**Practice "Terrain and Digger"**

Digger was once one of the most advanced and interesting computer games out there. In this block of tasks, we will recreate some of its subset using OOP.

You have to fill the finished workpiece with game elements. Each element should be able to:

• Return the name of the file containing the corresponding picture (for example, "Terrain.png")

• Report the priority of rendering. The higher the priority, the earlier the corresponding element is drawn, this is important for animation.

• Act - return the direction of movement and, if the object turns into something on the next move, then the result of the transformation.

• Allow collisions of two elements in one cell.

Terrain

Make a Terrain class by implementing ICreature. Make it so that it doesn't do anything.

Player

Make Player class by implementing ICreature.

Make the digger step in different directions depending on the pressed arrow key (Game.KeyPressed). Make sure the digger does not leave the playing field.

Make the ground disappear in the places where the digger passed.

Start the project - the game should work!

In the Game.CreateMap method, you can change the map on which the game will run. Use this feature for debugging.

// Paste the final content of the DiggerTask.cs file here

**Contains of Game.cs file**

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);

}

}

}

**Contains of DiggerTask.cs file**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Digger

{

// Write here classes Player, Terrain and others.

}

**Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Digger

{

// Write here classes Player, Terrain and others.

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";

}

}

}