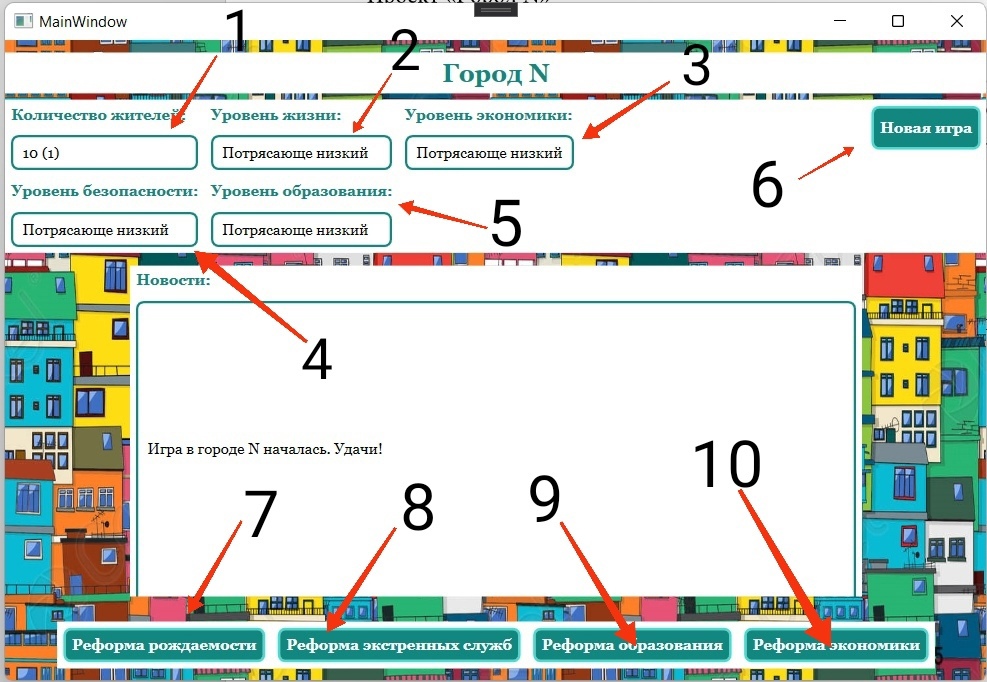
Проект «Город N»

***Введение***

Город N постигло стихийное бедствие. Вас направили из столицы восстанавливать город. Приведите его к процветанию!

***Правила игры:***



1. Для начала игры нажмите кнопку, обозначенную на рисунке номером 6 («Новая игра»)
2. На первом шаге игры от предшественника вам досталось 10 жителей. Изменение количества которых на следующем ходу указаны в скобках. (Смотри на рисунке номер 1)
3. Пункты 2, 3, 4, 5 – обозначают уровни развития сфер жизни города. В игре существует такая градация уровней:

* Потрясающе низкий
* Низкий
* Средний
* Высокий
* Превосходный

1. Вы можете проводить реформы.

* Реформа рождаемости (рис №7).
* Реформа экстренных служб (рис №8)
* Реформа образования (рис №9)
* Реформа экономики (рис №10)

Их влияние на сферы жизни показано в таблице (“+”-повышает, “-”-понижает, “0”- не влияет).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Характеристики | | | | |
| Реформа | Количество жителей | Безопасность | Уровень жизни | Образование | Экономика |
| Рождаемости | + | - | - | - | - |
| Экстренных служб | 0 | + | - | - | - |
| Образование | - | + | - | + | - |
| Экономики | + | 0 | + | - | + |

1. Помимо реформ в городе происходят случайные события. На вероятность их возникновения влияет только уровень развития той или иной сферы жизни. Они происходят на следующий день после реформы.

***Условия окончания игры:***

1. Количество жителей упало ниже 0
2. Уровень одной из сфер жизни снизился до минимума.
3. Все сферы жизни достигли максимального уровня.

Выполнение первого или второго условия обозначает поражение в игре. Только выполнение третьего пункта способно привести к победе.

***Код:***

/// <summary>

/// Класс описывающий сферу жизни

/// </summary>

public class Sphere

{

/// <summary>

/// Сообщение при завершении игры

/// </summary>

private readonly string message;

/// <summary>

/// Счётчик условных единиц

/// </summary>

private int \_Count = 10;

public int Count

{

get => \_Count;

set

{

\_Count = value;

UpdateLevel();

}

}

/// <summary>

/// Достигнут ли максимальный уровень

/// </summary>

public bool IsLevelMax => Level == Level.Unnattainable;

/// <summary>

/// Условный уровень

/// </summary>

public Level Level { get; set; }

/// <summary>

/// Повышение уровня характеристики

/// </summary>

private void UpdateLevel()

{

switch (Count)

{

case <= 5:

{

LevelMax?.Invoke(message);

break;

}

case <= 30:

{

Level = Level.None;

break;

}

case <= 50:

{

Level = Level.Low;

break;

}

case <= 100:

{

Level = Level.Medium;

break;

}

case <= 200:

{

Level = Level.High;

break;

}

case <= 255:

{

Level = Level.Unnattainable;

break;

}

case > 255:

{

LevelMax?.Invoke("Поздравляем! Вы привели свой город в прекрасное будущее.");

break;

}

default:

throw new ArgumentException("Выход за пределы значений");

}

}

/// <summary>

/// Перевод в строку

/// </summary>

/// <returns></returns>

public override string ToString()

{

return Level switch

{

Level.None => "Потрясающе низкий",

Level.Low => "Низкий",

Level.Medium => "Средний",

Level.High => "Высокий",

Level.Unnattainable => "Превосходный",

\_ => null,

};

}

#region Constructor

/// <summary>

/// Создание счётчика с выводом сообщения при проигрыше

/// </summary>

/// <param name="message">Сообщение при проигрыше по причине этого события</param>

public Sphere(string message)

{

this.message = message;

}

#endregion

#region events

public delegate void Game(string txt);

public event Game LevelMax;

#endregion

}

}

using System;

namespace WpfApp.Model

{

/// <summary>

/// Класс описывающий город

/// </summary>

public class Town

{

#region Constante

/// <summary>

/// Сообщение при проигрыше из-за снижения безопасности

/// </summary>

private const string fail\_security = "Террористы взорвали бомбу в центре города. Все жители погибли.";

/// <summary>

/// Сообщение при проигрыше из-за снижения уровня жизни

/// </summary>

private const string fail\_life = "Уровень безысходности достиг предела. Все жители погибли.";

/// <summary>

/// Сообщение при проигрыше из-за снижения уровня образования

/// </summary>

private const string fail\_education = "Жители разучились добывать еду. Город вымер.";

/// <summary>

/// Сообщение при проигрыше из-за снижения уровня экономики

/// </summary>

private const string fail\_economy = "Из-за инфляции деньги обесценились. Город наводнили грабители. Всё население было убито";

/// <summary>

/// Сообщение при происшествии с образованием

/// </summary>

private const string education\_event = "Преподаватели утонули в отчётах. Уровень образования снизился на 20%.";

/// <summary>

/// Сообщение при происшествии с экстренными службами

/// </summary>

private const string epidemic\_event = "В городе эпидемия. Погибло 20% жителей";

/// <summary>

/// Сообщение при происшествии с экстренными службами

/// </summary>

private const string mafia\_event = "Город захватила мафия. Безопасность снижена на 20%.";

/// <summary>

/// Сообщение при происшествии с экстренными службами

/// </summary>

private const string life\_event = "В городе сгорели ТЦ. Уровень жизни упал на 20%.";

/// <summary>

/// Сообщение при мирной жизни

/// </summary>

private const string calm\_event = "В городе всё спокойно.";

/// <summary>

/// Сообщение при происшествии с экономикой

/// </summary>

private const string inflation\_event = "Деньги обесценились. Экономика упала на 20%.";

#endregion

#region Constraction

public Town()

{

SubscriptionEvent(SecurityObj);

SubscriptionEvent(LifeObj);

SubscriptionEvent(EconomyObj);

SubscriptionEvent(EducationObj);

}

/// <summary>

/// Подписка на событие

/// </summary>

/// <param name="c"></param>

private void SubscriptionEvent(Sphere c)

{

c.LevelMax += Subscr;

}

/// <summary>

/// Обработчик события при достижении максимального уровня в одной из характеристик

/// </summary>

/// <param name="txt"></param>

private void Subscr(string txt)

{

if (SecurityObj.IsLevelMax && EconomyObj.IsLevelMax && LifeObj.IsLevelMax && EducationObj.IsLevelMax)

OnGameEnd?.Invoke(txt);

}

#endregion

#region Variables

/// <summary>

/// Наименование города

/// </summary>

public string NameTown { get; set; } = "Город N";

/// <summary>

/// Количество жителей

/// </summary>

private int unitCount = 10;

public int UnitCount

{

get => unitCount;

set

{

if (value <= 0) OnGameEnd("Все жители умерли.");

unitCount = value;

}

}

/// <summary>

/// Динамика изменения

/// </summary>

public int DynamicUnit { get; set; } = 1;

/// <summary>

/// Уровень безопасности

/// </summary>

public string Security => SecurityObj.ToString();

/// <summary>

/// Счётчик уровеня безопасности

/// </summary>

private readonly Sphere SecurityObj = new(fail\_security);

/// <summary>

/// Уровень жизни

/// </summary>

public string Life => LifeObj.ToString();

/// <summary>

/// Счётчик уровеня жизни

/// </summary>

private readonly Sphere LifeObj = new(fail\_life);

/// <summary>

/// Уровень образования

/// </summary>

public string Education => EducationObj.ToString();

/// <summary>

/// Счётчик уровня образования

/// </summary>

private readonly Sphere EducationObj = new(fail\_education);

/// <summary>

/// Уровень экономики

/// </summary>

public string Economy => EconomyObj.ToString();

/// <summary>

/// Счётчик уровня экономики

/// </summary>

private readonly Sphere EconomyObj = new(fail\_economy);

#endregion

#region Methods

/// <summary>

/// Реформа образования

/// </summary>

public void DoEducationReform()

{

UnitCount += DynamicUnit;

DynamicUnit -= 3;

SecurityObj.Count++;

LifeObj.Count++;

EducationObj.Count += 4;

EconomyObj.Count--;

OnMessage?.Invoke("Проведена реформа образования. Уровень образования повышен.");

RandomEvent();

}

/// <summary>

/// Реформа рождаемости

/// </summary>

public void DoFertilityReform()

{

UnitCount += DynamicUnit;

DynamicUnit += 2;

SecurityObj.Count -= 2;

LifeObj.Count -= 2;

EducationObj.Count -= 2;

EconomyObj.Count -= 2;

OnMessage?.Invoke("Проведена реформа рождаемости. Количество жителей увеличивается.");

RandomEvent();

}

/// <summary>

/// Реформа экономики

/// </summary>

public void DoEconomyReform()

{

UnitCount += DynamicUnit;

DynamicUnit += 5;

LifeObj.Count += 3;

EducationObj.Count -= 3;

EconomyObj.Count += 4;

OnMessage?.Invoke("Проведена реформа экономики. Уровень экономики повышен.");

RandomEvent();

}

/// <summary>

/// Реформа экстренных служб

/// </summary>

public void DoExtraReform()

{

UnitCount += DynamicUnit;

SecurityObj.Count += 3;

LifeObj.Count--;

EducationObj.Count--;

EconomyObj.Count -= 2;

OnMessage?.Invoke("Проведена реформа экстренных служб. Безопасность усилилась.");

RandomEvent();

}

/// <summary>

/// С некоторой вероятностью создаёт случайное событие

/// </summary>

public void RandomEvent()

{

Random random = new();

int numberEvent = random.Next(0, 10);

switch (numberEvent)

{

//Эпидемия

case 0:

{

UnitCount = DoEvent(UnitCount, LifeObj.Count, epidemic\_event);

break;

}

//Мафия

case 1:

{

SecurityObj.Count = DoEvent(SecurityObj.Count, SecurityObj.Count, mafia\_event);

break;

}

//Пожар

case 2:

{

LifeObj.Count = DoEvent(LifeObj.Count, SecurityObj.Count, life\_event);

break;

}

//Инфляция

case 3:

{

EconomyObj.Count = DoEvent(EconomyObj.Count, EconomyObj.Count, inflation\_event);

break;

}

//Проблемы с образованием

case 4:

{

EducationObj.Count = DoEvent(EducationObj.Count, EducationObj.Count, education\_event);

break;

}

default:

{

OnMessage(calm\_event);

break;

}

}

}

/// <summary>

/// Выполняет событие, если оно возможно

/// </summary>

/// <param name="field">Изменяемый параметр</param>

/// <param name="parametr">Параметр по которому проверяется вероятность</param>

/// <param name="message">Сообщение при выполнении</param>

/// <returns>Изменённое значение field</returns>

private int DoEvent(int field, int parametr, string message)

{

if (CheckEvent(parametr))

{

field -= field / 20;

OnMessage(message);

}

return field;

}

/// <summary>

/// Проверяет вероятность возникновения события

/// </summary>

/// <param name="parametr">Параметр проверки</param>

/// <returns>true-событие произойдёт, false-если нет</returns>

private static bool CheckEvent(int parametr)

{

Random random = new();

int prop = parametr - random.Next(0, 101);

return prop < 0;

}

#endregion

#region events

public delegate void MethodContainer(string txt);

public event MethodContainer OnGameEnd;

public event MethodContainer OnMessage;

#endregion

}

}