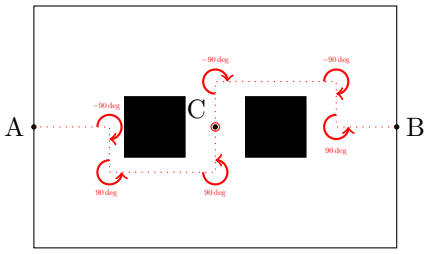
1. I, Robot

This summer I had the pleasure to work with three very gifted students on an ambitious project to code our NAO robot[1], *Astro*. The first goal was to have Astro perform some fun autonomous tasks: playing word games, moving around obstacle courses picking objects, or reading music from a partiture and playing it for us. We are even planing to write code that will allow Astro to look for a set of chess pieces, place them all on a board in the right position, and physically play against us.

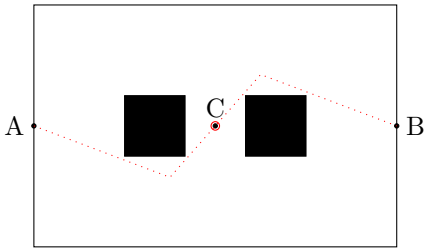
One of the advantages of a NAO robot is that it requires very little knowledge of actual programming to code it. The students took a 13-hour online beginner course on `python`, and in about one week they were able to control most of Astro's basic functionality. This allowed them to focus on the *mathematics* alone. In that sense, they had to spend some time reviewing their background geometry and algebra: sequences, lines and circles, systems of linear equations, polynomials and their roots, nonlinear equations and their solutions, polynomial interpolation, etc. They also needed some Calculus (nothing more complicated than derivatives and tangent lines), and some basic statistics.

This allowed them right away to solve some fun problems in obstacle avoidance, for example.

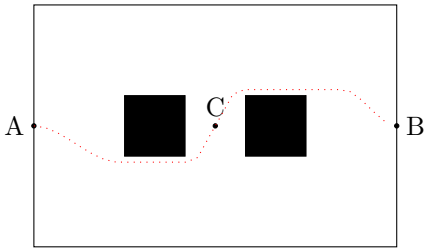
The diagram shows two ways in which the robot solves the obstacle race. The *easy* way is by walking forward in a fixed direction, then stopping, rotating in place to look in a new direction, and walk straight again. This is efficient, but too robot-like. We can do much better: by employing simple interpolation techniques, the robot learns to take a smoother approach, reminiscent of how humans walk.



A naïve straightforward way



A smoother straightforward way



A better way to do it

Figure 1. Simple obstacle race. Go from A to B, picking and object at C in the way.

[1] `http://www.aldebaran.com/en/humanoid-robot/nao-robot`