**Министерсво образования и нАукИ украины**

**Одесский национальный МОРСКОЙ университет**

Кафедра «Техническая кибернетика»

Расчетно-графическое задание

По дисциплине: «Кросс-платформенное программирование»

Тема: «Клиент-серверное приложение на примере игры»

Выполнили:

студент 3 к. 2 гр. КСФ

Шипков В.И.

студент 3 к. 2 гр. КСФ

Иванченко М.М.

Руководитель:

Ст. преподаватель

Рублев И.С.

# 

# Одесса 2018

ОГЛАВЛЕНИЕ

1. [ЗАДАЧА 3](#_Toc482690009)

2. [[РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ПРИЛОЖЕНИЯ](#_Toc482690012)](#_Toc482690011) 8

3. [РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ ПРИЛОЖЕНИЯ](#_Toc482690012) 10

[ВЫВОД 1](#_Toc482690013)2

[ПРИЛОЖЕНИЕ 1](#_Toc482690015)4

1. **ЗАДАЧА**

В данном расчетно-графическом задании, необходимо выполнить следующие поставленные цели:

1. Разработать систему, которая будет поддерживать обмен данными между сервером и клиентом. Причем передавать сразу целый набор сведений с поддержкой многопользовательского подключения.
2. Разработать графический интерфейс для программы-клиента.
3. Запустить и отладить рассматриваемое приложение в целях проверки работоспособности программы в целом и закреплении полученных знаний и навыков.
4. **РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ПРИЛОЖЕНИЯ**

Создание серверной части в данной системе ведется в **Visual Studio** с использованием языка программирования **C#**. Пример выполнения приведен на рис.2.1.

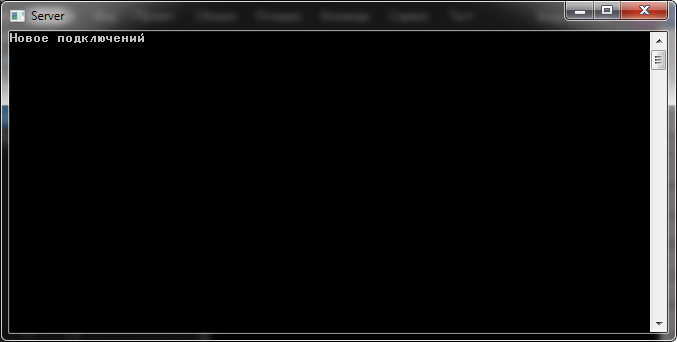


Рисунок 2.1 – Отображение серверной части приложения

В ходе разработки выделяются следующие основные компоненты:

**Class Client** – класс, отвечающий за вызов сокета для подключения к серверу, передачу и генерацию id адресов пользователей, а так же содержащий список клиентов находящихся в игре.

**void Main** - основной компонент серверной части, выводящий заголовок и сведения о подключенных пользователях, а так же определяющий по какому порту работает приложение.

**void HandleClient** – содержит прослушку подключенных пользователей, формирование пакетов для передачи данных и считывание полученных пакетов от клиента, а так же вывод соответствующих ошибок о сбоях в подключении при их возникновении.

Подробный листинг реализации приведен в приложении.

1. **РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ ПРИЛОЖЕНИЯ**

Клиентское приложение так же создавалось в среде разработки **Visual Studio 2015** на языке программирования **C#**. Данное клиентское приложение разработано с помощью технологии **Console Visual C#** и имеет 1 основное окно. В главном окне приложения располагается простой консольный интерфейс нарисован при помощи символов: заставка, подключение к серверу, выбор спрайта и цвета, а так же меню и игровой процесс (прохождение лабиринта), как показано на рис.3.1.



Рисунок 3.1. – Вывод меню игры

В ходе разработки выделяются следующие основные компоненты:

**class Player** – класс, который содержит координаты местоположения в консоли, спрайт, цвет и id каждого клиента в отдельности(все параметры игрока).

**void Draw** – отрисовка игрока на игровом поле.

**void Remove** – стирание игрока на поле.

**void Main** - основной компонент клиентской части, содержащий заставку и вывод меню, а так же отвечающий за подключение к серверу по TCP, выбор спрайта игроком и проверками на возможные движения в игровом поле. После успешной работы выводит игровое поле с лабиринтом для прохождения, как показано на рис.3.2.



Рисунок 3.2. – Вывод игроков и лабиринта

**void SendPacket** – отправка данных о пользователе на сервер при помощи сборки соответствующего пакета.

После того как один из игроков первым проходит лабиринт выводится соответствующая информация и игра прекращается, как отображено на рис.3.3.

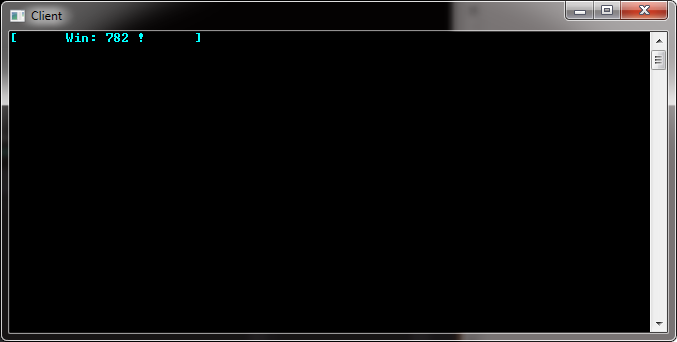


Рисунок 3.3. – Вывод сообщения о победе

**ВЫВОД**

В данном расчетно-графическом задании разработан проект, а именно клиент-серверная игра в жанре головоломка, по прохождению лабиринта. Было проведено проектирование и моделирование всех компонентов касающихся реализации игрового процесса и обмена данными между сервером и клиентом. Приложения было разработано в среде Visual Studio 2015 в соответствии с заданием проекта.

Благодаря проделанной работе мы закрепили полученные теоретические знания на практике.

**ПРИЛОЖЕНИЕ**

Листинг реализации программы:

**Серверная часть**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net.Sockets;

using System.Net;

using System.Text;

using System.Threading.Tasks;

using System.IO;

using System.Threading;

namespace ConsoleApplication1clser

{

class Program

{

class Client

{

public Socket Socket { get; set; }

public int ID { get; set; }

public Client(Socket socket)

{

Socket = socket;

}

}

static Socket socket = new Socket(SocketType.Stream, ProtocolType.Tcp);

static List<Client> clients = new List<Client>();//Список клиентов на сервере

static Random random = new Random();

static void Main(string[] args)

{

Console.Title = "Server";

socket.Bind(new IPEndPoint(IPAddress.Any, 2048));

socket.Listen(0);

socket.BeginAccept(AcceptCallback, null);

Console.ReadLine();

}

static void AcceptCallback(IAsyncResult ar)

{

Client client = new Client(socket.EndAccept(ar));

Thread thread = new Thread(HandleClient);

thread.Start(client);

clients.Add(client);

Console.WriteLine("Новое подключений");

socket.BeginAccept(AcceptCallback, null);

}

static void HandleClient(object o)

{

Client client = (Client)o;

MemoryStream ms = new MemoryStream(new byte[256], 0, 256, true, true);

BinaryWriter writer = new BinaryWriter(ms);

BinaryReader reader = new BinaryReader(ms);

while(true)

{

ms.Position = 0;

try

{

client.Socket.Receive(ms.GetBuffer());

}

catch(Exception e)

{

Console.WriteLine("ERROR : " + e.Message);

socket.Shutdown(SocketShutdown.Both);

socket.Close();

};

//|0

int code = reader.ReadInt32();

//0|

switch (code)

{

case 0:

while (true)

{

int id = random.Next(0, 1001);

if (clients.Find(c => c.ID == id) == null)

{

writer.Write(id);

//0, id|

client.Socket.Send(ms.GetBuffer());

client.ID = id;

break;

}

}

break;

case 1:

foreach (var c in clients)

{

if (c != client)

c.Socket.Send(ms.GetBuffer());

}

break;

}

}

}

}

}

**Клиентская часть**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net.Sockets;

using System.Net;

using System.Text;

using System.Threading.Tasks;

using System.IO;

using System.Threading;

namespace ConsoleApplication1client

{

class Program

{

class Player

{

public int X { get; set; }

public int Y { get; set; }

public char Sprite { get; set; }

public ConsoleColor Color { get; private set; }

public int ID { get; private set;}

public Player(int x, int y, char sprite, ConsoleColor color, int id)

{

X = 1;

Y = 0;

Sprite = sprite;

Color = color;

ID = id;

} //параметры игрока

public void Draw()

{

Console.ForegroundColor = Color;

Console.SetCursorPosition(X, Y);

Console.Write(Sprite);

}//отрисовка игрока

public void Remove()

{

Console.SetCursorPosition(X, Y);

Console.Write(" ");

}//стирание игрока

}

static Socket socket = new Socket(SocketType.Stream, ProtocolType.Tcp);

static MemoryStream ms = new MemoryStream(new byte[256], 0, 256, true, true);

static BinaryWriter writer = new BinaryWriter(ms);

static BinaryReader reader = new BinaryReader(ms);

static List<Player> players = new List<Player>();

static Player player;

static Random random = new Random();

enum PacketInfo

{

ID, Position

}

static void Main(string[] args)

{

Console.Title = "Client";

Console.CursorVisible = false;

Console.WriteLine("███████████████████████████████████████████████████████████████████████████████");

Console.WriteLine("█████████████████████░░░░▀█░░░░███████████████████▀▄▄▄▄▄▄▄▄▄▄▄▀████████████████");

Console.WriteLine("██████████████████████░░░░▀█░░░░████████████████▀░▄███████████▄░▀██████████████");

Console.WriteLine("███████████████████████░░░░░░░░░▄███▀▀░░░██████░▄███████████████▄░█████████████");

Console.WriteLine("████████████████████████░░░▄▄▄█▀▀░░░░▄▄▄███████░█████████████████░█████████████");

Console.WriteLine("██████████████████████████▀▀▀░░░░▄▄████████████░█████████████████░█████████████");

Console.WriteLine("██████████████████████▀▀░░░░▄▄████▀▀███████████░█████████████████░█████████████");

Console.WriteLine("███████████████████▀░░░▄▄▄█▀▀░░▀▀█▄▄▄██████████░█░█████████████░█░█████████████");

Console.WriteLine("███████████████████▄▄█████▀░░▄▄░▄▄█▀▀░▀████████░░▄███▒▒███▒▒███▄░░█████████████");

Console.WriteLine("██████████████████████████░░░██▀█▀░░▄▄█████████░▄██▒▒▒▒███▒▒▒▒██▄░█████████████");

Console.WriteLine("████████████████████████▀█░░░▀█░▀██▀▀░▄████████░██▒▒▄▒▒███▒▒▄▒▒██░█████████████");

Console.WriteLine("███████████████▀▀▀▀░░░░░░▀█░░░▀█░█▄▄██▀▀███████░██▒▒▒▒█████▒▒▒▒██░█████████████");

Console.WriteLine("████████████░░░░░░░░░░░░░░▀█░░░▀█░▀█░▄▄████████░▀███████████████▀░█████████████");

Console.WriteLine("████████████░░░░░░░░░░░░░░░█▄░░░██▄█▀▀▀█████████▄░░░░██║█║██░░░░▄██████████████");

Console.WriteLine("████████████░░░░░░░░░░░░░░░░███████▄░░░████████████░█████████░█████████████████");

Console.WriteLine("████████████░░░░░░░░░░░░░▄▄███████████▀▀▀██████████░█░██░██░█░█████████████████");

Console.WriteLine("████████████░░░░░░░░░▄▄██████████████░▄▄███████████░░░░░░░░░░░█████████████████");

Console.WriteLine("████████████░░░░▄▄▄████████████████████████████████████████████████████████████");

Console.WriteLine("███████████████████████████████████████████████████████████████████████████████");

Console.WriteLine("███████░░░█░██░█░███░█░█░███░░░█░░░█░░░█░░░█░░░█░░░█░░░█░░░█░█░░░█░██░█████████");

Console.WriteLine("███████░█░█░░█░█░░█░░█░░░███░███░█░█░█░█░█░█░█░█░█░█░█░██░██░█░█░█░░█░█████████");

Console.WriteLine("███████░█░█░█░░█░█░█░███░███░███░█░█░░██░░██░█░█░░██░░░██░██░█░█░█░█░░█████████");

Console.WriteLine("███████░░░█░██░█░███░█░░░███░░░█░░░█░█░█░███░░░█░█░█░█░██░██░█░░░█░██░█████████");

Console.WriteLine("███████████████████████████████████████████████████████████████████████████████");

Thread.Sleep(1000);

Console.Clear();

///////////////////////////////////////////Меню

Console.WriteLine("[ █ ████ ███ █ ███ █ █ █ █████ ]");

Console.WriteLine("[ █ █ █ █ █ █ █ █ █ ██ █ █ █ ]");

Console.WriteLine("[ █ ████ ███ █ ███ █ █ █ █ █ ███ ]");

Console.WriteLine("[ █ █ █ █ █ █ █ █ █ █ █ ██ █ █ ]");

Console.WriteLine("[ ████ █ █ ████ █ █ █ █ █ █ █ ]");

Console.WriteLine(" ");

Console.WriteLine(" ");

Console.WriteLine("[ Press Y to continue ]");

Console.WriteLine("[ Or N to exit ]");

ConsoleKeyInfo key = Console.ReadKey(true);

if (key.Key == ConsoleKey.N) { Environment.Exit(0); }

Console.ReadLine();

Console.Clear();

if (key.Key == ConsoleKey.Y)

{

Console.WriteLine("Подключение к серверу...");

socket.Connect("127.0.0.1", 2048);

Console.WriteLine("Подключено");

Thread.Sleep(2000);

Console.Clear();

Console.WriteLine("Введите спрайт:");

string tchar = Console.ReadLine();

char spr = Convert.ToChar(tchar[0]);

Console.Clear();

Console.WriteLine("Выберите цвет:");

for (int i = 0; i <= 14; i++)

{

Console.ForegroundColor = (ConsoleColor)i;

Console.WriteLine(i);

}

Console.ResetColor();

ConsoleColor clr = (ConsoleColor)int.Parse(Console.ReadLine());

Console.Clear();

int x = random.Next(1, 5);

int y = random.Next(1, 5);

Console.WriteLine("Получение идентификатора");

SendPacket(PacketInfo.ID);

int id = ReceivePacket();

Console.WriteLine("Получен ID:" + id);

Thread.Sleep(1000);

Console.Clear();

player = new Player(x, y, spr, clr, id);

SendPacket(PacketInfo.Position);

//////////////////лабиринт

Console.ForegroundColor = (ConsoleColor)7;

bool sost = false;

int[,] nums2 = { {1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,},

{1,0,1,0,0,0,0,1,0,0,0,1,1,1,1,0,0,0,0,1,},

{1,0,1,0,1,1,0,1,0,1,0,0,0,0,1,0,1,1,0,1,},

{1,0,0,0,0,0,0,1,0,1,1,1,1,0,1,0,0,1,0,1,},

{1,1,1,1,1,1,0,1,0,1,1,1,1,0,1,1,0,1,0,1,},

{1,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,1,},

{1,0,1,1,0,1,1,1,0,0,0,0,1,0,1,1,1,1,0,1,},

{1,0,0,0,0,0,0,0,0,1,1,1,1,0,1,0,1,1,1,1,},

{1,1,1,1,1,1,0,1,1,1,0,0,0,0,1,0,1,1,1,1,},

{1,1,0,0,0,1,0,1,1,1,0,1,1,1,1,0,0,0,0,1,},

{1,0,0,1,0,0,0,0,0,1,0,0,0,0,1,1,1,1,0,1,},

{1,1,1,1,1,1,0,1,1,1,1,1,1,0,1,0,0,0,0,1,},

{1,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,1,1,1,},

{1,0,1,1,0,1,1,1,1,1,1,1,1,0,1,0,0,0,0,1,},

{1,0,1,0,0,0,1,0,0,0,1,0,0,0,0,0,1,1,0,1,},

{1,0,1,1,1,1,1,0,1,0,1,1,1,0,1,0,0,0,0,1,},

{1,0,1,0,0,0,1,0,1,0,0,0,1,0,1,1,1,1,1,1,},

{1,0,1,0,1,0,1,0,1,1,1,0,0,0,0,0,0,0,0,1,},

{1,0,1,0,1,0,1,0,1,0,1,1,1,1,1,1,1,1,0,1,},

{1,0,1,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,1,},

{1,0,1,0,1,0,0,0,0,0,1,1,1,1,1,1,0,1,1,1,},

{1,0,1,0,1,1,1,1,1,0,0,0,1,0,0,0,0,0,0,1,},

{1,0,1,0,0,0,0,0,1,1,1,0,0,0,1,1,1,1,0,1,},

{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,3,1,} };

for (int i = 0; i < nums2.GetLength(0); i++)

{

for (int j = 0; j < nums2.GetLength(1); j++)

{

if (nums2[i, j] == 0) { Console.Write(" "); }

if (nums2[i, j] == 1) { Console.Write("█"); }

if (nums2[i, j] == 3) { Console.Write("░"); }

}

Console.WriteLine();

}

//¯\\_(ツ)\_/¯

//∩( ・ω・)∩

Task.Run(() => { while (true) ReceivePacket(); });

while (true)

{

player.Draw();

switch (Console.ReadKey(true).Key)

{

case ConsoleKey.LeftArrow:

if (nums2[player.Y, player.X - 1] == 1) { break; }

player.Remove(); player.X--;

break;

case ConsoleKey.RightArrow:

if (nums2[player.Y, player.X + 1] == 1) { break; }

player.Remove(); player.X++;

break;

case ConsoleKey.UpArrow:

if (nums2[player.Y - 1, player.X] == 1) { break; }

player.Remove(); player.Y--;

break;

case ConsoleKey.DownArrow:

if (nums2[player.Y + 1, player.X] == 1) { break; }

player.Remove(); player.Y++;

break;

}//switch

if (nums2[player.Y, player.X] == 3)

{

Console.Clear();

Console.WriteLine("[ Win: " + id + " ! ]");

Thread.Sleep(3000);

break;

}

if (player.X < 0)

{ player.X++; }

if (player.Y < 0)

{ player.Y++; }

if (player.Y >= 25)

{ player.Y--; }

if (player.X >= 80)

{ player.X--; }

player.Draw(); SendPacket(PacketInfo.Position);

}//while

Console.ReadKey(true);

}

else

Console.WriteLine("The wrong key is pressed");

}//main

static void SendPacket(PacketInfo info)

{

ms.Position = 0;

switch (info)

{

case PacketInfo.ID:

writer.Write(0);

socket.Send(ms.GetBuffer());

break;

case PacketInfo.Position:

writer.Write(1);

writer.Write(player.ID);

writer.Write(player.X);

writer.Write(player.Y);

writer.Write(player.Sprite);

writer.Write((int)player.Color);

socket.Send(ms.GetBuffer());

break;

}

}

static int ReceivePacket()

{

ms.Position = 0;

socket.Receive(ms.GetBuffer());

//|0, id

int code = reader.ReadInt32();

//0, |id

int id;

int x;

int y;

char sprite;

ConsoleColor color;

switch(code)

{

case 0: return reader.ReadInt32();

case 1:

id = reader.ReadInt32();

x = reader.ReadInt32();

y = reader.ReadInt32();

Player plr = players.Find(p=> p.ID == id);

if (plr != null)

{

plr.Remove();

plr.X = x;

plr.Y = y;

plr.Draw();

}

else

{

sprite = reader.ReadChar();

color = (ConsoleColor)reader.ReadInt32();

plr = new Player(x, y, sprite, color, id);

players.Add(plr);

plr.Draw();

}

break;

}

return -1;

}

}

}