Surf-to-Shore Visualization

Ryan B. Foss
BAE Systems

Motivation

Visualization via 3D rendering employing heuristic-based[1] physics models can provide insight into environmental behavior that is not always available through conventional analysis and simulation models. A visualization approach can also reveal missing behaviors or corner cases not exposed by a quantitative model. This is particularly the case with highly complex environments such as a surf-zone. And of course, driving simulations have long been useful as supplemental sources of information to model-based design. In the following, we describe a surf-to-shore visualization that extends from a conventional driving simulation (both of these documented and released as part of the project deliverables).

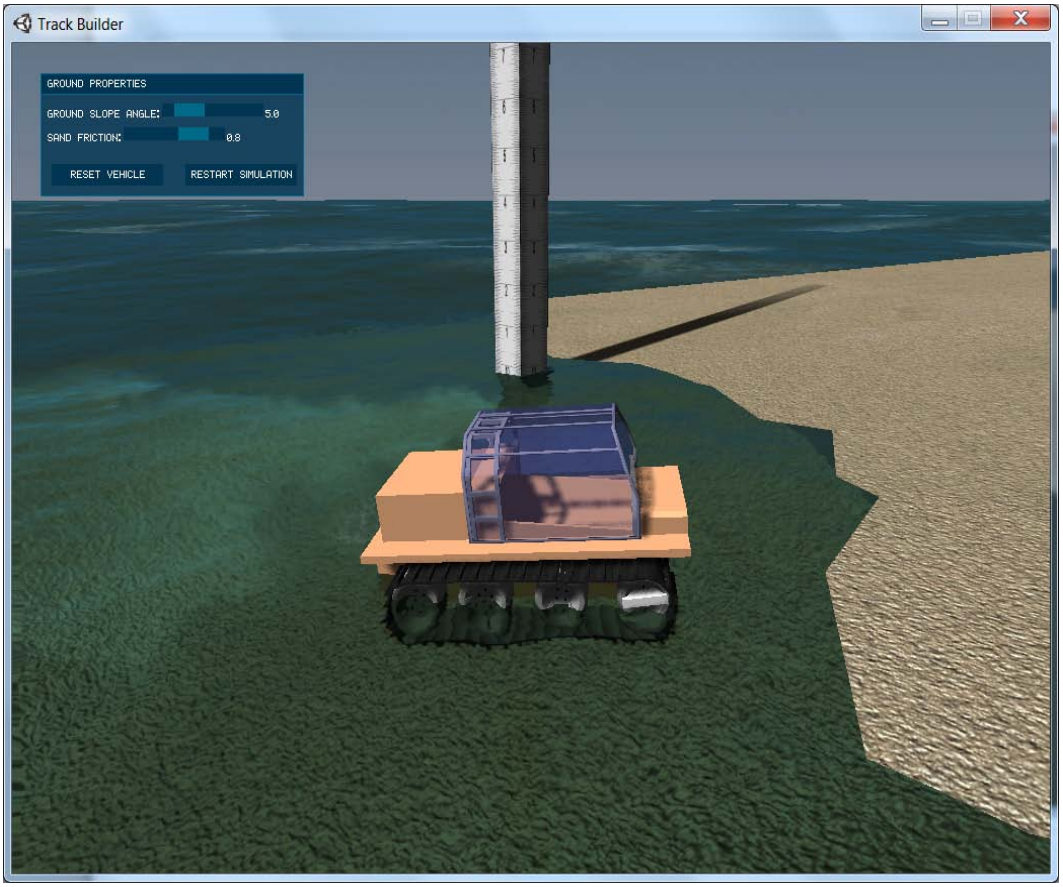

Figure 1: Surf-to-shore visualization context

[1] These are referred to as "gaming" physics models, yet gaming physics often contains a significant fundamental underpinning in real mathematical-based physics. See [1] for example.

Basic Detail

The vehicle is simulated using physics provided by the Unity 3D engine[2]. Physical properties are assigned to the geometry in the scene. Properties include static and dynamic frictions. Forces and torques are applied to a rigid body, which cause the vehicle to move. Buoyancy forces act while the body of the vehicle is in water. Friction properties react between the wheels and linkages making the track, as well as the track with the ground when it is in contact.

Additional interaction and forces have not been added yet. The current state only considers the friction of the land regardless of its interaction with the water.

Water Model (Ocean)

Many real-time variations exist for ocean visualization. In this case, a publicly available open source ocean was used. It uses a typical Fourier FFT to deform a mesh representing the ocean. The ocean is can be queried for wave height at a position, which is what is used for physical interaction calculations in the simulation.

Buoyancy

The buoyancy is calculated based on a simpler representation of the shape of the vehicle, a rectangular volume. This volume is divided into multiple sections and for each a buoyancy calculation is performed. The buoyancy force is based on how much of the volume is submerged.

For the sake of simplicity, the calculations are assumed to counter act the forces of gravity, thus making the volume float on the water when essentially half submerged.

Additionally, the drag impacted on the vehicle changes depending on the amount that is submerged.

Vehicle, how it works

Many approaches were considered and tested when building the vehicle. This particular iteration uses much of the internal Unity physics to provide motion and physical interaction. The tracks are modeled using sectional linkages that are powered by wheels. A simple drive train was added to ramp up and down the control inputs, specifically providing torque to the wheels and propellers.

When the vehicle is in the water, the propellers provide force to the vehicle only if it is submerged.

Note that the Unity vehicle used in the amphibious settings differs from the vehicle used in rigid course and track based simulations, described briefly next.

Terrain Model (Track)

Figure 3 below illustrates the 3D track generation and Figure 4 the visual rendering

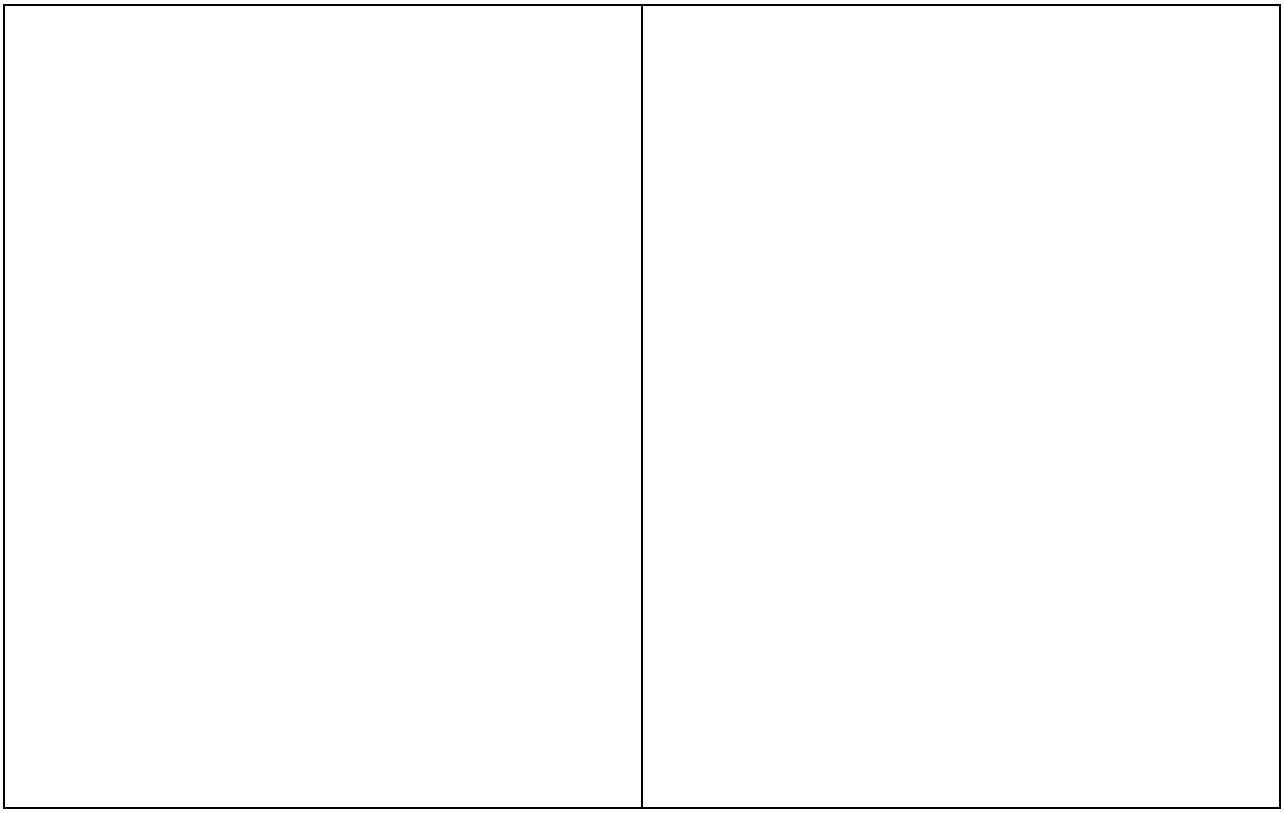

Figure 2: Overview of visual rendering of tracks via a 3D track generation process

Figure 3: Driving visualization where either stochastic roughness (left) or deterministic obstacles (right) track profiles are rendered, with vehicle interactions applied.

The usage of these models are described in two documents

`\Models\Aquatic\features\surf\doc\UserGuide-SurfToShore.docx`
`\Models\Land\terrains\grades_slopes\doc\UserGuide-TerrainBuilder.docx`

Concerns

I attempted a number of different models to represent this effect and settled on this one. It is a satisfying simulation, but not without limitations and frustrations. I would consider it to be a first iteration which provides insight into the complexity of the problem. Additionally, I have much better understanding of the physics engine Unity uses and the methods available, as well as potential limitations it contains.

The physical interaction between the linkages representing the rubber track and the powered wheels produces a very satisfying simulation, but its relation to a real world interaction is questionable. At the moment, the tracks limit the vehicle speed regardless of inputting larger torques. I suspect this is because the wheels are pulling and pushing, and because the track is constructed the way it is, it's like pushing a rope.

The drive train and propeller drives are separate from each other, and produce the same forces regardless of being engaged or not. For instance, if the engine is powering the propellers and tracks it could be assumed that the power is divided between the two systems. The drive train does supply some ramping and smoothing to the inputs as the engine is engaged and disengaged.

Steering is a very simplistic representation that does not work well.

Not Addressed

A number of elements are not addressed in this simulation.

- Sink,

- Slip,
- Impact of water on the ground with regards to traction,
- Steering,
- Wave direction and body movement

Future Improvements

There are a number of ways the simulation could be improved. The terrain could be modeled with various sections or transition zones with different physical properties, such as slippery earth closer to the shore.

The drive train needs improvement to provide more appropriate torques to the tracks and propellers.

Investigate and improve the track model so forces are translated through the system properly.

Conclusions

These kinds of tools are useful for rapid validation of high level design performance on test tracks and in aquatic environments.

References

[1] J. Tessendorf, "Simulating ocean water," *Simulating Nature: Realistic and Interactive Techniques. SIGGRAPH*, 2001.

[2] "Unity (game engine) - Wikipedia, the free encyclopedia." [Online]. Available: http://en.wikipedia.org/wiki/Unity_(game_engine). [Accessed: 12-Feb-2013].