

SWS assignment: Computer-controlled Kart Racing

1. Context
2. Safety First
3. Hardware basics
4. Software basics
5. Assumptions
6. The assignment



Context

KidsGoKarting Corporation is planning to roll out kart-racing circuits in amusement parks, in which kids (ages 6 to 10) can compete for fun.

The motion of the karts, which are like miniature race cars, will be fully computer-controlled — although still subject to the laws of physics; for example, a fast-moving kart cannot halt instantaneously.

The junior kart drivers will have an illusion of control in that they can accelerate, brake, change direction, etcetera, but all such driver “control” actually just generates a data stream of input to a central processor whose output stream forms the actual control of the karts.

Safety first

Crashes might happen in two ways: a faster kart could crash into the back of a slower kart ahead of it, or a kart switching lanes could crash into the side of a kart in the next lane.

Physical limitations of the set-up already guarantee that karts cannot go off track, and also that kart drivers will not be subjected to potentially harmful strong forces, as long as it can be ensured that there are no crashes between carts.

It is imperative that the karts will not crash — for the safety of the kids, but also to prevent damage to the karts. Whatever the input to the processor, its output must be such that there can be no crashes.

Hardware basics

There are sensors informing the control processor every millisecond or so of the position and speed of the karts, which may be assumed to be known to sufficient accuracy. The processor can transmit a “reference speed” to a kart, telling it to accelerate or decelerate to that speed. How fast and to what speed the karts can accelerate and other such physical characteristics are dependent on the make of the karts and coded into a kart-modeling module; for the purpose of this assignment that module functions like an oracle.

The circuit track consists of a number of parallel lanes, and a kart stays in one lane until the processor directs it to switch lanes — which the processor should only do if it is safe.

Software basics

The software process as envisaged is a simple control loop:

- Find limits on safe speeds, using a safety margin.
- If (close to) unsafe, slow down¹ one or more karts (sensibly²).
- If karts want to change lanes, let them if it is safe.
- If karts want to change speed, let them if it is safe.

The module determining the safe speeds should ensure that crashes can always be avoided by appropriate decelerations.

¹ Karts are never speeded up for safety purposes, only slowed down.

² For example, if a kart is slow and a faster kart is about to crash into its back, only the faster kart should be slowed down.

Assumptions

The following may be assumed:

- The processor is sufficiently fast and reliable.
- The hardware (sensors and controls) are also reliable.
- The “oracle” module faithfully models the physical karts.

Under these assumptions, actual physical safety — the absence of crashes — can be guaranteed by proper software control. The control should not be too cautious; otherwise the whole thing is not a fun experience. Instead, the control should allow going to the limits — with a comfortable safety margin.

The assignment

Specify an *internalized* notion of safety, that is, some notion of limits, or an envelope, such that the basic control loop can always keep the system in a safe state — a system invariant.

This safety spec will eventually become part of the documentation that is required to get a permit for this amusement park attraction; it will be scrutinized by safety experts and engineers.

Different circuits will use different lay-outs and different makes of karts, so the specification must be general. The design of the software control loop as sketched is preliminary and may need adjustment, so the spec needs to be adaptable.