Министерство образования и молодежной политики Свердловской области ГБОУ СО КШИ «Екатеринбургский кадетский корпус»

**Паспорт проекта**

**практико-ориентированный проект**

 **Программная модель теплового движения молекул идеального газа**

**Автор:** Старостин Александр Андреевич, кадет 211 взвода ГБОУ СО КШИ «Екатеринбургский кадетский корпус»

 **Руководитель:** Ковалева Ирина Эдуардовна, учитель физики ГБОУ СО КШИ «Екатеринбургский кадетский корпус»

Екатеринбург 2021 г.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc98273837)

[ГЛАВА 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 5](#_Toc98273838)

[1.1. Основы молекулярно-кинетической теории идеального газа. 5](#_Toc98273839)

[1.2. Объединенный газовый закон 6](#_Toc98273840)

[1.3. Изопроцессы 8](#_Toc98273841)

[1.4 Столкновение молекул 10](#_Toc98273842)

[1.4.1 Центральное столкновение шариков 10](#_Toc98273843)

[1.4.2 Нецентральное столкновение шариков 12](#_Toc98273844)

[ГЛАВА 2. ПРАКТИЧЕСКАЯ ЧАСТЬ 14](#_Toc98273845)

[2.1. Язык программирования 14](#_Toc98273846)

[2.2. Разработка интерфейса программы 14](#_Toc98273847)

[2.2. Разработка программного кода 16](#_Toc98273848)

[2.3. Разработка инструкции пользователя 18](#_Toc98273849)

[ЗАКЛЮЧЕНИЕ 20](#_Toc98273850)

[СПИСОК ИСТОЧНИКОВ И ЛИТЕРАТУРЫ 21](#_Toc98273851)

[ПРИЛОЖЕНИЯ 22](#_Toc98273852)

# ВВЕДЕНИЕ

Современный мир все больше использует на благо человечества компьютерное моделирование процессов: умный дом, голосовые помощники, симуляторы вождения и даже школьное обучение включает в себя работу с программами, например, тестирование, виртуальные лаборатории, виртуальные экскурсии. Все стараются донести до нас знания в интересной, часто игровой форме.

В этом году мы изучали тепловое движение идеального газа. В воображении сложно представить, как именно поведут себя молекулы газа при разных условиях. Исходя из выше сказанного, я определил **проблему исследования,** как упростить получение знаний о тепловом движении идеального газа.

Я увлекаюсь программированием и разработкой простых программ для компьютеров. Поэтому решил объединить программное моделирование, анимацию и новые знания о тепловом движении, которые сможет получить любой школьник, поработав в моей программе. Программа позволяет упростить этот процесс получения информации, быстро увидеть изменение поведения газа при разных условиях.

**Объектом исследования является:** тепловое движение идеального газа.

**Предмет исследования –** модель поведения идеального газа при разных условиях.

**Цель работы**: создание компьютерной программы расчета и визуализации движения молекул идеального газа.

Проблема и цель исследования определили его **задачи:**

1) изучить тепловое движение идеального газа;

2) разработать программу движения молекул на языке программирования python;

3)провести тестирование программы;

4) разработать инструкцию пользователя.

**Дорожная карта проекта**

|  |  |  |  |
| --- | --- | --- | --- |
| Ключевые задачи | сентябрь-октябрь | ноябрь-декабрь | январь-март |
| Изучение теории | Изучение понятия идеального газа,  |  |  |
| Подготовка оборудования | Установка необходимого программного обеспечения |  |  |
| Расчет формул |  | Теоретическое обоснование программы |  |
| Создание программы |  | Разработка интерфейса | Написание кода программы |

# ГЛАВА 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

**1.1. Основы молекулярно-кинетической теории идеального газа.**

Молекулярно-кинетическая теория (сокращённо МКТ) — теория, возникшая в XIX веке и рассматривающая строение вещества, в основном газов, с точки зрения трёх основных приближенно верных положений. Положения были сформулированы М.В. Ломоносовым в XVII веке.

Основные положения МКТ:

* Все тела состоят из частиц: атомов и молекул;
* Эти частицы беспрерывно движутся;
* Частицы взаимодействуют между собой.

Идеальный газ – это теоретическая модель газа, которая соответствует следующим требованиям:

* расстояние между молекулами намного больше их размеров (молекулы можно считать материальными точками);
* силами взаимодействия, кроме сил, возникающих при соударениях, можно пренебречь (потенциальная энергия взаимодействия молекул по сравнению с кинетической энергией хаотического движения пренебрежимо мала);
* столкновения молекул друг с другом и со стенками сосуда являются абсолютно упругими;
* движение каждой молекулы подчиняется законам классической динамики Ньютона

**Основное уравнение Молекулярно-кинетической теории**: давление идеального газа пропорционально произведению массы молекулы, концентрации молекул и среднему квадрату скорости движения молекул.

**p = 1/3·m0·n·v2**

m0 - масса одной молекулы газа;

n = N/V – концентрация молекул;

v2- средняя квадратичная скорость движения молекул.

   Так как средняя кинетическая энергия поступательного движения молекул E = m0\*v2/2, то домножив основное уравнение МКТ на 2, получим p = 2/3· n·(m0· v2)/2 = 2/3·E·n

**p = 2/3·E·n**

Давление газа равно 2/3 от средней кинетической энергии поступательного движения молекул, которые содержатся в единичном объеме газа.

Так как m0·n = m0·N/V = m/V = ρ,   где ρ – плотность газа, то имеем     **p = 1/3· ρ· v2**

**1.2. Объединенный газовый закон**

Макроскопические величины, однозначно характеризующие состояние газа, называют **термодинамическими параметрами газа.**

Важнейшими термодинамическими параметрами газа являются его**объем V, давление р и температура Т.**

   Всякое изменение состояния газа называется **термодинамическим процессом.**

   В любом термодинамическом процессе изменяются параметры газа, определяющие его состояние.

   Соотношение между значениями тех или иных параметров в начале и конце процесса называется **газовым законом**.

   Газовый закон, выражающий связь между всеми тремя параметрами газа называется **объединенным газовым законом.**

**p = nkT**

   Соотношение ***p* = *nkT***связывающее давление газа с его температурой и концентрацией молекул, получено для модели идеального газа, молекулы которого взаимодействуют между собой и со стенками сосуда только во время упругих столкновений. Это соотношение может быть записано в другой форме, устанавливающей связь между макроскопическими параметрами газа – объемом *V*, давлением *p*, температурой *T* и количеством вещества ν. Для этого нужно использовать равенства

 $n= \frac{N}{V}$

где n – концентрация молекул, N – общее число молекул, V – объем газа

Тогда получим  $n= \frac{N}{V}$ или  $p\frac{V}{T}=Nk$

   Так как при постоянной массе газа N остается неизменным, то Nk – постоянное число, значит $\frac{pV}{t}=const$

При постоянной массе газа произведение объема на давление, деленное на абсолютную температуру газа, есть величина одинаковая для всех состояний этой массы газа.

Уравнение, устанавливающее связь между давлением, объемом и температурой газа было получено в середине XIX века французским физиком Б. Клапейроном и часто его называют **уравнением Клайперона**.

Уравнение Клайперона можно записать в другой форме.

*p* = *nkT,*

учитывая, что

 $n= \frac{N}{V}= \frac{vN\_{A}}{V}= \frac{mN\_{A}}{MV}$

   Здесь *N* – число молекул в сосуде, ν – количество вещества, *N*А – постоянная Авогадро, *m* – масса газа в сосуде, *M* – молярная масса газа. В итоге получим:

$pV=vN\_{A}kT =\frac{m}{M}N\_{A}kT $

Произведение постоянной Авогадро NА на постоянную Больцмана k называется **универсальной газовой постоянной** и обозначается буквой R.

Ее численное значение в СИ   ***R* = 8,31 Дж/моль·К**

Соотношение

$pV=vRT= \frac{m}{M}RT$

называется **уравнением состояния идеального газа**.

В полученной форме оно было впервые записано Д. И. Менделеевым. Поэтому уравнение состояния газа называется **уравнением Клапейрона–Менделеева.`**

Для одного моля любого газа это соотношение принимает вид: **pV=RT**

1.3. Изопроцессы

**Изопроцессы** — термодинамические процессы, во время которых количество вещества и один из параметров состояния: давление, объём, температура. При этом масса должна быть постоянной величиной.

**Изотермический процесс**– процесс перехода идеального газа из одного состояния в другое без изменения температуры. Закон, описывающий связь меду параметрами газа при таком процессе, называется закон Бойля-Мариотта в честь двух учёных, практически одновременно выведших его: англичанина Роберта Бойля и француза Эдма Мариотта

Выведем его: P \* V = v \* R \* T

А теперь учитывая: v = const и T = const

Получаем: P1 \* V1 = P2 \* V2 для любых различных состояний газа, или же просто:

 P \* V = const - закон Бойля-Мариотта

**Изобарный** **процесс**– процесс перехода идеального газа из одного состояния в другое при постоянном значении давления. Впервые такой процесс рассмотрел французский учёный Жозеф-Луи Гей-Люссак, по этой причине закон носит его имя.

Выведем его:

Запишем обычное уравнение состояния: P \* V = v \* R \* T

А теперь учитывая:  и 

Получаем: $\frac{V\_{1}}{T\_{1}}= \frac{V\_{2}}{T\_{2}}$  для любых различных состояний газа, или же просто:

$\frac{V}{T}=const$ - закон Гей-Люссака

**Изохорный** **процесс**– процесс перехода идеального газа из одного состояния в другое при постоянном значении объёма. Процесс рассмотрен впервые французом Жаком Шарлем, поэтому закон носит его имя.

Выведем его:

Снова запишем обычное уравнение состояния: 

А теперь учитывая: v = const и V = const

Получаем:   для любых различных состояний газа, или же просто:

$\frac{P}{T}=const$

$\frac{P\_{1}}{T\_{1}}= \frac{P\_{2}}{T\_{2}}= …$ - закон Шарля

**1.4. Столкновение молекул**

Прежде чем перейти к вычислениям, примем простейшую модель для молекул. Будем представлять их в виде упругих шариков.

**1.4.1. Центральное столкновение шариков**

Рассмотрим два сферических объекта с массами m1 и m2. Предположим, что эти шарики движутся без вращения по одной оси и испытывают центральное упругое соударение. В этом случае закон сохранения импульса запишется в виде:

m1v1i + m2v2i = m1v1 + m2v2

где v1i и v2i - начальные скорости каждого объекта, а v1 и v2 - их конечные скорости. Закон сохранения энергии записывается в виде:

m1v1i2 / 2 + m2v2i2 / 2 = m1v12/ 2 + m2v22/ 2

Закон сохранения импульса может быть преобразован следующим образом:

m1 (v1i - v1) = m2 (v2 - v2i)

Также преобразуем выражение для закона сохранения энергии

m1 (v1i2 - v12) = m2 (v22 - v2i2)

Если разница между начальной и конечной скоростями не равна нулю (то есть столкновение действительно произошло), мы можем разделить второе из двух последних уравнений на первое, что дает:

v1i + v1 = v2 + v2iили
v1i - v2i = v2 - v1

Другими словами,  в одномерном случае упругих столкновений относительная скорость движения объектов после столкновения равняется относительной скорости движения до столкновения.

Чтобы получить конечные скорости движения объектов через их начальные скорости и массы, нужно выразить v2 из последнего уравнения и подставить его в уравнение для закона сохранения импульса. Окончательно получаем:

v1 = v1i (m1 - m2) / (m1 + m2) + v2i (2 m2) / (m1 + m2)

Таким же способом находим выражение для  v2

v2 = v1i (2 m1) / (m1 + m2) + v2i (m2 - m1) / (m2 + m1)

Далее предположим, что сталкиваются объекты с одинаковой массой, т.е. m1=  m2 = m. В этом случае:

v1 = v1i (m - m) / (m + m) + v2i (2 m) / (m + m)
v2 = v1i (2 m) / (m + m) + v2i (m - m) / (m + m)

Окончательно получаем, что v1 = v2i и v2 = v1i

Это означает, что в случае центрального упругого соударения объектов с равными массами, они будут просто обмениваться скоростями. Если один из объектов до столкновения покоился, то после столкновения он остановится, а второй объект начнёт движение. При этом скорость движения второго объекта будет равна скорости первого объекта до столкновения.

**1.4.2. Нецентральное столкновение шариков**

Рассматривая в предыдущем разделе движение нескольких шаров на нитях, мы предполагали, что их скорости в момент столкновения направлены вдоль линии, соединяющей центры масс. Такие удары называются центральными. Картина соударения при нецентральном ударе будет совсем иной.  Здесь во время удара имеет место как приближение центров шаров друг к другу вследствие их деформации, так и скольжение поверхности одного шара по поверхности другого. Очевидно, что вследствие скольжения поверхностей возникнут силы трения, которые вместе с упругими силами взаимодействия определят изменение скорости шаров после удара. Кроме того, силы трения вызовут вращение шаров относительно их центров масс.

Если силы трения очень малы по сравнению с упругими силами, то действием сил трения можно пренебречь и в этом случае задача о нецентральном столкновении шаров решается достаточно просто. Действительно, соединим центры масс сталкивающихся шаров прямой и разложим скорость каждого шара на нормальную составляющую, направленную вдоль линии центров, и тангенсальную составляющую, перпендикулярную к ней. Так как согласно нашему предположению силы трения отсутствуют,  то тангенсальные силы во время столкновения не возникают и, следовательно, тангенсальные скорости шаров изменяться не будут. Нормальные же составляющие скорости после удара можно определить на основании закона сохранения количества движения и закона сохранения энергии таким же путем, как и при центральном ударе

# ГЛАВА 2. ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1. Язык программирования

Python – известный язык программирования, используемый как для разработки самостоятельных программ, так и для создания прикладных сценариев в самых разнообразных областях применения. Это мощный, переносимый, легкий в использовании и свободно распространяемый язык. Программисты, функционирующие в самых различных областях, считают, что уклон Python на эффективность разработки и высокое качество программного обеспечения предоставляет ему стратегическое преимущество, как в маленьких, так и в больших проектах.

Писать программу я буду на языке программирования Python. При написании программы я буду использовать библиотеку pygame.

**2.2. Разработка интерфейса программы**

 Интерфейс моей программы будет состоять из 2 окон. В первое окно – “Меню”, здесь пользователь должен будет ввести параметры, по которым будет работать программная модель. После ввода параметров, для запуска работы программы пользователю необходимо нажать кнопку “Старт”, после чего откроется второе окно. В окне меню присутствует кнопка “Инструкция”, при нажатии которой откроется файл \*.pdf в котором содержится подробная инструкция по использованию программы.



Рис 1. Окно меню

Второе окно – основная часть программы. В этом окне будет отображаться модель идеального газа и его текущие параметры.

Рис 2. Окно приложения

2.2. Разработка программного кода

Программа разработана на языке программирования python с использованием библиотек “tkinter”–для разработки интерфейса и“pygame” для отрисовки движения молекул.

В первую очередь нужно создать шарики, движение которых я буду программировать. Для этого я определил количество шариков и создал списки, в которых будут храниться все характеристики каждого шарика.

# Списки значений
kol\_mol = 30

rad\_mol = [] # радиус
x\_mol = [] # Х
y\_mol = [] # У
c\_mol = [] # Цвет
# вектора скоростей
dx = []
dy = []

Затем с помощью цикла я заполнил списки.

# Вводимзначения
for i in range(kol\_mol + 1):
rad\_mol.append(3)
v\_mol.append(random.randint(2, 5))
x\_mol.append(random.uniform(200, WIDTH - 500))
y\_mol.append(random.uniform(100, HEIGHT - 100))
c\_mol.append((191, 63, 63))
dx.append(random.random() % 10 \* ter \* 0.1)
dy.append(random.random() % 10 \* ter \* 0.1)

После определения характеристик шариков, я создал основной цикл работы программы, работающий до тех пор, пока переменная “start” имеет значение “True”.

while start:

for t in range(1, kol\_mol + 1):
x\_mol[t] += dx[t]
y\_mol[t] += dy[t]
pygame.draw.circle(screen, c\_mol[t], (x\_mol[t], y\_mol[t]), rad\_mol[t])
pygame.draw.rect(screen, (140, 169, 207), (600, 0, 400, 500))

pygame.time.delay(10)
pygame.display.update()

После того как я “научил” шарики двигаться, нужно было определить правила столкновения со стенами. Для этого в основном цикле создал второй цикл, который проверял каждый шарик на столкновение со стенами. Если шарик сталкивался с верхней стеной или нижней стеной, то вектор по оси X менялся на противоположный, если с правой или левой стеной, то на противоположный менялся вектор по оси Y.

forq in range(1, kol\_mol + 1):
 if (x\_mol[q] - rad\_mol[q] <= 0 or x\_mol[q] + rad\_mol[q] >= WIDTH - 400):
 dx[q] = -dx[q]
if (y\_mol[q] - rad\_mol[q] <= 0 or y\_mol[q] + rad\_mol[q] >= HEIGHT):
dy[q] = -dy[q]

Последним необходимым правилом была закономерность вычислений при столкновении самих шариков. Для этого я создал двойной цикл внутри основного. Затем проверял на столкновение. Если оно происходило, я изменял направление движения шариков в соответствии физическим описанием столкновения, описанного в теоретической части.

# Столкновения с другими шарами
forz in range(1, kol\_mol + 1): # i1 p2
forx in range(z + 1, kol\_mol + 1):

if(z == x): # проверка одинаковости
continue

del\_x = x\_mol[z] - x\_mol[x]
del\_y = y\_mol[z] - y\_mol[x]

os = (del\_x \*\* 2 + del\_y \*\* 2) \*\* 0.5
ifos == 0: os = 0.01

s = del\_x / os# sin
c = del\_y / os# cos

if(os<= rad\_mol[z] + rad\_mol[x]):
 Vn1 = dx[x] \* s + dy[x] \* c
 Vn2 = dx[z] \* s + dy[z] \* c

 Vt1 = -dx[x] \* c + dy[x] \* s
 Vt2 = -dx[z] \* c + dy[z] \* s

vr = Vn2
Vn2 = Vn1
Vn1 = vr

 dx[z] = Vn2 \* s - Vt2 \* c
dy[z] = Vn2 \* c + Vt2 \* s
 dx[x] = Vn1 \* s - Vt1 \* c
dy[x] = Vn1 \* c + Vt1 \* s

Полный код программы См. Приложение 1

2.3. Разработка инструкции пользователя

Программа разработана для ПК со следующими минимальными системными требованиями:

1. Операционная система Microsoft Windows XP и выше.

2. Оперативная память от 2 Гб.

3. Манипулятор компьютерная мышь и клавиатура.

Программа запускается двойным кликом мыши по файлу \*.exe

После запуска открывается окно “Меню”. Для работы программы необходимо ввести параметры, по которым будет построена модель.

После того как все поля будут заполнены, для запуска модели необходимо нажать кнопу “Старт”. После запуска откроется второе окно, в котором и будет отображаться модель идеального газа. Помимо модели во втором окне будут отображаться параметры состояния смоделированного идеального газа. Параметры газа будут описаны в единицах измерения СИ.

#

# ЗАКЛЮЧЕНИЕ

В результате выполнения работы мною была разработана программа моделирования идеального газа.

Программа позволяет неквалифицированному пользователю:

* осуществлять ввод исходных данных с клавиатуры;
* отслеживать изменение поведения молекул при разных параметрах;
* рассчитывать количество молекул;
* рассчитывать концентрацию молекул;

Созданная программа позволяет производить вычисления для любых наборов исходных данных.

Программа работает в режиме диалога с пользователем. Пользователь выполняет необходимые действия путем нажатия соответствующей кнопки.

Таким образом, я достиг поставленной цели моего проекта.

Во время работы над проектом мною был изучен теоретический материал по поведению идеального газа при разных условиях и применены основные приемы программирования на языке python.

# СПИСОК ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Г. Я. Мякишев, Б. Б. Буховцев, Н. Н. Сотский Физика. - 6 изд. - Москва: Просвещение, 2019. - 426 с.
2. Идеальный газ // skysmart URL: https://skysmart.ru/articles/physics/idealnyj-gaz (дата обращения: 02.11.2021).
3. Марк Лутц Программирование на Python. Том 1. - 4 изд. - Санкт-Петербург: Символ-Плюс, 2011. - 992 с.

# ПРИЛОЖЕНИЯ

**Приложение 1**

fromtkinter import \*

import subprocess

import sys

import pygame

import random

import math

pygame.init()

menu = Tk()

menu.geometry('700x600')

menu.title('Меню')

menu.resizable(False, False)

menu.image = PhotoImage(file='fon.png')

def validate(new\_value):

returnnew\_value == "" or new\_value.isnumeric()

vcmd = (menu.register(validate), '%P')

# ---------------------------------------------------------------------------------------------------------------------------

def click():

 WIDTH = 600 + 400

 HEIGHT = 500

 FPS = 100

 n\_mass1 = Label(menu, text=" ", font=("Arial Bold", 10), fg='#ff0000')

 n\_mass1.place(x=150, y=220 + we)

 n\_mass2 = Label(menu, text=" ", font=("Arial Bold", 10), fg='#ff0000')

 n\_mass2.place(x=150, y=120 + we)

 n\_mass3 = Label(menu, text=" ", font=("Arial Bold", 10), fg='#ff0000')

 n\_mass3.place(x=150, y=170 + we)

 n\_mass1.config(text=' ')

 n\_mass2.config(text=' ')

 n\_mass3.config(text=' ')

 # Функции

defrast(x1, y1, x2, y2):

 r = (((x1 - x2) \*\* 2) + ((y1 - y2) \*\* 2)) \*\* 0.5

return (r)

pygame.init()

pygame.mixer.init()

screen = pygame.display.set\_mode((WIDTH, HEIGHT))

pygame.display.set\_caption("Движение молекул")

clock = pygame.time.Clock()

 # Списки значений

kol\_mol = 30

rad\_mol = [] #радиус

x\_mol = [] # Х

 y\_mol = [] # У

v\_mol = [] # Скорость

c\_mol = [] # Цвет

 # вектора скоростей

dx = []

dy = []

sten = []

shar = []

font = pygame.font.Font('freesansbold.ttf', 18) # Шрифт

start = True

ifmasa\_entry.get() == '':

 n\_mass1.config(text = '\*Введитезначение')

start = False

exit()

ifterm.get() == '':

 n\_mass2.config(text = '\*Введите значение')

start = False

exit()

ifob\_entry.get() == '':

 n\_mass3.config(text = '\*Введите значение')

start = False

exit()

ter = float(term.get())

mass = float(masa\_entry.get()) / 1000

ob = float(ob\_entry.get()) / 1000

print(mass)

 # Защитаотдурака

if mass > 0.1:

 n\_mass1.config(text = '\*Слишком большое значение')

start = False

if mass <= 0:

 n\_mass1.config(text = '\*Слишкоммаленькоезначение')

start = False

ifter> 1000:

 n\_mass2.config(text = '\*Слишкомбольшоезначение')

start = False

ifter< 0:

 n\_mass2.config(text = '\*Слишкоммаленькоезначение')

start = False

ifob> 0.1:

 n\_mass3.config(text = '\*Слишкомбольшоезначение')

start = False

ifob<= 0:

 n\_mass2.config(text = '\*Слишкоммаленькоезначение')

start = False

# ifob

# ifter

 # Расчеты

 N = 6.02 \* (mass / 16)

 n = N / ob

 # Вводимзначения

fori in range(kol\_mol + 1):

rad\_mol.append(3)

v\_mol.append(random.randint(2, 5))

x\_mol.append(random.uniform(200, WIDTH - 500))

y\_mol.append(random.uniform(100, HEIGHT - 100))

c\_mol.append((191, 63, 63))

dx.append(random.random() % 10 \* ter \* 0.1)

dy.append(random.random() % 10 \* ter \* 0.1)

sten.append(0)

shar.append(0)

c\_mol[1] = [225, 0, 0]

ter = float(term.get())

mass = float(masa\_entry.get()) / 1000

ob = float(ob\_entry.get()) / 1000

print(mass)

 # Расчеты

dav = (mass \* 8.14 \* ter) / (0.016 \* ob)

 N = 6.02 \* (mass / 16)

 n = N / ob

ter = str(ter)

dav = str(dav)

mass = str(mass)

ob = str(ob)

 N = str(N)

 n = str(n)

print(N)

while start:

clock.tick(FPS)

ter = float(ter)

dav = float(dav)

mass = float(mass)

ob = float(ob)

 N = float(N)

 n = float(n)

for event in pygame.event.get():

ifevent.type == pygame.QUIT:

start = False

elifevent.type == pygame.KEYDOWN:

ifevent.key == pygame.K\_LEFT:

ter -= 1

elifevent.key == pygame.K\_RIGHT:

ter += 1

elifevent.key == pygame.K\_UP:

dav += 1

elifevent.key == pygame.K\_DOWN:

dav -= 1

ter = str(ter)

dav = str(dav)

mass = str(mass)

ob = str(ob)

 N = str(N)

 n = str(n)

 t\_term1 = font.render("Температура:", 1, (225, 225, 0), (0, 0, 225))

 t\_dav1 = font.render("Давление:", 1, (225, 225, 0), (0, 0, 225))

 t\_term2 = font.render(ter, 1, (225, 225, 0), (0, 0, 225))

 t\_dav2 = font.render(dav, 1, (225, 225, 0), (0, 0, 225))

 t\_mas1 = font.render("Массагаза:", 1, (225, 225, 0), (0, 0, 225))

 t\_kol1 = font.render("Числомолекул:", 1, (225, 225, 0), (0, 0, 225))

 t\_mas2 = font.render(mass, 1, (225, 225, 0), (0, 0, 225))

 t\_kol2 = font.render(N, 1, (225, 225, 0), (0, 0, 225))

 t\_kon1 = font.render("Концентрация:", 1, (225, 225, 0), (0, 0, 225))

t\_kon2 = font.render(n, 1, (225, 225, 0), (0, 0, 225))

 t\_t1 = t\_term1.get\_rect()

 t\_t1.center = (400 // 2, 400 // 2)

 t\_t2 = t\_term2.get\_rect()

 t\_t2.center = (400 // 2, 400 // 2)

 t\_d1 = t\_dav1.get\_rect()

 t\_d1.center = (400 // 2, 400 // 2)

 t\_d2 = t\_dav2.get\_rect()

 t\_d2.center = (400 // 2, 400 // 2)

 t\_d2 = t\_dav2.get\_rect()

 t\_d2.center = (400 // 2, 400 // 2)

 t\_m1 = t\_mas1.get\_rect()

 t\_m1.center = (400 // 2, 400 // 2)

 t\_m2 = t\_mas2.get\_rect()

 t\_m2.center = (400 // 2, 400 // 2)

 t\_k1 = t\_kol1.get\_rect()

 t\_k1.center = (400 // 2, 400 // 2)

 t\_k2 = t\_kol2.get\_rect()

 t\_k2.center = (400 // 2, 400 // 2)

 t\_n1 = t\_kon1.get\_rect()

 t\_n1.center = (400 // 2, 400 // 2)

 t\_n2 = t\_kon1.get\_rect()

 t\_n2.center = (400 // 2, 400 // 2)

screen.fill((200, 214, 230))

for t in range(1, kol\_mol + 1):

x\_mol[t] += dx[t]

y\_mol[t] += dy[t]

pygame.draw.circle(screen, c\_mol[t], (x\_mol[t], y\_mol[t]), rad\_mol[t])

pygame.draw.rect(screen, (140, 169, 207), (600, 0, 400, 500))

 # текстсправа

screen.blit(t\_term1, (650, 10))

screen.blit(t\_term2, (850, 10))

screen.blit(t\_dav1, (650, 30))

screen.blit(t\_dav2, (850, 30))

screen.blit(t\_mas1, (650, 50))

screen.blit(t\_mas2, (850, 50))

screen.blit(t\_kol1, (650, 70))

screen.blit(t\_kol2, (850, 70))

screen.blit(t\_kon1, (650, 90))

screen.blit(t\_kon2, (850, 90))

pygame.time.delay(10)

pygame.display.update()

# Столкновения со стенами

forqinrange(1, kol\_mol + 1):

if (sten[q] == 1):

sten[q] = 0

continue

if (x\_mol[q] - rad\_mol[q] <= 0 or x\_mol[q] + rad\_mol[q] >= WIDTH - 400):

dx[q] = -dx[q]

if (y\_mol[q] - rad\_mol[q] <= 0 or y\_mol[q] + rad\_mol[q] >= HEIGHT):

dy[q] = -dy[q]

ifx\_mol[q] + rad\_mol[q] < -4:

print('стена')

x\_mol[q] += 12

ifx\_mol[q] + rad\_mol[q] > WIDTH + 10:

x\_mol[q] -= 12

print('стена')

ify\_mol[q] + rad\_mol[q] < -4:

y\_mol[q] += 12

print('стена')

ify\_mol[q] + rad\_mol[q] > HEIGHT + 4:

print('стена')

y\_mol[q] -= 12

sten[q] += 1

tr = 2

 # Столкновения с другими шарами

for z in range(1, kol\_mol + 1): # i1 p2

for x in range(z + 1, kol\_mol + 1):

if (z == x): # проверка одинаковости

continue

del\_x = x\_mol[z] - x\_mol[x]

del\_y = y\_mol[z] - y\_mol[x]

os = (del\_x \*\* 2 + del\_y \*\* 2) \*\* 0.5

ifos == 0: os = 0.01

 s = del\_x / os # sin

 c = del\_y / os # cos

if (os<= rad\_mol[z] + rad\_mol[x]):

print("есть", os)

 Vn1 = dx[x] \* s + dy[x] \* c

 Vn2 = dx[z] \* s + dy[z] \* c

print("Vn1 =", Vn1)

print("Vn2 =", Vn2)

 Vt1 = -dx[x] \* c + dy[x] \* s

 Vt2 = -dx[z] \* c + dy[z] \* s

print("Vt1 =", Vt1)

print("Vt2 =", Vt2)

vr = Vn2

 Vn2 = Vn1

 Vn1 = vr

dx[z] = Vn2 \* s - Vt2 \* c

dy[z] = Vn2 \* c + Vt2 \* s

dx[x] = Vn1 \* s - Vt1 \* c

dy[x] = Vn1 \* c + Vt1 \* s

pygame.quit()

# --------------------------------------------------------------------------------------------------------------------------------

we = 100

title = Label(menu, text="Введитепараметрыработыпрограммы", font=("Arial Bold", 18))

title.place(x=130, y=20 + we)

s\_term = Label(menu, text="Температура", font=("Arial Bold", 15))

s\_term.place(x=20, y=95 + we)

s\_rm = Label(menu, text="Кельвенов", font=("Arial Bold", 15))

s\_rm.place(x=340, y=95 + we)

term = Entry(menu, validatecommand=vcmd, validate='key', font=("Arial Bold", 13))

term.place(x=150, y=100 + we)

s\_ob\_entry = Label(menu, text="Объем", font=("Arial Bold", 15))

s\_ob\_entry.place(x=20, y=145 + we)

ob\_entry = Entry(menu, validatecommand=vcmd, validate='key', font=("Arial Bold", 13))

ob\_entry.place(x=150, y=150 + we)

s\_oby = Label(menu, text="Литров", font=("Arial Bold", 15))

s\_oby.place(x=340, y=145 + we)

s\_masa = Label(menu, text="Масса", font=("Arial Bold", 15))

s\_masa.place(x=20, y=195 + we)

masa\_entry = Entry(menu, validatecommand=vcmd, validate='key', font=("Arial Bold", 13))

masa\_entry.place(x=150, y=200 + we)

s\_as = Label(menu, text="Грамм", font=("Arial Bold", 15))

s\_as.place(x=340, y=195 + we)

st = Button(menu, text='Старт', font=("Arial Bold", 18), width=20, command=click)

st.place(x=385, y=300 + we)

en = Button(menu, text='Инструкция', font=("Arial Bold", 18), width=20, command=click)

en.place(x=20, y=300 + we)

menu.mainloop()