﻿using System;

using System.Collections.Generic;

using System.IO;

using System.Threading;

namespace PacMan {

public class Game {

private const string HIGHSCOREPATH = "./scores.txt";

private const int UIWIDTH = 0;

public static Game Instance { get; set; }

public List<Tile> Tiles;

public List<Character> Characters;

public Map Map;

public Pacman Player;

public Vector2 GameArea;

public bool Restart;

private Dictionary<string, int> \_highscores;

private readonly FrameBuffer \_frameBuffer;

public Game() {

Instance = this;

\_frameBuffer = FrameBuffer.Instance;

Characters = new List<Character>();

Tiles = new List<Tile>();

Console.Clear();

Console.CursorVisible = false;

\_frameBuffer.Clear();

Map = new Map();

Console.SetWindowSize(Map.MapSize.X + UIWIDTH, Map.MapSize.Y);

Console.SetBufferSize(Console.WindowWidth, Console.WindowHeight);

DrawGameArea();

//GameArea = new Vector2(Console.WindowWidth - UIWIDTH - 1, Console.WindowHeight);

}

public bool Update() {

KeyHandler.UpdateKeys();

DrawUI();

DrawGameArea();

if (Restart)

return false;

FrameBuffer.Instance.DrawFrame();

Thread.Sleep(25);

return true;

}

private void DrawUI() {

}

private void DrawGameArea() {

foreach (Tile tile in Tiles) {

if (tile.Intersection)

tile.Chixel.BackgroundColor = ConsoleColor.Yellow;

\_frameBuffer.SetChixel(tile.Position, tile.Chixel, FrameBuffer.BufferLayers.Obstacles);

}

foreach (Character character in Characters) {

character.Update();

}

}

private void ExitGame() {

Restart = false;

Environment.Exit(0);

}

private void RestartGame() {

Restart = true;

}

public void End() {

RestartGame();

}

public Tile[,] Get2DArray(Vector2 size) {

Tile[,] tiles = new Tile[size.X, size.Y];

for (int i = 0; i < size.Y; i++) {

for (int j = 0; j < size.X; j++) {

tiles[i, j] = Tiles[i \* size.X + j];

}

}

return tiles;

}

private void SaveScores(string name, int score) {

if (\_highscores.ContainsKey(name))

\_highscores[name] = score;

else

\_highscores.Add(name, score);

StreamWriter writer = null;

try {

writer = new StreamWriter(HIGHSCOREPATH, false);

foreach (KeyValuePair<string, int> entry in \_highscores) {

writer.WriteLine(entry.Key + "|" + entry.Value);

}

} catch (Exception e) {

Console.WriteLine(e.Message);

writer?.Close();

}

writer?.Close();

}

private void LoadScores() {

\_highscores = new Dictionary<string, int>();

if (!File.Exists(HIGHSCOREPATH)) return;

StreamReader reader = null;

try {

reader = new StreamReader(HIGHSCOREPATH);

string line;

while ((line = reader.ReadLine()) != null) {

string[] vals = line.Split('|');

int score = int.Parse(vals[1]);

\_highscores.Add(vals[0], score);

}

} catch (Exception e) {

Console.WriteLine(e.Message);

reader?.Close();

}

reader?.Close();

}

}

}