CENG216 – EE204
Joint Interdisciplinary Homework: Two-body Numerical Simulation

This document is a draft and is subject to revisions until all teams are formed.

Announcement date: 24 May 2021
Due date for team selections: 23:59, **26 May 2021**
Due date for submissions: 23:59, 04 June 2021

The two-body problem is to predict the motion of two planetary objects which are viewed as points. See the list of hints for details of the problem. In this homework, design and implement a two-body simulator that adheres to the Model-View-Controller design pattern as follows:

1. **Simulation:** Develop two applications one of which is written in **two_body_simulation.c** and the other **two_body_simulation.py**. These programs must be functionally equivalent and include the following parts separately:
    a. **Model**: a C structure and Python class (or, ideally a data class) named **TwoBodyModel** that stores the necessary data such as position information of both planetary bodies to be simulated.
    b. **Controller:** a C structure and a Python class named **TwoBodyController** that stores the necessary data such as eccentricity and mass ratio to specify the simulation details, and related functions/methods to perform the simulation by numerically solving the related ordinary differential equations using the Euler's method or the fourth order Runge-Kutta method.
    c. **App:** a simple command-line interface which allows for specifying the parameters of the simulation such as T, δt, method (Eurler/Runge-Kutta) and others. It must run the simulation from t = 0 to t = T in steps of δt, and then save the location and velocity vectors in a text file. The velocity vectors are saved due to debugging purposes but the location vectors must be used in the animation application specified below.

2. **Animation:** Develop an application consists of a single source file written in Python and named as **two_body_animation.py**. This program must be completely independent of the simulation part and include the following parts separately:

   a. **View**: a class named **TwoBodyView** that can draw the two-bodies on a window interactively. You may implement the drawing functionality using OpenCV (a computer vision library), TkInter (a built-in GUI toolkit), PyQt (an advanced GUI toolkit), PyGame (a library designed for video games), Matplotlib (a plotting library) or a library of your choice. This application must allow user to play/pause, rewind and exit the simulation using keyboard input (e.g., hitting space to play/pause) or graphical buttons (e.g., a button whose label toggles between "play" and "pause").
   b. **App**: a simple program which reads the text file generated in the simulation step and visualize the data using the view component.

Please submit your C and Python source files in a zip archive named ⟨TeamName⟩.zip (omit the angle brackets) with a text file describing the C (e.g., C99), compiler (e.g., gcc 7.5.0) and Python (e.g., 3.9.4) versions, the Python library names (ideally names of the pip packages), any information necessary to use the applications (e.g., which keyboard buttons are used for which functionalities, and how the text file should be interpreted), and team members (Student ID, department, full name). This file must be named **readme.txt**.

Note: This homework requires an interdisciplinary team of three to four students. At least one member in each team must be from the CENG and EE departments. Fill “teams.xlsx” under the section “Files”. Follow the instructions given in the upper left cell.

Hints:

- If you do not set a meaningful step size the solution method of ODE can have a high error that may lead divergence from the stable orbit limits.
- In C, you should be careful about the precedence and associativity of the arrow (->) operator while dealing with structures and use parentheses whenever necessary. Mistakes regarding this may not result in a syntax error which makes them hard to detect.
- You can check the example simulation written as a web application: Example Simulator
- In order to learn the concept of MVC, check: MVC Design Pattern Tutorial
- You can take advantage of the book: Numerical Analysis (Chapter 6 ODEs), 2nd Ed., T. Sauer