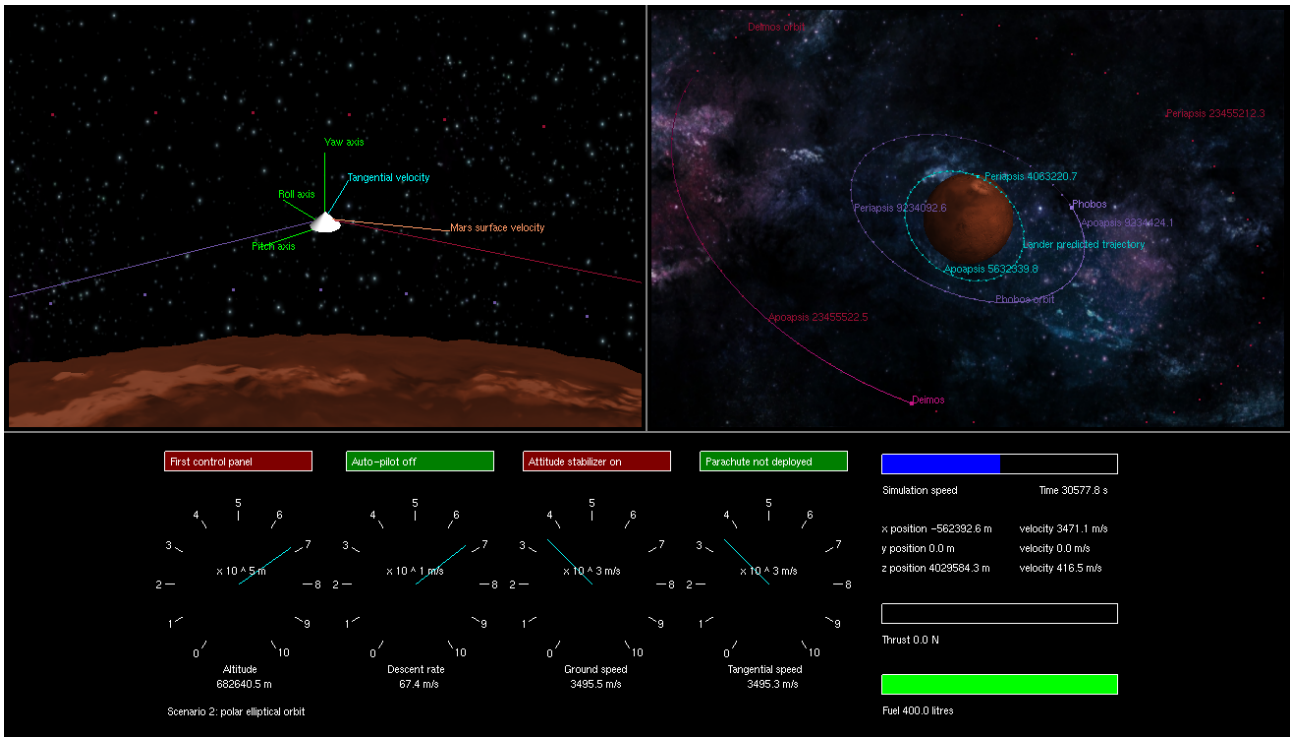
MARS LANDER PROJECT REPORT

Do Vinh Anh Nguyen
dvan2@cam.ac.uk



The simulator in full screen

Index

1/ Overview

This document is meant to be an informal report and a quick 'user manual' documenting some features of the lander and how they work. This Mars Lander exercise is definitely one of the more interesting assignments I have been given this summer. It introduces me to a whole lot of new and exciting fields, for example orbital mechanics, numerical simulation, and control theories.

2/ User guide

2.1/ Installation

- The Mars lander simulator is located at: https://github.com/dovinhanhnguyen/lander_dvan2
- The simulator uses SOIL (Simple OpenGL Image Library) to load texture images, which can be found at: http://www.lonesock.net/soil.html.

  For Windows: The lander program has not been tested on a Windows PC yet, but according to http://stackoverflow.com/questions/13009049/soil-set-up-in-visual-studio-2010 the necessary steps are:

  - Rename libSOIL.a to SOIL.lib
  - Add the path to SOIL.h to 'Additional Include Directories'
  - Add the path to SOIL.lib to 'Additional Library Directories'
  - Add the path to SOIL.lib to 'Properties → Linker → Input → Additional Dependecies' list

  For Mac OS X: The lander program (again) has not been tested on a Mac yet, but in my opinion the SOIL.h file should go to /usr/include and the libSOIL.a should go to /usr/lib

  For Linux: `$ sudo apt-get install libsoil-dev`
- For sound effects, I use irrKlang (which can be found at http://www.ambiera.com/irrklang/). However, I haven't been able to make the audio to work on Windows and Mac. On Linux, the way I did it is:

- Download the zip from the website and unpack it.
- Since I used `#include <irrklang/irrKlang.h>` I renamed the /include/ directory inside the unpacked directory to irrklang, then copy this directory to /usr/include.
- After that, I copied the libIrrKlang.so to /usr/lib.
- Because the lander simulator also uses mp3 files for its audio, I copied the plugin ikpMP3.so to the project working directory (apparently this plugin can only be loaded properly from the current working directory).

- NB: Thanks to this Mars Lander project, I have learnt about how to use external libraries, what shared and static libraries are, how different platforms are structured and a whole lot of other exciting things related to computers that I did not know before.

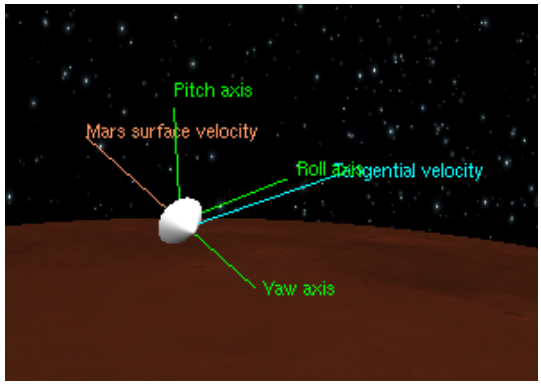
2.2/ Manual control

When the autopilot (see 2.3) is diengagaed, the lander is fully under your control.

- Use **a** to toggle autopilot.
- Use **up arrow** to increase thrust.
- Use **down arrow** to decrease thrust.
- Use **p** to deploy parachute.

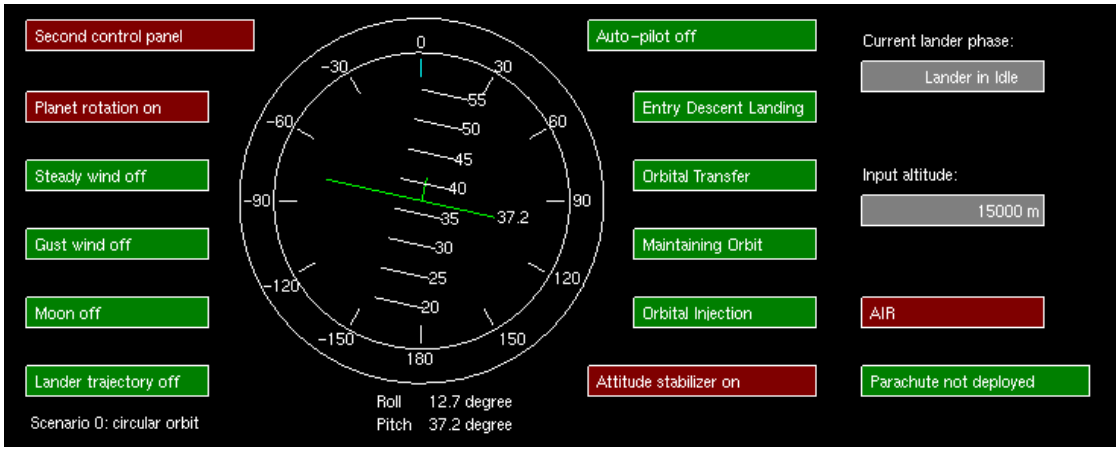
When the autopilot is not active, you can toggle the attitude stabilizer. Enabling it gives you control of how the lander is oriented in space.

- Use **i** to pitch up, **k** to pitch down.
- Use **u** to roll left, **o** to roll right.
- Use **j** to yaw left, **l** to yaw right.
- Use **Home** to point the lander upwards.



In the closeup view, there should be three green axes corresponding to the lander's pitch, roll, and yaw axes. Additionally, you might want to change to the second control panel to read the pitch and roll angles from the attitude indicator.

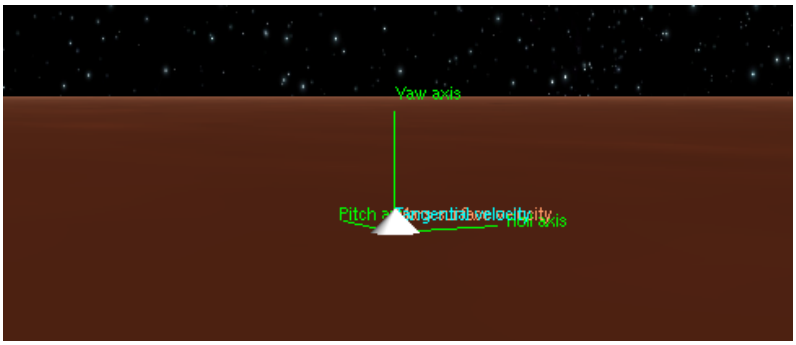
- Use **Page Up** to switch control panel.



There are in total ten simulation scenarios, of which the last three (7 to 9) are about launching the lander from different locations on Mars. When you run one of these orbital injection simulations in manual mode, it is recommended that you first pause the simulator, increase the thrust and then release the lander (otherwise the lander might fall towards ground before the thrust is large enough). Furthermore, the attitude stabilizer should be activated so that you can tilt the lander nose and gain sufficient tangential speed.

- Use **W** to unhold the lander when it is ready

On the second control panel, there is a small lamp indicating whether the lander is still held on ground ('GROUND' in a green box) or the lander is already airborne.

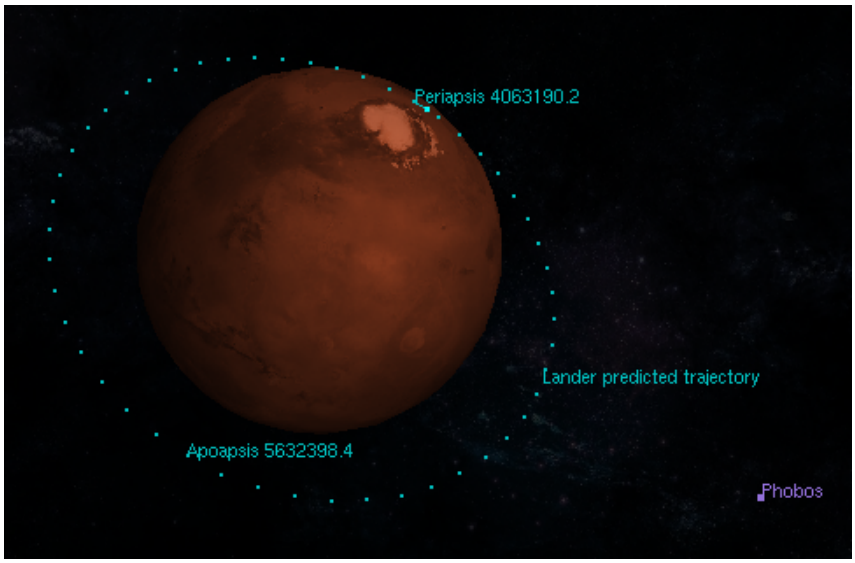


Scenario 7 – polar launch

To help with launching as well as navigating the lander, a predicted trajectory based on the lander's current position and velocty is provided in the orbital view.

- Use **n** to toggle lander trajectory in orbital view.

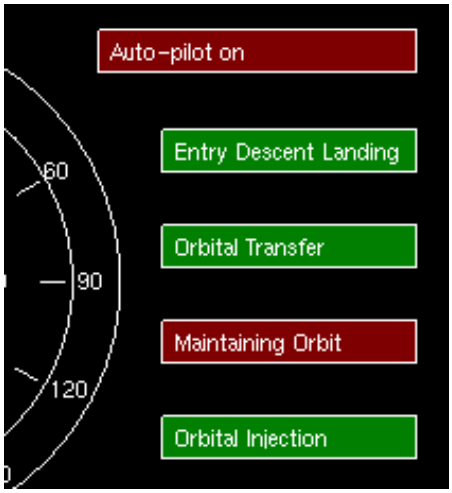
Note that this predicted trajectory is only available when the lander is 15000 metres above Mars surface. It is drawn in cyan and should look like this



If the trajectory clips the atmosphere, the display will turn yellow. If the trajectory results in a collision between Mars and the lander, the display turns red.

2.3/ Autopilot

The autopilot has four modes: Entry Descent Landing mode, Orbital Transfer mode, Maintaining Orbit mode, and Orbital Injection mode. The autopilot only accepts user input when it is in Maintaining Orbit mode.

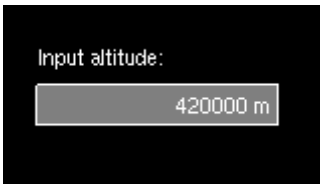


- When in orbit, press **d** to 'autoland' the lander.

The autopilot will automatically decelerate the lander and land it softly on Mars surface.

If you want to change the orbit, you will need to specify the altitude of the new orbit radius. In order to do this:

- Use **v** to inform the lander of the desired altitude. The 'Input altitude' box should turn blue, indicating that it is waiting for your input.
- Type in the new altitude. You can use **Backspace** to erase the units digit of your input, or **Delete** to clear the whole box.
- When finish, press **v** again so that the autopilot knows of the new altitude. The 'Input altitude' should now turn grey.



- Then press **c** to initiate the transfer sequence.

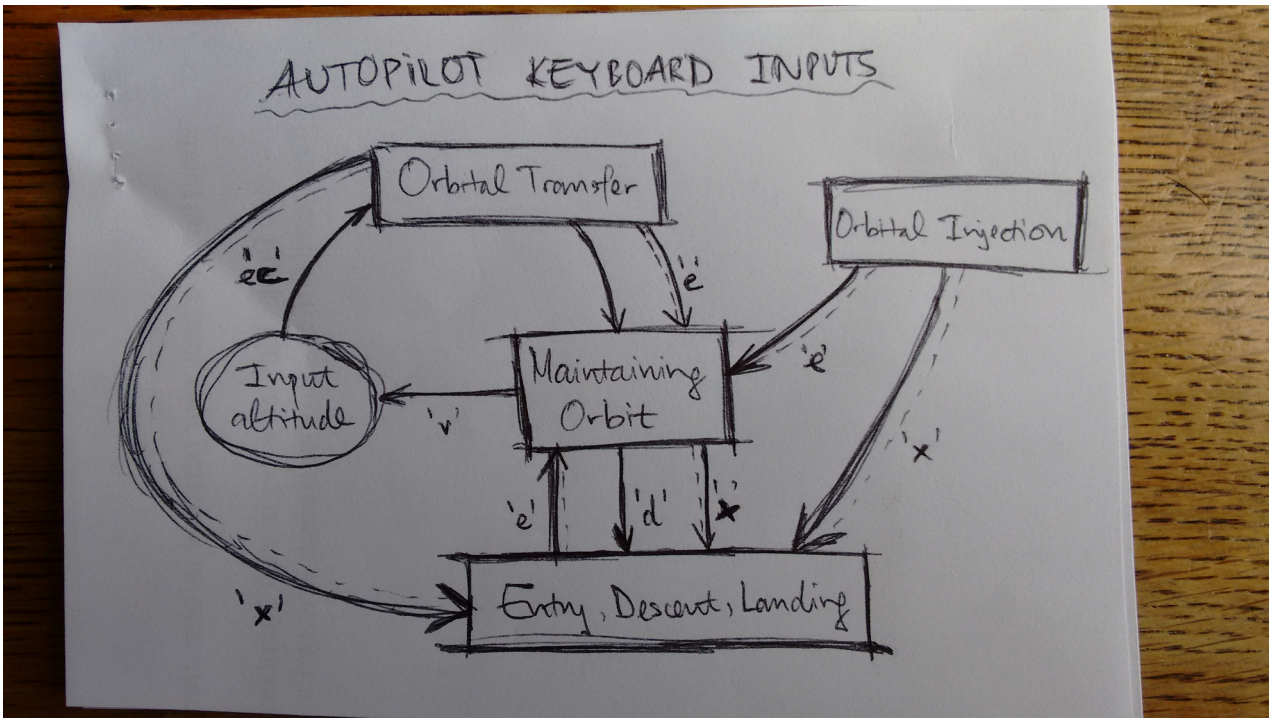
If somehow you find the lander falling towards Mars uncontrolably:

- Use **X** to put the autopilot in its Entry Descent Landing mode.

If the autopilot is not taking in your input:

- Use **E** to reset the autopilot to its Maintaining Orbit mode.

As before, you can still use **n** to view the lander trajectory, but you no longer have control of the thruster, the stabilizer, and the parachute.

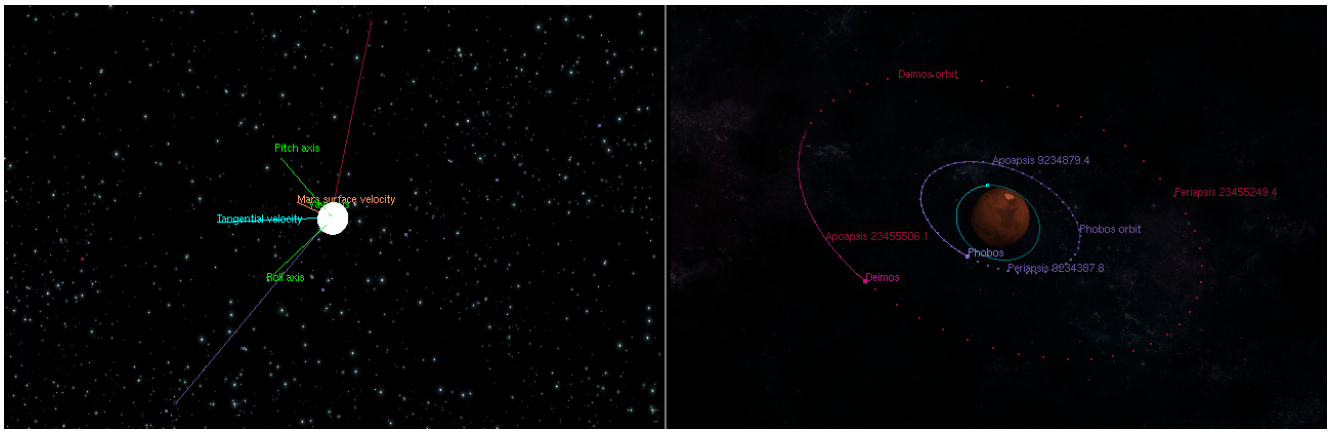


Autopilot keyboard inputs

2.4/ Other features

The simulator also takes into account Mars atmospheric rotation, steady and gust wind on Mars, and Mars moons (Phobos and Deimos).

- Use **r** to toggle planet rotation.
- Use **f** to toggle steady wind, and **g** to toggle gust wind. Note that they cannot be enabled at the same time.
- Use **m** to track Phobos and Deimos as well as view their orbits in the orbital view. (NB this is supposed to distort the lander's orbit, but nothing happens. I think it's either because the gravitational force is too small or there is an error in my code.)



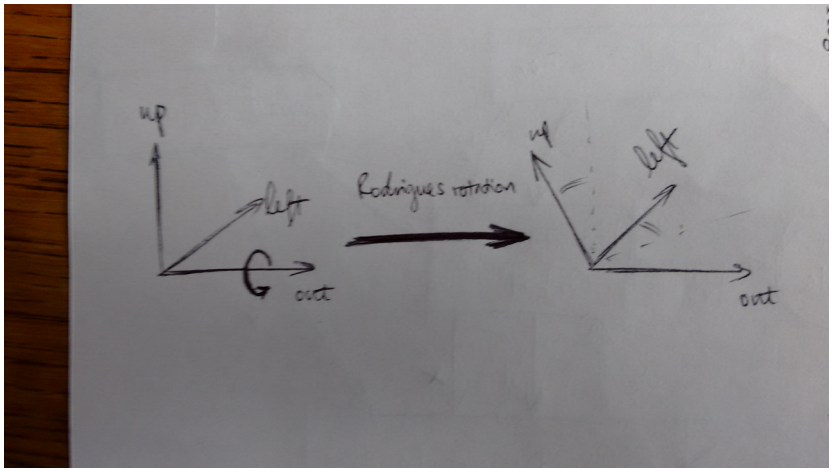
Phobos and Deimos

Appendix 1/ How it works - Attitude Control

The function that handles the attitude stabilizer is in *lander_graphics.cpp*. (NB attitude control is done in closeup view)

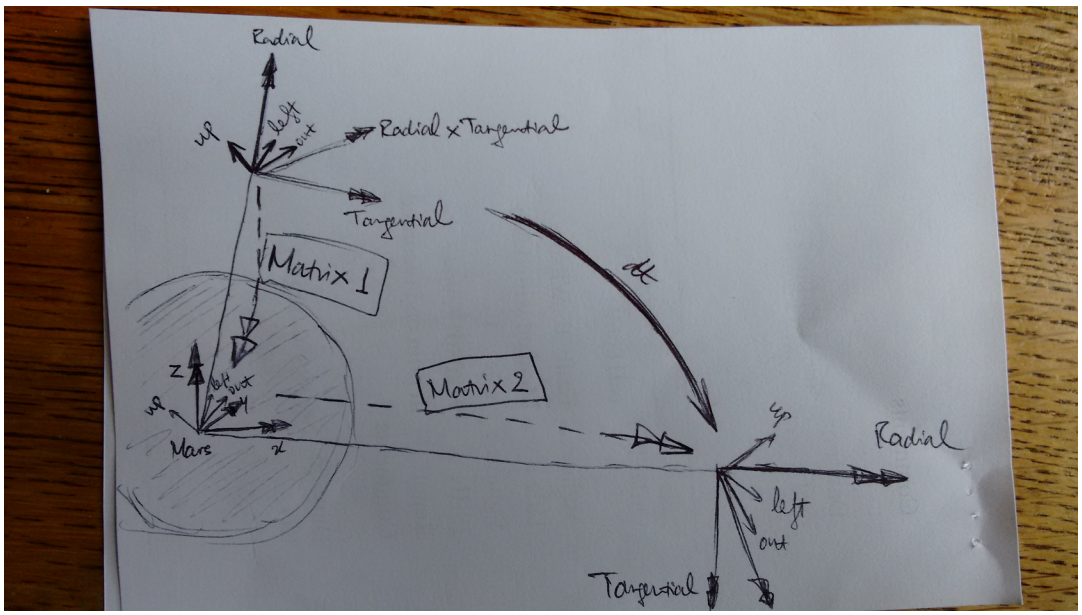
Appendix 1.1/ Manual

In order to pitch, roll, or yaw the lander, the function takes the current lander axes and rotates them to a new orientation according to the command from the user, then decomposes these axes into a corresponding set of Euler angles.



Pitch, roll, yaw

Keeping the lander's orientation fixed in the closeup view (the view which a person sitting inside the lander would see) is a little bit trickier. The way it is done is that the function rotates the lander's previous axes to the reference frame (Mars frame) then rotates them again so that their relationship with the closeup view's axes is the same as before.

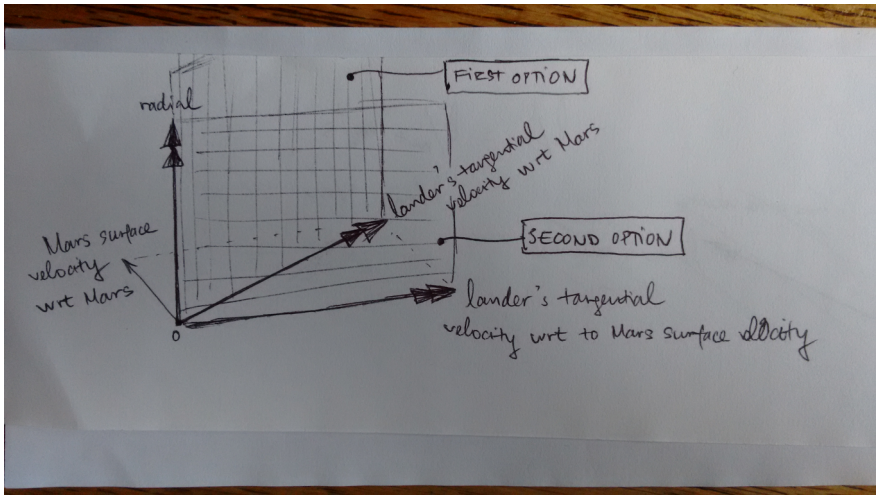


Stabilizing lander's attitude in closeup view

Appendix 1.2/ Autopilot

The autopilot makes use of the attitude stabilizer to control where the nose of the lander is pointing to. The stabilizer in this case allows the lander to pitch up and down in two planes, the first one is defined by the radial vector and the lander's tangential

velocity vector, whereas the second one is defined by the radial vector and the lander's velocity vector with respect to Mars. The diagram below may be more helpful in explaining this:



Attitude stabilizer when used by autopilot

Appendix 2/ How it works - Predicting trajectory

The Kepler_solver class is the one in charge of this. It can be found in the kepler_solver.h and kepler_solver.cpp.

Appendix 2.1/ Solving for Keplerian elements

Assuming we have a two-mass problem, the trajectory of an object moving under the influence of Mars gravity can be determined given its position and velocity. Firstly, we solve for the angular momentum vector, which represents the orbit plane's normal vector. Secondly, we solve for the eccentricity vector, which is the vector pointing from the apoapsis to the periapsis of the trajectory. The magnitude of this vector indicates how 'pointy' the trajectory is.

I adapt the calculations from the suggestions on this website http://space.stackexchange.com/questions/1904/how-to-programmatically-calculate-orbital-elements-using-position-velocity-vecto

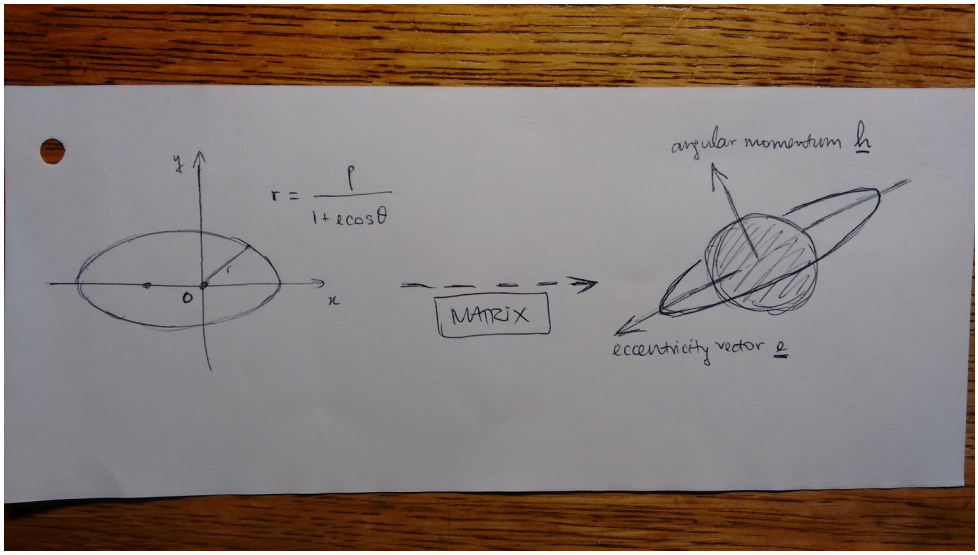
and the formulae on this website http://www.bogan.ca/orbits/kepler/orbteqtn.html

Appendix 2.2/ How trajectories are drawn

After figuring out the attributes of a trajectory, that trajectory is first drawn in a reference plane, where its eccentricity vector aligns with the plane x-axis and its angular momentum vector aligns with the plane z-axis, using the normalized polar-coordinate form

$$r = p/(1+e*\cos(\text{theta}))$$

All of the points on this plane are then multiplied by a matrix to get the trajectory its correct orientation in 3D space.



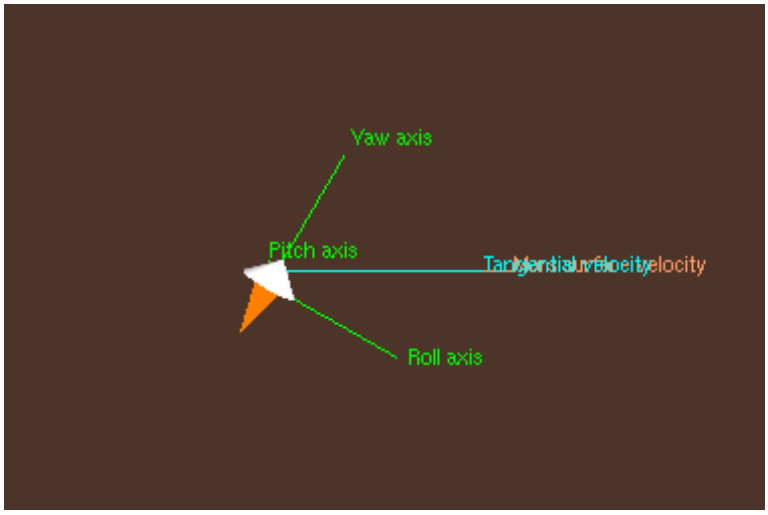
All of the functions that handle the drawing of future trajectories are located in the miscellaneous_functions.cpp file.

Appendix 3/ How it works – Autopilot

All autopilot manoeuvres are determined by the autopilot function in the lander_dynamics.cpp file.

Appendix 3.1/ Ascent guidance

For the first 1000m, the thruster is set at maximum level and the lander is pointed vertically upwards. From 1000m until the lander reaches the highest point of the exosphere, the lander is pitched 30 degrees in order to gain tangential speed.



30 degrees pitch to gain tangential speed

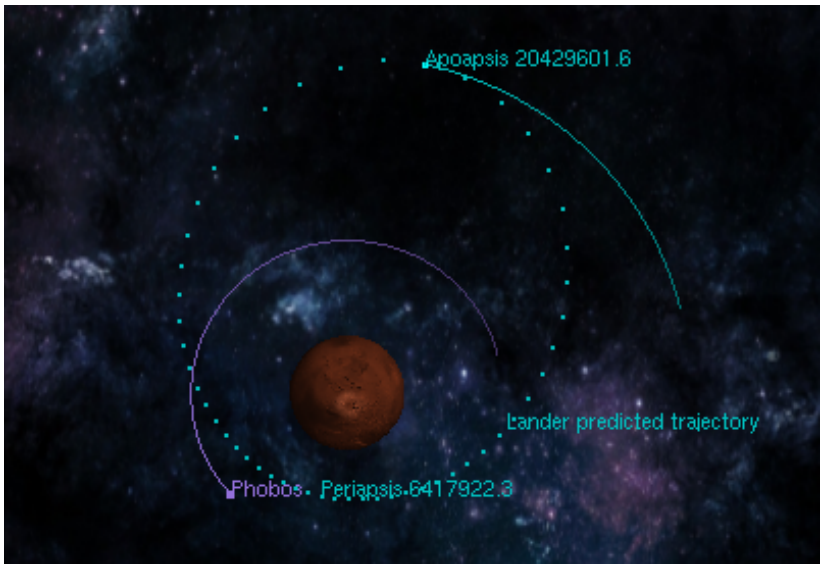
The thruster runs at full power if the apoapsis of the orbit is lower than 300,000m above Mars surface, otherwise it is switched off. When the lander successfully gets out of Mars atmosphere, the autopilot enters its Orbital Transfer mode (see below) to circularize the orbit.

Prepare to circularize lander's orbit

Appendix 3.2/ Orbital transfer

The autopilot is only able to perform a simple Hohmann transfer between two (approximately) circular orbits.

When a desired altitude is acquired, the autopilot will wait until the lander reaches either the orbit's apoapsis or periapsis to ignite the engine, putting the lander in an intermediate transfer orbit. The thruster is then shut down.



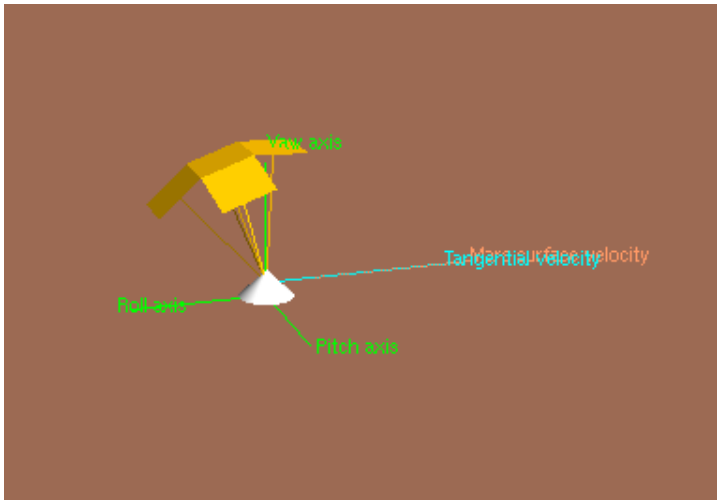
Lander about to enter the elliptical transfer orbit

Once the lander reaches the point where the transfer orbit meets the target orbit, the engine is then switched on once again to circularize the orbit.

I got the formulae required to calculate different speeds involved in different stages of a Hohmann transfer from https://en.wikipedia.org/wiki/Hohmann_transfer_orbit

Appendix 3.3/ Entry, descent, and landing guidance

Once the lander enters Mars atmosphere, the autopilot will use Mars atmosphere to slow the lander down. If the atmosphere drag is not enough, the autopilot will switch on the thruster. At 12,000m above Mars surface, the vertical speed must be approximately 480 m/s or lower in order for the parachute to be deployed later on.



Parachute fully deployed.

After the lander falls past the 100m mark, the parachute is released in order for the autopilot to deal with the engine lag and delay more easily. If the ground speed for some reasons is greater than 0.5 m/s (maybe because the initial tangential velocity is too high or there is wind), the autopilot will pitch the lander nose in the opposite direction to decrease the ground speed. At Mars surface, both the vertical speed and ground speed should be approximately 0.5 m/s and not higher than 1.0 m/s.

Appendix 4/ How it works - Mars rotation and wind

Mars atmospheric rotation causes some drag on the lander. On top of this, we can have either steady wind, which is always 10 m/s in the direction of Mars rotation, or random gust, whose speed follows a Weibull distribution.

The functions used for modelling different types of Mars winds can be found in lander_dynamics.cpp, lander_graphics.cpp and miscellaneous_functions.cpp.

Appendix 5/ How it works - Phobos and Deimos

Mars has two moons, Phobos and Deimos. Their mechanical dynamics are wrapped in the Orbiting_object class, which can be found in orbiting_object.h and orbiting_object.cpp, and are updated in the numerical_dynamics function in lander_dynamics.cpp.

Appendix 6/ How it works – Mars terrain

In order to generate some realistic Mars terrain, I used Blender (https://www.blender.org/) to create a UV sphere and then distorted the sphere (according to Mars topology map) using the Displacement Modifier.

In order to load the .obj file exported from Blender, I adapt the simple .obj loader from http://openglsamples.sourceforge.net/ and improve it a little bit to take in the UV coordinates data from .obj files. Mars texture (loaded from images using SOIL library) is then mapped on top of the 3D model to make things look prettier.

The Model_obj class (model_obj.h and model_obj.cpp) is responsible for handling .obj files and drawing 3D models onto the screen.

More on .obj format can be found at
https://en.wikipedia.org/wiki/Wavefront_.obj_file

Mars terrain

Appendix 7/ Room for improvement

- The orbital reentry guidance is pretty simple right now. I would love to implement a better autopilot that can actively change the controlled parameters (Kp, maybe with the addition of Ki and Kd) to minimize fuel consumption, time taken, and maximum acceleration.
- An analytical way to minimize the effect of the engine's lag and delay, instead of changing the value of Kh and Kp until they work.
- A more accurate orbital transfer stratergy.
- Draw Mars terrain when the lander is below the transition altitude and do some sort of collision detection between the lander and a hilly Mars surface.
- A rendezvous maneuver where the lander attempts to dock to a space station.
- A skydome or some sort of environment mappings to make an impression that we are actually in space.

Cambridge, 26 September 2016