Pacman Game Code

Driver Program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using EZInput;
using PacMan.GameGL;

namespace PacMan
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            GameGrid grid = new GameGrid("maze.txt", 23, 71);
            GameCell start = new GameCell(12, 22, grid);
            GamePacManPlayer pacman = new GamePacManPlayer('p', start);
            printMaze(grid);
            printGameObject(pacman);


            bool gameRunning = true;
            while (gameRunning)
            {
                Thread.Sleep(90);
                if (Keyboard.IsKeyPressed(Key.UpArrow))
                {
                    moveGameObject(pacman, GameDirection.Up);
                }

                if (Keyboard.IsKeyPressed(Key.DownArrow))
                {
                    moveGameObject(pacman, GameDirection.Down);
                }

                if (Keyboard.IsKeyPressed(Key.RightArrow))
                {
                    moveGameObject(pacman, GameDirection.Right);
                }

                if (Keyboard.IsKeyPressed(Key.LeftArrow))
                {
                    moveGameObject(pacman, GameDirection.Left);
                }
            }
        }
        1 reference
        static void clearGameCellContent(GameCell gameCell, GameObject newGameObject)
        {
            gameCell.CurrentGameObject = newGameObject;
            Console.SetCursorPosition(gameCell.Y, gameCell.X);
            Console.Write(newGameObject.DisplayCharacter);

        }
```

```
55          static void printGameObject(GameObject gameObject)
56          {
57              Console.SetCursorPosition(gameObject.CurrentCell.Y, gameObject.CurrentCell.X);
58              Console.Write(gameObject.DisplayCharacter);
59
60          }
61
            4 references
62          static void moveGameObject(GameObject gameObject, GameDirection direction)
63          {
64              GameCell nextCell = gameObject.CurrentCell.nextCell(direction);
65              if (nextCell != null)
66              {
67                  GameObject newGO = new GameObject(GameObjectType.NONE, ' ');
68                  GameCell currentCell = gameObject.CurrentCell;
69                  clearGameCellContent(currentCell, newGO);
70                  gameObject.CurrentCell = nextCell;
71                  printGameObject(gameObject);
72              }
73          }
74
            1 reference
75          static void printMaze(GameGrid grid)
76          {
77              for (int x = 0; x < grid.Rows; x++)
78              {
79                  int abc = 0;
80                  for (int y = 0; y < grid.Cols; y++)
81                  {
82                      GameCell cell = grid.getCell(x, y);
82                      GameCell cell = grid.getCell(x, y);
83                      printCell(cell);
84                  }
85
86              }
87          }
88
            1 reference
89          static void printCell(GameCell cell)
90          {
91              Console.SetCursorPosition(cell.Y, cell.X);
92              Console.Write(cell.CurrentGameObject.DisplayCharacter);
93          }
94
95      }
96  }
97
```

Game Cell

```
namespace PacMan.GameGL
{
    22 references
    class GameCell
    {
        int x;
        int y;
        GameObject currentGameObject;
        GameGrid grid;
        2 references
        public GameCell(int x, int y, GameGrid grid)
        {
            this.x = x;
            this.y = y;
            this.grid = grid;
        }
        2 references
        public GameCell nextCell(GameDirection direction)
        {

            if (direction == GameDirection.Left)
            {
                if (this.y > 0)
                {
                    GameCell ncell = grid.getCell(x, y - 1);
                    if (ncell.CurrentGameObject.GameObjectType != GameObjectType.WALL)
                    {
                        return ncell;
                    }
                }
            }

            if (direction == GameDirection.Right)
            {
                if (this.y < grid.Cols - 1)
                {
                    GameCell ncell = grid.getCell(this.x, this.y + 1);
                    if (ncell.CurrentGameObject.GameObjectType != GameObjectType.WALL)
                    {
                        return ncell;
                    }
                }
            }

            if (direction == GameDirection.Up)
            {
                if (this.x > 0)
                {
                    GameCell ncell = grid.getCell(this.x - 1, this.y);
                    if (ncell.CurrentGameObject.GameObjectType != GameObjectType.WALL)
                    {
                        return ncell;
                    }
                }
            }

            if (direction == GameDirection.Down)
```

```
55              {
56                  if (this.x < grid.Rows - 1)
57                  {
58                      GameCell ncell = grid.getCell(this.x + 1, this.y);
59                      if (ncell.CurrentGameObject.GameObjectType != GameObjectType.WALL)
60                      {
61                          return ncell;
62                      }
63                  }
64              }
65              return this; // if can not return next cell return its own reference
66          }
            3 references
67          public int X { get => x; set => x = value; }
            3 references
68          public int Y { get => y; set => y = value; }
            8 references
69          public GameObject CurrentGameObject { get => currentGameObject; set => currentGameObject = value; }
70      }
71
72
```

Game Grid

```
using System;
using System.IO;

namespace PacMan.GameGL
{
    6 references
    class GameGrid
    {
        GameCell[,] cells;
        int rows;
        int cols;

        1 reference
        public GameGrid(String fileName, int rows, int cols ) {
            //Numbers of rows and cols should load from the text file
            this.rows = rows;
            this.cols = cols;
            cells = new GameCell[rows, cols];
            this.loadGrid(fileName);
        }

        public GameCell getCell(int x, int y) {
            return cells[x, y];
        }

        public int Rows { get => rows; set => rows = value; }

        public int Cols { get => cols; set => cols = value; }

        1 reference
        void loadGrid(string fileName)
          {

              StreamReader fp = new StreamReader(fileName);
              string record;
              for (int row=0;row< this.rows;row++)
              {
                  record = fp.ReadLine();
                  for (int col = 0;col < this.cols; col++)
                  {
                      GameCell cell = new GameCell(row,col,this);
                      Char displayCharacter = record[col];
                      GameObjectType type = GameObject.getGameObjectType(displayCharacter);
                      GameObject gameObject = new GameObject(type, displayCharacter);
                      cell.CurrentGameObject = gameObject;
                      cells[row, col] = cell;
                  }
              }

              fp.Close();
          }



    }
}
```

Game Object

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace PacMan.GameGL
8   {
        13 references
9       class GameObject
10      {
11          char displayCharacter;
12          GameObjectType gameObjectType;
13          GameCell currentCell;
            3 references
14          public GameObject(GameObjectType type, char displayCharacter)
15          {
16              this.displayCharacter = displayCharacter;
17              this.gameObjectType = type;
18
19          }
20
            1 reference
21          public static GameObjectType getGameObjectType(char displayCharacter)
22          {
23
24              if (displayCharacter == '|' || displayCharacter == '%' || displayCharacter == '#')
25              {
26                  return GameObjectType.WALL;
27              }
28
29              if (displayCharacter == '.')
30              {
31                  return GameObjectType.REWARD;
32              }
33
34              return GameObjectType.NONE;
35          }
            3 references
36          public char DisplayCharacter { get => displayCharacter; set => displayCharacter = value; }
            4 references
37          public GameObjectType GameObjectType { get => gameObjectType; set => gameObjectType = value; }
            7 references
38          public GameCell CurrentCell
39          {
40              get => currentCell;
41              set
42              {
43                  currentCell = value;
44                  currentCell.CurrentGameObject = this;
45              }
46          }
47      }
48  }
49
```

Game Direction

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace PacMan.GameGL
8   {
        11 references
9       enum GameDirection
10      {
11          Left,
12          Right,
13          Up,
14          Down
15      }
16  }
17
```

Game Object Type

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace PacMan.GameGL
8   {
        14 references
9       enum GameObjectType
10      {
11          WALL,
12          PLAYER,
13          ENEMY,
14          REWARD,
15          NONE
16      }
17  }
18
```

Pacman

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace PacMan.GameGL
8   {
        3 references
9       class GamePacManPlayer : GameObject
10      {
            1 reference
11          public GamePacManPlayer(char displayCharacter,GameCell startCell)
12              :base (GameObjectType.PLAYER,displayCharacter) {
13              this.CurrentCell = startCell;
14          }
            0 references
15          public GameCell move(GameDirection direction) {
16              return this.CurrentCell.nextCell(direction);
17          }
18      }
19
20
21  }
22
```