**Физические симуляции**

Для проекта будут использоваться библиотеки PyGame и PyMunk(pymunk.pygame\_util).

Ссылка на git проекта ([E7425/Physics-simulations (github.com)](https://github.com/E7425/Physics-simulations/tree/main))

Работа будет проводится Егором(Тимлид) и Кириллом (неТимлид)

Каждый будет делать примерно равно кол-во работы. Распределение может измениться по ходу работы, но скорее всего Егор будет делать

Главный файл, столкновение тел, движение планет и силу Архимеда. Кирилл – сила трения, преломления луча, энергия и реализацию окон для ввода и паузы.

Проект будет состоять из 8 основных частей:   
главного меню и 7 физических симуляций.

1. **Главное меню**

Выглядит следующим образом

Преломление луча в разных средах

Движение планет по орбитам

Потенциальная и Кинетическая энергия

Сила Архимеда

Абсолютно неупругое столкновение тел

Абсолютно упругое столкновение тел

Сила трения

Код главного меню и непосредственно главного цикла находится в файле main.py. Основные элементы файла – функция main и класс MenuButton.

Кнопок-экземпляров класса будет 7. 2 из них будут вызывать функцию из файла collision с разными аргументами. Остальные 5 кнопок будут при нажатии вызывать каждый из 5 оставшихся файлов. (planets, archimeds, retraction, energy, friction).

В этих файлах будут использоваться экземпляры класса InputVal, который написан в файле input.py.

Функция main вызывается при запуске программы. В ней находиться основной цикл, а также создаются и размещаются кнопки – экземпляры класса **MenuButton**.

Данный класс имеет следующий **атрибуты**:

x – координата левого верхнего угла кнопки по оси x.

y – координата левого верхнего угла кнопки по оси y.

width – ширина(длинна) кнопки

height – высота кнопки

func – функция, которую будет вызывать кнопка, при нажатии на неё

text – текст кнопки.

**Методы**:

render\_button – рисует кнопку и её текст на экране.

check\_clicked – принимает позицию мыши, при ее нажатии. Вызывает self.func если нажатие было совершено по кнопке.

В большинстве последующих модулей будет использоваться метод pause, ставящий симуляцию на паузу.

В описание дальнейших модулей понадобится поле для ввода.

Оно будет реализовано в файле input.py

**2.Поле для ввода**

Файл input.py содержит класс InputVal.

**Атрибуты**:   
rect – Объект типа rect, создается на основе данных, переданных при создании экземпляра класса.

mx\_sym – максимальное кол-во символов, которое можно ввести.

color – цвет текста и рамки

text – текст, который нужно отобразить

text\_surface - отображающийся текст

font – шрифт, значение по умолчанию (None, 28)

active – активна ли кнопка в данный момент (Начальное значение - False)

**Методы:**

render\_input – принимает surface (экран), отображает поле для ввода на surface.

event\_handler – принимает event – событие, в зависимости от события либо стирает символ, либо добавляет введенный символ, либо возвращает напечатанный текст.

**3. Cтолкновение тел**

Реализовано в файле collision.py.

Содержит функции create\_square и collision\_simulation.

collision\_simulation принимает аргумент elastic(по умолчанию равен 1)

Содержит основной цикл отрисовки.

Будет иметь 4 поля для ввода (Экземпляры класса InputVal).

Вводятся масса и скорость двух объектов. На основе этих данных создаются два кубика с помощью функции create\_square.

Функция принимает space(пространство PyMunk), pos( координаты создаваемого кубика), mass, speed, elastic(по умолчанию 1).

Во время симуляции скорости тел будут выводиться в верхней части экрана.

Будет так же кнопка для удаления кубиков, а также кнопка для показа *справочной информации*\* (Экземпляры класса menuButton)

\*Описание принципа работы симуляции, а также формулы.

**4.Сила Архимеда**

Реализовано в файле archimedes.py

Содержит основную функцию archimedes\_simulation, также функции get\_ archimedes и get\_gravity для получения силы Архимеда и силы тяжести соответственно. Они будут выводиться на экран и на их основе определяться координата объекта.

Также функция get\_pos, которая получается силу тяжести. А также силу Архимеда, плотности тела и жидкости, а также объем тела.

Эти значение вводятся с помощью полей для ввода (Экземпляры класса InputVal).

Функция render\_object будет рисовать объект на экране по координатам, полученным из функции get\_pos.

Функция render\_vessel рисует на экране сосуд с жидкостью (прямоугольник с голубой заливкой) на фиксированных координатах.

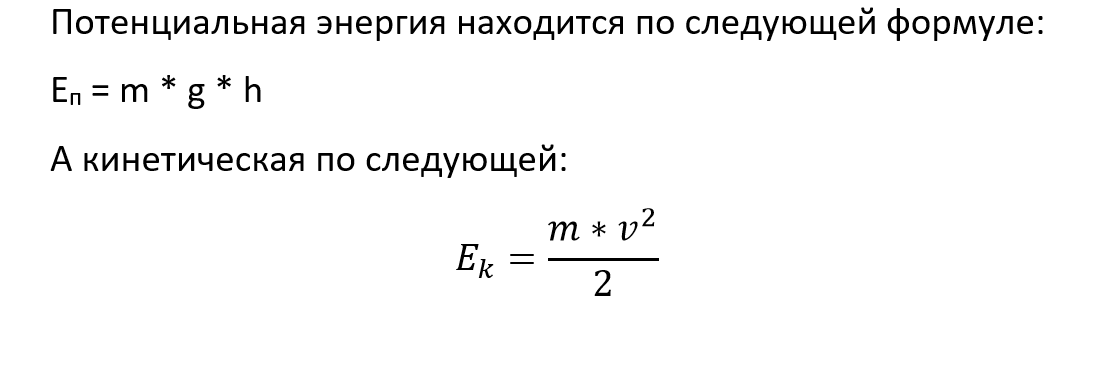
Будет также кнопка для получения справочной информации.

**5.Кинетическая и потенциальная энергия.**

Будет реализовано в файле energy.py.

(В данном разделе физическая симуляции будет проводиться вручную без библиотеки PyMunk для бОльшей точности)

Будут основная функция falling\_simulation, функции get\_kinetic и get\_potential для получения кинетический и потенциальной энергии соответственно. Они будут выводиться на экран.



Функции move и render\_object нужны для изменения координат объекта и отрисовки объекта соответственно.

Как обычно будет кнопка для сброса симуляции и ввода исходных значений (массы тела и его исходной высоты).

**6.Сила трения.**

**Будет реализовано в файле friction.py**

Основная функция – friction\_simulation.

Пользователь с помощью полей для ввода определяет массу объекта, его начальную скорость и коэффициент трения с поверхностью.

Функция create\_object – создаст динамический объект-квадрат с заданной массой, координатами и начальной скоростью.

Метод get\_speed будет получать скорость объекта.

На экран во время всего эксперимента будут выводиться скорость объекта и его ускорение.

Сам объект будет двигаться по платформе в соответствии с его массой, начальной скоростью и ускорением.

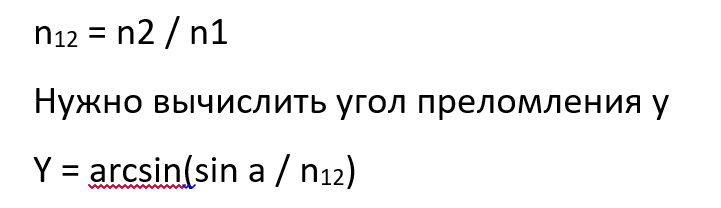
Конечно будет кнопка для вывода справочной информации по теме.

**7.Преломление луча**

**Будет реализовано в файле** refraction.py.

Основная функция – refraction\_simulation.

Функция calc\_angle – возвращает угол преломления при известном углу падения.



Функция draw\_line – рисует линию из заданной начальной координаты под заданным углом (относительно нормали).

Функция draw\_rect рисует прямоугольник на заданных координатах. Также принимает цвет, ширину и высоту прямоугольника.

С помощью полей для ввода пользователь задает показатель преломления среды из который исходит луч, показатель преломления среды, в которую входит луч, а также угол относительно нормали, под которым луч соприкасается с границей двух сред.

На экран выводится угол преломления и рисуется два луча. Также рисуются два прямоугольника, изображающие среды, в которых находится луч.

**8. Движение планет.**

Будет реализовано в файле planets.py

Основной цикл – функция planet\_moving.

**Класс Planet** для создания объектов-планет.

Метод *инициализации* будет содержать координаты центра планеты, её радиус, цвет и массу.

Метод *render* будет рисовать саму планету и её орбиту на экране.

(экран передается в функцию)

Метод get\_force вернут силу притяжения между двумя объектами.

Метод move будет высчитывать ускорение и передвижение планеты.

( Размер солнца будет сильно уменьшен для удобности показа симуляции).

На этом всё, но возможно, что будет принято решение сделать еще одну тему.