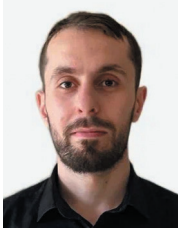


Пуртов Борис Альбертович
Инженер-системотехник, автор курса «Компьютерная физика»
Лауреат II степени XIV Международного конкурса
«Инновационные информационно-педагогические технологии в системе ИТ-образования» — ИП-2024

Джумаев Мансур Рамзанович
Врач-кибернетик, преподаватель Подготовительных курсов МФТИ и Учебного центра ВМК МГУ
Автор курса «Компьютерная физика и биофизика»
Лауреат II степени XIV Международного конкурса
«Инновационные информационно-педагогические технологии в системе ИТ-образования» — ИП-2024



Междисциплинарная информатика для школьников

Одним из направлений развития современной науки и образования является междисциплинарность — подход, при котором знания из различных дисциплин (предметов) интегрируются для решения комплексных задач. В статье рассматриваются некоторые разделы информатики, прежде всего компьютерная инфографика и программирование на языке Python, которые могут стать связующим звеном, соединяющим различные школьные дисциплины, такие как математика, физика, биология. Акцент делается на формирование ключевых программных блоков, которые позволяют программными методами создавать визуальные динамические модели из физики, биологии, географии. Статья рассчитанана учащихся, которые знакомы с основами программирования (в частности, python) и естественными науками.

Введение

Раскроем содержание основных понятий, которые используются в статье.

Междисциплинарность — это подход, который подразумевает объединение знаний из разных дисциплин (предметов) для решения комплексных задач. Междисциплинарность является отражением целостности окружающего нас мира. Междисциплинарный подход является одной из тенденций развития современной науки и необходим, чтобы решать многие проблемы, такие как, например, развитие новых технологий, изменение климата, здоровье населения и др. Понятно, что каждая из таких проблем выходит за рамки возможностей отдельно взятой области знания. Соответственно, нужны и междисциплинарные специалисты. И начинать подготовку таких специалистов нужно еще в школе.

Тем не менее, дисциплинарный (предметный) способ обучения, в котором содержание обучения делится на отдельные дисциплины (учебные

предметы), развивается и сохраняется как в науке, так и в системе подготовки на протяжении столетий. В школе дисциплинарный подход закреплен системой ЕГЭ.

Но настоятельные требования времени заставляют начать подготовку междисциплинарных специалистов еще в школе. В статье предлагается использовать для этогоинформационно-коммуникационные технологии (информатику) для организации межпредметных связей на базе алгоритмизации и кибернетических представлений. В качестве сверхзадачи имеется в виду формирование у школьников междисциплинарного мышления.

Информатика. Информационные технологии в широком смысле включают в себя такие составляющие, как кибернетика, информатика, алгоритмизация, программирование, компьютерная инфографика. В статье ставится задача превратить информатику в инструмент междисциплинарного взаимодействия для получения знаний о естественных и точных науках, для более полного и целостного описания окружающего мира. Из множества возможностей информатики в данной работе упор делается на программирование на языке Python, который имеет следующие особенности:

- один из самых популярных и распространенных языков программирования.
- относительно простой синтаксис.
- широкая сфера применения: обработка данных, создание сайтов, игры, нейросети и пр.
- большое количество библиотек и модулей для решения самых разных задач.

Одной из многочисленных библиотек python'а является turtle.

Библиотека turtle — графическая библиотека, встроенная в язык программирования Python. Она предназначена для создания простых графических изображений с помощью виртуальной черепахи (turtlegraphics). Суть довольно проста: перемещать черепашку (или курсор, или кисточку — все это синонимы) на определенное расстояние (в пикселях) в определенном направлении, меняя параметры черепашки. Библиотека turtle имеет преимущества: интуитивно понятные команды, которые можно по-разному комбинировать; наличие ее в базовой версии python (то есть не нужно отдельно устанавливать); возможности библиотеки (создание геометрических фигур и узоров разной степени сложности, разработка простых игр, построение графиков и диаграмм и др). Она также подходит для создания динамических анимационных инфографических моделей.

Инфографика. Наше сознание устроено так, что лучше всего воспринимаются зрительные (визуальные) образы. Инфографика — это визуализация знаний. Заметим, что инфографика является самой древней визуальной коммуникационной системой, возраст которой составляет около 50 000 лет [1]!

Большинство проектов, которые выполняют учащиеся, представляют из себя создание программных графических, динамических, анимационных моделей, раскрывающих обобщенные и функциональные свойства предмета изучения [3]. Создается наглядный образ «вторичной реальности».

Основная цель — формирование целостной картины окружающего мира. При этом междисциплинарность не отрицает отдельные учебные предметы. Они не утрачивают своей специфики, а вносят свой вклад в развитие междисциплинарной идеи. Общей связующей нитью являются два универсальных языка описания знаний — инфографика и программирование.

Цель статьи — ознакомить слушателей с принципами создания междисциплинарных курсов на базе информатики, которая является связующим звеном для разных дисциплин (предметов). Именно с ее помощью создаются «программные перемычки (окна)» в границах (пусть во многом условных),разделяющих различные естественнонаучные и точные дисциплины.

Проекты, описанные в статье, могут быть использованы при преподавании информатики в средней школе и системе дополнительного образования. Раскроем суть междисциплинарной информатики на примере нескольких заданий и проектов.

1. Измерение размера пикселя экрана программно-лабораторным методом

Междисциплинарная лабораторная работа, совмещающая информатику и физику. Изучаетсяметод измерения малых тел, известный как «метод рядов». В качестве объекта исследования выбран пиксель экрана монитора компьютера, часто невидимый невооруженным глазом. Для решения этой нестандартной задачи используется графическая программная модель картины «Черный квадрат» К.Малевича (рис.2) в масштабе 1:10 как дополнительного инструмента [2]. Основная идея — написать программу черчения на экране квадрата с заданным числом пикселей (360 пк) по каждой стороне (рис.1). Измерив изображение квадрата на экране компьютера линейкой, вычисляем размеры одного пикселя.

Проделав всю работу, вычисляем размер пикселя (и по ширине, и по высоте): 87 мм / 360 пк = 0,24 мм. От-

```
File  Edit  Format  Run  Options  Window  Help
import turtle as t

t.setup(500, 500)

t.up()
t.goto(-180, -180)
t.down()

t.begin_fill()  # начало заливки
for _ in range(4):  # цикл для квадрата
    t.forward(360)  # сторона квадрата
    t.left(90)  # угол квадрата
t.end_fill()  # конец заливки

t.hideturtle()  # скрыть курсор
```

Рис.1. Код программы черчения черного квадрата

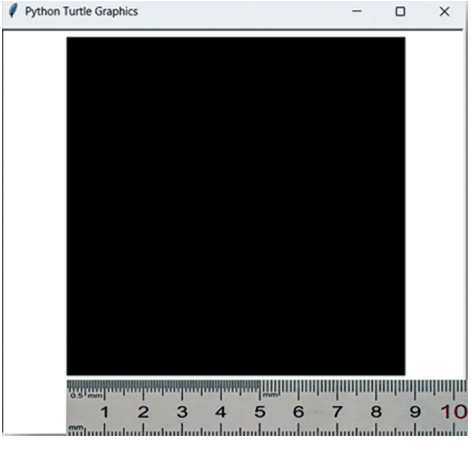


Рис. 2. Измерение квадрата линейкой на экране компьютера

сюда сразу делаем вывод, что пиксель имеет квадратную форму. Размеры субпикселя (пиксель состоит из 3-х субпикселей: красного, зеленого и синего) в таком случае будут равны: 0,24 мм (высота) и 0,24/3 = = 0,08 мм (ширина).

Приобретаемые по итогам выполнения работы знания, умения и навыки:

- Написание первой простой программы в *turtle* с использованием цикла.
- Формирование универсального навыка — работы с измерительным инструментом (линейкой).
- Умение совмещать преимущества «цифрового» (виртуального) и материального миров для междисциплинарных работ.

2. Геометрическая прогрессия в биофизике

Междисциплинарная работа, в основе которой лежит знание и практическое применение свойств геометрической прогрессии для описания и программной визуализации увеличения количества бактерий при их бинарном делении.

Бактерии делятся бинарно (из каждой текущей клетки образуются 2 новые). То есть, получается геометрическая прогрессия, в которой знаменатель равен 2 (количество клеток увеличивается с каждым делением в 2 раза). Задача — отобразить графически принцип деления бактерий, в основе которой лежит геометрическая прогрессия со знаменателем 2.

Для наглядности визуализации будем выводить количество бактерий в виде монетных столбиков в каждый условный момент времени (см. рис. 3), где каждый столбик — это количество бактерий в текущий момент времени.

В начальный (нулевой) момент времени (обозначим t_0) их количество равно 1 (одна исходная материнская клетка).

В следующий момент времени (обозначим t_1) их количество равно уже 2, так как из исходной материнской клетки образовались 2 новые

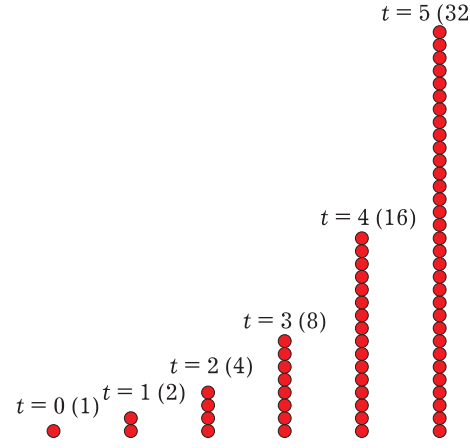


Рис. 3. Программная визуализация увеличения количества бактерий при их бинарном делении

(геометрическая прогрессия со знаменателем 2).

В момент времени t_2 их будет уже 4.

В момент времени t_3 — 8 клеток.

В момент времени t_4 — 16 клеток.

В момент времени t_5 — 32 клетки и т.д.

Ядро программы (рис. 4) представляет собой вложенный цикл, в котором внутренний цикл отвечает за количество бактерий в столбике в текущий момент времени (2^{time}, где time — текущий момент време-

```
for time in range(6):
    for n in range(2**time):
        t.up()
        t.goto(-400 + 130*time, -300 + 20*n)  # координаты бактерии
        t.down()
        coccus(10)  # функция для отрисовки круглой бактерии (радиус – 10 пикселей)
    t.up()
    t.goto(-400 + 130*time - 50, -300 + 20*n + 30)  # координаты надписи
    t.down()
    t.write(f"t = {time} ({2**time})", font = ('Arial Black', 15) )  # надписи
```

Рис. 4. Ядро программы изменения количества бактерий при делении

ни), а внешний — за количество таких столбиков (в нашем случае 6) (рис. 4). Данная модель с внесением небольших правок может использоваться в физике для описания изменения массы радиоактивного вещества с течением времени, но уже на основе убывающей геометрической прогрессии.Также возможно применение модели для описания деления ядер урана под действием нейтронов (физика) или роста популяции организмов за определенный промежуток времени (биофизика).

3. Цифровой секундомер

Суть работы — создание программного цифрового секундомера и на его базе программы измерения и оценки частоты сердечных сокращений (ЧСС или пульса).

Возникает вопрос: зачем создавать секундомер, если можно воспользоваться встроенной программой в смартфоне или на компьютере. Мы ставим перед собой учебно — практическую задачу — показать, как самостоятельно создать прибор для измерения времени, настроить его и использовать для исследования биофизических процессов.

Ядром программы является вложенный цикл (рис. 5). Внутренний цикл отсчитывает десятые доли секунды, внешний — секунды. Важнейшим элементом является команда задержки времени time.sleep(x), где x — время задержки. Без этой команды получился бы простой счетчик времени. Изменяя время задержки, ученики должны самостоятельно настроить секундомер, взяв в качестве эталона секундомер в смартфоне. Процесс настройки прибора называется «поверкой».

Следующий этап — использование созданного программно-аппаратного прибора для измерения частоты сердечных сокращений (ЧСС или пульса). Для этого необходимо научить учащихся измерять пульс и объяснить опасность нарушений сердечного ритма, которые являются признаками тяжелых заболеваний.

```
n = 15
for j in range(n):  # цикл отсчета секунд
    for i in range(10):  # цикл отсчета десятых долей секунды
        t.write(f'{j}.{i}', font = ('Arial Black', 50) )  # вывод времени на экран
        time.sleep(0.1)  # скорость смены значений секундомера
        t.clear()  # очищение окна для вывода следующего значения
t.write(f'{n}.{0}', font = ('Arial Black', 50) )
```

Рис. 5. Ядро программы «Цифровой секундомер»



Рис. 6. Окна интерфейса цифрового секундомера

Процесс вычисления ЧСС и выдачи «заключения» о состоянии ЧСС можно автоматизировать. Для этого необходимо:

- Настроить секундомер на отсчет 15 сек.
- Измерить пульс за 15 сек и ввести полученное значение в окно ввода.
- Перевести значение пульса за 15 сек в значение пульса за 1 мин.
- Вывести на экран количество ударов за 1 мин (значение минутного пульса).
- Выдать «заключение» о вашем состоянии: «Ваш пульс понижен» (при значении пульса< 60 уд/мин), «Ваш пульс в норме» (при значении пульса 60-85 уд/мин), «Ваш пульс повышен» (при значении пульса > 85 уд/мин).

Окна интерфейса программы вы видите на рис. 6. Одно окно отвечает за счет времени секундомера, другое — за оценку измеренного значения пульса.

Данная модель с внесением небольших правок может использоваться для измерения и оценки других важных показателей, например частоты дыхания или частоты моргания глаз.

4. Моделирование атомов и молекул

Суть работы — программное графическое моделирование атомов и молекул на примере молекулы воды и вычисление масштаба созданной модели.

При создании модели учитываются следующие реальные параметры:

- химический состав: 1 атом кислорода и 2 атома водорода.
- отношение радиусов атомов (радиус атома кислорода примерно в 2.5 раза больше радиуса атома водорода).
- угол между атомными связями «водород — кислород — водород» равен 104,5 град.

Ядро программы (рис. 7) и графический результат запуска программы (рис. 8).

После создания графической модели производится расчет масштаба созданной модели. Для этого берется средний диаметр реальной молекулы (примерно 0,3 нм), который со-

```
import turtle as t

t.setup(800, 600)
t.speed(100)

r = 100  # радиус атома кислорода (в пикс)

def atom(r, color):
    t.fillcolor(color)
    t.begin_fill()
    t.circle(r)  # окружность радиуса r
    t.end_fill()

# атом кислорода
atom(r, 'green')
```

Рис. 7. Ядро программы «Молекула воды»

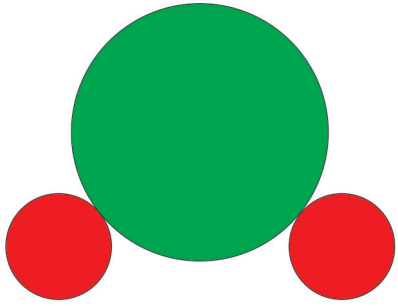


Рис. 8. Графическая модель молекулы воды

относится с диаметром полученной молекулы.

В последующем модель может быть «оживлена» для других разделов физики и химии, например: модель идеального газа, образование кристаллов, протекание химической реакции и т. д.

Полученный опыт можно использовать для других разделов физики, геометрии, инженерной графики.

Что учащиеся получили по итогам выполнения работы:

- Начальные знания атомно-молекулярной теории (строение вещества: молекулы и атомы).
- Изучили практическое применение функции с аргументами для создания модели атома.
- Научились вычислять масштаб модели.

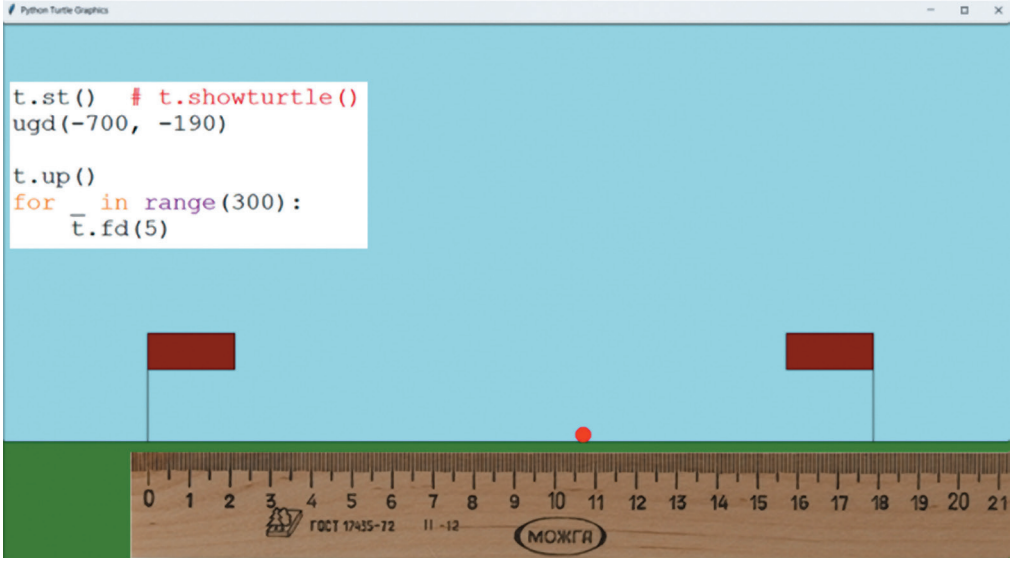


Рис.9. Ядро программы (в левом верхнем углу) и графический результат запуска программы

5. Анимационная модель движения тела

Суть работы состоит в создании программной анимационной модели прямолинейного движения тела на примере шарика и измерении скорости его движения [3].

Для вычисления скорости движения тела нужно знать:

1) Расстояние (определяется при помощи линейки)

2) Время движения (вычисляется при помощи секундомера)

Ядро программы (именно тот блок, который создает на экране иллюзию движения) и графический результат запуска программы можно увидеть на рис.9.

Как видно, код довольно простой: достаточно одного цикла for, в котором задается сдвиг изображения на определенное количество пикселей на каждой итерации.

В результате по экрану будет двигаться курсор, которому заранее придана форма шарика при помощи команды t.shape('circle').

На следующем этапе проводятся измерения пути от флажка до флажка при помощи линейки и времени движения шарика с помощью секундомера.

После вычисления времени движения и расстояния определяется значение скорости движения тела по известной формуле и производится перевод его значения в км/ч.

В качестве дополнительного задания учащимся предлагается внести изменения в программу таким образом, чтобы скорость движения шарика составила определенное значение (например, 1 км/ч). Это достигается изменением количества пикселей, на которое передвигается шарик на каждой итерации цикла в строчкеt.fd(5).

Получается универсальная базовая модель для отображения движения различных тел с различными скоростями для различных областей знаний, например:

- Физика: в сочетании с моделью молекулы — модель идеального газа.
- Биофизика: движение вирусов и бактерий.
- География: дрейф континентов.

6. Динамическая модель нагревания воды

В предыдущих примерах моделировались объекты и процессы, которые относительно легко визуализировать: объекты разных размеров; тело, двигающееся с определенной скоростью и т. д. А как визуализировать такие физические явления/величины, какэлектричество, энергия, магнетизм и т.д.? Для этого используются цветокодированные шкалы, в которых каждому цвету (или оттенку цвета) в шкале соответствует определенное значение физической величины.

В рассматриваемом примере создается цветокодированная шкала температуры воды от 0 до 100 градусов Цельсия. Современные компьютерные технологии позволяют прорисовать все 100 оттенков цвета, соответствующих 100 градусам. Но наше зрение не в состоянии четко определить границы этих оттенков. Поэтому мы делаем шкалу всего с 10

```
# нагревание воды в стакане
for i in range(100):
    red, blue = 0, 1
    color = (red + 0.01*i, 0, blue - 0.01*i)
    glass(200, 320, color)
    time.sleep(0.1)
```

Рис. 10. Ядро программы, обеспечивающее плавное изменение цвета

цветами и присваиваем им значения с шагом 10 градусов.

Изменение температуры воды в стакане при нагреваниибудем отображать с шагом 1 градус. При этом изменение цвета визуально будет происходить плавно.

Ядро программы состоит из одного цикла (рис. 10), на каждой итерации которогопроисходит изменение соотношения красного и синего цветов от 0 до 100% в цветовой модели RGB (информатика!). Уменьшение значения синего цвета сопровождается увеличением значения красного. Каждое повторение цикла «нагревает» воду на 1 градус Цельсия.

В начальный момент времени температура воды в стакане равна 0 град С (цвет холодной воды — синий). Далее по мере нагревания температура увеличивается, цвет воды изменяется в сторону красного. В RGB-коде это отображено так: уменьшение значения синего цвета и увеличение значения красного на каждой итерации цикла.

Скриншот графической динамической модели нагревания воды приведен на рис. 11. По сути, мы создали простой программный модуль,

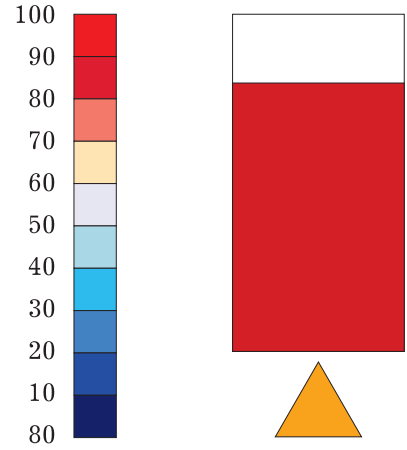


Рис. 11. Скриншот динамической модели нагревания воды

который является универсальным и может применяться для отображения многих физических (и не только) процессов. Достаточно изменить градуировку цветовой шкалы, например, заменить градусы на промилле, и мы получим модель, которая описывает изменение солености морской воды.

Или же можно адаптировать модель под задачи биологии, заменив градусы на проценты. Тогда получим модель, которая может иллюстрировать насыщение артериальной крови кислородом в организме человека.

7. Моделирование разряда молнии

Одна из самых необычных тем курса — моделирование разряда молнии.

Суть работы — создание простейшей динамической модели разряда молнии основе вложенного цикла.

Для этого нужно:

- Изучить природу электрических явлений.
- Графически изобразить грозовые облака и землю.
- Начертить динамически несколько веток молнии, комбинируя модули time, random и turtle.

Ядро программы приведено на рис. 12. Для моделирования разряда молнии используется вложенный цикл, в котором внешний цикл отвечает за количество веток (в нашем случае 5), а внутренний — за отрисовку одной ветки молнии.

Для отрисовки одной ветки молнии ученики представляют ее как ломаную линию, состоящую из определенного количества отрезков случайной длины заданного диапазона, которые идут вниз (от облаков к земле), квазислучайно меняя свое направление (опять-таки, в пределах заданного диапазона).

Задача учеников — меняя значения параметров (угол поворота и длину отрезка ломаной), добиться эффекта разряда молнии.

Скриншот динамического разряда молнии приведен на рис. 13.

Что дало изучение этой темы учащимся:

- Умение комбинировать разные модули: turtle, random и time.
- Изучение основ электричества.
- Начальные знания из теории фракталов (ветки молнии — это предфракталы).

Данная модель с внесением необходимых изменений может использоваться, например, в биологии (анатомии) для визуализации в простейшем виде бронхиального дерева легких или же сети кровеносных сосудов.

```
# разряд молнии
t.width(3)
t.color('white')

for _ in range(5):
    t.up()
    t.goto(0, 300)
    t.down()
    for _ in range(100):
        an = random.randint(-150, -80)  # угол поворота ломаной
        d = random.randint(10, 40)  # длина отрезка ломаной
        t.seth(an)
        t.fd(d)
```

Рис. 12. Код динамического разряда молнии

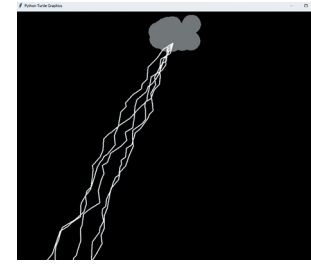


Рис. 13. Скриншот разряда молнии

8. Графическая модель Солнечной системы

Последний пример, который будет рассмотрен в данной статье — это пример работы (точнее сказать, часть более комплексной работы) по созданию графической модели Солнечной системы (на примере планет земной группы).

Суть работы — создание программной (сначала статической, а затем динамической) модели Сол-

```
for i in range(4):
    t.up()
    t.goto(0, -70 - r*i)
    t.down()
    t.circle(140 + i*r)

planet(m, r_merk, 140, 30, 'gray')  # отрисовка меркурия
planet(v, r_ven, -90, -60, 'lightyellow') # отрисовка венеры
planet(z, r_zem, 120, -120, 'cyan')  # отрисовка земли
planet(p, r_mars, -150, -180, 'red')  # отрисовка марса
```

Рис. 14. Код, задающий диаметры и орбиты планет

нечной системы для планет земной группы.

Планеты земной группы — это Меркурий, Венера, Земля, Марс.

Отношения радиусов планет и их положение на орбитах должны соответствовать реальности.

Ядро программы и графический результат запуска программы приведены на рис. 14 и рис. 15 соответственно.

На первом этапе при помощи цикла создаются концентрические орбиты, по которым движутся планеты (для удобства они имеют форму окружности). На втором этапе на орбитах располагаются планеты земной группы (каждая планета на своей орбите).

Заметьте, что отношения радиусов планет соответствуют реальным данным! На третьем, заключительном этапе планетыприводятся в движение.

Что дает изучение этой темы учащимся:

- Изучение движения тела по окружности.
- Более продвинутое применение функций с аргументами (у функции 4-5 аргументов!).
- Изучаются основы космонавтики.

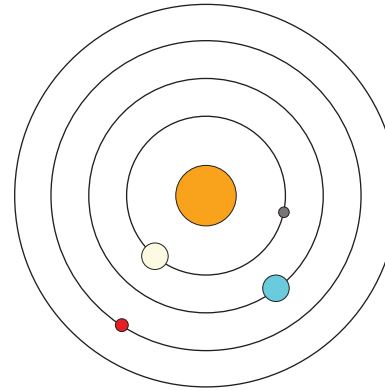


Рис.15. Графическая модель Солнечной системы для планет земной группы

- Производится настройка периодов обращения планет (в старших классах)за счет заранее встроенных программных «регуляторов» движения.

Данную модель можно трансформировать в более сложный вариант — модель Солнечной системы для всех планет (и земной группы, и группы планетов-гигантов). Также можно ее взять за основу при создании модели вращения спутников вокруг планеты (например, вращение спутников вокруг Марса).

Заключение

- Междисциплинарность является отражением целостности окружающего нас мира и одной из тенденций развития современной науки и образования.
- Готовить междисциплинарных специалист,ов необходимо со школы, формируя особый тип мышления (синтетическое мышление) и навыки переноса знаний из одной предметной области в другую.
- В качестве инструментов, позволяющих объединять различные дисциплины в единое целое, можетвыступать информатика (программирование) в сочетании с инфографикой.
- В статье приведены примеры различных междисциплинарных проектов. В каждом проекте выделено ядро программной модели. Именно с его помощью создаются «программные перемычки (окна)» в искусственных барьерах, разделяющих информатику и различные естественнонаучные дисциплины.
- Все проекты предусматривают самостоятельную их доработку учащимися, что возможно только при их глубоком погружении в изучаемую тему.
- У учащихся формируются и восприятие, и способности к решению проблем в целом; происходит развитие всех видов мышления: логического, абстрактного, образного, пространственного.
- Проекты, приведенные в статье, всего лишь базис для разработки других моделей различных процессов разной степени сложности для междисциплинарных курсов в рамках школьного образования.

Литература

1. Пуртов, Б.А., Джумаев, М.Р., 50 000 лет развития инфографики: от наскальной к компьютерной. // Потенциал — Москва, 2024 — №9 — С. 40-50
2. Буянин А., Голзари А., Калюжина С., Попов И. Определение размеров сверхмалых невидимых объектов (пикселей экрана) методом рядов. Потенциал №6, 2022г
3. Пуртов Б.А., Джумаев М.Р., Жуков Т.А., Крушанов А.А., Попов И.В. Лабораторная работа: Создание виртуальной программной анимационной модели движения шарика и измерение скорости его движения в среде лабораторной совмещенной реальности (ЛСР) // Потенциал — Москва, 2023 — №2 — С. 34-48

ЮМОР

— Сколько будет, если восемь разделить пополам? — спрашивает учитель.

— Если вдоль, то две тройки, если поперек, то два нулика, — ответил ученик.

* * *

Почему считается, что ученые любят витать в облаках?

— Потому что идеи носятся в воздухе.