Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по домашнему заданию

Выполнил:
студент группы ИУ5-35Б
Бронникова М. Е.
Подпись и дата:

Проверил:
Подпись и дата:

Москва, 2024 г

Задание:
Разработать игру Pacman, где пэкмэн перемещается по карте, ограниченной стенами и туннелем, на которой он убегает от 4 призраков, перемещающихся в случайном направлении. Победа присваивается при «съедении» точек.
Выполнение:
Program.cs

```
using PacmanGame;  // Подключаем пространство имен с классом Game1

class Program
{
   static void Main()
   {
      using var game = new Game1();  // Создаем объект класса Game1
      game.Run();  // Запускаем игру
   }
}
```

Game1.cs

```
using System;
using System.Collections.Generic;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using System.Linq;

namespace PacmanGame
{
   public class Game1 : Game
   {
      private GraphicsDeviceManager _graphics;
      private SpriteBatch _spriteBatch;

      private Texture2D pacmanTexture, wallTexture, floorTexture, dotTexture;
      private List<Ghost> ghosts = new List<Ghost>();

      private Vector2 pacmanPosition;
      private int score = 0;
      private int totalDots;
      private bool gameWon = false;
      private bool gameLost = false;

      //private int[,] map;
      private int[,] map = new int[21, 19]
      {
         {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
         {1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 3, 2, 2, 2, 2, 1},
```

```
            {1, 2, 1, 1, 1, 2, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1},
            {1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1},
            {1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1},
            {1, 3, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1},
            {1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1},
            {0, 0, 0, 0, 1, 2, 1, 1, 2, 1, 2, 1, 1, 2, 1, 0, 0, 0, 0},
            {0, 0, 0, 0, 1, 2, 2, 2, 2, 0, 2, 2, 2, 2, 1, 0, 0, 0, 0},
            {1, 1, 1, 1, 1, 2, 1, 1, 1, 0, 1, 1, 1, 2, 1, 1, 1, 1, 1},
            {2, 2, 2, 2, 2, 2, 1, 1, 0, 0, 0, 1, 1, 2, 2, 2, 2, 2, 2},
            {1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1},
            {0, 0, 0, 0, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 0, 0, 0, 0},
            {0, 0, 0, 0, 1, 2, 1, 1, 2, 1, 2, 1, 1, 2, 1, 0, 0, 0, 0},
            {1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1},
            {1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1},
            {1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1},
            {1, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1},
            {1, 2, 1, 1, 1, 2, 1, 1, 2, 1, 2, 1, 1, 2, 1, 1, 1, 2, 1},
            {1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 3, 1},
            {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
        };

        private int maxscore = 0;

        private bool ghostsReleased = false;
        private float ghostReleaseTime = 5.0f; // Время до выхода призраков (в секундах)
        private float elapsedTime = 0.0f;

        public Game1() //ох уж этот жалкий конструктор инициализации
        {
            _graphics = new GraphicsDeviceManager(this);
            Content.RootDirectory = "Content";
            IsMouseVisible = true;

               // генерация окна
            _graphics.PreferredBackBufferWidth = 608;  // 19 * 32 пикселей (ширина карты)
            _graphics.PreferredBackBufferHeight = 672; // 21 * 32 пикселей (высота карты)

            pacmanPosition = new Vector2(32, 32); //generation pacman posit
            totalDots = CountDots(); //food
        }
```

```
private int CountDots()
{
   int dots = 0;
   for (int y = 0; y < map.GetLength(0); y++)
      for (int x = 0; x < map.GetLength(1); x++)
         if (map[y, x] == 2) dots++;
   return dots;
}
bool isBigDotActive = false;

private void CollectBigDot()
{
   isBigDotActive = true; // заготовка под frightened
}

private Texture2D bigDotTexture;
public Vector2 StartPosition(int x, int y)
{
   int cellSize = 32; // Assuming each cell is 32x32 pixels
   return new Vector2(x * cellSize, y * cellSize);
}
private List<Texture2D> pacmanFrames; //for animation
private List<Texture2D> pacmanFramesDown;
private List<Texture2D> pacmanFramesUp;
private List<Texture2D> pacmanFramesLeft;
private int currentFrame;
private double frameTime;
private double timeElapsed;

protected override void LoadContent() //загрузка всех ресурсов
{
      pacmanFrames = new List<Texture2D>
   {
      Content.Load<Texture2D>("frame0"),
      Content.Load<Texture2D>("frame1"),
      Content.Load<Texture2D>("frame2"),
      Content.Load<Texture2D>("frame3"),
      Content.Load<Texture2D>("frame4"),
      Content.Load<Texture2D>("frame3"),
      Content.Load<Texture2D>("frame2"),
      Content.Load<Texture2D>("frame1"),
      Content.Load<Texture2D>("frame0")
   };

      pacmanFramesDown = new List<Texture2D>
```

```
{
    Content.Load<Texture2D>("frameD0"),
    Content.Load<Texture2D>("frameD1"),
    Content.Load<Texture2D>("frameD2"),
    Content.Load<Texture2D>("frameD3"),
    Content.Load<Texture2D>("frame4"),
    Content.Load<Texture2D>("frameD3"),
    Content.Load<Texture2D>("frameD2"),
    Content.Load<Texture2D>("frameD1"),
    Content.Load<Texture2D>("frameD0")
};
    pacmanFramesUp = new List<Texture2D>
{
    Content.Load<Texture2D>("frameU0"),
    Content.Load<Texture2D>("frameU1"),
    Content.Load<Texture2D>("frameU3"),
    Content.Load<Texture2D>("frame4"),
    Content.Load<Texture2D>("frameU3"),
    Content.Load<Texture2D>("frameU1"),
    Content.Load<Texture2D>("frameU0")
};
    pacmanFramesLeft = new List<Texture2D>
{
    Content.Load<Texture2D>("frameL0"),
    Content.Load<Texture2D>("frameL1"),
    Content.Load<Texture2D>("frameL2"),
    Content.Load<Texture2D>("frameL3"),
    Content.Load<Texture2D>("frame4"),
    Content.Load<Texture2D>("frameL3"),
    Content.Load<Texture2D>("frameL2"),
    Content.Load<Texture2D>("frameL1"),
    Content.Load<Texture2D>("frameL0")
};
currentFrame = 0;
frameTime = 0.1; // 0.1 секунды на кадр

_spriteBatch = new SpriteBatch(GraphicsDevice);
bigDotTexture = Content.Load<Texture2D>("fruit");
pacmanTexture = CreateTexture(Color.Yellow);
wallTexture = CreateTexture(Color.Purple);
floorTexture = CreateTexture(Color.Black);
dotTexture = Content.Load<Texture2D>("dot");

// Добавляем 4 призраков с разными цветами
```

```
            ghosts.Add(new Ghost(StartPosition(9, 9),
Content.Load<Texture2D>("red"), map, new Vector2(0, 0))); // Угол карты
вверху слева
            ghosts.Add(new Ghost(StartPosition(9, 10),
Content.Load<Texture2D>("pink"), map, new Vector2(map.GetLength(1) - 1, 0)));
// Угол карты вверху справа
            ghosts.Add(new Ghost(StartPosition(10, 10),
Content.Load<Texture2D>("orange"), map, new Vector2(0, map.GetLength(0) -
1))); // Угол карты внизу слева
            ghosts.Add(new Ghost(StartPosition(8, 10),
Content.Load<Texture2D>("cyan"), map, new Vector2(map.GetLength(1) - 1,
map.GetLength(0) - 1))); // Угол карты внизу справа


        }//public Ghost(Vector2 startPosition, Vector2 exitTarget, Texture2D
ghostTexture, int[,] gameMap, Color color)

        public Texture2D CreateTexture(Color color)
        {
            Texture2D texture = new Texture2D(GraphicsDevice, 32, 32);
            Color[] data = new Color[32 * 32];
            for (int i = 0; i < data.Length; i++) data[i] = color;
            texture.SetData(data);
            return texture;
        }
        private Vector2 pacmanDirection = new Vector2(1, 0); // Начальное
направление вправо
        private bool gameStarted = false; // Флаг для проверки, началась ли игра
        int mapWidth = 576;  // Примерная ширина карты (в клетках) 608
        int mapHeight = 672;
        //Vector2 pacmanPosition = new Vector2(50, 50);
    protected override void Update(GameTime gameTime) //состояние игры на
каждый кадр
    {
            if (Keyboard.GetState().IsKeyDown(Keys.Escape))
            {
                Exit(); // Выход из игры
            }

            if (gameLost || gameWon)
            {
                if (Keyboard.GetState().IsKeyDown(Keys.Space))
                {
                    RestartGame(); // Рестарт игры
                }
```

```
            return;
        }
    timeElapsed += gameTime.ElapsedGameTime.TotalSeconds; //для анимашек

    if (timeElapsed >= frameTime)
    {
        if (pacmanDirection == new Vector2(0, -1))  // Если движение вниз
        {
            currentFrame = (currentFrame + 1) % pacmanFramesDown.Count; // Используем pacmanFramesDown
        }
        if (pacmanDirection == new Vector2(0, 1))  // Если движение вверх
        {
            currentFrame = (currentFrame + 1) % pacmanFramesUp.Count;
        }
        if (pacmanDirection == new Vector2(-1, 0))  // Если движение вktdj
        {
            currentFrame = (currentFrame + 1) % pacmanFramesLeft.Count;
        }
        else
        {
            currentFrame = (currentFrame + 1) % pacmanFrames.Count;
        }

        // без него оно не хотело запускаться
        if (currentFrame >= pacmanFrames.Count || currentFrame >= pacmanFramesDown.Count
         || currentFrame >= pacmanFramesLeft.Count || currentFrame >= pacmanFramesUp.Count)
        {
            currentFrame = 0;  // Сброс индекса в на/чало, если он вышел за пределы
        }

        timeElapsed -= frameTime;
    }

    if (gameLost || gameWon)
    {
        if (Keyboard.GetState().IsKeyDown(Keys.Space))
            Exit(); // Выход, если игра закончена и нажата пробел
        return;
    }
```

```
        if (!gameStarted)
        {
            if (Keyboard.GetState().IsKeyDown(Keys.Space)) // Игрок нажал пробел
для старта
            {
                gameStarted = true; // Игра началась
            }
            return;
        }

        KeyboardState keyboardState = Keyboard.GetState();

        // Выбираем новое направление, если пользователь нажал клавишу
        Vector2 desiredDirection = pacmanDirection;
        if (keyboardState.IsKeyDown(Keys.Up)) desiredDirection = new Vector2(0, -
1);
        if (keyboardState.IsKeyDown(Keys.Down)) desiredDirection = new
Vector2(0, 1);
        if (keyboardState.IsKeyDown(Keys.Left)) desiredDirection = new Vector2(-
1, 0);
        if (keyboardState.IsKeyDown(Keys.Right)) desiredDirection = new
Vector2(1, 0);

        // Проверяем, если новое направление допустимо
        Vector2 nextPosition = pacmanPosition + desiredDirection * 2;
        if (IsValidMove(nextPosition, 32)) // 32 - размер шага
        {
            pacmanDirection = desiredDirection; // Обновляем направление
        }

        Vector2 movePosition = pacmanPosition + pacmanDirection * 2;
        if (IsValidMove(movePosition, 32))
        {
            pacmanPosition = movePosition; // Обновляем позицию
            CollectDot(); // Собираем точку
        }

        if (nextPosition.X < 0)
        {
            // Пакман выходит слева, появляется справа
            pacmanPosition.X = mapWidth - 32;
            pacmanDirection = new Vector2 (-1, 0);
        }
        else if (nextPosition.X >= mapWidth)
        {
```

```
            // Пакман выходит справа, появляется слева
            pacmanPosition.X = 32;
            pacmanDirection = new Vector2 (1, 0);
        }

        if (nextPosition.Y < 0) //на будущее
        {
            // Пакман выходит сверху, появляется снизу
            pacmanPosition.Y = mapHeight - 1;
        }
        else if (nextPosition.Y >= mapHeight)
        {
            // Пакман выходит снизу, появляется сверху
            pacmanPosition.Y = 0;
        }

        // Обновляем призраков
        for (int i = 0; i < ghosts.Count; i++) // Проходим по всем призракам
        {
            var ghost = ghosts[i];
            if (ghost.position.X < 0)
            {
                // Призрак выходит слева, появляется справа
                ghost.position.X = mapWidth - 32; // 32 — ширина спрайта призрака, подставьте нужное значение
                ghost.direction = new Vector2(1, 0); // Меняем направление на правое
            }
            else if (ghost.position.X >= mapWidth)
            {
                // Призрак выходит справа, появляется слева
                ghost.position.X = 32; // 32 — начальная позиция слева
                ghost.direction = new Vector2(-1, 0); // Меняем направление на левое
            }

        // Обновляем движение призрака
            ghost.Update(gameTime, pacmanPosition, ghost.ScatterTarget ); //GameTime gameTime, Vector2 pacmanPosition, Vector2 scatterTarget

            // Проверка на столкновение с призраком
            if (ghost.CheckCollision(pacmanPosition))
            {
                if (ghost.CurrentMode == Ghost.Mode.Frightened)
                {
                    //не удалось
                }
```

```
            else
            {
                gameLost = true; // Проигрыш при столкновении с обычным призраком
                break;
            }
        }
    }

    base.Update(gameTime); // Вызов метода базового класса
}


    //чек границ,чтобы никто по полю не летал
    private bool IsValidMove(Vector2 position, int textureSize)
    {
        // Границы карты
        float minX = 0;
        float minY = 0;
        float maxX = map.GetLength(1) * textureSize;
        float maxY = map.GetLength(0) * textureSize;

        // Проверяем, что позиция Пакмэна не выходит за пределы карты
        if (position.X < minX || position.Y < minY || position.X + textureSize > maxX || position.Y + textureSize > maxY)
            return false;

        int startX = (int)(position.X / textureSize);
        int startY = (int)(position.Y / textureSize);
        int endX = (int)((position.X + textureSize - 1) / textureSize); // Последний элемент по X
        int endY = (int)((position.Y + textureSize - 1) / textureSize); // Последний элемент по Y


        if (startX < 0 || startY < 0 || endX >= map.GetLength(1) || endY >= map.GetLength(0))
            return false;


        for (int x = startX; x <= endX; x++)
        {
            for (int y = startY; y <= endY; y++)
            {
```

```
                if (map[y, x] == 1) // Если в какой-то клетке стена, возвращаем
false
                    return false;
                }
            }

            return true; // Все проверки пройдены!
        }

        public int count = 0;
        private void CollectDot()
        {
            int mapX = (int)(pacmanPosition.X / 32);
            int mapY = (int)(pacmanPosition.Y / 32);

            // Check if the position contains a dot (2)
            if (map[mapY, mapX] == 2)
            {
                map[mapY, mapX] = 0; // Remove the dot
                score += 10;
                count++;
            }
            // Check if the position contains a big dot (3)
            else if (map[mapY, mapX] == 3)
            {
                map[mapY, mapX] = 0;
                score += 50;
                CollectBigDot();     // Trigger big dot effects FAIL

            }
            if(score > maxscore)
            {
                maxscore = score; //6:24
            }

            // Check for win condition
            if (totalDots == count)
            {
                gameWon = true; // Mark the game as won
            }
        }


        protected override void Draw(GameTime gameTime) //отрисовка всего
        {
```

```
GraphicsDevice.Clear(Color.Black);
Texture2D currentpacFrame = null;

_spriteBatch.Begin();

// Отрисовка карты
for (int y = 0; y < map.GetLength(0); y++){
   for (int x = 0; x < map.GetLength(1); x++)
   {
      Vector2 position = new Vector2(x * 32, y * 32);
      if (map[y, x] == 1)
         _spriteBatch.Draw(wallTexture, position, Color.White);
      else if (map[y, x] == 2)
         _spriteBatch.Draw(dotTexture, position, Color.White);
      else if (map[y, x] == 3) // Для больших пиллетов
         _spriteBatch.Draw(bigDotTexture, position, Color.White);
      else
         _spriteBatch.Draw(floorTexture, position, Color.White);
   }
}
//animation
if (pacmanDirection == new Vector2 (0, 1))
{
   currentpacFrame = pacmanFramesDown[currentFrame];
}
else if (pacmanDirection == new Vector2 (0, -1))
{
   currentpacFrame = pacmanFramesUp[currentFrame];
}
if (pacmanDirection.X == -1)
{
   currentpacFrame = pacmanFramesLeft[currentFrame];
}
else if(pacmanDirection.X == 1)
{
   currentpacFrame = pacmanFrames[currentFrame];
}
   if (currentpacFrame != null)
{
   _spriteBatch.Draw(currentpacFrame, pacmanPosition, Color.White);
}

// Отрисовка призраков
foreach (var ghost in ghosts)
{
```

```
            ghost.Draw(_spriteBatch);
        }
                // Отрисовка текста с очками
        string scoreText = $"Score: {score}";
        var font = Content.Load<SpriteFont>("DefaultFont");
        _spriteBatch.DrawString(font, scoreText, new Vector2(10, 10),
Color.White);

        string MaxscoreText = $"MaxScore: {maxscore}";
        //_spriteBatch.DrawString(font, MaxscoreText, new Vector2(450, 10),
Color.White);

        if (gameWon)
            DrawMessage("YOU WIN!");
        else if (gameLost)
            DrawMessage("YOU LOSE!");

        _spriteBatch.End();

        base.Draw(gameTime);
    }

    private void DrawMessage(string message)
    {
        SpriteFont font = Content.Load<SpriteFont>("DefaultFont");
        Vector2 size = font.MeasureString(message);
        Vector2 position = new Vector2(
            (_graphics.PreferredBackBufferWidth - size.X) / 2,
            (_graphics.PreferredBackBufferHeight - size.Y) / 2
        );
        _spriteBatch.DrawString(font, message, position, Color.White);
    }


  private void RestartGame() //обнуление всего, что менялось
  {
    // Сбрасываем состояние игры
    gameWon = false;
    gameLost = false;
    gameStarted = true;

    // Сбрасываем позицию пакмана
    pacmanPosition = new Vector2(32, 32);
    pacmanDirection = new Vector2(1, 0); // Начальное направление вправо
```

```
        // Сбрасываем счет
        score = 0;

        // Возвращаем призраков в начальные позиции
        ghosts.Clear(); // Очищаем список призраков
        ghosts.Add(new Ghost(StartPosition(9, 9), Content.Load<Texture2D>("red"),
map, new Vector2(0, 0)));
        ghosts.Add(new Ghost(StartPosition(9, 10),
Content.Load<Texture2D>("pink"), map, new Vector2(map.GetLength(1) - 1, 0)));
        ghosts.Add(new Ghost(StartPosition(10, 10),
Content.Load<Texture2D>("orange"), map, new Vector2(0, map.GetLength(0) -
1)));
        ghosts.Add(new Ghost(StartPosition(8, 10),
Content.Load<Texture2D>("cyan"), map, new Vector2(map.GetLength(1) - 1,
map.GetLength(0) - 1)));

        // Восстанавливаем карту
        map = new int[21, 19]
        {
            {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
            {1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 3, 2, 2, 2, 2, 1},
            {1, 2, 1, 1, 1, 2, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1},
            {1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1},
            {1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1},
            {1, 3, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1},
            {1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1},
            {0, 0, 0, 0, 1, 2, 1, 1, 2, 1, 2, 1, 1, 2, 1, 0, 0, 0, 0},
            {0, 0, 0, 0, 1, 2, 2, 2, 2, 0, 2, 2, 2, 2, 1, 0, 0, 0, 0},
            {1, 1, 1, 1, 1, 2, 1, 1, 1, 0, 1, 1, 1, 2, 1, 1, 1, 1, 1},
            {2, 2, 2, 2, 2, 2, 1, 1, 0, 0, 0, 1, 1, 2, 2, 2, 2, 2, 2},
            {1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1},
            {0, 0, 0, 0, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 0, 0, 0, 0},
            {0, 0, 0, 0, 1, 2, 1, 1, 2, 1, 2, 1, 1, 2, 1, 0, 0, 0, 0},
            {1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1},
            {1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1},
            {1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1},
            {1, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1},
            {1, 2, 1, 1, 1, 2, 1, 1, 2, 1, 2, 1, 1, 2, 1, 1, 1, 2, 1},
            {1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 3, 1},
            {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
        };

        // Сбрасываем количество точек
        totalDots = CountDots();
    }
```

```
}
}
```
Ghost.cs:

```
using System;
using System.Collections.Generic;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using System.Linq;
using PacmanGame;

namespace PacmanGame{
   public class Ghost
   {
      public enum Mode
      {
         Normal,     // Обычное поведение
         Frightened, // Испуганный режим
         Chase,      // Преследование
         Scatter     // Рассеивание
      }

      public Vector2 position;
      public Vector2 direction;
      private int[,] map;
      private Texture2D texture;
      private float speed;
      private float targetX;  // Целевая позиция по X
      private float targetY;  // Целевая позиция по Y
      private bool isMovingToTarget;
      private bool isAtGridCell;  // Флаг, указывающий, что призрак на границе клетки

      public Mode CurrentMode { get; private set; } // Добавляем свойство для режима

      public Ghost(Vector2 startPosition, Texture2D texture, int[,] map, Vector2 position)//initialize
      {
         this.position = startPosition;
         this.map = map;
         this.texture = texture;
         this.speed = 2.0f;
         this.direction = new Vector2(0, 0);
```

```
            this.CurrentMode = Mode.Normal; // Устанавливаем начальный режим
            this.isMovingToTarget = false;
            this.isAtGridCell = true; // Изначально считаем, что призрак находится на границе клетки
        }
        private float behaviorTimer = 0f;
        public Vector2 ScatterTarget { get; private set; }
        private float behaviorChangeInterval = 10f; // Интервал смены поведения (в секундах)
        //private desiredDirection currentBehavior = GhostBehavior.Scatter; // Начальное поведение
        public void Update(GameTime gameTime, Vector2 pacmanPosition, Vector2 scatterTarget) //изменения за кадр+логика
        {
            ScatterTarget = scatterTarget;
            behaviorTimer += (float)gameTime.ElapsedGameTime.TotalSeconds;
                if (behaviorTimer >= behaviorChangeInterval)
                {
                    behaviorTimer = 0f; // Сбрасываем таймер

                    // Переключаем режим
                    if (CurrentMode == Mode.Scatter) //жалкие попытки
                    {
                        CurrentMode = Mode.Chase;
                    }
                    else if (CurrentMode == Mode.Chase)
                    {
                        CurrentMode = Mode.Scatter;
                    }
                }
            // Если мы находимся на границе клетки, выбираем новое направление
            if (isAtGridCell)
            {
                //Console.WriteLine($"Mode: {CurrentMode}, Direction: {direction}, Position: {position}");
                // Выбираем новое направление в зависимости от режима
                Vector2 desiredDirection = direction;
                switch (CurrentMode)
                {
                    case Mode.Normal:
                        desiredDirection = ChooseDirectionRandomly();
                        break;
                    case Mode.Chase:
                        desiredDirection = GetDirectionToPacman(pacmanPosition);
                        break;
```

```
                case Mode.Scatter:
                    desiredDirection = GetScatterDirection();
                    break;
                case Mode.Frightened:
                    desiredDirection = GetOppositeDirection();
                    break;
            }

            // Проверяем, если новое направление допустимо
            Vector2 nextPosition = position + desiredDirection * speed;
            if (IsValidMove(nextPosition, 32)) // 32 - размер шага
            {
                direction = desiredDirection; // Обновляем направление
            }
        }

        // Двигаем призрака в выбранном направлении
        Vector2 movePosition = position + direction * speed;
        if (IsValidMove(movePosition, 32))
        {
            position = movePosition; // Обновляем позицию
        }

        // Проверяем, если мы достигли конца клетки, чтобы изменить
направление
        if (IsAtGridCell())
        {
            isAtGridCell = true;
        }
        else
        {
            isAtGridCell = false;
        }
    }

    private bool IsAtGridCell()
    {
        // Проверка, что призрак находится на границе клетки (целое число
координат)
        Console.WriteLine($"Mode: {CurrentMode}, Direction: {direction},
Position: {position}");

        return (Math.Abs(position.X % 32) < 0.5f && Math.Abs(position.Y % 32)
< 0.5f);
```

```
}

private Vector2 ChooseDirectionRandomly()
{
   // Список возможных направлений: вверх, вниз, влево, вправо
   Vector2[] possibleDirections = {
      new Vector2(1, 0),  // Вправо
      new Vector2(-1, 0), // Влево
      new Vector2(0, 1),  // Вниз
      new Vector2(0, -1)  // Вверх
   };

   Random rand = new Random();
   return possibleDirections[rand.Next(possibleDirections.Length)];
}

private Vector2 GetDirectionToPacman(Vector2 pacmanPosition)
{
   Vector2 delta = pacmanPosition - position;
   if (Math.Abs(delta.X) > Math.Abs(delta.Y))
   {
      return new Vector2(Math.Sign(delta.X), 0); // Двигаемся по X
   }
   else
   {
      return new Vector2(0, Math.Sign(delta.Y)); // Двигаемся по Y
   }
}

private Vector2 GetScatterDirection()
{
   Vector2 delta = ScatterTarget - position;
   if (Math.Abs(delta.X) > Math.Abs(delta.Y))
   {
      return new Vector2(Math.Sign(delta.X), 0);
   }
   return new Vector2(0, Math.Sign(delta.Y));
}

private Vector2 GetOppositeDirection()
{
   return -direction;
}

private bool IsValidMove(Vector2 position, int textureSize)
```

```
        {
            // Границы карты
            float minX = 0;
            float minY = 0;
            float maxX = map.GetLength(1) * textureSize;
            float maxY = map.GetLength(0) * textureSize;

            // Проверяем, что позиция призрака не выходит за пределы карты
            if (position.X < minX || position.Y < minY || position.X + textureSize > 
maxX || position.Y + textureSize > maxY)
                return false;

            // Проверяем, не выходит ли призрак за границы карты с учетом 
размера
            int startX = (int)(position.X / textureSize);
            int startY = (int)(position.Y / textureSize);
            int endX = (int)((position.X + textureSize - 1) / textureSize); // Последний 
элемент по X
            int endY = (int)((position.Y + textureSize - 1) / textureSize); // Последний 
элемент по Y

            // Проверяем, что призрак не столкнется со стенами (предположим, 
что 1 — это стена)
            for (int x = startX; x <= endX; x++)
            {
                for (int y = startY; y <= endY; y++)
                {
                    if (map[y, x] == 1) // Если в какой-то клетке стена, возвращаем 
false
                        return false;
                }
            }

            return true; // Все проверки пройдены
        }

        public void SetMode(Mode mode)
        {
            CurrentMode = mode;
        }

        public void Draw(SpriteBatch spriteBatch) //отрисовка
        {
            spriteBatch.Draw(texture, position, Color.White);
        }
```

```
        public bool CheckCollision(Vector2 pacmanPosition)
        {
            float collisionDistance = 16; // Задаём допустимое расстояние для столкновения
            return Vector2.Distance(position, pacmanPosition) < collisionDistance;
        }
    }
}
```

Результаты: