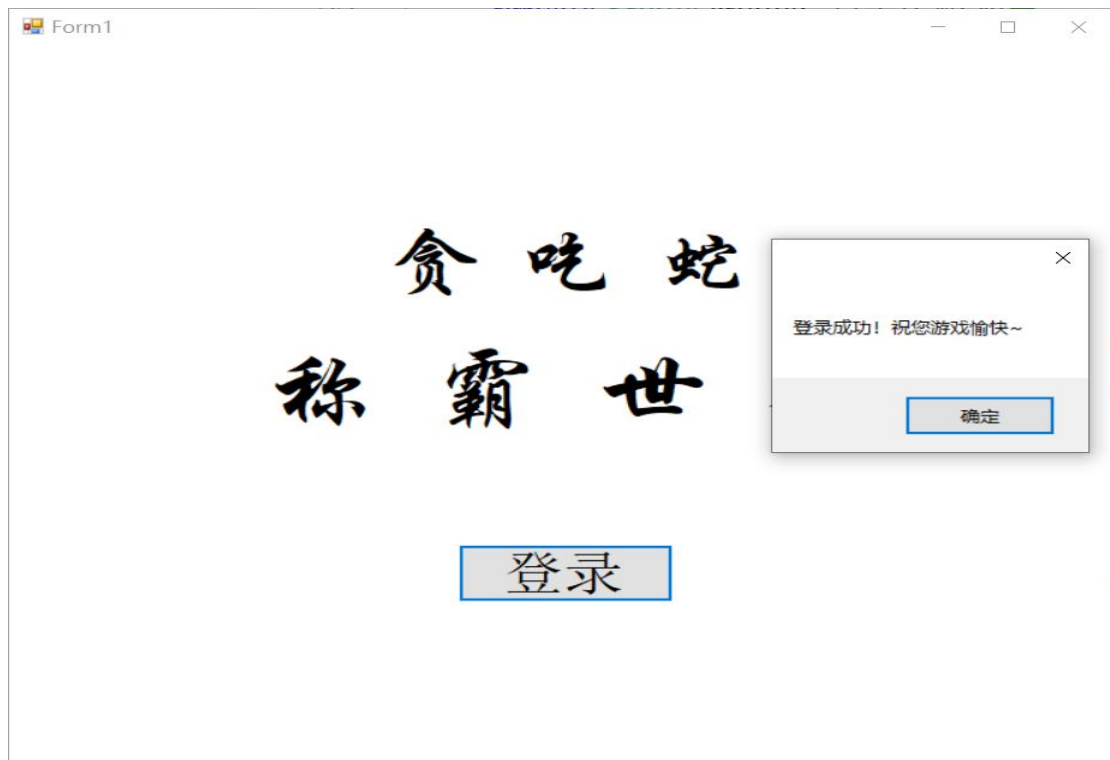
## 2.1 游戏登录失败



## 2.2 游戏登录失败三次



## 2.3 游戏登录成功

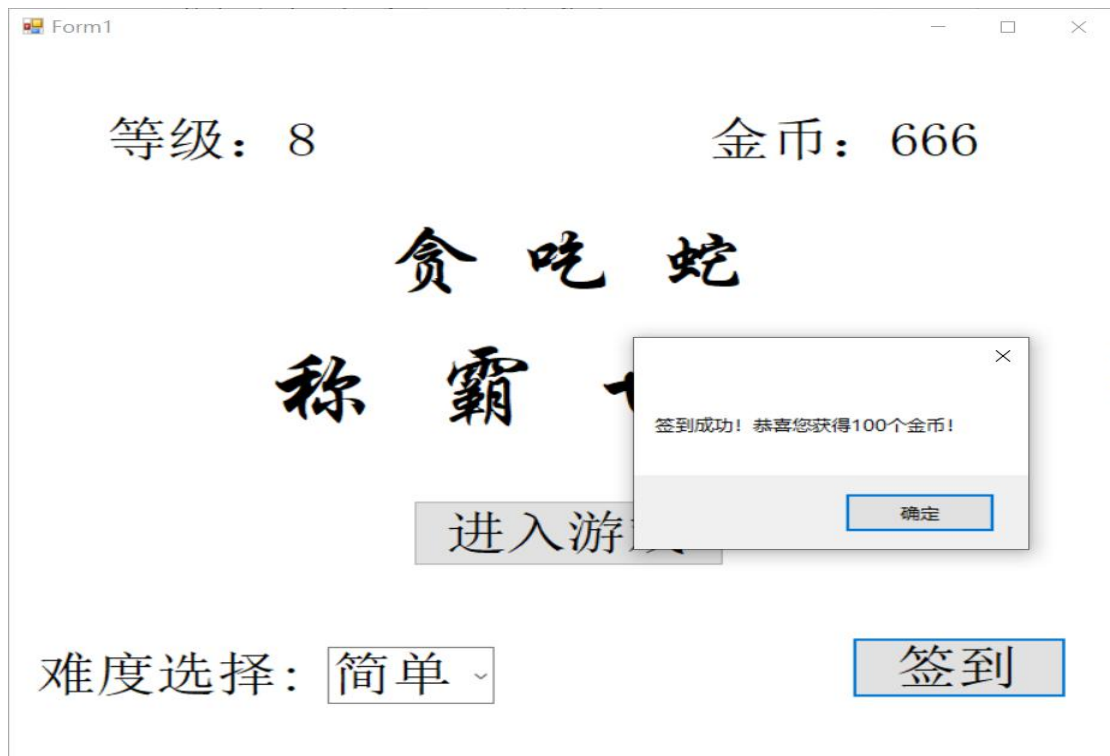


## 3. 游戏主页面






#### 4. 签到成功



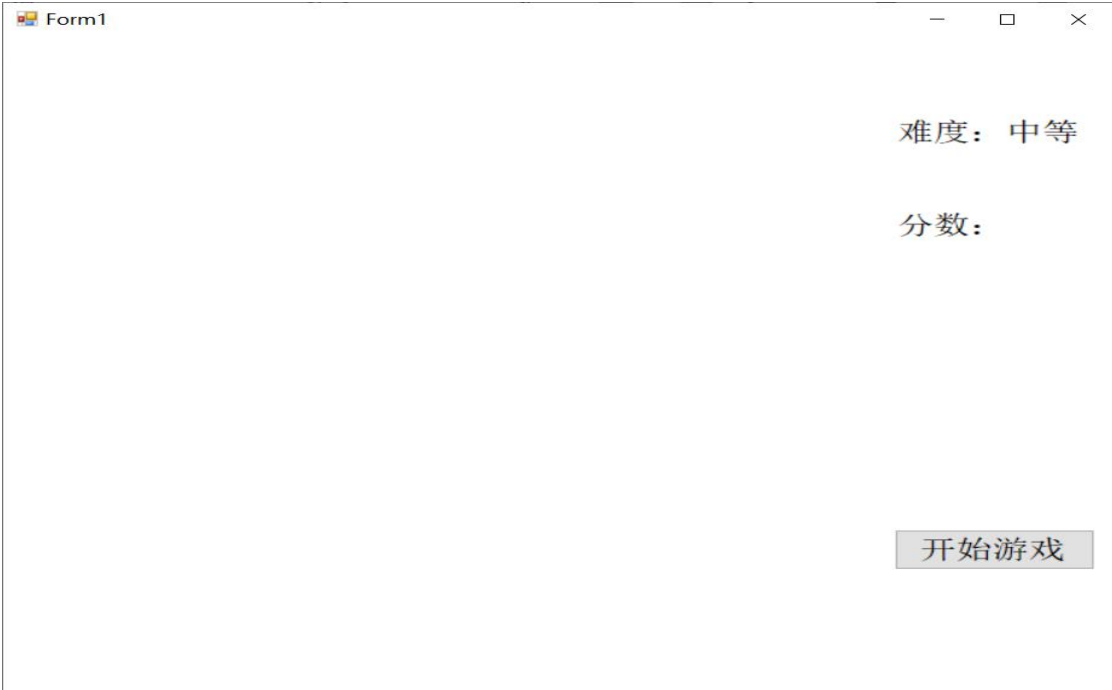
The screenshot shows a game window titled 'Form1'. At the top, it displays '等级: 8' (Level: 8) and '金币: 666' (Gold: 666). The main title '贪吃蛇' (Snake Game) is prominently displayed in a large, stylized font, followed by '称霸' (Dominate). Below this, there is a button labeled '进入游戏' (Enter Game). At the bottom, there is a '难度选择' (Difficulty Selection) dropdown menu currently set to '简单' (Easy), and a '签到' (Login) button. A small dialog box is open in the center, displaying the message '签到成功! 恭喜您获得100个金币!' (Login successful! Congratulations on obtaining 100 gold coins!) with a '确定' (Confirm) button.

#### 5. 难度选择



The screenshot shows the same game window 'Form1'. The '等级' (Level) is still 8, but the '金币' (Gold) has increased to 766. The main title '贪吃蛇' (Snake Game) and '称霸' (Dominate) remain. The '进入游戏' (Enter Game) button is still present. The '难度选择' (Difficulty Selection) dropdown menu is now open, showing four options: '中等' (Medium), '简单' (Easy), '中等' (Medium), and '困难' (Difficult). The '签到' (Login) button is still visible at the bottom right.

## 6. 进入游戏




Form1

难度：中等

分数：

开始游戏

## 7. 游戏进行中

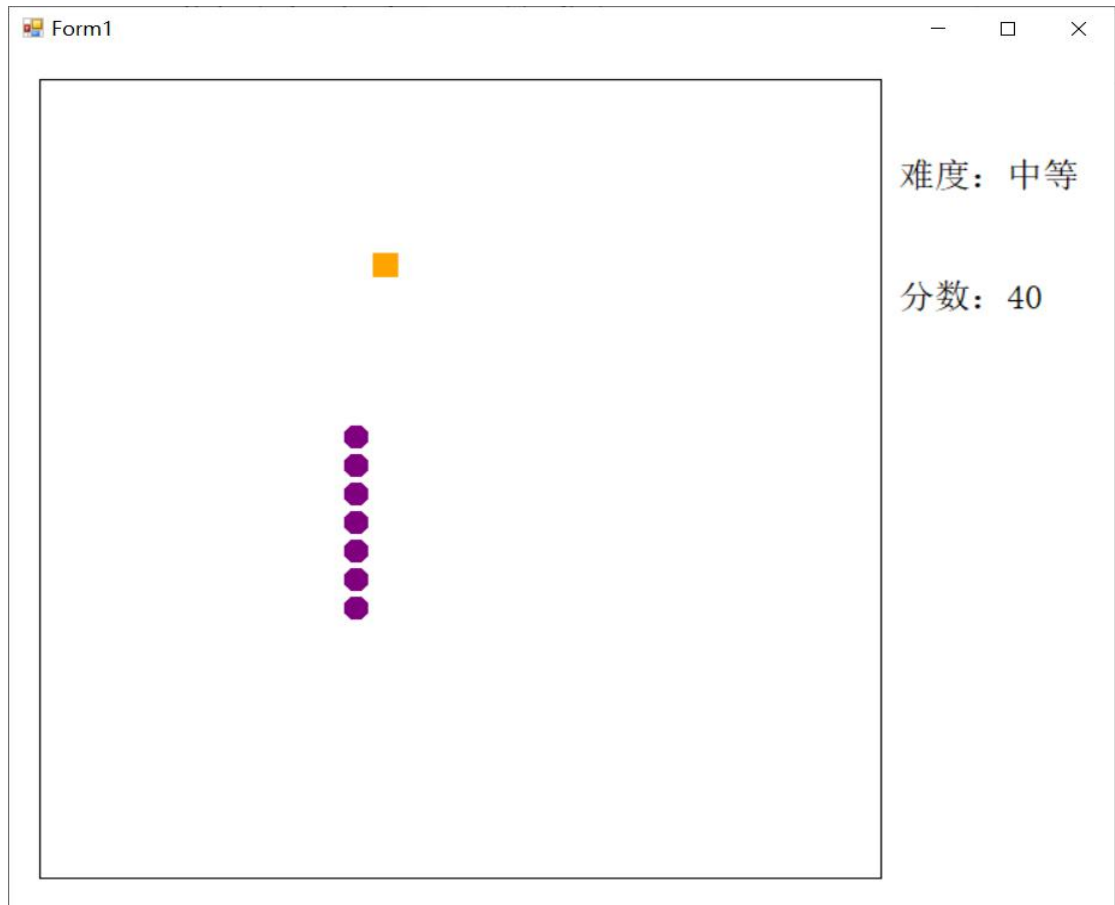


Form1

难度：中等

分数：0

## 8. 蛇增长



## 9. 游戏结束



10. 挑战成功



7. 系统出错设计

问题	解决办法
蛇进入游戏时, 方向键无法控制方向	取消开始游戏按钮
Label 字样显示	Label.visible=false
无法更改蛇的移动速度	设置 SelectedIndexChanged 事件
游戏结束时蛇无法正常清除	设置新属性来判断结束, 同时刷新页面

8. 附录

Block.cs

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;
```

```

using System.Threading.Tasks;
using System.Drawing;

namespace EatSnake
{
    class Block
    {
        // 每一个块的位置和大小
        private int x;
        private int y;
        private int size;
        // 每一个块的相关属性的封装
        public int X { get => x; set => x = value; }
        public int Y { get => y; set => y = value; }
        public int Size { get => size; set => size = value; }

        // 构造方法
        public Block() { }
        public Block(int x, int y, int objSize)
        {
            X = x;
            Y = y;
            Size = objSize;
        }

        // 绘画
        public virtual void Draw(Graphics g, Color c)
        {
            SolidBrush brush = new SolidBrush(c);
            g.FillEllipse(brush, X * Size + 2, Y * Size + 2, Size - 3, Size - 3);//绘制圆形身体
        }

        // 清除
        public void clear(Graphics g, Color c)
        {
            SolidBrush Brush = new SolidBrush(c);
            g.FillRectangle(Brush, X * Size + 1, Y * Size + 1, Size - 1, Size - 1);//填充矩形来清除
        }

    }
}

```

## Snake.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;
//游戏中的主角， 玩家通过上下左右四个键来控制蛇的运动。

namespace EatSnake
{
    class Snake:Block
    {
        //用于存放蛇的集合
        public List<Block> snakeList = new List<Block>();
        // 蛇头
        public Point Head = new Point();
        //方向, 1 蛇上 2 蛇下 3 蛇左 4 蛇右, 也可以用来认知下一个蛇方块的位置
        private int forld;
        // 方向字段封装
        public int Forld { get => forld; set => forld = value; }

        // public Snake() { }
        public Snake(int snake_size, int snakeX, int snakeY, int[][] mapGrid)
        {
            // 蛇块大小
            Size = snake_size;
            //初始化蛇 集合
            Head = new Point(snakeX, snakeY); //蛇头位置
            snakeList.Add(new Block(Head.X - 2, Head.Y, Size)); // 蛇尾
            snakeList.Add(new Block(Head.X - 1, Head.Y, Size));
            snakeList.Add(new Block(Head.X, Head.Y, Size)); //蛇头
            //蛇的初始方向, 赋值代表朝向
            //二维数组 snakeBody[1][2]代表 1 行二列 即先纵向再横向=>二维数组 snakeBody[Y][X]
            mapGrid[snakeList[0].Y][snakeList[0].X] = 4;
            mapGrid[snakeList[1].Y][snakeList[1].X] = 4;
            mapGrid[snakeList[2].Y][snakeList[2].X] = 4;
            Forld = 4;
        }
        // 画蛇
        public override void Draw(Graphics g, Color c)
```

```

{
    foreach (Block item in snakeList)
    {
        item.Draw(g, c);
    }
}

```

// 清除蛇

```

public void Clear(Graphics g, Color c, int[][] mapGrid)
{
    foreach (Block item in snakeList)
    {
        mapGrid[item.Y][item.X] = 0;
        item.clear(g, c);
    }
}

```

// 清除蛇尾

```

public void MoveTail(Graphics g, Color c, int[][] mapGrid)
{
    mapGrid[snakeList[0].Y][snakeList[0].X] = 0;
    snakeList[0].clear(g, c);
    snakeList.RemoveAt(0);
}

```

// 新生蛇头, 保证蛇不断移动

```

public void MoveHead(Graphics g, Color c)
{
    Block B = new Block(Head.X, Head.Y, Size);
    snakeList.Add(B);
    B.Draw(g, c);
}

```

// 检查是否咬到自己

```

public bool IsTouchMyself(int x, int y)
{
    for (int i = 0; i < snakeList.Count; i++)
    {
        if ((x == snakeList[i].X) && y == snakeList[i].Y)
        {
            return true;
        }
    }
    return false;
}

```

```

    }

    // 移动方向
    public void moveUp() { Head.Y--; Forld = 1; }
    public void moveDown() { Head.Y++; Forld = 2; }
    public void moveLeft() { Head.X--; Forld = 3; }
    public void moveRight() { Head.X++; Forld = 4; }

    }
}

```

## Food.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;

namespace EatSnake
{
    class Food:Block
    {
        // 初始化食物，设置食物的大小
        public Food(int foodSize) { Size = foodSize; }

        // 对绘画方法进行重写
        public override void Draw(Graphics g, Color c)
        {
            SolidBrush brush = new SolidBrush(c);
            g.FillRectangle(brush, X * Size + 2, Y * Size + 2, Size - 3, Size - 3);
        }

        // 设置 Food 的位置
        public Food NewFood()
        {
            Random r = new Random();
            Food food = new Food(Size);
            food.X = r.Next(1, 19);
            food.Y = r.Next(1, 19);
            return food;
        }
    }
}

```



```
    }  
}
```

## Form1.cs

```
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 EatSnake  
{  
    public partial class Form1 : Form  
    {  
        public static int unit = 20;    //设定每一个网格的大小  
        public int[][] mapGrid = new int[30][];    // 设定地图的网格  
        public bool isGamePalying = false;    //标记游戏开始  
        public bool isGameOver = false;    //标记游戏结束  
        public int score;    // 定义游戏分数  
  
        private Food food = new Food(unit);    //定义 食物 对象  
        private Snake snake;    //定义 蛇 对象  
        public enum Direction { Left, Right, Up, Down };    //定义上下左右方向  
        public Direction snakeDirection = Direction.Right;    //设置蛇初始方向  
  
        public Form1()  
        {  
            InitializeComponent();  
        }  
        #region 登录  
        // 创建 FrmPwdCheck 对象  
        FrmPwdCheck frmPwdCheck = new FrmPwdCheck();  
        private void btEnter_Click(object sender, EventArgs e)  
        {  
            DialogResult result = frmPwdCheck.ShowDialog();  
            //密码验证成功  
            if (result == DialogResult.OK)  
            {  
                MessageBox.Show("登录成功! 祝您游戏愉快~");  
            }  
        }  
    }  
}
```

```

        // 取消 登录按钮的显示
        btEnter.Visible = false;
        // 显示主页面
        label_level.Visible = true;
        label_money.Visible = true;
        label_Select.Visible = true;
        btSign.Visible = true;
        btEnterGame.Visible = true;
        comboBox1.Visible = true;
    }
}

```

#endregion

#region 游戏初始化

```

private void Form1_Load(object sender, EventArgs e)
{

```

```

    btStart.Visible = false;
    for (int i = 0; i < mapGrid.Length; i++)
    {
        mapGrid[i] = new int[30]; //初始化地图范围
    }

```

```

    snake = new Snake(unit, 10, 9, mapGrid); //蛇的初始位置

```

```

    food = food.NewFood(); //食物随机出现的位置

```

```

    timer1.Enabled = true; // 游戏能正常刷新

```

```

    timer1.Interval = 230; //与上一事件间隔时间 => 蛇的速度，可根

```

据 combox 进行修改

```

    score = 0; //初始游戏分数

```

```

    button_return.Visible = false;

```

```

}

```

#endregion

#region 游戏运行

```

void GamePlay()
{

```

```

    Graphics g = this.CreateGraphics();

```

```

    // 游戏开始

```

```

    if (isGamePalying)
    {

```

```

    {

```

```

        // 蛇移动

```

```

        if (snakeDirection == Direction.Right)
        {

```

```

        {

```

```

            snake.moveRight();

```

```

    }
    else if (snakeDirection == Direction.Down)
    {
        snake.moveDown();
    }
    else if (snakeDirection == Direction.Left)
    {
        snake.moveLeft();
    }
    else if (snakeDirection == Direction.Up)
    {
        snake.moveUp();
    }
    // 检查是否挑战成功
    if (score == 150)
    {
        isGamePalying = false;
        isGameOver = true;
        return;
    }

    // 检查蛇是否碰到墙
    if (snake.Head.X >= 29 || snake.Head.X <= 0 ||
        snake.Head.Y >= 29 || snake.Head.Y <= 0)
    {
        isGamePalying = false;
        isGameOver = true;
        return;
    }

    // 检查蛇是否咬到自己
    if (snake.IsTouchMyself(snake.Head.X, snake.Head.Y))
    {
        isGamePalying = false;
        isGameOver = true;
        return;
    }

    // 检查蛇是否吃到食物
    if (snake.Head.X == food.X && snake.Head.Y == food.Y)
    {
        food.clear(g, BackColor); // 清除已有食物
        food = food.NewFood(); // 初始化新食物
        food.Draw(g, Color.Orange); // 重画食物
    }

```

```

        score += 10; // 游戏加分
    }
    else
    {
        snake.MoveTail(g, BackColor, mapGrid); //清除蛇尾
    }
    snake.MoveHead(g, Color.Purple); //蛇向前移动
}
}

void GameOver()
{
    Graphics g = CreateGraphics();
    g.FillRectangle(new SolidBrush(Color.FromArgb(255, 255, 255)), 0, 0,
1000, 1000);
    snake.Clear(g, BackColor, mapGrid); // 清除已有的蛇
    food.clear(g, BackColor); // 清除食物
    label_over.Visible = true;
    btStart.Visible = true;
    btStart.Location = new Point(300, 400);
    btStart.Text = "重新开始";
    if (score == 150)
    {
        label_over.Text = "挑    战    成    功    ! ";
    }
    label_EndScore.Visible = true;
    label_EndScore.Text = "分数: " + score.ToString();
    label_difficulty.Visible = false;
    label_score.Visible = false;
    button_return.Visible = true;
}

private void timer1_Tick(object sender, EventArgs e)
{
    if (isGameOver)
    {
        GameOver();
        timer1.Enabled = false;
    }
    if (isGamePalying)
    {

```

```

        // 游戏相关页面的准备
        // 绘画
        Bitmap bitmap = new Bitmap(this.ClientSize.Width,
this.ClientSize.Height);
        Graphics g = Graphics.FromImage(bitmap);
        snake.Draw(g, Color.Purple);
        food.Draw(g, Color.Orange);
        // 画地图
        Graphics fg = this.CreateGraphics();
        fg.DrawImage(bitmap, this.ClientRectangle);
        Pen penMap = new Pen(Color.Black);
        fg.DrawRectangle(penMap, unit, unit, unit * (mapGrid.Length -
2), unit * (mapGrid.Length - 2));
        // 显示游戏相关标签
        label_score.Visible = true;
        label_score.Text= "分数: " + score.ToString();
        label_difficulty.Visible = true;
        // 默认显示简单
        if (label_difficulty.Text == "难度: ")
        {
            label_difficulty.Text = "难度: 简单";
        }
        // 因为点击之后, 焦点在 button 不在窗体, 取消显示即可控
制蛇的移动

        btStart.Visible = false;
        // 隐藏标签
        label_over.Visible = false;
        label_EndScore.Visible = false;
        // 游戏运行
        GamePlay();
    }
}

```

#endregion

#region 开始游戏

```

private void btStart_Click(object sender, EventArgs e)
{
    isGamePalying = true;
    timer1.Enabled = true; //启动 timer
    if (isGameOver)
    {
        // 重置 Snake
        //snake.Clear(this.CreateGraphics(), this.BackColor, mapGrid);
    }
}

```

```

// 清除已有的蛇
snake = new Snake(unit, 10, 9, mapGrid); // 实例化新蛇新蛇
snake.Draw(this.CreateGraphics(), Color.Purple); // 画 新蛇
//重置 food
//food.clear(this.CreateGraphics(), this.BackColor);//食物所在矩
形内部为背景色
food = food.NewFood();
food.Draw(this.CreateGraphics(), Color.Orange);//画食物
//重置方向为 right
snakeDirection = Direction.Right;
isGameOver = false;
// 重置分数
score = 0;
// 取消显示返回大厅
button_return.Visible = false;
    }

}
#endregion

#region 方向按键设置
private void Form1_KeyDown(object sender, KeyEventArgs e)
{
    // 不能在同一水平线进行更改方向
    if (e.KeyCode == Keys.Up && snakeDirection != Direction.Down)
    {
        snakeDirection = Direction.Up;
    }
    if (e.KeyCode == Keys.Down && snakeDirection != Direction.Up)
    {
        snakeDirection = Direction.Down;
    }
    if (e.KeyCode == Keys.Left && snakeDirection != Direction.Right)
    {
        snakeDirection = Direction.Left;
    }
    if (e.KeyCode == Keys.Right && snakeDirection != Direction.Left)
    {
        snakeDirection = Direction.Right;
    }
}
#endregion

```

```

#region 取消游戏主页面
private void btEnterGame_Click(object sender, EventArgs e)
{
    //取消 游戏主页面的显示
    label_level.Visible = false;
    label_money.Visible = false;
    label_Select.Visible = false;
    btSign.Visible = false;
    btStart.Visible = false;
    comboBox1.Visible = false;
    label_welcome1.Visible = false;
    label_welcome2.Visible = false;
    btEnterGame.Visible = false;
    // 显示游戏页面
    btStart.Visible = true;
    btStart.Text = "开始游戏";
    btStart.Location = new Point(590, 450);
    label_score.Visible = true;
    label_difficulty.Visible = true;
    // 画地图
    Bitmap bitmap = new Bitmap(this.ClientSize.Width,
this.ClientSize.Height);
    Graphics fg = this.CreateGraphics();
    fg.DrawImage(bitmap, this.ClientRectangle);
    Pen penMap = new Pen(Color.Black);
    fg.DrawRectangle(penMap, unit, unit, unit * (mapGrid.Length - 2),
unit * (mapGrid.Length - 2));
}
#endregion

#region 签到
private void btSign_Click(object sender, EventArgs e)
{
    MessageBox.Show("签到成功! 恭喜您获得 100 个金币! ");
    label_money.Text = "金币: 766";
    btSign.Enabled = false;
}
#endregion

private void comboBox1_SelectedIndexChanged(object sender, EventArgs
e)
{
    // 根据 combox 的选项, 然后更改 timer.Interval, 进而更改蛇的速
度

```

```

// timer 值越小，速度越快
// 默认是 230
switch (comboBox1.SelectedIndex)
{
    case 0:
        timer1.Interval = 230;
        label_difficulty.Text = "难度：简单";
        break;
    case 1:
        timer1.Interval = 150;
        label_difficulty.Text = "难度：中等";
        break;
    case 2:
        timer1.Interval = 50;
        label_difficulty.Text = "难度：困难";
        break;
}
}

private void button_return_Click(object sender, EventArgs e)
{
    Graphics g = CreateGraphics();
    g.FillRectangle(new SolidBrush(Color.FromArgb(255, 255, 255)), 0, 0,
1000, 1000);
    // 显示主页面
    label_welcome1.Visible = true;
    label_welcome2.Visible = true;
    label_level.Visible = true;
    label_money.Visible = true;
    label_Select.Visible = true;
    btSign.Visible = true;
    btEnterGame.Visible = true;
    comboBox1.Visible = true;
    // 取消结算标签
    label_EndScore.Visible = false;
    label_over.Visible = false;
    btStart.Visible = false;
    button_return.Visible = false;
}
}
}

```

**FmPwdCheck.cs**



```

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 EatSnake
{
    public partial class FrmPwdCheck : Form
    {
        //总共次数
        int MAXPWDCOUNT = 3;
        //已输次数
        int pwdcount = 1;
        public FrmPwdCheck()
        {
            InitializeComponent();
        }
        //在输入密码时，设置 textBox_pwd 的 PasswordChar 的值为 *，可保证每一次输入的值都为 *
        // 确定登录按钮
        private void btOK_Click(object sender, EventArgs e)
        {
            if (textBox_pwd.Text == "10086")
            {
                //密码正确，验证成功
                this.DialogResult = DialogResult.OK;
            }
            else
            {
                //密码错误
                pwdcount++;
                if (pwdcount > MAXPWDCOUNT)
                {
                    //验证失败
                    MessageBox.Show("您登录过于频繁，请休息一下");
                    this.DialogResult = DialogResult.Cancel;
                }
            }
        }
    }
}

```

```
        //还剩 MAXPWDCOUNT-pwdcount 次机会
        MessageBox.Show("密码错误, 请重新输入", "提示",
        MessageBoxButtons.OK);
        //将密码框置空
        textBox_pwd.Text = "";
        //重新将光标聚集到输入框
        textBox_pwd.Focus();
    }
}

private void btCancel_Click(object sender, EventArgs e)
{
    MessageBox.Show("退出成功! ");
    this.DialogResult = DialogResult.Cancel;
}
}
```