УДК 65.011.56

ВИЗУАЛИЗАЦИЯ МОДЕЛИРОВАНИЯ ФИЗИЧЕСКИХ ЗАКОНОВ

Черников А.С., Горбунова Т.Н.
Московский политехнический университет (107023, г. Москва, ул. Б.Семёновская, д. 38. email: mospolytech@mospolytech.ru)

Процесс визуализации физических процессов состоит из выбора математической модели с необходимым количеством параметров, оптимизации процесса анимации и выбора эффективного средства реализации сформулированной модели. Основная задача данной работы была в разработке графического приложения моделирования полета снаряда из пушки. В качестве языка программирования был выбран Python. Подобные задачи актуальны в образовательной среде, а также в сфере интерактивного взаимодействия. В статье рассмотрена математическая модель полета снаряда на плоскости с использованием абстрактной пушки без учета сопротивления воздуха, а также была произведена адаптация ее к компьютерному моделированию процесса. Также были проведены исследования, связанные с самим процессом анимации, а именно с изучением зависимости скорости смены кадров от траектории движения снаряда с учетом физиологических особенностей восприятия человека с целью оптимизации программного кода, а именно времени его выполнения. Для данной задачи был разработан алгоритм и программный код его реализующий, который представлен в работе с необходимыми комментариями. Результаты тестирования программы показали хороший уровень визуализации физической модели. Так же были предложены дальнейшие этапы его развития.

Ключевые слова: Python, графическое приложение, математическая модель, рендеринг, анимация, алгоритм

VISUALIZATION OF PHYSICAL LAWS SIMULATION

Chernikov A. S., Gorbunova T. N.
Moscow Polytechnic University (Bolshaya Semenovskaya str., 38, Moscow, 107023 email: mospolytech@mospolytech.ru)

The process of visualizing the physical process consists of selecting a mathematical model with the required number of parameters, the optimization of the animation process and finding effective means of implementation of the formulated model. The main objective of this work was to develop a graphical modeling application of flight of the projectile from the gun. As a programming language chosen was Python. Such problems are relevant in the educational environment, as well as in the field of interactive communication. The article considers the mathematical model of the projectile in the plane with an abstract of the gun without taking into account air resistance, and was also produced its adaptation to the computer modeling process. Also, studies have been conducted, associated with the animation process, namely with the study of the dependence of the rate of change of frames from the trajectory of the projectile, taking into account the physiological characteristics of human perception to optimize the program code, namely the time it is done. For this task the algorithm was developed and the code that it implements, which is represented in the work with the necessary comments. The results of the testing program showed a satisfactory level of visualization of the physical model. Also proposed further stages of its development.

Keywords: Python, GUI application, mathematical model, rendering, animation, algorithm

Введение

Современный уровень развития и распространения компьютерных технологий дает возможность, а также, в свою очередь, определяет способ представления информации в графическом виде как наиболее оптимальной для восприятия и анализа [1,2]. Также особо следует выделить образование и сферы научного исследования, где моделирование различных процессов является эффективным и широко используемым инструментом. В связи с этим была сформулирована задача визуализации криволинейных процессов. В качестве средства реализации был выбран популярный сегодня языка Python, возможности которого позволяют реализовать поставленные задачи [8].

Целью работы было разработать на языке программирования Python графическое приложение для моделирования стрельбы из пушки на плоскости. Сопротивление воздуха не

учитывать. Пушка должна обладать прицелом, отображающем расчетную точку попадания снаряда.

Математическая модель

В качестве модели, описывающей движение снаряда, была выбрана простейшая модель, отвечающая заданным в постановке задачи условиям, описывающая положение снаряда на момент времени t согласно следующим уравнениям:

$$y(t) = v_0 \cdot t \cdot \sin(a) - \frac{g \cdot t^2}{2}$$

$$x(t) = v_0 \cdot t \cdot \cos(a)$$

Где $x(t)$ и $y(t)$ координаты снаряда по осям x и y в момент времени t соответственно, a – угол наклона выстрела к горизонту, v_0 – начальная скорость снаряда, t – время полета снаряда, g – ускорение свободного падения, взято равным 9,8 м/с^2.

Для управления наклоном ствола абстрактной пушки была составлена система уравнений, полученной из:

1. общей формулы прямой между двумя точками, использующейся при создании алгоритма Брезенхэма рисования прямой линии на ЭВМ [4],
2. формулы расстояния между двумя точками (в случае задачи это основание ствола пушки и конца пушки).

Учитывая, что в рамках данной задачи можно ограничиться решением для первой координатной четверти, т.к. мы всегда можем выполнять преобразования систем координат по типу аффинных, если нам потребуется, например, масштабирование одной системы координат в другой или их поворот и перемещения. А также очевидно, что в рамках данной системы по смыслу $l \geq 0$, $x_0 = y_0 = 0$, $x_1 > 0$ и $y_1 > 0$. Таким образом, система

$$\begin{cases} y - y_0 = \dfrac{y_1 - y_0}{x_1 - x_0}(x - x_0) \\ l = \sqrt{(x - x_0)^2 + (y - y_0)^2} \end{cases}$$

может быть упрощена до

$$\begin{cases} \mathrm{c} = \dfrac{y_1}{x_1} \\ x = \sqrt{l^2 - y^2} \\ y = \dfrac{cl}{\sqrt{c^2 + 1}} \end{cases}$$

где l – длина отрезка между точкой начала координат, совпадающей с точкой начала ствола и курсором мыши, а x и y – координаты этого курсора.

Анимация

Задача анимации заключается в комфортном для зрителя восприятии отображаемых процессов. Для плавной анимации полета снаряда будет производиться рендеринг при помощи смены некоторого числа кадров. Был исследован вопрос об оптимальном числе кадром. Если сделать их число фиксированным, то при стрельбе по длинной траектории, снаряд будет лететь намного быстрее, чем при стрельбе по короткой. Таким образом, количество кадров при стрельбе должно зависеть от длины траектории полета снаряда.

Однако, снаряд летит по параболе, а вычисление её длинны довольно ресурсоемкая процедура, требующая численных методов интегрирования, что не оптимально с точки зрения производительности задачи программного рендеринга. Поскольку человеческий глаз и восприятие не идеальны и обладают естественными для людей особенностями [3, 7], то на практике для расчета числа кадров оказалось достаточным аппроксимировать параболу простой ломаной линией с одной единственной точкой излома в самой верхней её точке, тогда экспериментальным путем было подобрано количество кадров порядка 30 в секунду.

Разработка алгоритма программы

После получения минимально необходимой математической модели был разработан алгоритм для приложения. В общих чертах его логику можно описать следующим образом:

1. Создание необходимых объектов для отрисовки и переменных, описывающих начальные параметры модели, таких как начальную скорость снаряда и ускорение свободного падения
2. Код для отрисовки начального положения пушки и прицела, а также поверхности по которой проводится стрельба
3. Реализация поворота пушки, согласно наведению прицела при помощи курсора мыши с помощью функции rotate_cannon, которая наводит ствол пушки на одну прямую с курсором мыши, а также отмечает расчетное место попадания снаряда на поверхности
4. Для стрельбы была разработана функция shoot_ball, которая отрисовывает анимацию полета снаряда по траектории, описываемой моделью [5,6].

Таким образом, было получено всё необходимое для разработки кода нашей программы, который приведен ниже в полном объеме с поясняющими комментариями на языке Python:

```
# -*- coding: utf-8 -*-

import math

import time

from tkinter import *


scene_width = 640

scene_height = 480
```

```
c = Canvas(width=scene_width, height=scene_height, bg='grey80')
c.pack()
cannon_length = 70
v = 70
g = 9.8

def rotate_cannon(x_i, y_i):
    mn = y_i/x_i
    y = (mn * cannon_length) / (math.sqrt(mn * mn + 1))
    x = math.sqrt(cannon_length * cannon_length - y * y)
    c.coords(cn, 0, scene_height-10, int(x), scene_height-10-int(y))
    angle = math.atan2(x_i, y_i)  # угол наклона ствола пушки
    t = 2*v*math.sin(1.5707963267948966-angle)/g
    x_t = v*t*math.cos(1.5707963267948966-angle)  # координата прицела
    c.coords(cursor, int(x_t-5), scene_height-15, int(x_t+5), scene_height-5)
    return

def motion(event):
    if event.x > 0 and event.y < scene_height-10:
        rotate_cannon(float(event.x), float(470-event.y))
    return

def shoot_ball(event):
    ball = c.create_oval(-5, scene_height-15, 5, scene_height-5, fill='yellow')
    angle = math.atan2(float(event.x), float(470-event.y))  # угол выстрела
    delta_t = 2*v*math.sin(1.5707963267948966-angle)/g  # время полета снаряда
    x_t = v*delta_t*math.cos(1.5707963267948966-angle)/10  # расстояние приземления
снаряда
    y_max = v*delta_t/2*math.sin(1.5707963267948966-angle)-g*delta_t*delta_t/8  #
максимальная высота полета снаряда
```

"""Теперь, чтобы не проводить вычисления без лишней необходимости, длину кривой по которой летит снаряд для увеличения зависимости количества кадров от параметров выстрела, мы будем аппроксимировать ломаной кривой из 2-ух отрезков, точка начала полета снаряда, точка максимума по у и точка приземления снаряда.

Это позволит нам сделать более реалистичной анимацию полета снаряда, что будет комфортно для пользователя.

"""

```
    length = 2*math.sqrt(x_t*x_t/4+y_max*y_max)  # грубая оценка длины траектории

    num_frames = int(length/delta_t)  # количество кадров для отрисовки

    delta_t = delta_t / num_frames

    for i in range(num_frames):  # цикл отрисовки

        x_ball = v*(i+1)*delta_t*math.cos(1.5707963267948966-angle)

        y_ball = v*(i+1)*delta_t*math.sin(1.5707963267948966-angle)-
g*(i+1)*delta_t*(i+1)*delta_t/2

        c.coords(ball, int(x_ball-5), scene_height-y_ball-15, int(x_ball+5), scene_height-
y_ball-5)

        c.update()

        time.sleep(0.02)

    c.coords(ball, int(x_ball - 5), scene_height - y_ball - 15, int(x_ball + 5), scene_height -
y_ball - 5)

    return


c.bind('<Motion>', motion)

c.bind('<Button 1>', shoot_ball)

c.create_oval(-25, 445, 25, 495, fill='#141414')

c.create_rectangle(25, scene_height-10, 0, scene_height, fill='#141414')

cn = c.create_line(0, scene_height-10, cannon_length, scene_height-10, width=10,
fill='#141414')

c.create_line(0, scene_height-10, scene_width, scene_height-10, width=1, fill='black')

cursor=c.create_oval(cannon_length, scene_height-15, cannon_length+10, scene_height-
5, fill='red')


mainloop()
```

Конечно, данный код можно оптимизировать, например, вынеся расчеты часто проводимых фрагментов функций в отдельные переменные, но это усложнит простоту понимания примера, а скорость работы кода достаточна для плавной анимации на ЭВМ, предоставленных для данной работы.

Скриншоты работы программы приведены ниже. На рисунке 1 показаны пушка, прицел и указатель мыши для настройки параметров задачи в начальный момент времени. На рисунке 2 – пушка и снаряд в конечный момент времени.

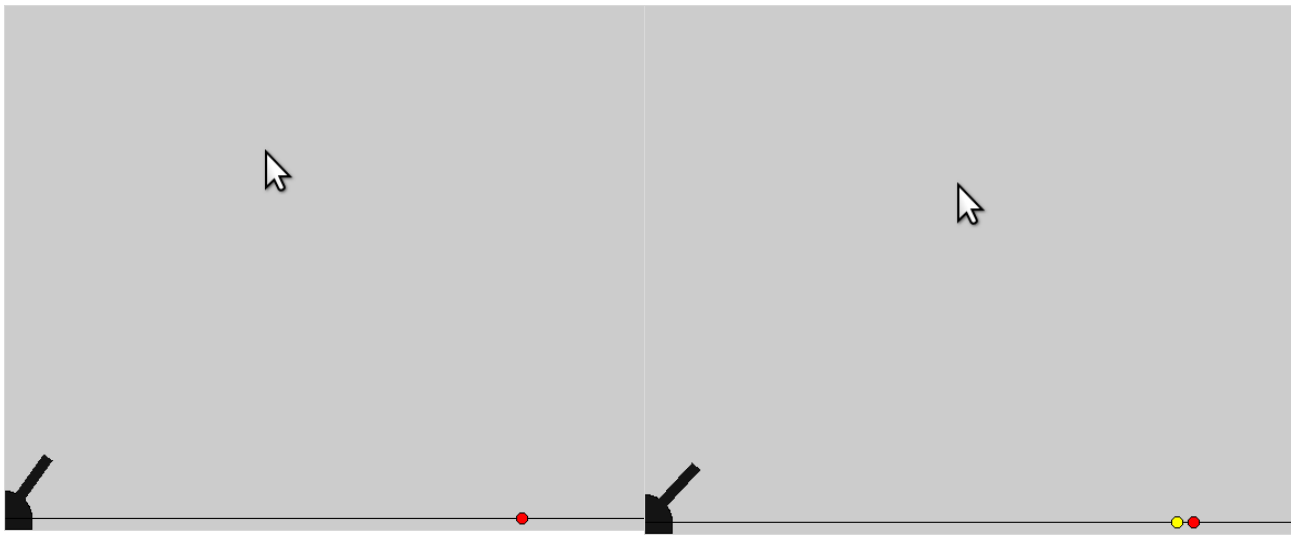

Рис. 1 Пушка и прицел

Рис. 2 Пушка после выстрела

Выводы

В результате данной работы была выработана математическая модель, адаптированная к процессу визуализации на компьютере. Исследован процесс анимации и его особенности с моделированием криволинейных процессов. Разработана программа на Python с программной библиотекой tkinter.

В ходе выполнения задачи было разработано приложение, отвечающее требованиям поставленной задачи. Данное приложение можно рассматривать как базовое для дальнейшей реализации процесса визуализации как с дополнительными параметрами этой модели, так и для других задач этого класса.

Список литературы

1. Горбунова Т.Н. Инфографика в процессе обучения / Актуальные проблемы развития вертикальной интеграции системы образования, науки и бизнеса: экономические, правовые и социальные аспекты: материалы V Международной научно- практической конференции 1-2 декабря 2016г. - Т. 2 / под ред. С.Л. Иголкина. – Воронеж: ВЭПИ, 2016. – 169-172с.
2. Данилова Н.Н. Физиология высшей нервной деятельности: учебник для студентов вузов / Н.Н. Данилова, А.Л. Крылова; отв. ред. Е. Баранчикова. - Ростов н/Д.: Феникс, 1999. - 480с.: ил. - (Серия "Учебники и учебные пособия").

3. Кудрина, М.А. Компьютерная графика: учеб. / М.А. Кудрина, К.Е. Климентьев. – Самара: Изд-во Самар. гос. аэрокосм. ун-та, 2013. – 138 с.
4. Роджерс Д. Алгоритмические основы машинной графики. — М.: Мир, 1989. — С. 54-63.
5. Саммерфилд Марк. Python на практике. — Перевод с английского. — М.: ДМК Пресс, 2014. — 338 с. — ISBN 978-5-97060-095-5.
6. Федоров Д. Ю. Основы программирования на примере языка Python : учеб.пособие / Д. Ю. Федоров. – СПб., 2016. – 176 с.
7. Восприятие Информации Человеком. Человеческое Восприятие [Электронный ресурс] - Электрон. дан. - cop. 2010-2011. - Режим доступа: http://mywebpro.ru/psihika/vospr-infor-chelov-chelov-vospr.html
8. [Электронный ресурс] https://www.python.org/doc/