**迷宫游戏**

目录

[一、游戏简介](#_Toc134216934)

[二、系统概述](#_Toc134216935)

[三、系统设计](#_Toc134216936)

[（一）总体设计思路](#_Toc134216937)

[（二）迷宫元素设计思路](#_Toc134216938)

[（三）类设计](#_Toc134216939)

[四、关键技术](#_Toc134216940)

[（一）迷宫数字状态地图的创建（MazeMap类）](#_Toc134216941)

[（二）迷宫数字状态通路的创建](#_Toc134216942)

[（三）迷宫的绘制](#_Toc134216943)

[（四）玩家移动的实现与判断](#_Toc134216944)

[（五）设置参数的判断（Setting类）](#_Toc134216945)

[五、效果及存在的问题](#_Toc134216946)

[（一）运行效果](#_Toc134216947)

[（二）存在的不足及预期的解决方法](#_Toc134216948)

[1、窗口大小不够灵活](#_Toc134216949)

[2、显示答案后答案路径被吞](#_Toc134216950)

[六、附录](#_Toc134216951)

[MainWindow.xaml](#_Toc134216952)

[MainWindow类](#_Toc134216953)

[Game.xaml](#_Toc134216954)

[Game类](#_Toc134216955)

[Setting.xaml](#_Toc134216956)

[Setting类](#_Toc134216957)

[MazeMap类](#_Toc134216958)

[七、参考文献](#_Toc134216959)

# 一、游戏简介

本次设计为迷宫游戏。

游戏过程中，玩家通过键盘进行移动，在规定时间内走出迷宫即可获得游戏的胜利。

# 二、系统概述

游戏主要由三个界面构成：主界面、游戏界面以及设置界面。

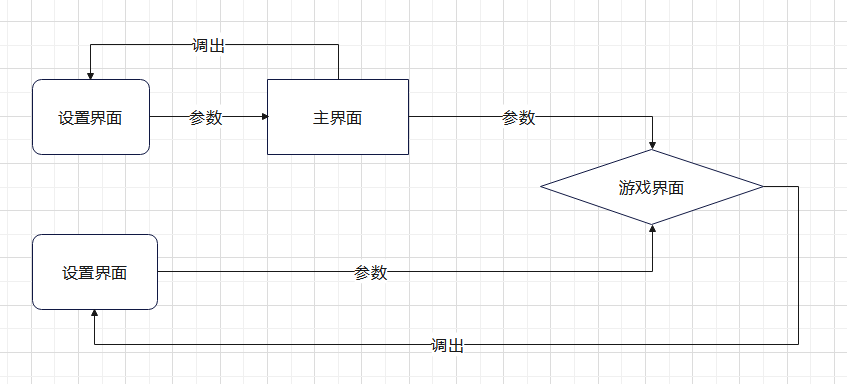
主界面主要功能为开始游戏，以及调出设置界面对游戏的相关参数（迷宫地图的大小，规定的时间长短等）进行预设置。

游戏界面主要功能为创建迷宫（即开始一轮新的游戏，包括重新开始），暂停游戏，显示迷宫答案以及调出设置界面对游戏相关参数进行修改（修改后将重新开始一轮新的游戏）。当玩家成功走出迷宫后系统会根据迷宫的大小以及所用的时间对玩家的操作进行评分。

设置界面主要功能为对迷宫大小，规定时间的长短进行设置。玩家在设置完毕后，系统会自行判断玩家所给的参数是否合法并给出相应反馈。

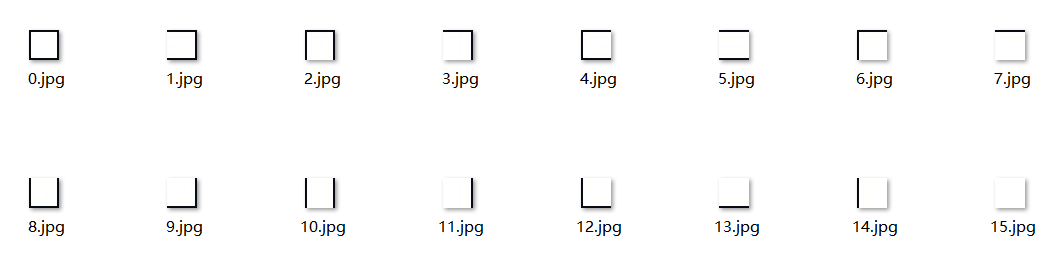
# 三、系统设计

## （一）总体设计思路

游戏一共分为三个界面，其中在主界面和游戏界面中均可以调出设置界面。设置界面在获取玩家提供的合法参数后需将参数传递回主界面或游戏界面，而若设置界面将参数传递回主界面，则主界面还需将参数传递给游戏界面，供游戏界面生成迷宫，其间具体关系如下：

## （二）迷宫元素设计思路

将迷宫视为由一个个小方格组成的。每个方格有上、下、左、右四个方向。每个方向上有两种可能：可通过/不可通过。我们规定，用1表示这个方向可以通过，0表示这个方向不可通过。这样，每个格子就有2X2X2X2=16种可能的状态。而当我们按上-右-下-左这一顺时针方向去表示每个方格四个方向上的状态时，我们可以得到一个四位二进制数（如：设一个方格的状态为：上-可通过，右-不可通过，下-可通过，左-不可通过，用01表示则为：1010）转换为十进制数后我们便可以得到0~15中的某个特定的数，因此我们就可以用 0 ~ 15 来表述每一种状态的方格，并且用该十进制数对其进行编号后，可得到如下关系：



而根据上面的规定，我们可以把上、右、下、左四个方向也表示为四位二进制数，即：

上：1000 右：0100 下：0010 左：0001

这样，对于任意一个格子，我们要判断它的某个方向是否可通过时，只需要进行一次同或操作，检查结果是否为0就可以了。

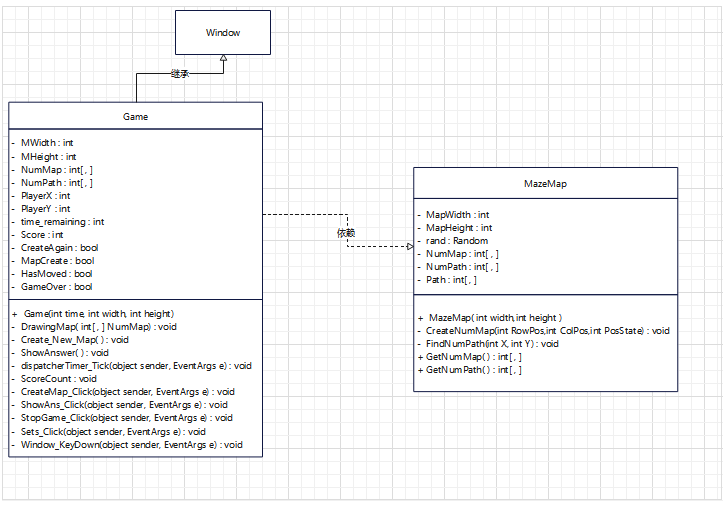
如：假设一个方格的状态为6（即：0110）

向左前进：0001 & 0110 = 0 不可通过

向右前进：0100 & 0110 = 0100 ！= 0 可以通过

## （三）类设计

该项目中最为重要的两大类便是Game类和MazeMap类。前者负责处理游戏界面的所有相关操作，包括根据参数创建迷宫（包括重新创建迷宫），显示迷宫答案，计时，判断玩家胜负，对玩家操作进行打分等；后者负责生成迷宫数字状态地图以及迷宫数字状态通路。其他的类有Setting类，负责对迷宫的相关参数进行设置调整，判断相关参数是否合法等；MainWindows类，负责显示主界面，以及调出设置界面对参数进行预设置



关键类之间的关系

# 四、关键技术

## （一）迷宫数字状态地图的创建（MazeMap类）

迷宫数字地图的创建主要运用了Random以及递归的思想。

其大致思路为：先将二维整数数组NumMap初始化（即将每个点的值设置为-1（-1表示该点尚未遍历）），该操作在本项目中由MazeMap构造函数实现），随后随便选择一个点作为初始点（本项目选择的是（0,0））创建一个存储int类型数据的List并赋值{0,1,2,3}，代表方向（0-上，1-右，2-下，3-左）。建设每个点的初始状态均为4面封闭（即0000）。利用random随机选择一个方向移动，并根据这个方向产生一个附加状态。随后抛弃该方向并判断移动后的点的位置是否合法以及是否已经被遍历过，若位置合法且没有被遍历过，则将该点的初始状态与附加状态进行异或（即将当前点这个方向打开），随后将移动后到的点以及附加状态作为参数进行迭代；若不合法则说明当前点四周已经没有可以继续前进的位置，结束返回。

迭代结束后将（0,0）位置的上面打开作为入口，（迷宫宽度-1，迷宫高度-1）位置的下面打开作为出口。（该操作在本项目中放至GetNumMap()函数实现）



数字状态迷宫生成代码

举例：假设生成的迷宫如图（1~9仅表示位置，而非状态）：



其思路为：

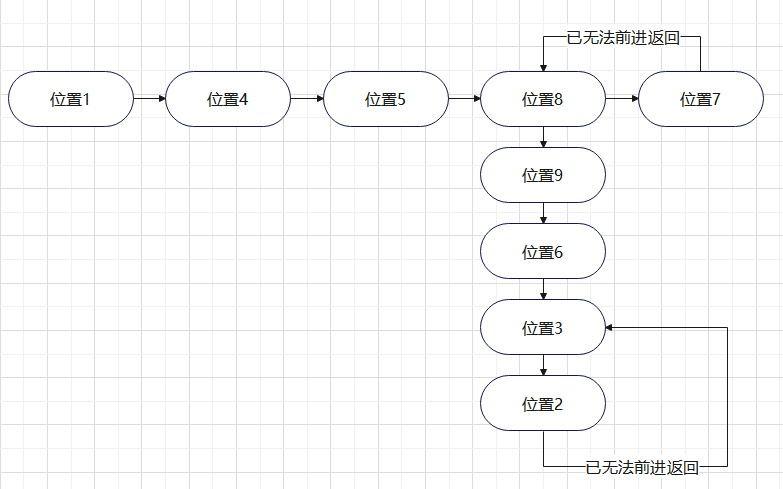
从位置1开始：

默认位置1的初始状态为（0000），random随机出的操作为向下，则附加状态为（0010），位置1向下可行，则进行异或，位置1的状态变为为（0010），随后将位置4和附加状态（0010）代入。

默认位置4的初始状态为（0000），根据附加状态（0010）将位置4的初始状态改为（1000）。random随机出的操作为向右，附加状态为（0100），位置4向右可行，则进行异或，位置4状态变为（1100），随后将位置5和附加状态（0100）代入…如此反复

最后将位置1的上面打开作为入口（即变为（1010）），位置9的下面打开作为出口（即变为（1011））。

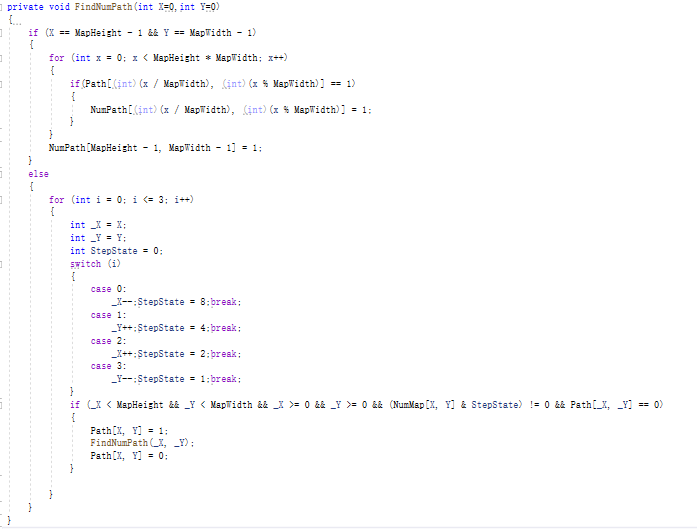
迷宫生成过程如下图：



## （二）迷宫数字状态通路的创建

寻找迷宫通路主要用到的方法为迭代。

其大致思路为：先将二维整数数组NumPath、Path初始化（即将每个点的值设置为0（0表示该点不是通路上的点）），该操作在本项目中由MazeMap构造函数实现）从起点开始，向四个方向前进，选取方向后通过对移动后位置是否合法、同或运算等方式判断该方向是否可以通行。若可以通行，则先将Path中该点标记为1，随后继续向其他行走，直至到达终点或进入死路。若进入死路，则不做任何操作若到达终点，则将Path中所有为1的点对应至NumPath中，再将NumPath的终点赋值为1，结束寻找。



寻找迷宫数字解代码

## （三）迷宫的绘制

迷宫的绘制主要利用了Image控件和Canvas控件

通过Canvas控件建立的虚拟坐标轴，通过NumMap将经过特定编号的图片绑定至Image控件实例image中，通过控制image的margin值将image添加至Canvas中

## （四）玩家移动的实现与判断

玩家移动的实现：利用了Canvas元素可以重叠的特性。重写键盘事件，通过PlayerX，PlayerY对玩家进行定位，将玩家所在点位替换为玩家的图片，将玩家离开的点位替换为迷宫原来的图片。

玩家移动的判断：重写键盘事件，当玩家进行移动时，通过同或运算等条件进行判断是否可以移动，玩家移动后判断是否已经到达终点。

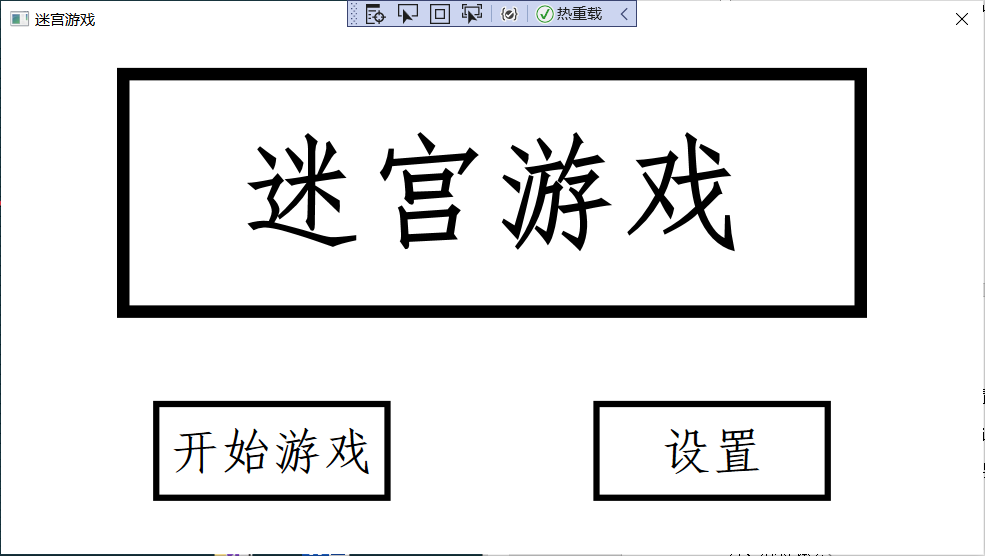
## （五）设置参数的判断（Setting类）

为防止系统在修改时便报错，将参数的检测放至Yes\_Btn\_Click()中而非用TextChanged()实时跟踪。同时因为改动涉及三个部分，所以设置了三个bool变量进行检测，只有当三个bool变量均为TRUE时，参数才会被记录。针对每个bool变量的判断除了基础的判断值是否在范围内，测试时还发现当输入不为数字时程序同样会报错，所以又增加了try-catch语句进行完善。

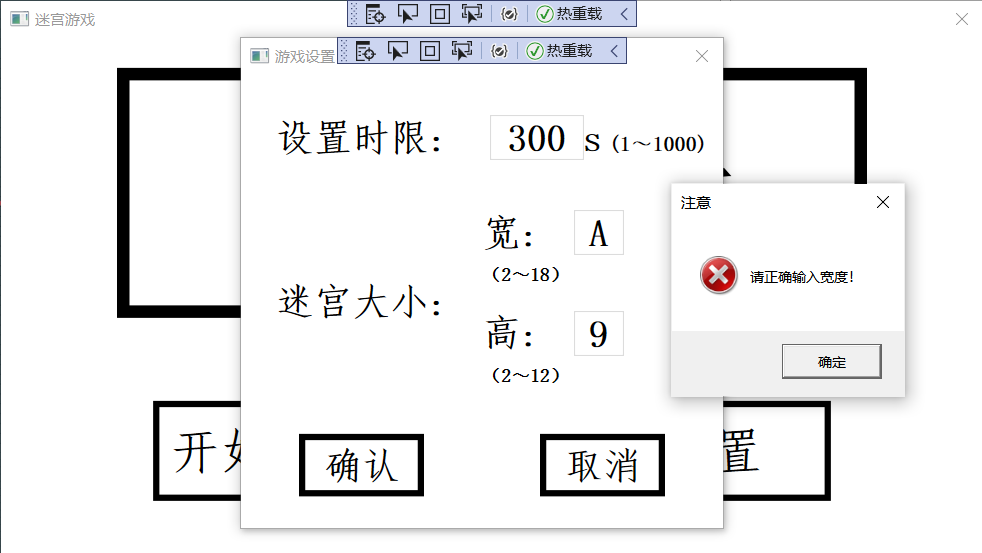
# 五、效果及存在的问题

## （一）运行效果

运行效果较好，当从主界面调出设置界面进行参数的预设置时，若参数错误会正确指出，若参数无误，则传回参数，改变系统的部分值，该值的改变能传递至游戏界面

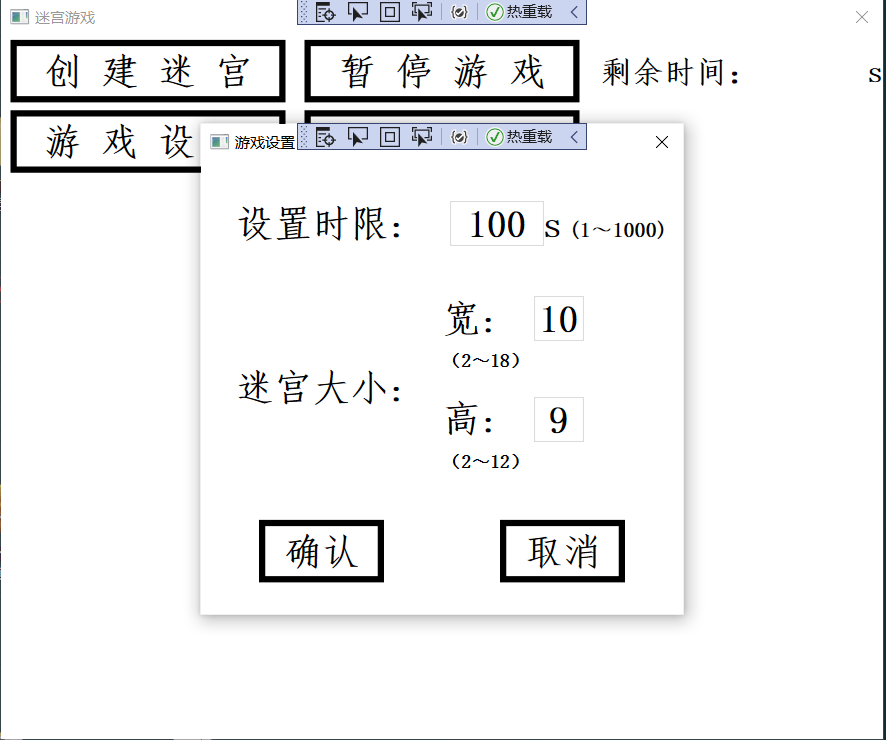


主界面

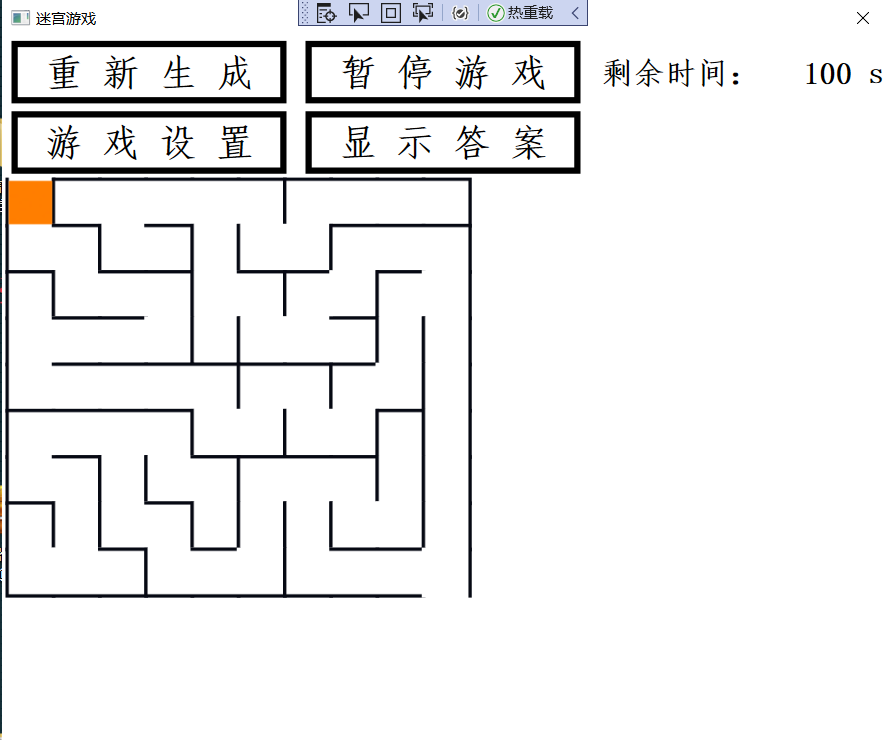




当相关值设置不对时，系统会自动报出错误，提醒玩家更改



在主界面设置完毕的值，会传递值游戏界面，此时调出设置界面也可以看到已经进行预设的值



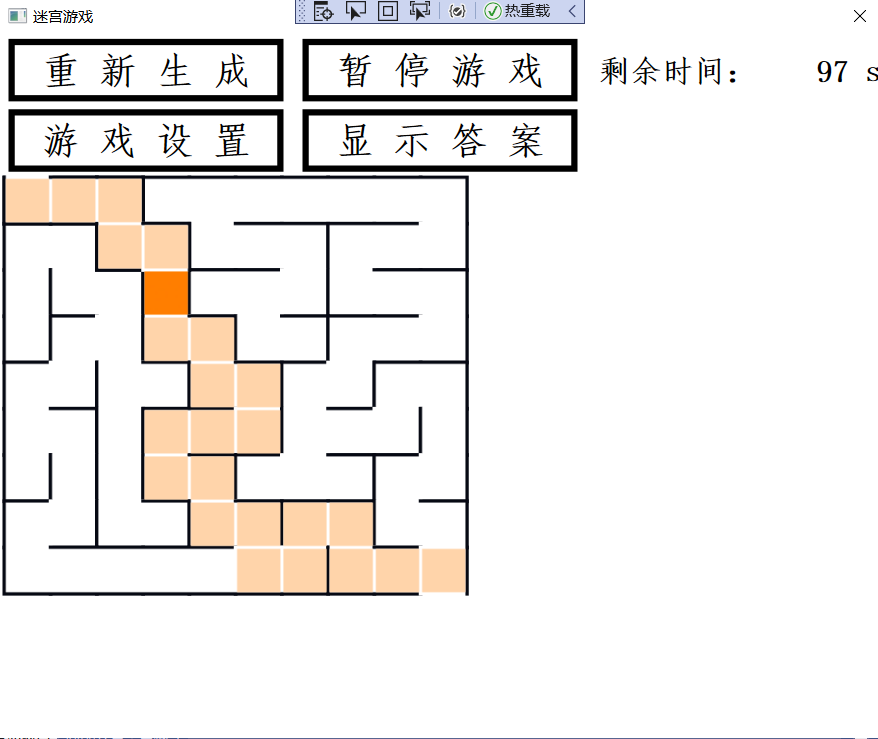
游戏界面



当玩家未移动时，时间不会开始倒计时，玩家移动后开始计时



玩家到达终点时，系统根据相关数据计算得分，并给出下一步操作的提示



当玩家点击显示答案时，系统将给出正确路径，同时视为游戏结束，倒计时暂停。此时玩家走到终点将无得分计算。

## （二）存在的不足及预期的解决方法

### 1、窗口大小不够灵活

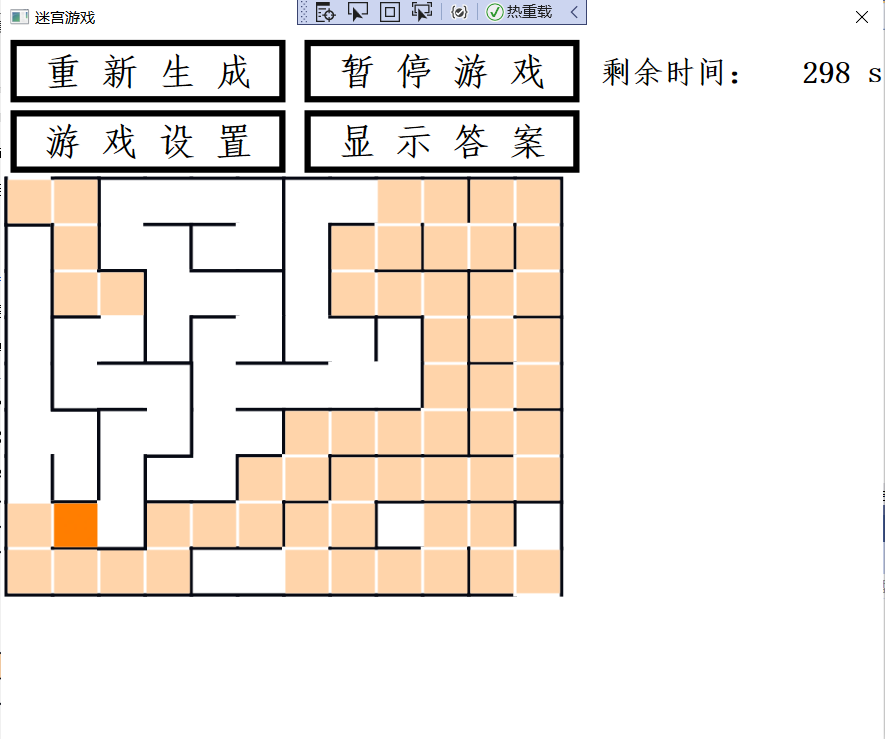
原先计划是只完成一个固定大小的迷宫设计，在编程过程中发现只要对代码进行一定的修改完善，创建一个大小可变的迷宫也是可以做到的。但这样修改后发现窗口的大小会导致迷宫显示不全，一时间又想不到解决的办法，便只好依据当前窗口的大小对迷宫的大小进行限制。

预期解法：尝试使用ViewBox以及滚条进行解决。或者视图会跟随玩家移动而移动

### 2、显示答案后答案路径被吞

玩家点击显示答案依旧可以移动，而玩家移动时会导致已经显示的答案被覆盖掉,如：





预期解决：充分利用GameOver变量，当GameOver变量为TRUE时，玩家移动不再覆盖迷宫原图

# 六、附录

## MainWindow.xaml

<Window x:Class="Maze.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Maze"

mc:Ignorable="d"

Title="迷宫游戏" Height="450" Width="800" ResizeMode="NoResize" WindowStartupLocation="CenterScreen">

<Grid x:Name="Main">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="0.3\*"/>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="0.3\*"/>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="0.3\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="3\*"/>

<RowDefinition Height="2\*"/>

</Grid.RowDefinitions>

<Grid.Resources>

<Style x:Key="Lable\_Style" TargetType="Label">

<Setter Property="Foreground" Value="Black"/>

<Setter Property="FontSize" Value="100"/>

<Setter Property="FontFamily" Value="仿宋"/>

<Setter Property="FontWeight" Value="Bold"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="Label">

<Grid>

<Rectangle x:Name="OuterR" Fill="Black" Height="200" Width="600"/>

<Rectangle x:Name="InnerR" Fill="White" Height="180" Width="580"/>

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Grid>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<Style x:Key ="Btn\_Style" TargetType ="{x:Type Button}">

<Setter Property ="Foreground" Value ="Black"/>

<Setter Property ="FontSize" Value ="40"/>

<Setter Property="FontFamily" Value="仿宋"/>

<Setter Property ="FontWeight" Value ="Bold"/>

<Setter Property ="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Grid>

<Rectangle Name="OuterR" Width="190" Height="80" Fill="Black"/>

<Rectangle Name="InnerR" Width="180" Height="70" Fill="White"/>

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Grid>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="InnerR" Property="Fill" Value="Black"/>

<Setter Property="Foreground" Value="White"/>

</Trigger>

<Trigger Property="IsMouseOver" Value="False">

<Setter TargetName="InnerR" Property="Fill" Value="White"/>

<Setter Property="Foreground" Value="Black"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Grid.Resources>

<Label HorizontalContentAlignment="Center" VerticalContentAlignment="Center" Grid.Row="0" Grid.ColumnSpan="5" Style="{StaticResource Lable\_Style}">迷宫游戏</Label>

<Button Name="Start" Content="开始游戏" Grid.Column="1" Grid.Row="1" Style="{StaticResource Btn\_Style}" Click="Start\_Click" />

<Button Name="Sets" Content="设置" Grid.Column="3" Grid.Row="1" Style="{StaticResource Btn\_Style}" Click="Sets\_Click" />

</Grid>

</Window>

## MainWindow类

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace Maze

{

/// <summary>

/// MainWindow.xaml 的交互逻辑

/// </summary>

public partial class MainWindow : Window

{

int time\_remaining=300;

int MHeight=9;

int MWidth=12;

public MainWindow()

{

InitializeComponent();

}

private void Start\_Click(object sender, RoutedEventArgs e)

{

Game NewGame = new Game(time\_remaining,MWidth,MHeight);

NewGame.Show();

this.Close();

}

private void Sets\_Click(object sender, RoutedEventArgs e)

{

Setting set = new Setting(time\_remaining,MWidth,MHeight);

set.ShowDialog();

bool result = (bool)set.DialogResult;

if (result)

{

time\_remaining = set.time\_remaining;

MHeight = set.MHeight;

MWidth = set.MWidth;

}

}

}

}

## Game.xaml

<Window x:Class="Maze.Game"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Maze"

mc:Ignorable="d"

Title="迷宫游戏" Height="600" Width="720" ResizeMode="NoResize" WindowStartupLocation="CenterScreen" KeyDown="Window\_KeyDown" >

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="2\*"/>

<RowDefinition Height="8\*"/>

</Grid.RowDefinitions>

<Grid x:Name="Btn\_Grid">

<Grid.Resources>

<Style x:Key ="Btn\_Style" TargetType ="{x:Type Button}">

<Setter Property ="Foreground" Value ="Black"/>

<Setter Property ="FontSize" Value ="30"/>

<Setter Property="FontFamily" Value="仿宋"/>

<Setter Property ="FontWeight" Value ="Bold"/>

<Setter Property="Focusable" Value="False"/>

<Setter Property ="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Grid>

<Rectangle Name="OuterR" Width="220" Height="50" Fill="Black"/>

<Rectangle Name="InnerR" Width="210" Height="40" Fill="White"/>

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Grid>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="InnerR" Property="Fill" Value="Black"/>

<Setter Property="Foreground" Value="White"/>

</Trigger>

<Trigger Property="IsMouseOver" Value="False">

<Setter TargetName="InnerR" Property="Fill" Value="White"/>

<Setter Property="Foreground" Value="Black"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Grid.Resources>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="1\*"/>

<RowDefinition Height="1\*"/>

</Grid.RowDefinitions>

<Button x:Name="CreateMap" Content="创 建 迷 宫" Style="{StaticResource Btn\_Style}" Grid.Column="0" Grid.Row="0" Click="CreateMap\_Click"/>

<Button x:Name="StopGame" Content="暂 停 游 戏" Style="{StaticResource Btn\_Style}" Grid.Column="1" Grid.Row="0" Click="StopGame\_Click"/>

<Button x:Name="Sets" Content="游 戏 设 置" Style="{StaticResource Btn\_Style}" Grid.Column="0" Grid.Row="1" Click="Sets\_Click" />

<Button x:Name="ShowAns" Content="显 示 答 案" Style="{StaticResource Btn\_Style}" Grid.Column="1" Grid.Row="1" Click="ShowAns\_Click"/>

<TextBlock Text="剩余时间：" Foreground="Black" FontSize="25" FontFamily="仿宋" FontWeight="Bold" VerticalAlignment="Center" HorizontalAlignment="Left" TextWrapping="Wrap" Grid.Column="2" Grid.Row="0" Margin="9,15,0,14.6"/>

<TextBlock x:Name="TimeBlock" Foreground="Black" FontSize="25" FontFamily="仿宋" FontWeight="Bold" VerticalAlignment="Center" HorizontalAlignment="Right" TextWrapping="Wrap" Grid.Column="2" Grid.Row="0" Width="110" TextAlignment="Right" Margin="0,12,25.6,11.6" />

<TextBlock Text="s" Foreground="Black" FontSize="25" FontFamily="仿宋" FontWeight="Bold" VerticalAlignment="Center" HorizontalAlignment="Right" TextWrapping="Wrap" Grid.Column="2" Grid.Row="0" />

</Grid>

<Canvas x:Name="Map" Grid.Row="2" Width="700" Height="450 " HorizontalAlignment="Center" VerticalAlignment="Center" >

</Canvas>

</Grid>

</Window>

## Game类

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Threading;

namespace Maze

{

/// <summary>

/// Game.xaml 的交互逻辑

/// </summary>

public partial class Game : Window

{

int MWidth = 12; //迷宫地图宽度 默认为12

int MHeight = 9; //迷宫地图高度 默认为9

int[,] NumMap; //数字地图

int[,] NumPath; //数字解

int PlayerX = 0; //记录玩家位置

int PlayerY = 0; //记录玩家位置

int time\_remaining=300; //时限 默认为300

int Score=0; //得分

bool CreateAgain = false; //判断是否为第二次创建地图

bool MapCreate = false; //判读地图是否已经生成

bool HasMoved = false; //判断玩家是否已经移动

bool GameOver = false; //判断一局游戏是否已经结束

private DispatcherTimer dispatcherTimer;

//窗口初始化，接收主窗口的三个参数

public Game(int time, int width, int height)

{

InitializeComponent();

dispatcherTimer = new DispatcherTimer();

dispatcherTimer.Tick += new EventHandler(dispatcherTimer\_Tick);

dispatcherTimer.Interval = new TimeSpan(0, 0, 1);

time\_remaining = time;

MWidth = width;

MHeight = height;

}

//方法--通过数字地图在Canvas面板绘制地图（包括绘制出玩家的起点位置）

private void DrawingMap(int[,] NumMap)

{

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

for (int j = 0; j < MWidth; j++)

{

Image image = new Image();

image.Margin = new Thickness(37 \* j,37 \* i, 0, 0);

string str = @"./ImageResource/Images/";

str += NumMap[i,j].ToString();str += ".jpg";

image.Source = new BitmapImage(new Uri(str, UriKind.Relative));

Map.Children.Add(image);

}

Image image1 = new Image();

image1.Margin = new Thickness(0, 0, 0, 0);

string str1 = @"./ImageResource/PlayerImages/p.png";

image1.Source = new BitmapImage(new Uri(str1, UriKind.Relative));

Map.Children.Add(image1);

}

//方法--创建一张新的地图（并标出时限）

//等价于 开启一轮新的游戏

private void Create\_New\_Map()

{

Map.Children.Clear();

PlayerX = 0;

PlayerY = 0;

HasMoved = false;

GameOver = false;

MazeMap Map1 = new MazeMap(MWidth, MHeight);

NumMap = Map1.GetNumMap();

NumPath = Map1.GetNumPath();

DrawingMap(NumMap);

MapCreate = true;

TimeBlock.Text = time\_remaining.ToString();

}

//方法--根据数字解绘制出迷宫解

private void ShowAnswer()

{

if (MapCreate)

{

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

for (int j = 0; j < MWidth; j++)

{

if (NumPath[i, j] == 1 && (i != PlayerX || j != PlayerY))

{

Image image = new Image();

image.Margin = new Thickness(37 \* j, 37 \* i, 0, 0);

string str = @"./ImageResource/PlayerImages/p1.png";

image.Source = new BitmapImage(new Uri(str, UriKind.Relative));

Map.Children.Add(image);

}

}

}

}

//方法--计时并判断玩家是否失败

private void dispatcherTimer\_Tick(object sender, EventArgs e)

{

if (int.Parse(TimeBlock.Text) - 1 >= 0)

{

TimeBlock.Text = (int.Parse(TimeBlock.Text) - 1).ToString();

}

else

{

GameOver = true;

dispatcherTimer.Stop();

string str = "时间到，游戏结束！\n是否开始新一轮游戏？";

MessageBoxResult messageBoxResult;

messageBoxResult = MessageBox.Show(str, "失败", MessageBoxButton.YesNo, MessageBoxImage.Exclamation);

switch (messageBoxResult)

{

case MessageBoxResult.Yes:

Create\_New\_Map(); break;

case MessageBoxResult.No:

{

MessageBoxResult messageBoxResult1;

messageBoxResult1 = MessageBox.Show("是否显示答案", "答案", MessageBoxButton.YesNo, MessageBoxImage.Exclamation);

switch (messageBoxResult1)

{

case MessageBoxResult.Yes: ShowAnswer(); break;

}

break;

}

}

}

}

//方法--计算分数

private void ScoreCount()

{

if (time\_remaining - int.Parse(TimeBlock.Text) != 0)

{

Score = (int)((MHeight \* MWidth) \* 100 + 1000 / (time\_remaining - int.Parse(TimeBlock.Text)));

}

else

{

Score = (int)((MHeight \* MWidth) \* 100 + 1000);

}

}

//按键--创建迷宫

private void CreateMap\_Click(object sender, RoutedEventArgs e)

{

if (!CreateAgain) { CreateMap.Content = "重 新 生 成";CreateAgain = true; }

Create\_New\_Map();

dispatcherTimer.Stop();

}

//按键-改写键盘事件，用于玩家走迷宫以及判断玩家是否成功

private void Window\_KeyDown(object sender, KeyEventArgs e)

{

if (MapCreate)

{

if (!HasMoved) { HasMoved = true; dispatcherTimer.Start(); ; }

int \_PlayerX = PlayerX;

int \_PlayerY = PlayerY;

int Direction = 0;

Key key = e.Key;

switch (key)

{

case Key.Up:

\_PlayerX--;

Direction = 8;

break;

case Key.Down:

\_PlayerX++;

Direction = 2;

break;

case Key.Left:

\_PlayerY--;

Direction = 1;

break;

case Key.Right:

\_PlayerY++;

Direction = 4;

break;

}

if (\_PlayerX < MHeight && \_PlayerY < MWidth && \_PlayerX >= 0 && \_PlayerY >= 0 && (NumMap[PlayerX, PlayerY] & Direction) != 0)

{

Image image1 = new Image();

image1.Margin = new Thickness(37 \* PlayerY, 37 \* PlayerX, 0, 0);

string str1= @".\ImageResource\Images\";

str1 += NumMap[PlayerX, PlayerY].ToString(); str1 += ".jpg";

image1.Source = new BitmapImage(new Uri(str1,UriKind.Relative));

Map.Children.Add(image1);

Image image2 = new Image();

image2.Margin = new Thickness(37 \* \_PlayerY, 37 \* \_PlayerX, 0, 0);

string str2 = @".\ImageResource\PlayerImages\p.png";

image2.Source = new BitmapImage(new Uri(str2, UriKind.Relative));

Map.Children.Add(image2);

PlayerX = \_PlayerX;

PlayerY = \_PlayerY;

}

if (PlayerX == MHeight - 1 && PlayerY == MWidth - 1)

{

dispatcherTimer.Stop();

if (!GameOver)

{

GameOver = true;

string str = "恭喜您成功过关！您的得分为： ";

ScoreCount();

str += Score.ToString();

str += "\n是否开始新一轮游戏？";

MessageBoxResult messageBoxResult;

messageBoxResult = MessageBox.Show(str, "成功", MessageBoxButton.YesNo, MessageBoxImage.Exclamation);

switch (messageBoxResult)

{

case MessageBoxResult.Yes:

Create\_New\_Map(); break;

}

}

}

}

}

//按键--显示答案

private void ShowAns\_Click(object sender, RoutedEventArgs e)

{

GameOver = true;

dispatcherTimer.Stop();

ShowAnswer();

}

//按键--暂停游戏

private void StopGame\_Click(object sender, RoutedEventArgs e)

{

if(MapCreate)

{

dispatcherTimer.Stop();

MessageBox.Show("游戏暂停中", "暂停", MessageBoxButton.OK, MessageBoxImage.Warning);

dispatcherTimer.Start();

}

}

//按键--游戏设置

private void Sets\_Click(object sender, RoutedEventArgs e)

{

if(MapCreate)

{

MessageBoxResult messageBoxResult;

dispatcherTimer.Stop();

messageBoxResult=MessageBox.Show("进行设置后迷宫将重新生成，是否继续？", "暂停", MessageBoxButton.YesNo, MessageBoxImage.Warning);

switch (messageBoxResult)

{

case MessageBoxResult.Yes:

{

Setting set = new Setting(time\_remaining, MWidth, MHeight);

set.ShowDialog();

bool result = (bool)set.DialogResult;

if (result)

{

time\_remaining = set.time\_remaining;

MHeight = set.MHeight;

MWidth = set.MWidth;

}

Create\_New\_Map();

break;

}

case MessageBoxResult.No:

dispatcherTimer.Start();

break;

}

}

else

{

Setting set = new Setting(time\_remaining, MWidth, MHeight);

set.ShowDialog();

bool result = (bool)set.DialogResult;

if (result)

{

time\_remaining = set.time\_remaining;

MHeight = set.MHeight;

MWidth = set.MWidth;

}

}

}

}

}

## Setting.xaml

<Window x:Class="Maze.Setting"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Maze"

mc:Ignorable="d"

Title="游戏设置" Height="400" Width="400" ResizeMode="NoResize" WindowStartupLocation="CenterScreen" >

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="1\*"/>

<RowDefinition Height="0.5\*"/>

<RowDefinition Height="0.3\*"/>

<RowDefinition Height="0.5\*"/>

<RowDefinition Height="0.3\*"/>

<RowDefinition Height="1\*"/>

</Grid.RowDefinitions>

<Grid.Resources>

<Style x:Key ="Tebl\_Style" TargetType ="{x:Type TextBlock}">

<Setter Property ="Foreground" Value ="Black"/>

<Setter Property ="FontSize" Value ="30"/>

<Setter Property="FontFamily" Value="仿宋"/>

<Setter Property ="FontWeight" Value ="Bold"/>

</Style>

<Style x:Key ="TeBox\_Style" TargetType ="{x:Type TextBox}">

<Setter Property ="Foreground" Value ="Black"/>

<Setter Property ="FontSize" Value ="30"/>

<Setter Property="FontFamily" Value="仿宋"/>

<Setter Property ="FontWeight" Value ="Bold"/>

</Style>

<Style x:Key ="Btn\_Style" TargetType ="{x:Type Button}">

<Setter Property ="Foreground" Value ="Black"/>

<Setter Property ="FontSize" Value ="30"/>

<Setter Property="FontFamily" Value="仿宋"/>

<Setter Property ="FontWeight" Value ="Bold"/>

<Setter Property ="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Grid>

<Rectangle Name="OuterR" Width="100" Height="50" Fill="Black"/>

<Rectangle Name="InnerR" Width="90" Height="40" Fill="White"/>

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Grid>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="InnerR" Property="Fill" Value="Black"/>

<Setter Property="Foreground" Value="White"/>

</Trigger>

<Trigger Property="IsMouseOver" Value="False">

<Setter TargetName="InnerR" Property="Fill" Value="White"/>

<Setter Property="Foreground" Value="Black"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Grid.Resources>

<TextBlock HorizontalAlignment="Center" TextWrapping="Wrap" Text="设置时限:" Style="{StaticResource Tebl\_Style}" Grid.Column="0" Grid.Row="0" VerticalAlignment="Center" />

<TextBlock HorizontalAlignment="Center" TextWrapping="Wrap" Text="迷宫大小:" Style="{StaticResource Tebl\_Style}" Grid.Column="0" Grid.Row="1" Grid.RowSpan="4" VerticalAlignment="Center" />

<TextBlock HorizontalAlignment="Left" TextWrapping="Wrap" Style="{StaticResource Tebl\_Style}" Grid.Column="1" Grid.Row="1" VerticalAlignment="Center" ><Run Text="宽"/><Run Text=":"/></TextBlock>

<TextBlock HorizontalAlignment="Left" TextWrapping="Wrap" Style="{StaticResource Tebl\_Style}" Grid.Column="1" Grid.Row="3" VerticalAlignment="Center" ><Run Text="高"/><Run Text=":"/></TextBlock>

<TextBlock HorizontalAlignment="Right" TextWrapping="Wrap" Text="s" Style="{StaticResource Tebl\_Style}" Grid.Column="1" Grid.Row="0" VerticalAlignment="Center" Margin="0,34,31.6,34.2" Width="81" />

<TextBlock HorizontalAlignment="Right" TextWrapping="Wrap" Text="(1～1000)" FontSize="17" Style="{StaticResource Tebl\_Style}" Grid.Column="1" Grid.Row="0" VerticalAlignment="Center" Margin="0,47,10.6,38.2" Width="81" />

<TextBlock HorizontalAlignment="Left" FontSize="15" TextWrapping="Wrap" Style="{StaticResource Tebl\_Style}" Grid.Column="1" Grid.Row="2" VerticalAlignment="Top"><Run Text="（"/><Run Text="2"/><Run Text="～18）"/></TextBlock>

<TextBlock HorizontalAlignment="Left" FontSize="15" TextWrapping="Wrap" Style="{StaticResource Tebl\_Style}" Grid.Column="1" Grid.Row="4" VerticalAlignment="Top"><Run Text="（"/><Run Text="2"/><Run Text="～12）"/></TextBlock>

<TextBox x:Name="TimeBox" Style="{StaticResource TeBox\_Style}" HorizontalAlignment="Left" TextWrapping="Wrap" Text="300" VerticalAlignment="Center" TextAlignment="Center" Width="75" Grid.Column="1" Grid.Row="0" BorderBrush="Gainsboro" Background="White" Margin="6.2,27,0,27.4" />

<TextBox x:Name="MWidthBox" Style="{StaticResource TeBox\_Style}" HorizontalAlignment="Right" TextWrapping="Wrap" Text="12" VerticalAlignment="Center" Width="40" TextAlignment="Center" Grid.Column="1" Grid.Row="1" BorderBrush="Gainsboro" Background="White" Margin="0,4.6,79.6,4.2" />

<TextBox x:Name="MHeightBox" Style="{StaticResource TeBox\_Style}" HorizontalAlignment="Right" TextWrapping="Wrap" Text="9" VerticalAlignment="Center" Width="40" TextAlignment="Center" Grid.Column="1" Grid.Row="3" BorderBrush="Gainsboro" Background="White" Margin="0,4.8,79.6,4" />

<Button x:Name="Yse\_Btn" Content="确认" HorizontalAlignment="Center" VerticalAlignment="Center" Style="{StaticResource Btn\_Style}" Grid.Column="0" Grid.Row="5" Click="Yse\_Btn\_Click"/>

<Button x:Name="No\_Btn" Content="取消" HorizontalAlignment="Center" VerticalAlignment="Center" Style="{StaticResource Btn\_Style}" Grid.Column="1" Grid.Row="5" Click="No\_Btn\_Click"/>

</Grid>

</Window>

## Setting类

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Maze

{

/// <summary>

/// Setting.xaml 的交互逻辑

/// </summary>

public partial class Setting : Window

{

public int time\_remaining=300;

public int MWidth =12;

public int MHeight =9;

public Setting(int time, int width, int height)

{

InitializeComponent();

time\_remaining = time;

MWidth = width;

MHeight = height;

TimeBox.Text = time\_remaining.ToString();

MWidthBox.Text = MWidth.ToString();

MHeightBox.Text = MHeight.ToString();

}

private void Yse\_Btn\_Click(object sender, RoutedEventArgs e)

{

bool InputRight0 = false;

bool InputRight1 = false;

bool InputRight2 = false;

try

{

if (1 <= int.Parse(TimeBox.Text) && int.Parse(TimeBox.Text) <= 1000)

{

time\_remaining = int.Parse(TimeBox.Text); InputRight0 = true;

}

else

{

MessageBox.Show("请正确输入时限！", "注意", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

catch (FormatException)

{

MessageBox.Show("请正确输入时限！", "注意", MessageBoxButton.OK, MessageBoxImage.Error);

}

try

{

if (2 <= int.Parse(MWidthBox.Text) && int.Parse(MWidthBox.Text) <= 18)

{

MWidth = int.Parse(MWidthBox.Text);InputRight1 = true;

}

else

{

MessageBox.Show("请正确输入宽度！", "注意", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

catch (FormatException)

{

MessageBox.Show("请正确输入宽度！", "注意", MessageBoxButton.OK, MessageBoxImage.Error);

}

try

{

if (2 <= int.Parse(MHeightBox.Text) && int.Parse(MHeightBox.Text) <= 12)

{

MHeight = int.Parse(MHeightBox.Text);InputRight2 = true;

}

else

{

MessageBox.Show("请正确输入高度！", "注意", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

catch (FormatException)

{

MessageBox.Show("请正确输入高度！", "注意", MessageBoxButton.OK, MessageBoxImage.Error);

}

if (InputRight1 && InputRight2 && InputRight0)

{

this.DialogResult = true;

}

}

private void No\_Btn\_Click(object sender, RoutedEventArgs e)

{

this.DialogResult = false;

}

}

}

## MazeMap类

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Maze

{

//迷宫地图类 用于生成数字地图

class MazeMap

{

int MapWidth=12; //迷宫地图宽度

int MapHeight=9; //迷宫地图长度

Random rand = new Random(); //随机数

int[,] NumMap; //数字迷宫

int[,] NumPath; //迷宫的数字解

int[,] Path;

//初始化地图，将所有位置赋值为-1（-1表示未遍历过）

//初始化数字解，将所有位置赋值为0（0表示该位点不是正确的路径点）

public MazeMap(int width,int height)

{

MapWidth = width;

MapHeight = height;

NumMap = new int[MapHeight, MapWidth];

for (int x = 0; x < MapHeight \* MapWidth; x++)

{

NumMap[(int)(x / MapWidth),(int)( x % MapWidth)] = -1;

}

NumPath = new int[MapHeight, MapWidth];

for (int x = 0; x < MapHeight \* MapWidth; x++)

{

NumPath[(int)(x / MapWidth), (int)(x % MapWidth)] = 0;

}

Path = new int[MapHeight, MapWidth];

for (int x = 0; x < MapHeight \* MapWidth; x++)

{

Path[(int)(x / MapWidth), (int)(x % MapWidth)] = 0;

}

}

//获取有起点和终点的数字地图

public int[,] GetNumMap()

{

CreateNumMap();

//创建起点和终点

NumMap[0, 0] = NumMap[0, 0] ^ 8;//打开上面，作为入口

NumMap[MapHeight - 1, MapWidth - 1] = NumMap[MapHeight - 1, MapWidth - 1] ^ 2;//打开下面，作为出口

return NumMap;

}

//获取数字解

public int[,] GetNumPath()

{

FindNumPath();

return NumPath;

}

//创建数字地图

private void CreateNumMap(int RowPos=0,int ColPos=0,int PosState=0)

{

List<int> Directions = new List<int> { 0, 1, 2, 3 }; //储存方向，0-上；1-右；2-下；3-左

int State = 0; //假设每个点位初始状态均为4面封闭

switch(PosState) //根据上一个点位的状态更新当前点位的状态

{

case 1: State = 4;break;

case 2: State = 8;break;

case 4: State = 1;break;

case 8: State = 2;break;

}

NumMap[RowPos, ColPos] = State;

while(Directions.Count>0) //随机选择一个方向前进，并产生一个附加状态

{

int X = RowPos;

int Y = ColPos;

int DirectNum = rand.Next(Directions.Count);

int AddState = 0;

switch(Directions[DirectNum])

{

case 0:

X--;AddState = 8;break;

case 1:

Y++;AddState = 4;break;

case 2:

X++;AddState = 2;break;

case 3:

Y--;AddState = 1;break;

}

Directions.RemoveAt(DirectNum); //抛弃已用方向

if(X<MapHeight && Y<MapWidth && X>=0 && Y>=0 && NumMap[X,Y] == -1) //判断遍历是否合法，合法则更新当前点位状态

{

State = State ^ AddState;

NumMap[RowPos, ColPos] = State;

CreateNumMap(X, Y, AddState);

}

}

}

//寻找数字解

private void FindNumPath(int X=0,int Y=0)

{

if (X == MapHeight - 1 && Y == MapWidth - 1)

{

for (int x = 0; x < MapHeight \* MapWidth; x++)

{

if(Path[(int)(x / MapWidth), (int)(x % MapWidth)] == 1)

{

NumPath[(int)(x / MapWidth), (int)(x % MapWidth)] = 1;

}

}

NumPath[MapHeight - 1, MapWidth - 1] = 1;

}

else

{

for (int i = 0; i <= 3; i++)

{

int \_X = X;

int \_Y = Y;

int StepState = 0;

switch (i)

{

case 0:

\_X--;StepState = 8;break;

case 1:

\_Y++;StepState = 4;break;

case 2:

\_X++;StepState = 2;break;

case 3:

\_Y--;StepState = 1;break;

}

if (\_X < MapHeight && \_Y < MapWidth && \_X >= 0 && \_Y >= 0 && (NumMap[X, Y] & StepState) != 0 && Path[\_X, \_Y] == 0)

{

Path[X, Y] = 1;

FindNumPath(\_X, \_Y);

Path[X, Y] = 0;

}

}

}

}

}

}

# 七、参考文献

1. JavaScript写的网页迷宫游戏

<https://oldj.net/article/2010/06/03/javascript-maze/> ，2010-06-03

2. C# 生成迷宫及寻路

<https://www.cnblogs.com/fmnisme/archive/2011/06/22/2087072.html>，2011-06-22