**МУНИЦИПАЛЬНОЕ АВТОНОМНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**СРЕДНЯЯ ШКОЛА № 59 «ПЕРСПЕКТИВА»**

Проектно – исследовательская работа

**Тема: «**Модель физических взаимодействий объектов

в двумерном пространстве**»**

Выполнил учащийся 10Б класса

Бабкин Максим Валентинович

Руководитель: Теряник Елена Анатольевна

Липецк, 2023г

**Содержание**

Введение………………………………………………………………………..3

Глава 1. Теоретическая часть………………………………………………….4

* 1. Физическое тело и его свойства ……………………………………4
  2. Взаимодействия тел………………………………………………….5
  3. Python…………………………………………………………………6

Глава 2. Практическая часть……………………………………………..........7

2.1. Создание модели………………………………………………..........7

Заключение………………………………………………………………........10

Список используемой литературы…………………………………………..11

Приложение…………………………………………………………………...12

**Введение.**

**Актуальность:** довольно сложно представить, как взаимодействуют тела, как они после столкновения двигаются, как деформируются или узнать, как происходят ситуации, которые возможны лишь в теории

**Цель:** создать модель физических взаимодействий объектов в двумерном пространстве.

**Задачи:**

1. Проанализировать текстовый и иллюстративный материал по данной теме.
2. Написать код.
3. Протестировать код.

**Гипотеза:** моя модель поможет понять, как взаимодействуют объекты

**Объект:** информатика физика

**Предмет:** код.

**Методы:** анализ, исследование, разработка кода

**Глава 1. Теория**

* 1. **Физическое тело и его свойства.**

Телом в физике считается материальный объект, отделенный конкретными собственными границами от других тел и характеризующийся конкретным объемом, массой, формой. Это понятие используется для упрощения математических расчетов с целью определения качественных или количественных параметров процессов, в которых участвует данный объект. Пример такого тела может послужить человек.

Объём – это количественная характеристика пространства, занимаемого телом или веществом. Объём тела определяется его формой и линейными размерами. (т.к. моя модель находится в двумерном пространстве, то так такого объёма у тел нет). Объём тела можно найти как отношение массы объекта на его плотность.

Масса – это Масса величина, показывающая, как тело сопротивляется изменению скорости (насколько оно инертно) и как участвует в гравитационном взаимодействии. Чем больше масса объекта, тем сильней он притягивает другой объект.

Так же у тела есть такой параметр как коэффициент трения об поверхность. Он представляет собой безразмерную скалярную величину, которая равна отношению силы трения между двумя телами и силы, прижимающей их друг к другу.

Так как тело находится в пространстве то оно имеет свои координаты. Эти параметры обозначают, где объект находится в данный момент. Так как мы ощущаем только 3 пространства то и координат 3, обычно их обозначают как x y z, но моей модели этих их всего 2, то есть нет перемещения по координате z в виду отсутствия 3-го измерения.

Тело в пространстве имеет потенциальную и кинетическую энергии.

Потенциальная энергия – это энергия взаимодействия. Иначе говоря, если тело за счёт взаимодействия с другим телом может совершить некоторую работу, то оно будет обладать потенциальной энергией. Кинетическая энергия – это энергия движения тела.

**1.2 Взаимодействие тел.**

Взаимодействие тел — это воздействие тел друг на друга. Тела могут взаимодействовать при помощи механического воздействия, инерции, свойств вещества электричества и прочих факторов.

Механическое воздействие – это воздействие, в результате которого могут происходить изменения скоростей тел без изменения их химического состава. Механическое воздействие может происходить при соприкосновении тел или на расстоянии – в результате действия электромагнитных или гравитационных полей.

* 1. **Python**

Python – это высокоуровневый язык программирования общего назначения с динамической типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным  в том плане, что всё является объектами. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Сам же язык известен как интерпретируемый и используется в том числе для написания скриптов. Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках.

Pygame – библиотека python созданная для написания мультимедийных приложений основанная на структуре SDL. Данная библиотека позволяет создавать окна в которых можно размещать интересующие пользователя канвасы(картинки находящиеся в памяти pygame), а так же создаёт главный цикл приложения в котором можно отлавливать ивенты(нажатие клавиш клавиатуры или кнопок мыши) и производить дальнейшие действия исходя из заданного условия.

Pymunk – это библиотека python которая помогает производить вычисления физических свойств объектов. Данная библиотека даёт возможность создавать объекты в окне pygame имеющие физические свойства.

Json – это популярный формат текстовых данных, который используется для обмена данными в современных веб - и мобильных приложениях.

**Глава 2. Практическая часть.**

**2.1 Создание модели.**

Создаю главный файл main, в котором я делаю следующие шаги:

* Создаю физические константы, разрешение монитора
* Создаю функцию инициализации окна и запуска
* Создаю нужные объекты классов
* Запускаю главный цикл модели в которой происходят все события поля

Код главного цикла main (приложение 2)

Далее создаю класс objects\_fun в котором реализованы следующие функции:

* Поиск объекта
* Создание границ
* Сбор объектов
* Открытие окна настроек объекта
* Поворот объекта на колёсико мыши
* Поворот объекта на 90 градусов
* Поворот объекта на 45 градусов
* Создание связей между объектами
* Сохранение поля
* Загрузка поля
* Удаление всех объектов
* Удаление выбранного объекта
* Остановка всех объектов
* Отмена остановки всех объектов
* Добавление объекта
* Создание четырёхугольника
* Создание круга
* Получение информации об объекте
* Изменение объекта

Часть кода objects\_fun (приложение 3)

Далее создаю класс world в котором реализованы следующие функции:

* Поймать объект
* Отпустить объект
* Движение объекта, когда он пойман
* Отрисовка всех объектов и информации на поле

Часть кода world (приложение 4)

Далее создаю класс menu в котором реализованы следующие функции:

* Инициализация
* Сохранение стандартных типов объектов
* Сохранение настроек
* Закрытие окон настроек стандартных объектов
* Открытие всех окон меню

Часть кода menu (Приложение 5)

Схема работы модели (приложение 1)

**Заключение.**

По итогам работы модель создана и способна правдоподобно отображать как взаимодействуют объекты в двумерном пространстве, а также создавать и обрабатывать те случаи, которые возможны лишь в теории.

**Список литературы.**

Python.org

<http://www.pymunk.org/en/latest/index.html>

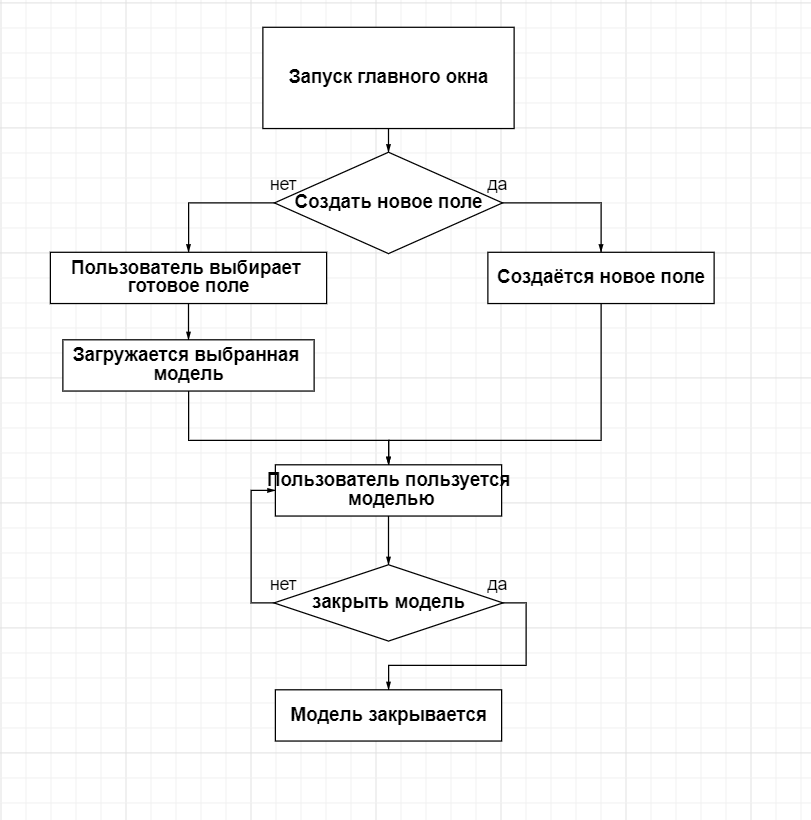
<https://pygame-menu.readthedocs.io/en/4.1.3/index.html>

<https://pypi.org/project/pygame/>

Физика 10 класс О. Ф. Кабардин

**Приложение.**

Приложение 1:



Приложение 2:

while run:  
 mouse\_position = pg.mouse.get\_pos()  
 for event in pg.event.get():  
 if event.type == pg.QUIT:  
 run = False  
 break  
 if event.type == pg.MOUSEBUTTONDOWN:  
 if event.button == 3:  
 data = OCreator.openStandartS(type\_o)  
  
 OCreator.add\_obj(mouse\_position, type\_o, space, data[0], data[6], data[3], data[1], data[2],  
 data[-1] \* 3.1415926535 / 180, data[5])  
 elif event.button == 1:  
 if on:  
 world.resume\_object()  
 on = False  
 elif on == False and world.is\_shape(space, mouse\_position):  
 world.pick\_object(world.is\_shape(space, mouse\_position))  
 on = True  
 elif event.button == 4 and on:  
 OCreator.rotate\_object(OCreator.searchf(space, mouse\_position), 1)  
 elif event.button == 5 and on:  
 OCreator.rotate\_object(OCreator.searchf(space, mouse\_position), -1)  
 elif event.button == 4:  
 if type\_o == "квадрат":  
 type\_o = "круг"  
 else:  
 type\_o = "квадрат"  
 elif event.button == 5:  
 if type\_j == "пружина":  
 type\_j = "нить"  
 else:  
 type\_j = "пружина"  
 if event.type == pg.KEYDOWN:  
 if event.key == pg.K\_SPACE:  
 if PAUSE:  
 OCreator.resume\_all\_objects()  
 PAUSE = False  
 else:  
 OCreator.stop\_all\_objects()  
 PAUSE = True  
 if event.key == 115:  
 if on:  
 menu.SettingsMenu.enable()  
 menu.show\_settings(OCreator.get\_info(space, mouse\_position))  
 menu.SettingsMenu.mainloop(window)  
 menu.SettingsMenu.remove\_widget(menu.SettingsMenu.\_widgets[-1])  
 OCreator.edit\_object(space, mouse\_position)  
 world.resume\_object()  
 on = False  
 else:  
 menu.SaveMenu.mainloop(window)  
 name\_file = menu.save\_field()  
 if name\_file == "":  
 for i in sample("0123456789", k=5):  
 name\_file += str(i)  
 OCreator.save\_field(name\_file)  
 run = False  
 if event.key == 122:  
 menu.showStandartS('квадрат')  
 menu.StandartObjectS.enable()  
 menu.StandartObjectS.mainloop(window)  
 menu.saveStandartS('StandartS.json')  
 menu.StandartObjectS.remove\_widget(menu.StandartObjectS.\_widgets[-1])  
 if event.key == 120:  
 menu.showStandartS('круг')  
 menu.StandartObjectS.enable()  
 menu.StandartObjectS.mainloop(window)  
 menu.saveStandartS('StandartB.json')  
 menu.StandartObjectS.remove\_widget(menu.StandartObjectS.\_widgets[-1])  
 if event.key == 118:  
 menu.StandartConnection.enable()  
 menu.showStandartJ()  
 menu.StandartConnection.mainloop(window)  
 menu.saveStandartConnection()  
 if event.key == 100 and on:  
 OCreator.set\_90d\_object((OCreator.searchf(space, mouse\_position)))  
 elif event.key == pg.K\_DELETE:  
 if on:  
 OCreator.delete\_object(space, OCreator.searchf(space, mouse\_position))  
 on = False  
 elif event.key == pg.K\_TAB:  
 OCreator.delete\_all\_objects(space)  
 world.resume\_object()  
 on = False  
 elif event.key == pg.K\_LSHIFT and on:  
 OCreator.rotate\_object\_45(OCreator.searchf(space, mouse\_position))  
 elif event.key == 99:  
 OCreator.collect\_shapes(space, mouse\_position)  
 elif event.key == 106:  
 OCreator.connect\_shapes(space, type\_j)  
 elif event.key == 113:  
 run = False  
 world.move\_founded\_object(mouse\_position)  
 world.draw(space, window, draw\_options, type\_o, clock, type\_j)  
 space.step(1 / self.fps)  
 clock.tick(self.fps)  
pg.quit()

Приложение 3:

def load\_field(self, space, name):  
 if os.path.isfile(f"fields/{name}.json"):  
 with open(f"fields/{name}.json", 'r') as field:  
 field = load(field)  
 for i in field:  
 data = field[i]  
 if 'пружина' in data:  
 data2 = data['пружина']  
 SBODY = []  
 SSHAPE = []  
 for j in range(0, 2):  
 data = data2[str(j)]  
 body = pm.Body()  
 body.position = data['position']  
 if data['shape'] == "круг":  
 shape = pm.Circle(body, data['args'][0])  
 elif data['shape'] == 'квадрат':  
 shape = pm.Poly.create\_box(body, (data['args'][0] \* 2, data['args'][1] \* 2))  
 shape.elasticity = data['elasticity']  
 SSHAPE.append(shape)  
 body.body\_type = data['body\_type']  
 shape.color = data['color']  
 shape.friction = data['friction']  
 shape.mass = data['mass']  
 body.\_set\_angle(data['angle'])  
 space.add(body, shape)  
 self.bodyO.append(body)  
 self.shapeO.append(shape)  
 self.objects.append((shape, body))  
 SBODY.append(body)  
 joint = pymunk.DampedSpring(SBODY[0], SBODY[1], (0, 0), (0, 0), 10, field[i]['k'], 1)  
 self.sshapes.append(  
 [joint, (SSHAPE[0], self.bodyO[self.shapeO.index(SSHAPE[0])]),  
 (SSHAPE[1], self.bodyO[self.shapeO.index(SSHAPE[1])]), "пружина"])  
 space.add(joint)  
 self.objects.append(joint)  
 else:  
 body = pm.Body()  
 body.position = data['position']  
 if data['shape'] == "круг":  
 shape = pm.Circle(body, data['args'][0])  
 elif data['shape'] == 'квадрат':  
 shape = pm.Poly.create\_box(body, (data['args'][0] \* 2, data['args'][1] \* 2))  
 shape.elasticity = data['elasticity']  
 body.body\_type = data['body\_type']  
 shape.color = data['color']  
 shape.friction = data['friction']  
 shape.mass = data['mass']  
 body.\_set\_angle(data['angle'])  
 space.add(body, shape)  
 self.bodyO.append(body)  
 self.shapeO.append(shape)  
 self.objects.append((shape, body))  
 self.objects\_for\_save.append((shape, body))  
  
def delete\_all\_objects(self, space):  
 for i in self.objects:  
 try:  
 space.remove(i[0], i[1])  
 except:  
 space.remove(i)  
 self.objects = []  
 self.objects\_for\_save = []  
 self.cshapes = [None, None]  
 self.flag = True  
  
def stop\_all\_objects(self):  
 for i in self.objects:  
 if i != 'нить' and i != "пружина":  
 i[1].body\_type = pm.Body.STATIC  
  
def resume\_all\_objects(self):  
 for i in self.objects:  
 if i != 'нить' and i != "пружина":  
 i[1].body\_type = pm.Body.DYNAMIC

Приложение 4:

def pick\_object(self, search):  
 self.shape\_founded = search.shape  
  
def resume\_object(self):  
 self.shape\_founded = None  
  
def move\_founded\_object(self, pos):  
 if self.shape\_founded is not None:  
 if pos[1] < self.height:  
 self.shape\_founded.body.position = pos  
 self.shape\_founded.body.velocity = 0, 0  
  
  
def draw(self, space, window, draw\_options, type\_o,clock, type\_j):  
 window.fill("gray")  
 font1 = pg.font.SysFont('standart.ttf', 25)  
 fps = font1.render('{0:.0f}'.format(clock.get\_fps()) + " fps", True, [50, 50, 50])  
 text1 = font1.render(f'{type\_o}', True, (100, 100, 100))  
 text2 = font1.render(f'{type\_j}', True, (100, 100, 100))  
  
 textRect1 = text1.get\_rect()  
 textRect1.center = (self.width\*0.05 + 40, 50)  
  
 textRect2 = text2.get\_rect()  
 textRect2.center = (self.width \* 0.95 - 40, 50)  
  
 tfps = fps.get\_rect()  
 tfps.center = (self.width\*0.5, 50)  
  
 window.blit(text1, textRect1)  
 window.blit(fps, tfps)  
 window.blit(text2, textRect2)  
 space.debug\_draw(draw\_options)  
 pg.display.update()

Приложение 5:

def save\_settings(self):  
 color = [int(self.SettingsMenu.\_widgets[3].get\_value()), int(self.SettingsMenu.\_widgets[4].get\_value()),  
 int(self.SettingsMenu.\_widgets[5].get\_value()), int(self.SettingsMenu.\_widgets[6].get\_value())]  
 f = str(self.objects[0])[15:str(self.objects[0]).index(' ')]  
 if f == "Circle":  
 t = "круг"  
 size = float(self.SettingsMenu.\_widgets[7].get\_value())  
 elif f == "Poly":  
 t = "квадрат"  
 size = [float(self.SettingsMenu.\_widgets[7].get\_value()), float(self.SettingsMenu.\_widgets[8].get\_value())]  
 btype = self.SettingsMenu.\_widgets[-1].get\_value()[1]  
 d = {0: {  
 "mass": float(self.SettingsMenu.\_widgets[0].get\_value()),  
 "friction": float(self.SettingsMenu.\_widgets[1].get\_value()),  
 "elasticity": float(self.SettingsMenu.\_widgets[2].get\_value()),  
 "color": color,  
 'position': [float(self.SettingsMenu.\_widgets[10].get\_value()),  
 float(self.SettingsMenu.\_widgets[11].get\_value())],  
 'shape': t,  
 'body\_type': btype,  
 'args': size,  
 'angle': float(self.SettingsMenu.\_widgets[9].get\_value()) \* 3.1415926535 / 180  
 }}  
 data = json.dumps(d)  
 data = json.loads(str(data))  
 with open('object.json', "w") as file:  
 json.dump(data, file, indent=4, ensure\_ascii=False)  
 self.SettingsMenu.disable()  
  
def show\_settings(self, objects):  
 self.objects = objects  
 f = str(objects[0])[15:str(objects[0]).index(' ')]  
 if f == "Circle":  
 size = objects[0].radius  
 h = 0  
 self.SettingsMenu.\_widgets[7].set\_title("Радиус: ")  
 self.SettingsMenu.\_widgets[8].hide()  
 elif f == "Poly":  
 self.SettingsMenu.\_widgets[7].set\_title("Длина: ")  
 self.SettingsMenu.\_widgets[8].show()  
 size = abs(objects[0].get\_vertices()[0][0])  
 h = abs(objects[0].get\_vertices()[0][1])  
 self.SettingsMenu.\_widgets[0].set\_value(str(objects[0].mass))  
 self.SettingsMenu.\_widgets[1].set\_value(str(objects[0].friction))  
 self.SettingsMenu.\_widgets[2].set\_value(str(objects[0].elasticity))  
 self.SettingsMenu.\_widgets[3].set\_value(str(objects[0].color[0]))  
 self.SettingsMenu.\_widgets[4].set\_value(str(objects[0].color[1]))  
 self.SettingsMenu.\_widgets[5].set\_value(str(objects[0].color[2]))  
 self.SettingsMenu.\_widgets[6].set\_value(str(objects[0].color[3]))  
 self.SettingsMenu.\_widgets[7].set\_value(str(size))  
 self.SettingsMenu.\_widgets[8].set\_value(str(h))  
 self.SettingsMenu.\_widgets[9].set\_value(  
 str(float(str(objects[1].\_get\_angle())[0:10]) \* 180 / 3.1415926535)[0:7])  
 self.SettingsMenu.\_widgets[10].set\_value(str(objects[1].position[0])[0:6])  
 self.SettingsMenu.\_widgets[11].set\_value(str(objects[1].position[1])[0:6])  
  
 if objects[1].body\_type == 0:  
 self.SettingsMenu.add.selector('Состояние:', [("Динамическое", 0), ("Статичное", 1)])  
 else:  
 self.SettingsMenu.add.selector('Состояние:', [("Статичное", 0), ("Динамическое", 1)])