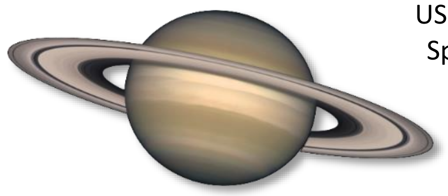
USU CS 4700
Spring 2015

Lua Lander



Introduction

The purpose of this assignment is to introduce you to programming in Lua. You will be creating a simulation to safely land an exploratory vehicle on a distant planetary surface. To do this you will not only need to setup the simulation, but also add in the dynamic logic which takes into account the lander's altitude, velocity, fuel reserves, and thruster strength in order to make a safe approach down to the surface.

Lua Setup

First you need to install Lua. There are many options to do this, I would suggest using one of the following:

- ZeroBrane Lua IDE: http://studio.zerobrane.com/
  - Windows, Mac, and Linux compatible
  - Nice standalone Lua IDE
- Eclipse Lua Development Tools: http://www.eclipse.org/ldt/
  - Develop your Lua code inside of Eclipse
- Lua Extension for Visual Studio 2012/2013: https://babelua.codeplex.com/
  - Windows only

Part 1 – Lander (20 pts)

First you need to create the Lander object which will be used by this simulation. I have created the Planet object for you as an example. You will using a simple object library (classLib.lua) which you can download from Canvas.

Your Lander object should contain the following data members which you should initialize in its constructor:

- velocity – The lander's velocity, with *negative values* indicating motion toward planet's surface
- altitude – Height above the planet's "sea level"
- fuelReserve – Remaining units of fuel
- thrusterStrength – Thruster's ability to counteract the planet's gravitational pull

Your Lander object must contain the following method:

- string Lander:toString()
  - Returns the current state of the lander as a string of the format:
    "Altitude: 15.0, Velocity: -10.5, Fuel: 12.0"

Part 2 – Simulation (35 pts)

Now you will need to create a Simulation object which runs the simulation. Your Simulation object must contain the following members which should be initialized via its constructor:

- lander – The Lander object being used for the simulation
- planet – The Planet object on which the lander will attempt to land
- strategy – The coroutine which implements the strategy on how to attempt to land (Part 3)

Your Simulation object should also contain the following methods:

- void Simulation:updateLander(burnRate)
    - Checks to make sure that the desired burn rate does not exceed the amount of fuel remaining, if so, limit the burn rate.
    - Updates the altitude and velocity of the lander in the simulation using these equations:
      *altitude = altitude + (velocity * DELTA_TIME)*
      *velocity = velocity + (((thrusterStrength * burnRate) - gravity) * DELTA_TIME)*
      DELTA_TIME is the constant time step between iterations of the simulation and is defined inside the provided source.
    - Update fuel remaining by subtracting the burn rate.
- bool Simulation:reachedSurface()
    - Checks if the lander has reached the planet's surface at the landing site elevation (safely, or not so safely)
- bool Simulation:landed()
    - Checks if the lander has reached the surface and did so at a safe landing velocity
- string Simulation:positionString()
    - Returns a string representation of the relative position of the lander to the planetary surface landing site with | representing planet surface (which might not be at height zero!) and * being the lander. For example: " | *"
- void Simulation:run()
    - While the lander has not yet reached the surface
      1) Print the position string
      2) Let your coroutine determine the fuel to burn for the next iteration (Part 3)
      3) Update the lander
    - Check if the lander successfully landed on the surface, if so, print a congratulatory message. If not, print a message informing of the crash.

Part 3 – Strategy (30 pts)

You will need to implement at least one strategy that will allow you to land on Pluto. A strategy is a coroutine which yields the next burn rate to use during the next time step of the descent. This coroutine will not be a member function of the Simulation class, but it will be passed into the Simulation class constructor and stored as a class member.

This strategy *must* be implemented as a coroutine. You may pass as much or as little information (altitude, fuel, gravity, etc) into your coroutine as you need.

To pass data to a coroutine you must do so using coroutine.resume(..) from within Simulation:run(), and to retrieve that data you should use coroutine.yield(..) inside your coroutine. The information at http://www.lua.org/pil/9.1.html might prove useful if coroutines aren't making sense to you.

Correctly implementing, resuming, and yielding your coroutine will give you full credit for this portion (regardless of landing outcome).

Part 4 – Safe Landing (15 pts)

If your strategy allows you to safely land on Pluto with the provided planetary constraints (Pluto will always be a planet in my book!), then you will receive these points. To land safely you must reach the surface at a descent velocity of no greater than -1.5.

Extra Credit Opportunities

- **Option #1 (5 pts):** Make your strategy also work (or create a separate one) to land safely on Mars using the provided planetary constraints.
- **Option #2 (15 pts):** Rewrite your strategy logic in C++ and have it called from inside your Lua coroutine. Include this as a separate folder in your submission.
- **Option #3 (15 pts):** In addition to the assignment, in a separate folder embed your Lua simulation inside of a C++ application so that it starts the simulation and retrieves the strings or sufficient information from Lua so that your C++ application can display all of the output.

Example Output

```
|                                                    *
|                                                    *
|                                                    *
|                                                     *
|                                                       *
|                                                         *
|                                                           *
|                                                              *
|                                                                  *
|                                                                      *
|                                                                          *
|                                                                               *
|                                                                                    *
|                                                                                        *
|                                                                                           *
|                                                                                              *
|                                                                                                 *
|                                                                                                   *
|                                                                                                    *
|                                                                                                     *
|                                                                                                      *
|                                                                                                      *
|                                                                                                     *
|                                                                                                    *
|                                                                                                   *
|                                                                                                 *
|                                                                                              *
|                                                                                           *
|                                                                                        *
|                                                                                    *
|                                                                               *
|                                                                          *
|                                                                     *
|                                                               *
|                                                        *
|                                                 *
|                                          *
|                                  *
|                         *
|                *
|       *
Altitude: -5, Velocity: -10.5, Fuel: 0
Crashed and Burned Sucker!
```