DAT154 – Assignment 2

Purpose:

C# and .NET.
Libraries, Graphical Elements, Timers, Events & Delegates.

Please read through the entire assignment to get an overview before starting on the individual tasks.

Part 1: The Library

Task 1:

Create a visual studio solution with a library and a console project to test out the example space objects program from lecture 13 (Code available at the end of this document). I recommend you retype the code yourself instead of using copy/paste to get a feel for using the C# language.

Make sure the program compiles and runs and provides the expected output.

Task 2:

Expand on the program. Add additional objects like comets, asteroids, asteroid belts, and dwarf planets. Make sure each of these new objects inherits correctly. Feel free to change how space objects inherits from each other though, don't let the example limit you, just make sure your result makes sense.

Task 3:

Add additional fields to the classes, like orbital radius (assume circular orbits for simplicity), orbital period, object radius, rotational period (length of a day) and object color. Make sure this information is added to the appropriate place in the inheritance hierarchy.

Remember to make accessor methods to get and set these values as needed.

Add methods for calculating a planets position with a given time as input. For simplicity, assume the sun is always in (0,0), and at time 0, all planets are lined up along the positive x-axis (y=0). The same goes for the moons in relation to their planets.
Use the orbital radius and orbital period to calculate the planet's position. Remember that the position of a moon should be relative to the planet it orbits.

Note that to calculate circular motion, you need to use trigonometry. Basically:

X = radius * Cos(angle)

Y = radius * Sin(angle)

The math library in -net want the angles in radians, so if you have them in degrees, multiply by Pi/180.

Task 4:

Resource: http://nineplanets.org/ + Google. A spreadsheet with some key information from nineplanets.org is available on Canvas.

In the main program, create a representation of our solar system by using the classes in the library you developed above. Add all major objects in our solar system (The sun, the planets, the dwarf planets, the asteroid belt, and the moons. For planets with a huge amount of moons, only use the ones where orbital period is listed @ http://nineplanets.org/data.html).
It is up to you how you wish to represent this internally in the program.

When run, the main program should ask for a time (number of days since time 0), and the name of a planet. It should then print out the details about the chosen planet, including its position (based on the given time), as well as all the details and positions of the moons belonging to this planet. If the user does not enter a planet, the program should instead default to the sun, and print out information about that, as well as all the planets.

Part 2: The Simulation

Task 4:

Add a project for a graphical .net application that should reference the library.

You are free to choose if you wish to make the graphical part a Windows Forms Application, a Windows Presentation Foundation Application or a MAUI application.

Pick what you want, here are some advantages and disadvantages.

- Windows Forms: Windows only, the drawing operations are similar to what you did in the SDK part, so the familiarity may make this one easier.
- WPF: Windows only. Here you need to treat the shapes representing the planets as persistent objects. This is a very different approach to the Windows Forms one, but may make more sense from an object oriented approach.
- MAUI: Multi-platform, can even be written using Visual Studio directly on a Mac, no need for a Virtual Machine. However, this technology is still partly in preview, not all tools are ready for it yet, and the teachers and assistants have VERY little experience with the technology meaning it may be more difficult for us to provide help.

Task 5:

Create a visual 2D representation of the solar system, using the planetary data to make it semi-accurate. Note that due to the fact that the distances are huge and many planets are small in relation, you should implement some kind of scaling scheme for displaying the distances and sizes, so that everything can be represented visually on the screen.

The 2D representation should include at least the sun and all the planets. You should also implement a way of zooming in to a planet to see the moons in orbit around the planet. The exact implementation is up to you, but suggestions include a drop-down box or menu where you can choose a planet, or clicking on a planet zooms in on it, while right-clicking takes you back to the solar system overview.

The system overview should be with the sun in the center, so that the planets can be seen at all times, but you can experiment with another view for the planetary zoom display if you wish, like showing the planet in one of the corners (and thus only showing about ¼ of the moon orbits)

There should be options of hiding and showing text labels (star/planet/moon names), and planetary orbits. When zoomed in on a planet, the full data about that planet can optionally be displayed in a corner.

The moons/planets can either be drawn by using bitmaps, or you can go with a simple filled circle.

Adapt the library as required, but do not drop it. Your applications should still rely on it.

Task 6:

Now it is time to animate the system. Use events and delegates to control the system. The event that the objects subscribe on is DoTick which is fired from a central object. All objects shall subscribe to this event. When this event occurs, the object shall update its position.

Do not use timers directly but make a central class which may use a timer to fire the DoTick event. Showing that you manage to create your own event is part of the requirement. Also make it possible to change the speed of the simulation (Key press, menu, etc…).

Make sure that the simulation runs correctly based on the orbital speeds of the planets.

ASSEMBLY SPACEOBJECT.CS:

```
using System;

namespace SpaceSim {
    public class SpaceObject {

        protected String name;

        public SpaceObject(String name) {
            this.name = name;
        }
        public virtual void Draw() {
            Console.WriteLine(name);
        }
    }

    public class Star : SpaceObject {
        public Star(String name) : base(name) { }
        public override void Draw() {
            Console.Write("Star  : ");
            base.Draw();
        }
    }

    public class Planet : SpaceObject {
        public Planet(String name) : base(name) { }
        public override void Draw() {
            Console.Write("Planet: ");
            base.Draw();
        }
    }

    public class Moon : Planet {
        public Moon(String name) : base(name) { }

        public override void Draw() {
            Console.Write("Moon  : ");
            base.Draw();
        }
    }
}
```

ASSEMBLY MAINPROG.CS:

```
using System;
using System.Collections.Generic;
using SpaceSim;

class Astronomy {
    public static void Main() {
        List<SpaceObject> solarSystem = new List<SpaceObject>
        {
            new Star("Sun"),
            new Planet("Mercury"),
            new Planet("Venus"),
            new Planet("Terra"),
            new Moon("The Moon")
        };

        foreach (SpaceObject obj in solarSystem) {
            obj.Draw();
        }

        Console.ReadLine();
    }
}
```