# Занятие 2. Реализация метода потенциальных полей с использованием Python/C#

Общая схема реализуемого модельного примера представлена на рис. 2.1. Черным прямоугольником обозначен робот, фиолетовым кругом – препятствие и красной звездой – цель. С точки зрения метода потенциальных полей все объекты воспринимаются как материальные точки, расположенные в геометрических центрах перечисленных фигур.

С каждой точкой сопоставляется заряд и масса. Векторы сил притяжения в каждой паре частиц определяются знаками двух зарядов и величиной обратно пропорциональной квадрату расстояния между частицами. Масса влияет на ускорение приобретаемое частицей под действием соответствующей силы.

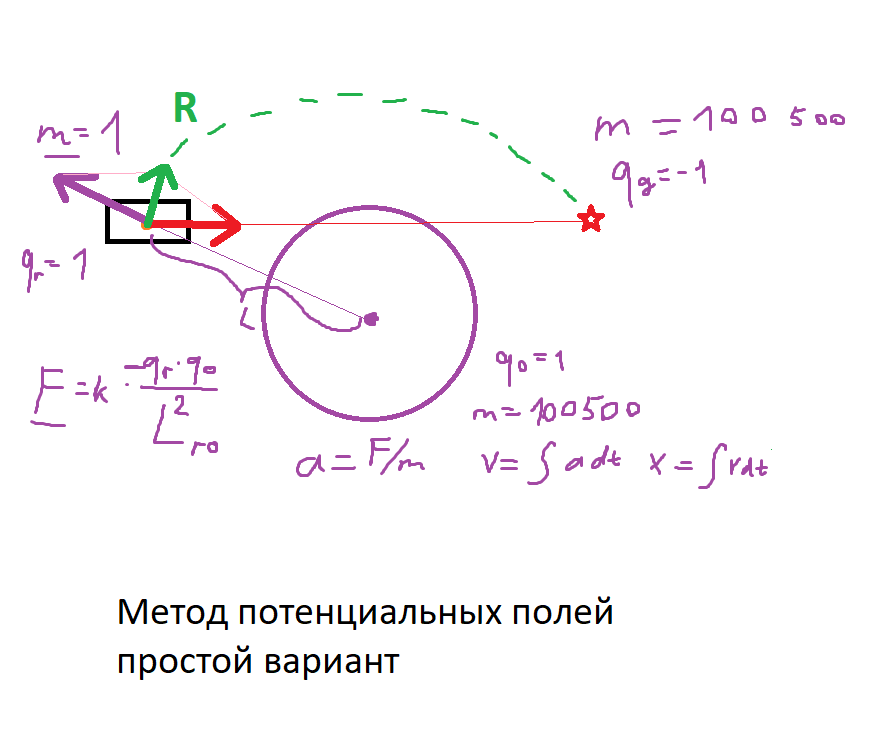


Рис. 2.1. Реализация метода потенциальных полей для системы робот-препятствие-цель. Зеленым цветом показан результирующий вектор сил, действующих на робота. Пунктиром обозначена траектория движения робота к цели.

Возможны и альтернативные версии метода потенциальных полей, как представлено на рис. 2.2.

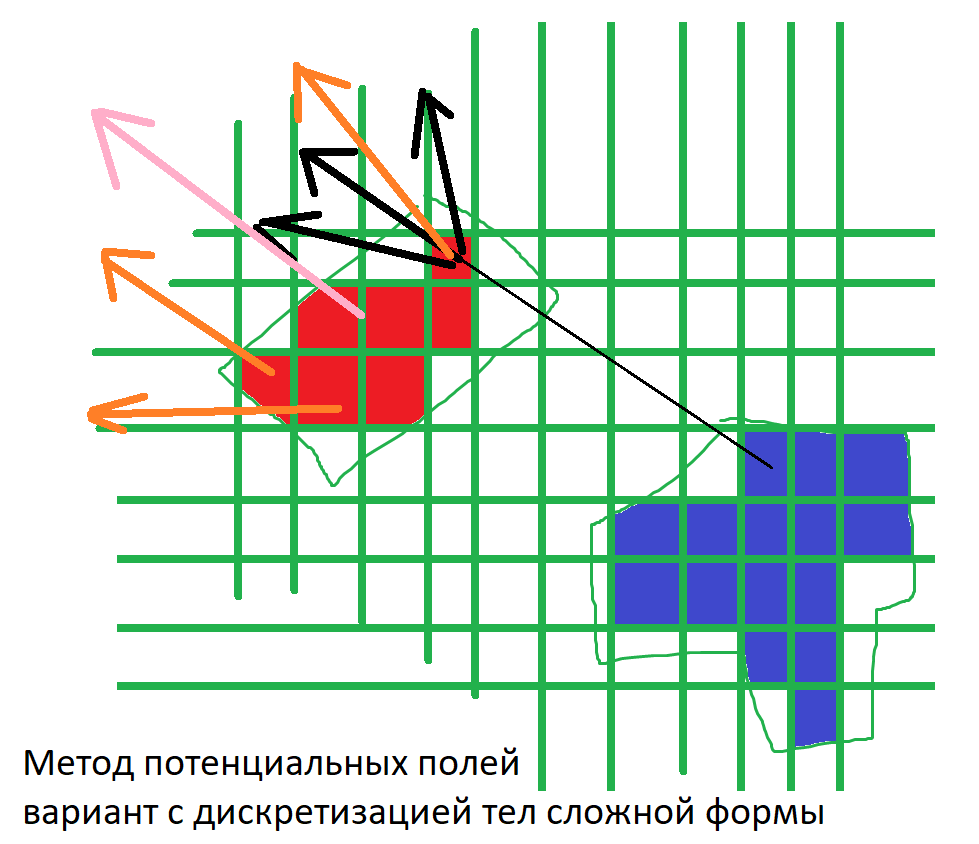


Рис. 2.2. Реализация альтернативной версии метода потенциальных полей, где каждое тело рассматривается как совокупность частиц.

В конечном счете, на выходе метода потенциальных полей получаем вектор R, в сторону которого должен поворачивать и двигаться робот (рис. 2.3)

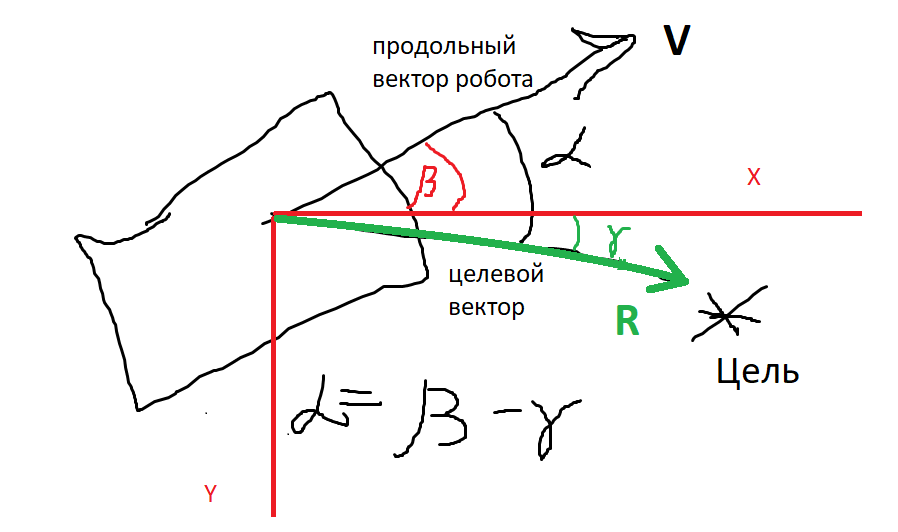


Рис. 2.3. Иллюстрация векторов и углов, описывающих положение робота и целевой точки

Для реализации подобного управления разработаем 2 программных модуля:

- сервер моделирования мобильного робота на языке C# (см. Практич. занятие 1);

- клиент управления мобильным роботом на языке Python.

Обобщенная схема проргаммного комплекса, включающего в свой состав данные модули, представлена на рис. 2.4. Обмен данными между модулями производится посредством сетевого протокола HTTP.

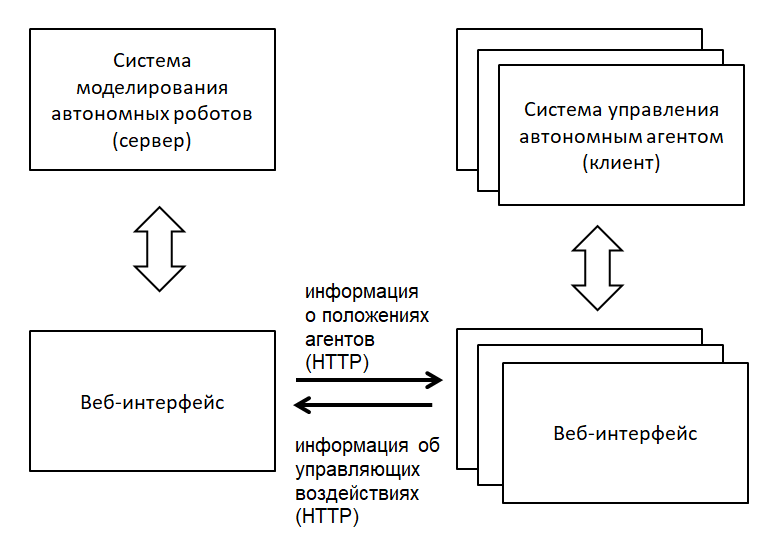


Рис. 2.4. Обобщенная структура системы по управлению виртуальной моделью мобильного робота

Пример практической реализации подобного ПО приведен на рис. 2.5.

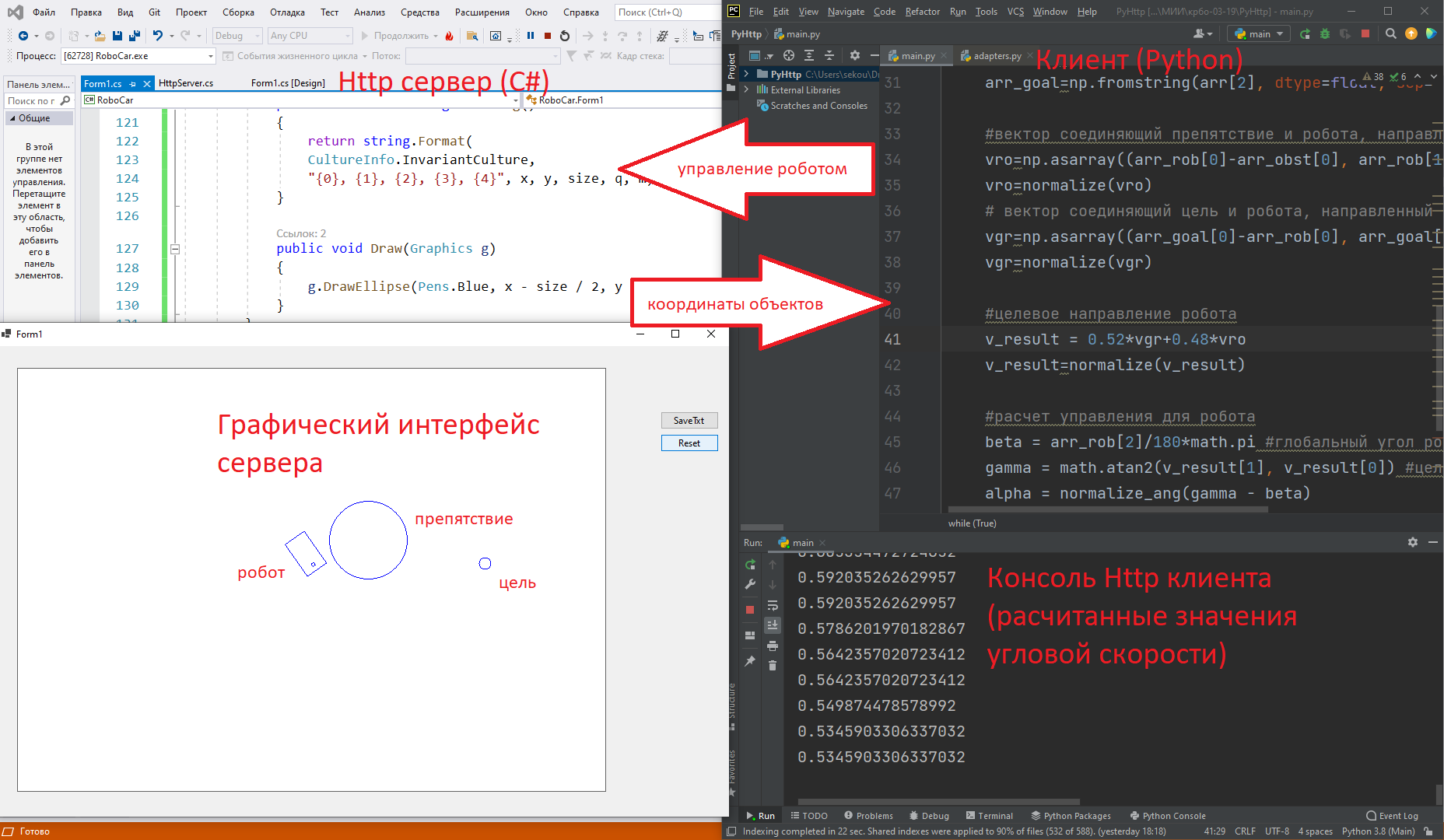
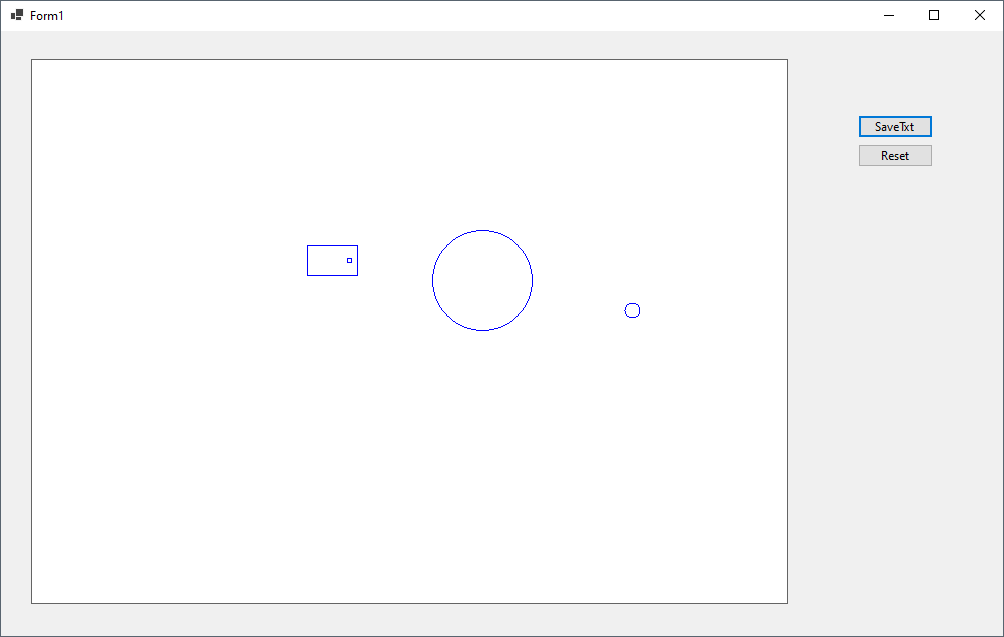


Рис. 2.5. Совместная работа в средах разработки Visual Studio (для создания моделирующего сервера) и PyCharm (для создания управляющей программы)

Код управления и обработки информации для клиентского приложения

import math  
import requests  
import numpy as np  
from urllib.parse import urlencode  
  
#сессия HTTP - чтоб избежать многократных переподключений к серверу  
session = requests.Session()  
  
def send(data):  
 str = urlencode(data)  
 r = session.get("http://127.0.0.1:8080/?" + str, timeout=1000)  
 return r.text.split("UserResult=")[1]  
  
def normalize(x):  
 norm = np.sqrt(x.dot(x))  
 x = x / norm if norm != 0 else x  
 return x  
  
def normalize\_ang(ang):  
 while(ang<-math.pi):ang+=2\*math.pi  
 while(ang>math.pi):ang-=2\*math.pi  
 return ang  
  
time=0  
#управляющие данные  
data = {'id': 0, 'v': 1, 'w': 0}  
  
while(True):  
 info=send(data)  
  
 #обработка ответа от сервера об объектах среды  
 arr=info.split('; ')  
 arr\_rob=np.fromstring(arr[0], dtype=float, sep=', ')  
 arr\_obst=np.fromstring(arr[1], dtype=float, sep=', ')  
 arr\_goal=np.fromstring(arr[2], dtype=float, sep=', ')  
  
 #вектор соединяющий препятствие и робота, направленный в сторону робота  
 vro=np.asarray((arr\_rob[0]-arr\_obst[0], arr\_rob[1]-arr\_obst[1]))  
 Lro=np.sqrt(vro.dot(vro)) #расстояниe до препятствия  
 Fro = 100\*\*2/Lro\*\*2 #сила обратно пропорциональна квадрату расстояния  
 vro=normalize(vro)\*Fro  
  
 #вектор соединяющий цель и робота, направленный в сторону цели  
 vgr=np.asarray((arr\_goal[0]-arr\_rob[0], arr\_goal[1]-arr\_rob[1]))  
 Fgr=1 #сила - константа  
 vgr=normalize(vgr)\*Fgr  
  
 #целевое направление робота  
 v\_result = 0.5\*vgr+0.5\*vro  
 v\_result=normalize(v\_result)  
  
 #расчет управления для робота  
 beta = arr\_rob[2]/180\*math.pi #глобальный угол робота  
 gamma = math.atan2(v\_result[1], v\_result[0]) #целевой угол  
 alpha = normalize\_ang(gamma - beta)  
  
 data['w'] = 2\*alpha  
  
 print(data['w'])  
 time+=1

Код сервера



using System;

using System.Collections.Generic;

using System.Drawing;

using System.Globalization;

using System.IO;

using System.Net;

using System.Threading;

using System.Windows.Forms;

namespace RoboCar

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

Graphics g;

RobCar r1 = new RobCar(), r2 = new RobCar();

Obj obst = new Obj();

Obj goal = new Obj();

private void Form1\_Load(object sender, EventArgs e)

{

bt\_reset\_Click(null, null);

pb.Image = new Bitmap(pb.Width, pb.Height);

g = Graphics.FromImage(pb.Image);

//g.DrawRectangle(Pens.Blue, 30, 30, 100, 50);

timer1.Enabled = true;

var srv = new MyServer(IPAddress.Loopback, 8080);

srv.onGET = p =>

{

//считываем параметры управления

//роботом из http-запроса

var str\_id = HttpHelper.

GetParam(p.http\_url, "id");

var str\_v = HttpHelper.GetParam(p.http\_url, "v");

var str\_w = HttpHelper.GetParam(p.http\_url, "w");

RobCar r = null;

if (str\_id == "0") r = r1;

if (str\_id == "1") r = r2;

try

{

r.vlin = float.Parse(str\_v, CultureInfo.InvariantCulture);

}

catch { }

try

{

r.vrot = float.Parse(str\_w, CultureInfo.InvariantCulture);

}

catch { }

string info = string.Format(CultureInfo.InvariantCulture,

"{0}; {1}; {2}", r.ToString(), obst.ToString(), goal.ToString());

return info;

};

var t = new Thread(new ThreadStart(srv.listen));

t.Start();

}

private void timer1\_Tick(object sender, EventArgs e)

{

r1.Move();// 1, 3);

r2.Move();// 2, -3);

g.Clear(Color.White);

r1.Draw(g);

//r2.Draw(g);

obst.Draw(g);

goal.Draw(g);

pb.Invalidate();

lines.Add(r1.ToString() + "; " + r2.ToString());

}

public class float2

{

public float x, y;

public float2() { }

public float2(float x, float y) { this.x = x; this.y = y; }

public float2 Rotate(float ang)

{

var v = new float2();

float s = (float)Math.Sin(ang);

float c = (float)Math.Cos(ang);

v.x = x \* c - y \* s;

v.y = y \* c + x \* s;

return v;

}

}

List<string> lines=new List<string>();

private void bt\_txt\_Click(object sender, EventArgs e)

{

File.WriteAllLines("lines.txt", lines);

}

public class Obj //произвольный объект (цель или препятствие)

{

public float x, y, size;

public float q, m;//заряд и масса

public override string ToString()

{

return string.Format(

CultureInfo.InvariantCulture,

"{0}, {1}, {2}, {3}, {4}", x, y, size, q, m);

}

public void Draw(Graphics g)

{

g.DrawEllipse(Pens.Blue, x - size / 2, y - size / 2, size, size);

}

}

private void bt\_reset\_Click(object sender, EventArgs e)

{

r1 = new RobCar { x = 300, y = 200, ang\_deg = 0 };

r2 = new RobCar { x = 300, y = 150, ang\_deg = 45 };

obst = new Obj { x = 450, y = 220, size = 100, m = float.MaxValue, q = 1 };

goal = new Obj { x = 600, y = 250, size = 15, m = float.MaxValue, q = -1 };

}

public class RobCar

{

public float L=50, w=30;

public float x, y, ang\_deg;

public float vrot, vlin;

public void Draw(Graphics g)

{

var T = g.Transform;

g.TranslateTransform(x, y);

g.RotateTransform(ang\_deg);

g.DrawRectangle(Pens.Blue, -L/2, -w/2, L, w);

g.DrawRectangle(Pens.Blue, L/3-2, -2, 4, 4);

g.Transform = T;

}

public void Move(float v, float w)

{

this.vlin = v; this.vrot = w;

Move();

}

public void Move()

{

var vec = new float2(1, 0).Rotate(ang\_deg / 180 \* (float)Math.PI);

x += vlin \* vec.x;

y += vlin \* vec.y;

ang\_deg += vrot;

}

public override string ToString()

{

return string.Format(CultureInfo.InvariantCulture, "{0}, {1}, {2}", x, y, ang\_deg);

}

}

}

}