**Тлеубергенова Камшат Уралбаевна**

**Учитель информатики КГУ «ОШ №1» г. Актау **

**Вращение Солнечной системы на языке программирование Python: программирование и визуализация**

Современные технологии позволяют ученикам не только изучать теоретические аспекты астрономии, но и самостоятельно моделировать движение небесных тел с помощью программирования. В данной статье мы рассмотрим, как ученик Ковалев Эрик реализовал на языке Python симуляцию вращения Солнечной системы.

Основная задача работы заключалась в создании визуальной модели движения планет вокруг Солнца с использованием популярной библиотеки Pygame. Такой проект помогает ученику закрепить знания о законах Кеплера, орбитальном движении и основах программирования.

**Используемые технологии**

Для реализации проекта были использованы:

* **Python** — основной язык программирования.
* **Pygame** — библиотека для создания анимации.
* **Математические функции** — для расчета орбитальных движений.

**Описание кода**

Программа создает окно, в котором на черном фоне отображается Солнце в центре. Вокруг него по эллиптическим орбитам движутся планеты. Каждая планета представлена объектом с определенными параметрами: радиусом орбиты, скоростью и цветом.

Основной код содержит:

1. **Определение параметров планет** (например, их орбитальные радиусы и скорости обращения).
2. **Функцию обрисовки** каждого объекта.
3. **Цикл анимации**, в котором координаты планет обновляются согласно их угловой скорости.

Пример ключевого фрагмента кода:

import pygame

import math

 Pygame

pygame.init()

WIDTH, HEIGHT = 800, 800

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

clock = pygame.time.Clock()

BLACK = (0, 0, 0)

YELLOW = (255, 255, 0)

BLUE = (0, 0, 255)

RED = (255, 0, 0)

class Planet:

 def \_\_init\_\_(self, x, y, radius, color, speed):

 self.x = x

 self.y = y

 self.radius = radius

 self.color = color

 self.angle = 0

 self.speed = speed

 def update\_position(self):

 self.angle += self.speed

 self.x = WIDTH//2 + math.cos(self.angle) \* 150

 self.y = HEIGHT//2 + math.sin(self.angle) \* 150

 def draw(self, screen):

 pygame.draw.circle(screen, self.color, (int(self.x), int(self.y)), self.radius)

sun = Planet(WIDTH//2, HEIGHT//2, 30, YELLOW, 0)

earth = Planet(WIDTH//2 + 150, HEIGHT//2, 10, BLUE, 0.02)

mars = Planet(WIDTH//2 + 200, HEIGHT//2, 8, RED, 0.015)

planets = [earth, mars]

running = True

while running:

 screen.fill(BLACK)

 sun.draw(screen)

 for planet in planets:

 planet.update\_position()

 planet.draw(screen)

 pygame.display.flip()

 clock.tick(60)

pygame.quit()



### Основные функции программы

1. **Инициализация Pygame и создание игрового окна**
В начале кода происходит инициализация библиотеки Pygame и создание основного окна размером 1920x1080 пикселей. Также задается цвет фона и основные переменные для управления камерой и масштабированием.
2. **Загрузка изображений планет**
Программа использует изображения реальных планет, предварительно загруженные в словарь. Они масштабируются для улучшенного визуального восприятия.
3. **Создание классов "Планета" и "Звезда"**
Для структурированности кода введены два класса:
	* Planet (Планета) — управляет положением и движением планеты по орбите.
	* Star (Звезда) — отвечает за генерацию фоновых звезд, создавая эффект глубины космоса.
4. **Обрисовка и обновление объектов**
В игровом цикле программа обновляет координаты планет, рисует их на экране, а также позволяет пользователю взаимодействовать с ними. Если выбран режим фокусировки, то камера плавно центрируется на выбранной планете.
5. **Реализация управления**
Пользователь может управлять симуляцией при помощи клавиатуры и мыши:
	* Пробел — пауза/возобновление движения.
	* Стрелки вверх/вниз — изменение скорости вращения.
	* Колесико мыши — увеличение и уменьшение масштаба.
	* ЛКМ по планете — выбор для подробного изучения.
	* Перетаскивание мыши — передвижение камеры.

**Ход работы и возникающие трудности**

Разработка симуляции включала несколько этапов:

1. **Разработка концепции** — Эрик изучил движение планет в Солнечной системе и принципы моделирования орбитальных движений.
2. **Выбор инструментов** — после анализа доступных библиотек было решено использовать Pygame из-за простоты работы с графикой и анимацией.
3. **Программирование базовой структуры** — создание графического окна, реализация движения объектов.
4. **Оптимизация кода** — улучшение плавности анимации, подбор реалистичных параметров для движения планет.

Одной из сложностей стало правильное масштабирование орбит и скоростей. В реальности расстояния между планетами значительно отличаются, и скорость их движения также варьируется. Для создания визуально понятной модели пришлось использовать приближенные значения, которые сохраняли реалистичность, но делали анимацию более удобной для восприятия.

Еще одна трудность заключалась в обработке большого количества объектов в реальном времени. Чтобы оптимизировать работу программы, Эрик реализовал обновление позиций планет с учетом их индивидуальной скорости обращения вокруг Солнца.

**Результаты и выводы**

Данный проект является отличным примером того, как программирование может помочь в изучении естественных наук. Создавая подобные модели, ученики развивают алгоритмическое мышление, учатся работать с графикой и математическими вычислениями.

Помимо образовательных целей, такой проект можно использовать как основу для более сложных симуляций, например, моделирования экзо планетных систем или изучения влияния различных параметров на орбитальное движение.

**Эмоции ученика и его вклад в проект**

Эрик с энтузиазмом подошел к проекту, проявляя интерес не только к программированию, но и к астрономии. В процессе работы он сталкивался с различными трудностями, но всегда находил пути их решения. Например, после первых тестов программы он заметил, что движение планет выглядит неестественно, и самостоятельно нашел способ исправить эту проблему. Также он проявил инициативу, предложив добавить новые функции, такие как изменение масштаба орбит.

Работа над проектом дала Эрику ценный опыт в программировании, научила его искать информацию и адаптировать код под новые задачи. Он не просто следовал инструкциям, а пытался разобраться в сути алгоритмов и улучшить свою симуляцию.

**Мнение учителя**

Этот проект демонстрирует высокий уровень владения программированием у ученика. Он использует сложные концепции, такие как объектно-ориентированное программирование, обработка событий и работа с графикой. Код хорошо структурирован, а его функционал позволяет пользователю не только наблюдать за движением планет, но и взаимодействовать с ними.

Работа над такими проектами помогает развивать не только навыки программирования, но и логическое мышление, внимание к деталям и творческий подход. Поздравляю Ковалева Эрика с отличной работой и надеюсь, что в будущем он продолжит совершенствовать свои навыки в разработке программного обеспечения!

Эрик не просто переписал код из учебников — он самостоятельно анализировал, исправлял ошибки и улучшал свою модель. Этот опыт помог ему не только освоить основы программирования, но и развить важные навыки решения проблем, критического мышления и работы с алгоритмами. Надеюсь, что этот проект вдохновит его на дальнейшее изучение программирования и науки в целом.