**Практика «Земля и Диггер»**

Когда-то Digger был одной из самых продвинутых и интересных компьютерных игр. В этом блоке задач мы воссоздадим некоторое её подмножество с помощью ООП.

Вам предстоит наполнить готовую заготовку игровыми элементами. Каждый элемент должен уметь:

* Возвращать имя файла, в котором лежит соответствующая ему картинка (например, "Terrain.png")
* Сообщать приоритет отрисовки. Чем выше приоритет, тем раньше рисуется соответствующий элемент, это важно для анимации.
* Действовать — возвращать направление перемещения и, если объект во что-то превращается на следующем ходу, то результат превращения.
* Разрешать столкновения двух элементов в одной клетке.

### Terrain

Сделайте класс Terrain, реализовав ICreature. Сделайте так, чтобы он ничего не делал.

### Player

Сделайте класс Player, реализовав ICreature.

Сделайте так, чтобы диггер шагал в разные стороны в зависимости от нажатой клавиши со стрелкой (Game.KeyPressed). Убедитесь, что диггер не покидает пределы игрового поля.

Сделайте так, чтобы земля исчезала в тех местах, где прошел диггер.

Запустите проект — игра должна заработать!

В методе Game.CreateMap вы можете менять карту, на которой будет запускаться игра. Используйте эту возможность для отладки.

// Вставьте сюда финальное содержимое файла DiggerTask.cs

**Содержимое файла Game.cs**

using System.Windows.Forms;

namespace Digger

{

public static class Game

{

private const string mapWithPlayerTerrain = @"

TTT T

TTP T

T T T

TT TT";

private const string mapWithPlayerTerrainSackGold = @"

PTTGTT TS

TST TSTT

TTTTTTSTT

T TSTS TT

T TTTG ST

TSTSTT TT";

private const string mapWithPlayerTerrainSackGoldMonster = @"

PTTGTT TST

TST TSTTM

TTT TTSTTT

T TSTS TTT

T TTTGMSTS

T TMT M TS

TSTSTTMTTT

S TTST TG

TGST MTTT

T TMTTTT";

public static ICreature[,] Map;

public static int Scores;

public static bool IsOver;

public static Keys KeyPressed;

public static int MapWidth => Map.GetLength(0);

public static int MapHeight => Map.GetLength(1);

public static void CreateMap()

{

Map = CreatureMapCreator.CreateMap(mapWithPlayerTerrain);

}

}

}

**Содержимое файла DiggerTask.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Digger

{

//Напишите здесь классы Player, Terrain и другие.

}

**Программа:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Digger

{

//Напишите здесь классы Player, Terrain и другие.

public class Terrain : ICreature

{

public CreatureCommand Act(int x, int y)

{

return new CreatureCommand { DeltaX = 0, DeltaY = 0 };

}

public bool DeadInConflict(ICreature conflictedObject)

{

return true;

}

public int GetDrawingPriority()

{

return 0;

}

public string GetImageFileName()

{

return "Terrain.png";

}

}

public class Player : ICreature

{

public CreatureCommand Act(int x, int y)

{

Keys key = Game.KeyPressed;

switch (key)

{

case Keys.Down:

if (y < Game.MapHeight - 1) return new CreatureCommand { DeltaX = 0, DeltaY = 1 };

break;

case Keys.Up:

if (y >= 1) return new CreatureCommand { DeltaX = 0, DeltaY = -1 };

break;

case Keys.Right:

if (x < Game.MapWidth - 1) return new CreatureCommand { DeltaX = 1, DeltaY = 0 };

break;

case Keys.Left:

if (x >= 1) return new CreatureCommand { DeltaX = -1, DeltaY = 0 };

break;

}

return new CreatureCommand { DeltaX = 0, DeltaY = 0 };

}

public bool DeadInConflict(ICreature conflictedObject)

{

return false;

}

public int GetDrawingPriority()

{

return 1;

}

public string GetImageFileName()

{

return "Digger.png";

}

}

}