About the IA:

* Strategy
* Detect horizontal surface for landing
* Compute trajectory
* Moving to a selected target

1. Code organization

Controller package: IA logic.

Pidcontroller package: implementation of a pid controller (one input and one output).

Path package: code responsible of generating a path that Mars\_Lander can follow.

1. Strategy

First, we thought about following a simple strategy which was to make the lander converge at an X position which corresponds to the target location, then going slowly to land safely. But this strategy is not general and kind of slow. A better approach is to compute a trajectory that the lander can follow. Once this trajectory is generated the lander will converge to each target (x,y) at a time. The finale target will be the landing position.

The lander needs to converge to x and y to be able to move to the next coordinates. The lander converges to x and y separately. For instance, let us assume that the Lander is in P0 = (0,0) and is moving to P1 = (25,45). If the lander converges first to y = 45, the next target coordinate won’t be set until he converges to x = 25. This is since we used an independent PID for each kind of movement (for x and y).

1. Detect horizontal surface for landing

The horizontal surface is an array of coordinates with the same Y value. With a for loop, once we detected that some consecutive coordinates have the same Y value, we exit the loop and compute the limit coordinates which are : pA.X and pB.X := pA.X + Terrain\_Object.getLength;

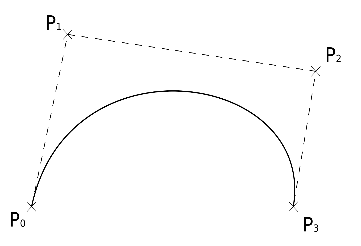
1. Compute trajectory

In order to land safely, mars lander needs to follow a predefined trajectory that can be calculated for every map. We implemented an algorithm that computes a famous curve named “ Bézier curve”, it allows us to find a trajectory from the starting point of the lander to the target location which is the surface. Every target is basically a coordinates ( x and y ). To generate this trajectory, we must define 4 control points that limits the surve. Those 4 controls are calculated based on the starting point of the Lander and the final target. The way this point are chosen allows us to follow a safe path without colliding with an unsafe location.

The associated code is found in Path.ads / Paths.adb where we can find the following procedures / functions:

* Procedure comptute & subdivide are responsible for generating an array of coordinates which is the final path to follow by the lander
* An entry get\_curvePoints to get the path
* A draw procedure to enable the drawing of the trajectory, which is good for testing or for the user the visualize the path to follow.
* An init procedure that calculate the 4 control points
* Since the path package needs to know what surface he’s dealing with, a setTerrain procedure is provided to set the terrain once it’s generated.

A visual example of a trajectory using 4 control points:



1. PID controller

We implemented a pid controller in pidcontroller package that will be used while controlling the movement of the lander.

|  |
| --- |
|  |

1. Moving to a selected target:

Once the path is generated, the mars lander can start to follow it. We’ve already defined a kind of interface that enables us to control the lander by only injecting Booleans. These Booleans are up, left and right. Every time the Boolean is true, the corresponding procedure will be triggered, and the lander will move. With this, we used two pid controllers that outputs Booleans to control the lander.

A first PID was used to control movement at y axis. This pid named XController works fine, the lander converges to a target Point smoothly without crashing and maintaining its position in space without any problem. This behavior was tested independently.

A second PID was used to control movement at x axis. As before, it outputs Booleans to move the lander to the left or the right. We find it very difficult to tune it and finally didn’t not manage to make it work. The lander converges to a target point but then starts to oscillate too much. This behavior makes a safe landing impossible.

The IA doesn’t know anything about the dynamics of the lander, the tuning of the pid and the generated coordinates to follow are the only thing that guarantees a smooth behavior of the lander.

1. Conclusion about the IA:

As a system, the IA is not perfectly working and that is due to the pid responsible of the converge to an x coordinate. But as components of the system, the trajectory is well computed, and the XController works fine.