УТВЕРЖДЕН

RU.17701729.05.03-01 ТЗ 01-1-ЛУ

**Разработка системы диагностики артроза по рентген снимкам коленного сустава**

**Текст программы**

**RU.17701729.05.03-01**

**Листов 27**



**Пермь 2024**

**Аннотация**

В данном документе приведен текст программы прототипа клона игры “Doom”. Текст программы приведён на исходном языке. Исходным языком является C#. Среда разработки – Microsoft Visual Studio.

# 1 Текст программы на исходном языке

## 1.1 Код, отвечающий за пулю

using System;

using System.Drawing;

using System.Runtime.Remoting.Messaging;

using COURSEPROJ.models;

namespace COURSEPROJ.entities

{

public class Bullet:steper

{

public static short globid = 499;

public short posx;

public short posy;

public Image curimg;

public float dposx;

public float dposy;

public short wait4Death = 0;

public bool friendly;

public short id;

public float x

{

get { return dposx; }

set { dposx = value; }

}

public short speed { get; set; }

public float angle { get; set; }

public float y

{

get { return dposy; }

set { dposy = value; }

}

public short ID

{ get { return id; } }

public Image img

{ get { return this.curimg; } }

public Bullet(short x, short y, short speed, float angle, bool friendly, ref int[,] MAP)

{

curimg = Hero.sprBullet;

posx = x;

posy = y;

dposx = x;

dposy = y;

id = ++globid;

MAP[y / 64, x / 64] = id;

this.speed = speed;

this.angle = angle;

Form1.stepersimgs[ID % 2048] = this.img;

wait4Death = 0;

this.friendly = friendly;

}

public void step(ref int[,] MAP, ref Hero hero)

{

Form1.stepersimgs[ID % 2048] = this.img;

if (curimg == Hero.sprBulletExploded)

{

if (wait4Death == 5)

{

Program.disposeObject(this);

return;

}

MAP[(int) y / 64, (int) x / 64] = id;

wait4Death++;

return;

}

float dx = (float)(speed\*Math.Cos(angle));

float dy = (float)(speed\*Math.Sin(angle));

MAP[(int)y / 64, (int)x / 64] = 0;

if ((y + dy) / 64 >= MAP.GetLength(1) |

(x + dx) / 64 >= MAP.GetLength(0) |

(y + dy) / 64 < 0 |

(x + dx) / 64 < 0|

speed<5)

{

curimg = Hero.sprBulletExploded;

return;

}

if (MAP[((int)y + (short)(Math.Sign(dy)\*speed)) / 64, ((int)x+(short)(Math.Sign(dx)\*speed)) / 64] >= 100 |

MAP[((int)y + (short)(Math.Sign(dy)\*speed)) / 64, ((int)x+(short)(Math.Sign(dx)\*speed)) / 64] == 0)

{

this.dposy += dy;

this.dposx += dx;

this.posx = (short) this.dposx;

this.posy = (short) this.dposy;

}

else

{

speed = (short)(speed / 1.5) ;

dx = (float)(speed\*Math.Cos(angle));

dy = (float)(speed\*Math.Sin(angle));

if (MAP[((int)y + (short)(Math.Sign(dy)\*speed)) / 64, ((int)x+(short)(Math.Sign(dx)\*speed)) / 64] >= 100 |

MAP[((int)y + (short)(Math.Sign(dy)\*speed)) / 64, ((int)x+(short)(Math.Sign(dx)\*speed)) / 64] == 0)

{

this.dposy += dy/2;

this.dposx += dx/2;

this.posx = (short) this.dposx;

this.posy = (short) this.dposy;

}

else

{

curimg = Hero.sprBulletExploded;

return;

}

}

MAP[(int) y / 64, (int) x / 64] = id;

}

public short sub { get; set; }

~Bullet()

{

Console.WriteLine(this.ID+" successfully destroyed");

}

}

}

## 1.2 Код, отвечающий за противника первого типа

using System;

using System.Drawing;

using COURSEPROJ.models;

namespace COURSEPROJ.entities

{

public class enemy1:steper

{

public static short globid = 99;

public float posx;

public float posy;

public short id, speed;

public double stepsub;

private bool shooted;

public float subspeed;

public Image[] curimg;

private Random rnd = new Random();

public float x

{

get { return posx; }

set { posx = value; }

}

public float y

{

get { return posy; }

set { posy = value; }

}

public short ID

{

get { return id; }

}

public enemy1(short x, short y, ref int[,] MAP)

{

curimg = Hero.sprEnemy2\_walk;

shooted = false;

subspeed = 0;

speed = 25;

sub = 0;

stepsub = 0;

posx = x;

posy = y;

id = ++globid;

MAP[y / 64, x / 64] = id;

}

public void step(ref int[,] MAP, ref Hero hero)

{

Form1.stepersimgs[ID % 2048] = this.img;

stepsub += subspeed;

sub = (short) stepsub;

short distToPlayer = (short)Math.Sqrt(Math.Pow(hero.posX - this.posx, 2) + Math.Pow(hero.posY - this.posy, 2));

double angleToPlayer = Hero.findAngle(this.posx, this.posy, hero.posX, hero.posY);

for (int i = 0; i < distToPlayer; i += 5)

{

int scanned = MAP[(short) (this.posy + i \* Math.Sin(angleToPlayer)) / 64, (short) (this.posx + i \* Math.Cos(angleToPlayer)) / 64];

if (

scanned!=0 & scanned<99

)

{

return;

}

}

if (this.sub % curimg.Length == 1 & this.curimg == Hero.sprEnemy2\_shoot&this.shooted==false)

{

Form1.stepers.Add(

new Bullet( (short)(this.posx+100\*Math.Cos(angleToPlayer)+10\*Math.Cos(angleToPlayer+90)), (short)(this.posy+100\*Math.Sin(angleToPlayer)+10\*Math.Sin(angleToPlayer+90)),

25,

(float)angleToPlayer-(float)Math.PI/180\*2,

false, ref MAP)

);

shooted = true;

}

if (sub % curimg.Length != 1 | curimg != Hero.sprEnemy2\_shoot)

{

shooted = false;

}

foreach (steper steper\_ in Form1.stepers.ToArray())

{

if (steper\_ is Bullet blt)

{

if (Math.Abs(blt.posx-this.posx)<64&

Math.Abs(blt.posy-this.posy)<64)

{

if (blt.friendly)

{

if (MAP[(int) y / 64, (int) x / 64] == id)

{

MAP[(int) y / 64, (int) x / 64] = 0;

}

Program.disposeObject(this);

}

else

{

blt.curimg=Hero.sprBulletExploded;

}

}

}

}

switch (distToPlayer)

{

case >600 and <2500:

subspeed = .2f;

speed = 15;

this.curimg = Hero.sprEnemy2\_walk;

float dx = (float) (speed \* Math.Cos(angleToPlayer));

float dy = (float) (speed \* Math.Sin(angleToPlayer));

MAP[(int) y / 64, (int) x / 64] = 0;

if (MAP[

((int) y + Math.Sign(dy) \* 16) / 64,

((int) x + Math.Sign(dx) \* 16) / 64

] == 0|

MAP[

((int) y + Math.Sign(dy) \* 16) / 64,

((int) x + Math.Sign(dx) \* 16) / 64

]>=100

)

{

this.posy += dy;

this.posx += dx;

}

MAP[(int) y / 64, (int) x / 64] = id;

break;

case <=600:

speed = 0;

subspeed = .1f;

this.curimg = Hero.sprEnemy2\_shoot;

break;

case >=2500:

speed = 0;

subspeed = 0;

this.curimg = Hero.sprEnemy2\_shoot;

stepsub = 0;

break;

}

}

public Image img

{

get

{

return curimg[sub % curimg.Length];

}

}

~enemy1()

{

Console.WriteLine(this.ID+" successfully destroyed");

}

public short sub { get; set; }

}

}

## 1.3 Код, отвечающий за противника второго типа

using System;

using System.Drawing;

using COURSEPROJ.models;

namespace COURSEPROJ.entities

{

public class enemy2:steper

{

public static short globid = 199;

public float posx;

public float posy;

public short id, speed;

private bool shooted;

public double stepsub;

public float subspeed;

public Image[] curimg;

private Random rnd = new Random();

public float x

{

get { return posx; }

set { posx = value; }

}

public float y

{

get { return posy; }

set { posy = value; }

}

public short ID

{

get { return id; }

}

public enemy2(short x, short y, ref int[,] MAP)

{

curimg = Hero.sprEnemy1\_walk;

shooted = false;

Form1.stepersimgs[ID % 2048] = this.img;

subspeed = 0;

speed = 15;

sub = 0;

stepsub = 0;

posx = x;

posy = y;

id = ++globid;

MAP[y / 64, x / 64] = id;

}

public void step(ref int[,] MAP, ref Hero hero)

{

Form1.stepersimgs[ID % 2048] = this.img;

stepsub += subspeed;

sub = (short) stepsub;

short distToPlayer = (short)Math.Sqrt(Math.Pow(hero.posX - this.posx, 2) + Math.Pow(hero.posY - this.posy, 2));

double angleToPlayer = Hero.findAngle(this.posx, this.posy, hero.posX, hero.posY);

for (int i = 0; i < distToPlayer; i += 5)

{

int scanned = MAP[(short) (this.posy + i \* Math.Sin(angleToPlayer)) / 64, (short) (this.posx + i \* Math.Cos(angleToPlayer)) / 64];

if (

scanned!=0 & scanned<99

)

{

return;

}

}

if (this.sub % curimg.Length == 1 & this.curimg == Hero.sprEnemy1\_shoot&this.shooted==false)

{

Form1.stepers.Add(

new Bullet( (short)(this.posx+100\*Math.Cos(angleToPlayer)+10\*Math.Cos(angleToPlayer+90)),

(short)(this.posy+100\*Math.Sin(angleToPlayer)+10\*Math.Sin(angleToPlayer+90)),

20,

(float)angleToPlayer-(float)Math.PI/180/3,

false,ref MAP)

);

shooted = true;

}

if (sub % curimg.Length != 1 | curimg != Hero.sprEnemy1\_shoot)

{

shooted = false;

}

foreach (steper steper\_ in Form1.stepers.ToArray())

{

if (steper\_ is Bullet blt)

{

if (Math.Abs(blt.posx-this.posx)<32&

Math.Abs(blt.posy-this.posy)<32)

{

if (blt.friendly)

{

if (MAP[(int) y / 64, (int) x / 64] == id)

{

MAP[(int) y / 64, (int) x / 64] = 0;

}

Program.disposeObject(this);

}

else

{

blt.curimg=Hero.sprBulletExploded;

}

}

}

}

switch (distToPlayer)

{

case <2500:

subspeed = .2f;

speed = 20;

this.curimg = Hero.sprEnemy1\_walk;

float dx = (float) (speed \* Math.Cos(angleToPlayer+Math.PI));

float dy = (float) (speed \* Math.Sin(angleToPlayer+Math.PI));

MAP[(int) y / 64, (int) x / 64] = 0;

if (MAP[

((int) y + Math.Sign(dy) \* 64) / 64,

((int) x + Math.Sign(dx) \* 64) / 64

] == 0)

{

this.posy += dy;

this.posx += dx;

}

else

{

speed = 0;

subspeed = .05f;

this.curimg = Hero.sprEnemy1\_shoot;

}

MAP[(int) y / 64, (int) x / 64] = id;

break;

case >=2500:

speed = 0;

subspeed = .1f;

this.curimg = Hero.sprEnemy1\_shoot;

break;

}

}

public Image img

{

get

{

return curimg[sub % curimg.Length];

}

}

~enemy2()

{

Console.WriteLine(this.ID+" successfully destroyed");

}

public short sub { get; set; }

}

}

## 1.4 Код, отвечающий за персонажа

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Drawing.Configuration;

using System.Drawing.Drawing2D;

using System.Drawing.Imaging;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Timers;

using System.Windows.Forms;

using COURSEPROJ.models;

using static COURSEPROJ.Form1;

namespace COURSEPROJ.entities

{

public class Hero

{

public double posX;

public double posY;

public float \_angle = 90;

public double speed = 20;

public short hspeed=0;

public short vspeed=0;

public double dirAngle;

public Image curImg;

public Image texStoneWall = new Bitmap(Path.Combine(

new DirectoryInfo(Directory.GetCurrentDirectory()).FullName, "sprites\\16px\_5.png"));

public Image texStoneWallDamaged = new Bitmap(Path.Combine(

new DirectoryInfo(Directory.GetCurrentDirectory()).FullName, "sprites\\16px\_4.png"));

public Image texFinalWall = new Bitmap(Path.Combine(

new DirectoryInfo(Directory.GetCurrentDirectory()).FullName, "sprites\\16px\_7.png"));

public Image texFinalSideWall = new Bitmap(Path.Combine(

new DirectoryInfo(Directory.GetCurrentDirectory()).FullName, "sprites\\16px\_6.png"));

public Image[] texFinalThingGif =

{

new Bitmap(Path.Combine(new DirectoryInfo(Directory.GetCurrentDirectory()).FullName,

"sprites\\16px\_10\_1.png")),

new Bitmap(Path.Combine(new DirectoryInfo(Directory.GetCurrentDirectory()).FullName,

"sprites\\16px\_10\_2.png"))

};

public static Image sprBulletExploded = new Bitmap(Path.Combine(

new DirectoryInfo(Directory.GetCurrentDirectory()).FullName, "sprites\\16px\_8.png"));

public static Image[] sprEnemy1\_walk =

{

new Bitmap(Path.Combine(new DirectoryInfo(Directory.GetCurrentDirectory()).FullName,

"sprites\\spr\_enemy1\_walk1.png")),

new Bitmap(Path.Combine(new DirectoryInfo(Directory.GetCurrentDirectory()).FullName,

"sprites\\spr\_enemy1\_walk2.png")),

new Bitmap(Path.Combine(new DirectoryInfo(Directory.GetCurrentDirectory()).FullName,

"sprites\\spr\_enemy1\_walk3.png")),

new Bitmap(Path.Combine(new DirectoryInfo(Directory.GetCurrentDirectory()).FullName,

"sprites\\spr\_enemy1\_walk4.png"))

};

public static Image[] sprEnemy2\_walk =

{

new Bitmap(Path.Combine(new DirectoryInfo(Directory.GetCurrentDirectory()).FullName,

"sprites\\spr\_enemy2\_walk1.png")),

new Bitmap(Path.Combine(new DirectoryInfo(Directory.GetCurrentDirectory()).FullName,

"sprites\\spr\_enemy2\_walk2.png")),

new Bitmap(Path.Combine(new DirectoryInfo(Directory.GetCurrentDirectory()).FullName,

"sprites\\spr\_enemy2\_walk3.png")),

new Bitmap(Path.Combine(new DirectoryInfo(Directory.GetCurrentDirectory()).FullName,

"sprites\\spr\_enemy2\_walk4.png"))

};

public static Image[] sprEnemy1\_shoot =

{

new Bitmap(Path.Combine(new DirectoryInfo(Directory.GetCurrentDirectory()).FullName,

"sprites\\spr\_enemy1\_shoot1.png")),

new Bitmap(Path.Combine(new DirectoryInfo(Directory.GetCurrentDirectory()).FullName,

"sprites\\spr\_enemy1\_shoot2.png")),

};

public static Image[] sprEnemy2\_shoot =

{

new Bitmap(Path.Combine(new DirectoryInfo(Directory.GetCurrentDirectory()).FullName,

"sprites\\spr\_enemy2\_shoot1.png")),

new Bitmap(Path.Combine(new DirectoryInfo(Directory.GetCurrentDirectory()).FullName,

"sprites\\spr\_enemy2\_shoot2.png")),

new Bitmap(Path.Combine(new DirectoryInfo(Directory.GetCurrentDirectory()).FullName,

"sprites\\spr\_enemy2\_shoot3.png")),

};

public static Image sprBullet = new Bitmap(Path.Combine(

new DirectoryInfo(Directory.GetCurrentDirectory()).FullName, "sprites\\16px\_9.png"));

public static double findAngle(double x1, double y1, double x2, double y2)

{

return Math.Atan2(

(y2-y1),(x2-x1)

);

}

public float Angle

{

get{return \_angle;}

set

{

\_angle = value%360;

while (\_angle<0){\_angle+=360;}

\_angle = \_angle%360;

}

}

public Hero()

{

}

public void handleKeysDown(KeyPressEventArgs e)

{

Console.WriteLine(e.KeyChar);

switch (e.KeyChar)

{

case 'w':

this.vspeed=1;

break;

case 'a':

this.hspeed=-1;

break;

case 's':

this.vspeed=-1;

break;

case 'd':

this.hspeed=1;

break;

}

}

public void handleKeysUp(KeyEventArgs e)

{

switch (e.KeyCode)

{

case Keys.W:

this.vspeed=0;

break;

case Keys.A:

this.hspeed=0;

break;

case Keys.S:

this.vspeed=0;

break;

case Keys.D:

this.hspeed=0;

break;

case Keys.Escape:

Environment.Exit(0);

break;

}

}

public void step(EventArgs e)

{

if (this.hspeed == this.vspeed & this.vspeed == 0) return;

double vangle = ((1 - vspeed)/2) \* Math.PI ;

double hangle = Math.PI / 2 \* this.hspeed ;

double newAngDelta = (vangle == 0 ? hangle : vangle);

if (vangle != 0 & hangle != 0)

{

newAngDelta = Math.Atan2(

Math.Sin(vangle) + Math.Sin(hangle),

Math.Cos(hangle) + Math.Cos(vangle)

);

}

double newAng = (Angle - 10)\*Math.PI/180 + newAngDelta;

double deltaX = this.speed \* Math.Cos(newAng);;

double deltaY = this.speed \* Math.Sin(newAng);;

if (Form1.MAP[(short)(this.posY+Math.Sign(deltaY)\*64)/64, (short)(this.posX+Math.Sign(deltaX)\*64)/64] == 0 |

Form1.MAP[(short)(this.posY+Math.Sign(deltaY)\*64)/64, (short)(this.posX+Math.Sign(deltaX)\*64)/64] >= 100)

{

this.posX += deltaX;

this.posY += deltaY;

}

}

public void drawSelf(Bitmap b)

{

}

public void watch(Bitmap b, Bitmap b2, ref int[,] MAP)

{

float[] lens = new float[720];

short[] flats = new short[720];

bool[] drawed = new bool[1000];

bool[] met = new bool[stepers.Count];

float[] steperCoords = new float[stepers.Count\*2];

float[] ids = new float[stepers.Count\*2];

short stepi = 0;

foreach (steper steper\_ in stepers)

{

steperCoords[stepi \* 2] = steper\_.x;

steperCoords[stepi \* 2 + 1] = steper\_.y;

ids[stepi] = steper\_.ID;

stepi++;

}

float[] getCollision(ref int[,] MAP\_, float angleInGrad, double startx, double starty)

{

angleInGrad\*=(float)Math.PI/180;

float x\_;

float y\_;

double cos = Math.Cos(angleInGrad);

double sin = Math.Sin(angleInGrad);

for (float c = 0f; c<640\*20; c+=1f) {

x\_ = Convert.ToSingle(startx + c\*cos);

y\_ = Convert.ToSingle(starty + c\*sin);

short spltdx = (short)(x\_ / 64);

short spltdy = (short)(y\_ / 64);

if (MAP\_[spltdy, spltdx] == 0)

{

continue;

}

if (MAP\_[spltdy,spltdx]==2|

MAP\_[spltdy,spltdx]==3|

MAP\_[spltdy,spltdx]==4|

MAP\_[spltdy,spltdx]==5|

MAP\_[spltdy,spltdx]==6) {

return new []

{

x\_,

y\_,

Convert.ToSingle(MAP\_[spltdy,spltdx])

};

}

for (short i1=0; i1<stepers.Count;i1++)

{

short i2 = (short)(i1 \* 2);

if (Math.Abs(steperCoords[i2]-x\_)<6 & Math.Abs(steperCoords[i2+1]-y\_)<6 & met[i1]==false)

{

met[i1] = true;

return new[]

{

steperCoords[i1\*2],

steperCoords[i1\*2+1],

ids[i1]

};

}

}

}

return new [] {

Convert.ToSingle(this.posX),

Convert.ToSingle(this.posY),

-1f};

}

Graphics g = Graphics.FromImage(b);

g.InterpolationMode=InterpolationMode.NearestNeighbor;

g.SmoothingMode=SmoothingMode.HighQuality;

Graphics g2 = Graphics.FromImage(b2);

g2.InterpolationMode=InterpolationMode.NearestNeighbor;

g2.SmoothingMode=SmoothingMode.HighQuality;

float[] coords;

Image tex = texStoneWall

for (short i=0; i<720;i++)

{

float angleForCoords = Angle-45+((float)i)/8

coords = getCollision(ref MAP, angleForCoords-10, this.posX, this.posY);

byte tex\_size = 16;

float columnHeight;

void drawVertical()

{

switch (coords[2])

{

case 2:

tex = texStoneWall;

break;

case 3:

tex = texStoneWallDamaged;

break;

case 4:

tex = texFinalSideWall;

break;

case 5:

tex = texFinalWall;

break;

case 6:

tex = texFinalThingGif[

(int)Math.Round(

(Math.Sin(Form1.elapsedtimers/3)+1)/2

)

];

break;

case >99 and <200:

break;

case >499 and <600:

break;

default:

tex = texFinalWall;

break;

}

//стена

if (coords[2] < 99f)

{

float len = Convert.ToSingle(Math.Sqrt(Math.Pow(coords[0] - posX, 2.0) +

Math.Pow(coords[1] - posY, 2.0)));

columnHeight =

Math.Max(Convert.ToSingle(7500 \* 2 / (len \* Math.Cos((45 - i / 8) \* Math.PI / 180))), 0);

short tex\_x = (short) (Math.Max(

coords[0] / 64 - (short) ((coords[0] + 1) / 64),

coords[1] / 64 - (short) ((coords[1] + 1) / 64)

) \* tex\_size);

while (tex\_x < 0)

{

tex\_x += tex\_size;

}

if (len < lens[i] | lens[i] == 0)

{

lens[i] = len;

}

g.DrawImage(tex, new Rectangle(i, 180 - (short) columnHeight, 1, (short) columnHeight \* 2),

tex\_x,

0, 1, tex\_size, GraphicsUnit.Pixel);

//тень

if (lens[i] > 5)

{

g.DrawLine(Form1.pens[Math.Min((short) (len / 7), (short) 150)],

i, 180 - (short) columnHeight, i, 180 + (short) columnHeight);

}

}

//монстр

if (coords[2] > 99f & drawed[(short) (coords[2]) % drawed.Length] == false)

{

float len\_ = Convert.ToSingle(Math.Sqrt(Math.Pow(coords[0] - posX, 2.0) +

Math.Pow(coords[1] - posY, 2.0)));

if (len\_ < lens[i] | lens[i] == 0)

{

lens[i] = len\_;

flats[i] = (short) coords[2];

}

drawed[(short) coords[2] % drawed.Length] = true;

}

}

drawVertical();

while (coords[2] > 99f)

{

coords = getCollision(ref MAP, angleForCoords-10, coords[0], coords[1]);

drawVertical();

}

}

//draw flat objects

while (!Enumerable.SequenceEqual(flats, new short[720] ))

{

float minlen = 0;

short minflat = 0;

short mini = 0;

for (short i = 0; i < 720; i++)

{

if (flats[i]==0){continue;}

if (lens[i] > minlen)

{

minlen = lens[i];

minflat = flats[i];

mini = i;

}

}

flats[mini] = 0;

byte spr\_size = (byte)Form1.stepersimgs[minflat].Width;

float columnHeight = Math.Max(Convert.ToSingle(7500 \* 2 / (minlen \* Math.Cos((45 - mini / 8) \* Math.PI / 180))), 0)/32\*spr\_size;

short start = (short) Math.Max(0, (int) (mini - columnHeight));

short end = (short) Math.Min(720, mini + columnHeight);

ImageAttributes attr = new ImageAttributes();

if (minflat > 99 & minflat < 200)

{

ColorMatrix colormatrix = new ColorMatrix(new float[][]

{

new float[] {0, 0, 0, 0, 0}, // red scaling factor of 2

new float[] {0, 0, 0, 0, 0}, // green scaling factor of 1

new float[] {0, 0, 0, 0, 0}, // blue scaling factor of 1

new float[] {0, 0, 0, (float)(Math.Min((short) (minlen / 7), (short) 150)) / 255, 0},

new float[] {0, 0, 0, 0, 1}

}

);

attr.SetColorMatrix(colormatrix);

}

short[] toDraw = new short[end - start];

float[] srcs = new float[end - start];

for (short j = start; j < end-1; j++)

{

if (minlen < lens[mini])

{

lens[mini] = minlen;

}

if (minlen <= lens[j]|minlen <= lens[j+1])

{

float srcx = ((j - mini + columnHeight) / (columnHeight \* 2)) \* spr\_size;

toDraw[j - start] = j;

srcs[j - start] = srcx;

Rectangle rect = new Rectangle(j, 180 - (short) columnHeight, 1, (short) (columnHeight) \* 2);

g2.DrawImage(Form1.stepersimgs[minflat%2048],

rect,

srcx,

0, 1, spr\_size, GraphicsUnit.Pixel);

if (minlen > 70)

{

g2.DrawImage(Form1.stepersimgs[minflat%2048],

rect,

j,

0, 1, spr\_size, GraphicsUnit.Pixel, attr);

}

}

}

}

}

}

}

## 1.5 Вспомогательный код для пули и противников

using System.Drawing;

using COURSEPROJ.entities;

namespace COURSEPROJ.models

{

public interface steper

{

float x { get; set; }

float y { get; set; }

short ID { get; }

void step(ref int[,] MAP, ref Hero hero);

Image img { get; }

short sub { get; set; }

}

}

## 1.6 Код для запуска программы

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

using COURSEPROJ.models;

namespace COURSEPROJ

{

public static class Program

{

public static void disposeObject(steper obj)

{

for (short i=0; i<Form1.stepers.Count;i++)

{

if (Form1.stepers[i] == null)

{

continue;

}

if (Form1.stepers[i].ID == obj.ID)

{

Form1.stepers.RemoveAt(i);

}

}

GC.Collect();

}

public static Form1 window;

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

window = new Form1();

Application.Run(window);

}

}

}