Why Force Feedback In Computer Simulators Does Not Work

Over the last decade we have seen dramatic improvements in PC hardware power, quality of video presentation, number and accuracy of simulation models and yet one thing still fails to impress. That's right, force feedback, or in short, FFB device. As a consumer product it has been around since mid 80's and greatly improved along the way. Steering wheels, joysticks, aircraft yokes - they all seem to push, pull and vibrate but still don't come close to feeling like controls in actual car or airplane.

So what is going on here? Is this a problem of design, manufacturing or the whole concept?

I have designed and built a number of FFB systems completely from scratch - both mechanically and electronically. These steering wheels are 10 times stronger and faster then consumer ones. These systems have low inertia, low friction and two orders of magnitude higher force control accuracy than what came before them. They can respond to force commands and send position data to PC up to 1000 times a second. So do these advanced wheels feel 10 times more real then the consumer wheel when plugged into a typical PC simulator? No. They are more powerful, fast and smooth but no more real...

Maybe the problem of realism lies somewhere else but not in helical gears, belt drives and precision torque control? Maybe somewhere along the whole concept of simulation there is something that just does not work is it should?

I will try to get down to the very basics of how we model the physical objects in virtual world and then interact with them from our real world. It is easier to work from an example then from formulas, so...

Let's consider a simple road car simulator. Road car - because its dynamic modelling is easier than aircraft or submarine. Simple - because we would strip everything nonessential from it leaving just bare minimum - just enough to understand the concept. It won't need to travel fast so we take out the engine. It won't need to slow down quickly so we remove the brakes. We will not slide into tight corners so we leave just one steerable front wheel. Since the cornering forces are low, even steering rack would be an overkill so we will leave it out too.

We have one and only push trike simulator!

Apollo has been sent to the Moon with a guidance computer that had 2 Kbytes memory and 2 MHz CPU speed. Surely a modern computer with 4 GBytes memory, four cores running at 3 GHz each and a steering wheel with special profile helical gears and dual motors can simulate this trike so well that were you to close your eyes you'd feel like you are 3 years old again? In reality our imaginary simulator feels nothing like real trike. Why?

Let's start from the beginning and look at how our push trike simulator has been written. The way most (or maybe all?) simulators are.

We have created precise mechanical model of the frame and all other important parts. We calculated or measured all the fixed to the frame and and moving masses, their moments of inertia, locations of their centres of gravity. We have placed constraints on the way these parts can move, rotate or slide with relation to each other. We have written down formulas tying together acting forces and movement of these parts including friction, elastic compliance and viscous damping. We have measured aerodynamic drag, and even lateral forces acting on each part - including the points where these forces are centred. We have run the tyres on a drum machine and plotted cornering force and aligning torques at all imaginable loads.

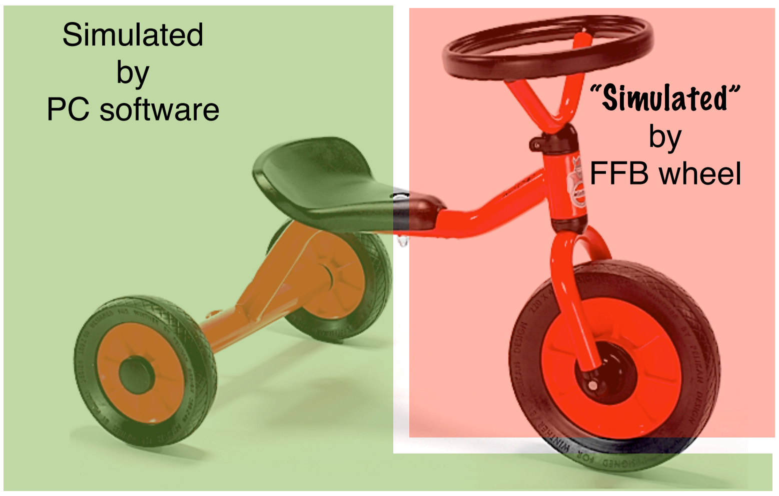
We run the simulator and test drive it. It still feels nowhere like a real thing. Why?!

Let's see how we handle force feedback in our simulator. Any FFB wheel works in a very simple way. It receives force demands from the simulator running on a PC and sends back reports with accurate position data to let simulator know what way and how much we have turned it. It sounds obvious but let's point this out again: simulator sends force commands to the wheel and receives position data back from it.

How do we decide what force to sent to the FFB wheel from our sim? We find the grand total of all forces acting on the steering axis based on front tyre vertical load, rotation speed, position angle. How do we know the front tyre position angle in relation to the vehicle? Simple! We get it from the steering wheel, right? We don't even have to divide it by the steering rack reduction ratio since we don't have a steering rack.

Let's step back and repeat this again. We send the forces acting on a car front wheels to the FFB steering wheel and moments later we receive back new position of our car front wheels. Doesn't this strike you as strange? Maybe not, so let's focus on this.

We have offloaded the whole process of simulation of the front wheels and steering rack dynamics onto the FFB steering wheel. What should have been precision simulation of acceleration, damping, friction of the front wheels steering movement is now outsourced to a device with a few plastic cogs and less then 7 bit force accuracy.



But even the best steering wheel is not designed to simulate anything. It just moves when you ask it to. If you hold it tight, it does not move. That's all. You might argue that wheel has some natural moment of inertia and damping and they can be considered scaled down properties of the front wheels. No, the moment of inertia of any FFB steering wheel is less than even our simple trike's steering and front wheel moment of inertia - let alone the inertia of front wheels and steering rack of actual car even reduced by steering rack gearing.

OK, moment of inertia simulation is pretty obvious - we have two massive front wheels on any car and they need some good effort to make them accelerate. They have "weight."

But why damping? Cars don't have dampers in steering systems. Sure, they don't but requirement for damping simulation comes from natural damping in steering rack and existence of gyroscopic couple - the result of forces acting on a front wheel spinning around horizontal axis and contact patch moving towards one side of the wheel while cornering or just compliant suspension (variable camber.) The resulting counterforce is proportional to the speed of steering wheel rotation. Which in the end is equivalent to damping. Gyroscopic couple damping effect is proportional to the vehicle travelling speed and is one of the reasons why steering "stiffens" with higher speeds and motorcycles remain stable at high speeds.

But we have digressed. We have concluded that maybe if steering wheel has some damping and inertia it can simulate the front wheels for us?

At best simulation of damping in FFB wheels is mediocre. Even worse, no consumer wheel I know of can simulate inertia. This is due to the fact that it requires very accurate angular acceleration estimation. This is just not possible with simple encoder even with few thousands counts per wheel turn.

And now is the worst bit. As soon as you gently place your hands on the wheel its damping and inertia increase several times. You grab the wheel tighter and suddenly the car has ten times more massive front tyres. No car in the world has tyres with variable moment of inertia!

So what would be the "right" way of getting around this? How can we simulate the car front wheels inside the sim? It won't be a problem if we had *all* the necessary input data - all the forces acting on the wheels, their position, moments of inertia, etc - and wouldn't let anybody else interfere with the accurate simulation.

Remember that we have noted earlier that FFB steering wheel traditionally receives the force demands from the sim and sends back the wheel position? One way of solving our simulation problem would be to turn it all on its head and instead *send position demand to the FFB wheel and receive back the force applied by driver*. From first glance it does not make sense. But look how the real car works. It presents you the steering wheel in a certain position and then you apply force to it in a hope that it will move where you want it to be. Then car then moves the steering wheel through its tyres according to physics laws. The wheel is happen to be in a certain position and you apply forces to it to make it change this position.

You control the car by applying forces and not by placing the wheel at a certain angle. Think about it. You are turning a corner and going a bit wide. What do you do? Do you add extra 10 degrees to steering? No, you add a little force and the wheel turns a bit more. How much? You probably have no idea and neither do I.

What if your street car had a wheel that had no force on it when you turn it? In other words you would control it by the *position* of the wheel. You probably wouldn't be able to drive it. But it would be perfect for any simulator to model because this is how simulators are written to work.

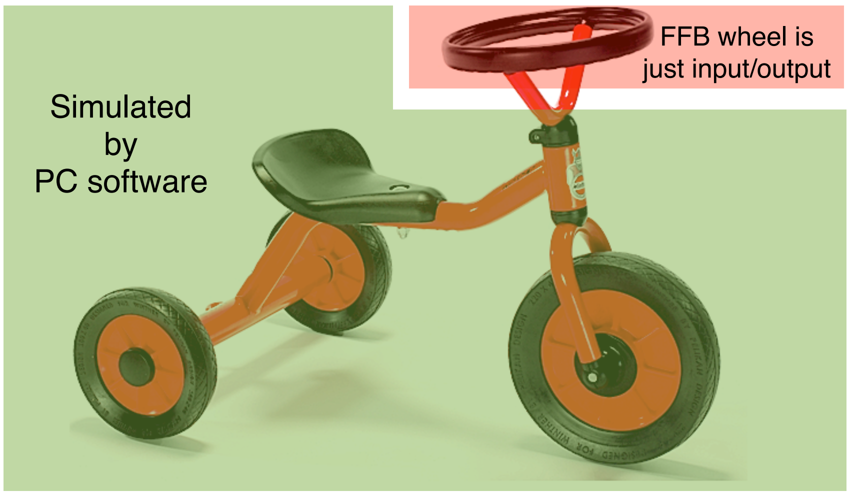
So once again, modern car presents the steering wheel to the driver and driver applies force to it to make the car change direction. This is completely the opposite as how FFB devices made to operate. Can we build such a device? Should it then be called Position Feedback? PFB wheel?

If you are familiar with industrial automation and controls then you probably already guessed that ideal system for our sim steering is a servo system with stiff position control where torque demand from PID loop not only applied to servo motor but also sent back to the sim as force applied by the human driver.

Our car sim physics update loop works in the following way. The software takes the front wheels position, adjusts it according to steering rack configuration - in other words scales it down by gearing ratio - and sends it to the steering system. This now becomes as servo system new holding setpoint. External force applied by the driver is measured and sent back to the sim. The sim combines this steering rack force with other known forces acting on front tyres and then calculates wheels vertical rotation dynamics - angular acceleration based on wheels moment of inertia, rotational speed, and finally arrives at new wheels position. Which is in turn sent to our steering device, becomes a new setpoint, new steering force duly arrives back and everything starts over again.

Perfect harmony. Our push trike sim finally feels as good as a real one.

Can this be done with an ordinary FFB wheel? Yes, to some extent. If you set up a very heavy spring effect (P component) accompanied by critical damping (D component) then you have some sort of a rudimentary servo system. Getting the motor torque out and into PC is more problematic as you can only guess it via displacement from a setpoint (P-component input.)



What is the morale of the story? The simulation software programmers rely on hardware manufacturers for being part of their system while hardware manufacturers never expected to be “in the loop” at all. Few people in the industry understand how “the other side” or even overall system works and there is no available information or discussions on the subject. I hope someday this will change for the better!

Leo Bodnar, July 2011