Computer-Controlled Cars in Vamos

Sam Varner
snick-a-doo@comcast.net

January 14, 2012

Battling an opponent can be more fun than racing alone. In order to make the opponents challenging, but not invincible, they need to drive like a person would. This document describes the techniques used to make computer-controlled opponents in Vamos.

1 Strategy

From the beginning, I wanted the robots to have the same interface to the cars as a human player. Robots can operate the throttle, brakes, steering, clutch, shift gears and that's it. The robots *do* have some advantages over human players. They have detailed information about the track and about the car's current state. They can process that information quickly and operate the controls precisely.

The strategy for making competitive robot opponents is to calculate a good path around the track, then to operate the controls to follow that path at the highest possible speed.

2 The Racing Line

It's easy to take a picture of a track and draw a reasonable racing line. You approach the turns on the outside and take a smooth line that cuts across to the inside and then back to the outside. If there are several turns close together, you have to compromise, making a line that would not be ideal for any one turn in isolation. But even for complicated sections, it's easy to freehand a decent racing line. But how can we describe this process precisely enough to allow a computer to calculate the racing line?

As an example, take a look at the track in figure 1. If you were trying to set a fast lap time, you would stick to the right-hand side of the road down the long straight and make a smooth curve through the first two left turns. You would cross the track to set up for the right-hand turn 3. This turn is followed closely by the sharp left-hand turn 4, so you can't take the ideal line through both; you need to compromise the exit of 3 to make a decent entrance to 4. After that

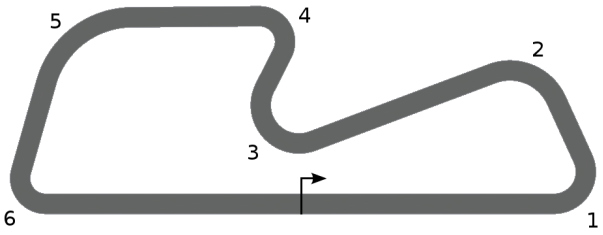

Figure 1: A simple example track. The total length of the centerline is 1133 m.

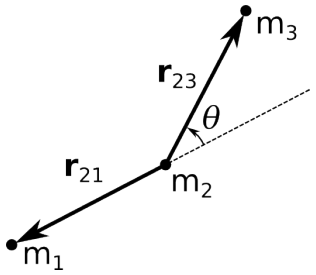

Figure 2: Three nodes on the calculated racing line. The angular displacement θ produces a restoring torque.

you sweep through 5 and set up to make the widest possible curve through turn 6 so that you carry as much speed as possible down the long straight.

From this exercise it seems that a good racing line is one that minimizes curvature. We can find a path that minimizes curvature by simulating a chain of masses on spring-loaded hinges that loops around the track and connects to itself. We will start by placing the masses on the centerline of the track and constrain them to stay on the track surface. As we propagate the system forward in time, the springs will open the hinges and the masses will shift until we approach a smooth curve that minimizes the energy stored in the springs.

The force acting on a given mass m_2 can be calculated from the relative positions of its neighbors as shown in figure 2. Let's start by assuming that closing the hinge produces a torque proportional to the angular displacement.

$$\mathbf{N}_2 = \mathbf{r}_{23} \times \mathbf{F}_3 = -k\theta\hat{\mathbf{n}} \tag{1}$$

where $\mathbf{r}_{23}$ is the vector from m_2 to m_3 and $\hat{\mathbf{n}}$ is the unit vector in the direction of $\mathbf{r}_{23} \times \mathbf{r}_{21}$. The force on m_3 is then

$$\mathbf{F}_3 = -\frac{k}{|\mathbf{r}_{23}|}\theta(\hat{\mathbf{n}} \times \hat{\mathbf{r}}_{23}) \tag{2}$$

We can calculate $\theta\hat{\mathbf{n}}$ from the cross product of $\mathbf{r}_{23}$ and $\mathbf{r}_{21}$.

$$\mathbf{r}_{23} \times \mathbf{r}_{21} \quad = \quad |\mathbf{r}_{23}||\mathbf{r}_{21}| \sin(\pi - \theta)\hat{\mathbf{n}} \tag{3}$$

$$
\begin{aligned}
&= |\mathbf{r}_{23}||\mathbf{r}_{21}| \sin\theta\hat{\mathbf{n}} && (4)\\
\sin\theta\hat{\mathbf{n}} &= \mathbf{r}_{23} \times \mathbf{r}_{21}/|\mathbf{r}_{23}||\mathbf{r}_{21}| && (5)\\
&= \hat{\mathbf{r}}_{23} \times \hat{\mathbf{r}}_{21} && (6)
\end{aligned}
$$

Since we're trying to model a smooth curve, we expect the bend at each node to be small so that we can use x for $\sin x$. If these conditions are not met then the number of nodes is insufficient to model a smooth curve and should be increased.

$$\theta\hat{\mathbf{n}} = \hat{\mathbf{r}}_{23} \times \hat{\mathbf{r}}_{21} \tag{7}$$

Substituting into equation 2 we get

$$\mathbf{F}_3 = \frac{k}{|\mathbf{r}_{23}|}(\hat{\mathbf{r}}_{23} \times \hat{\mathbf{r}}_{21}) \times \hat{\mathbf{r}}_{23} \tag{8}$$

By symmetry, we find that $\mathbf{F}_1 = \mathbf{F}_3$ and $\mathbf{F}_2 = -2\mathbf{F}_3$ provided that θ is small.

We can define the curvature vector $\mathbf{c}$ at m_3 to be

$$\mathbf{c} = \hat{\mathbf{n}}/R = \theta\hat{\mathbf{n}}/|\mathbf{r}_{23}| \tag{9}$$

where R is the radius of curvature of the racing line at m_3. The angle θ is the angle between m_2 and m_3 from the center of curvature. If m_1, m_2, and m_3 are equally spaced along a circular arc then this is the same as the angular displacement defined above. We expect these conditions to hold if the nodes are sufficiently close together. We can rewrite equation 8 as

$$\mathbf{F}_3 = -k\mathbf{c} \times \hat{\mathbf{r}}_{23} \tag{10}$$

We will use the magnitude of the curvature vector to determine the maximum speed for computer-controlled cars at a given point on the track.

The forces are calculated for each contiguous triplet of nodes. The total force on a given node is the sum of the forces exerted by its neighbor's hinges, and the reaction forces from the force of its own hinge on its neighbors. Instead of allowing the nodes to move freely in all directions, we constrain the nodes to move across the track. This constraint improves stability which allows us to use higher hinge torques which in turn gives faster convergence. The node's motion is damped to improve stability.

Once the total force on each node is calculated a new position for the node is calculated using Newton's laws of motion and the Euler method.

$$
\begin{aligned}
\mathbf{v}_{i+1} &= \mathbf{v}_i + (\mathbf{F}/m - c\mathbf{v}_i)\Delta t && (11)\\
\mathbf{r}_{i+1} &= \mathbf{r}_i + \mathbf{v}_{i+1}\Delta t && (12)
\end{aligned}
$$

After many iterations the positions should stabilize. The racing line can be a bit distorted at very sharp turns because the nodes become close together as they move to the apex. Deleting every other node after convergence helps to smooth it out.

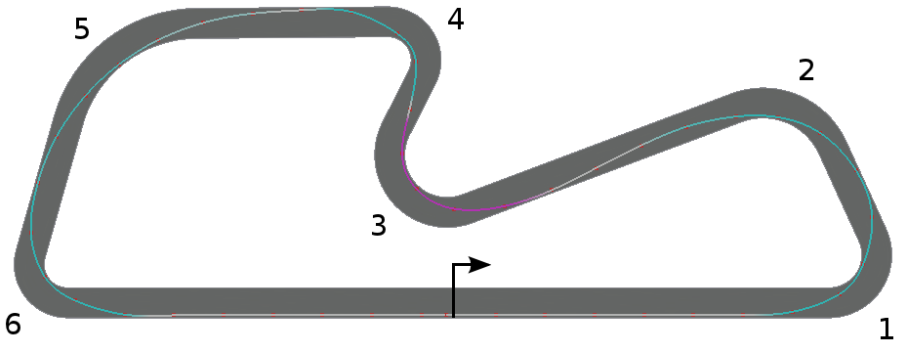

Figure 3: A calculated racing line for the example track. A line of 139 nodes was propagated for 800 iterations. The stiffness was 1.0 Nm/rad and the damping coefficient was 0.1 kg/s. The time step and mass were set to unity. The nodes are shown as red dots. Cyan tinting indicates the degree of left-hand curvature, magenta indicates right-hard curvature.

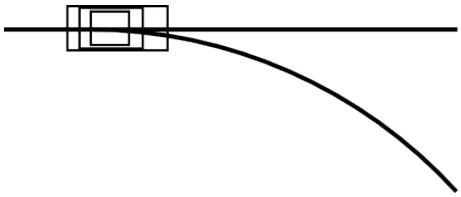

Figure 4: After the steering angle changes from zero to non-zero, the car's new path deviates from its old path slowly at first.

The curvature need not be calculated until propagation is finished, but equation 10 shows that it may be convenient to calculate curvature as the forces are calculated during propagation.

The racing line calculated for the example track shown in figure 3 matches well with what we would expect. However, there is still a little distortion around the sharp turn 4. The curve flattens out a bit after the apex.

3 Steering

The racing line tells where to go, the robot just needs to turn the steering wheel to keep the car centered on the line. However, simply setting the steering angle in proportion to distance between the car's centerline and the racing line does not work well. Changing the steering angle does not immediately affect its lateral position. The car travels in an arc that's tangent to its previous path. The old path and new path diverge slowly at first as shown in figure 4.

This delay between setting the steering angle and appearance of its desired

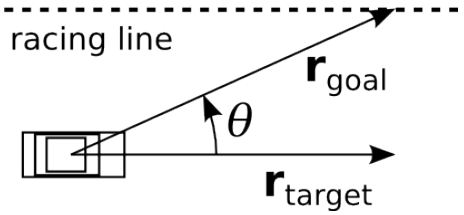

Figure 5: The steering angle is determined from $\mathbf{r}_{\text{goal}}$, which points straight ahead, and $\mathbf{r}_{\text{target}}$ which points to the racing line.

effect inevitably leads to oscillation. The change in steering angle doesn't immediately change what we're trying to control—the lateral position of the car—so the robot changes it some more. Eventually it finds itself quickly traveling across the desired position so it starts steering in the opposite direction.

Control becomes much more stable if we aim for a point farther down the road. Imagine a long pole extending in front of the car. Instead of trying to keep the car on the line, we try to keep the tip of the pole on the line. More precisely, if we define $\mathbf{r}_{\text{target}}$ to be the vector from the center of the car to the tip of our pole, and $\mathbf{r}_{\text{goal}}$ to be the vector from the center of the car to a point ahead of the car on the racing line, then we can use the angle between them as the steering angle.

$$\sin\theta = (\mathbf{r}_{\text{target}} \times \mathbf{r}_{\text{goal}})/|\mathbf{r}_{\text{target}}||\mathbf{r}_{\text{goal}}| \quad (13)$$

$$\theta \approx (\mathbf{r}_{\text{target}} \times \mathbf{r}_{\text{goal}}) \cdot \mathbf{z} \quad (14)$$

4 Speed Control

4.1 Cornering

The centripetal acceleration can be calculated from the car's speed and trajectory as $a = v^2 c(x)$ where x is the distance along the track and $c(x)$ is the curvature of its path at that distance. If the car is driving on the racing line, the curvature can be obtained as shown in section 2. The curvature as a function of distance for the example track (figure 1) is shown in figure 6.

Banking and elevation changes can affect the maximum safe speed for a corner. If a corner is at the crest of a hill, the car will get light and lose some traction. In general, gravity, normal, and frictional forces must sum to the centripetal force. Forces normal to the road can be ignored, so we have

$$\mathbf{F}_{\text{c}} \cdot \hat{\mathbf{q}} = \mathbf{F}_{\text{g}} \cdot \hat{\mathbf{q}} + \mathbf{F}_{\mu} \quad (15)$$

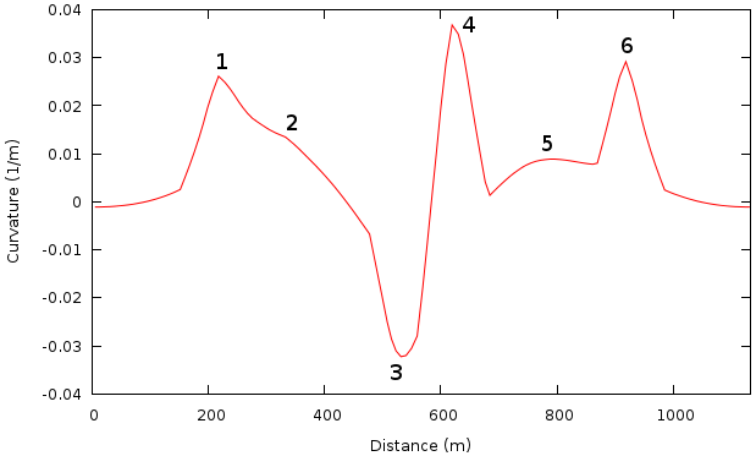

Figure 6: Curvature of the racing line for the example track. Curvature peaks near the turns. Curvature is positive for left turns.

where $\hat{\mathbf{q}}$ is the unit vector parallel to the road and away from the center of curvature. The frictional force is $-\mu F_n \hat{\mathbf{q}}$ where μ is the coefficient of static friction and the normal force given by

$$F_n = \mathbf{F}_\mathrm{g} \cdot \hat{\mathbf{n}} + F_a - \mathbf{F}_\mathrm{c} \cdot \hat{\mathbf{n}} \tag{16}$$

where F_a is aerodynamic downforce. We substitute the following expressions for the forces

$$\mathbf{F}_\mathrm{c} = -mv^2 c\hat{\mathbf{r}} \tag{17}$$

$$\mathbf{F}_\mathrm{g} = -mg\hat{\mathbf{z}} \tag{18}$$

$$F_a = \alpha v^2 \tag{19}$$

and solve for v.

$$v_\mathrm{max} = \left(\frac{g(\hat{\mathbf{z}} \cdot \hat{\mathbf{q}} + \mu \hat{\mathbf{z}} \cdot \hat{\mathbf{n}})}{c(\hat{\mathbf{r}} \cdot \hat{\mathbf{q}} + \mu \hat{\mathbf{r}} \cdot \hat{\mathbf{n}}) - \mu\alpha/m} \right)^{1/2} \tag{20}$$

The fastest way around the track is to stay as close as possible to v_max without going over. Figure 7 shows part the v_max curve for the example track.

4.2 Braking

If the car's speed is less than v_max it should be either accelerating as much as possible to try to reach v_max or braking to avoid exceeding v_max up the road. To find out what situation we're in, we need to know the car's speed as a function of distance under braking.

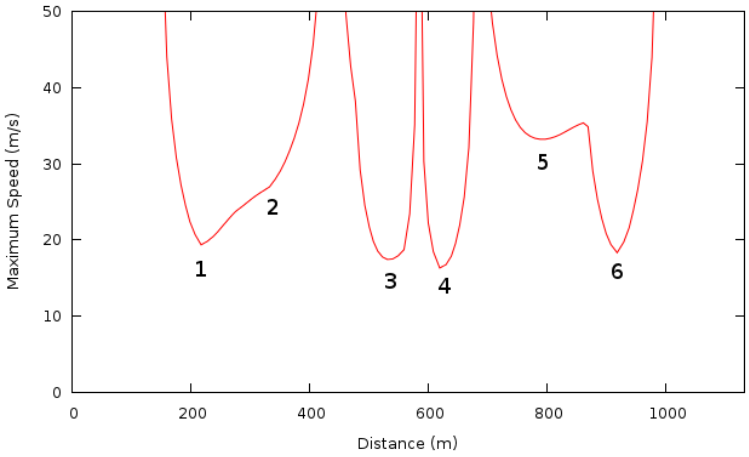

Figure 7: Maximum speed on the example track's racing line for the a car capable of 1 g lateral acceleration.

4.2.1 Constant Deceleration

To simplify matters for the moment, we will assume constant deceleration and ignore the contribution of wind resistance. We will also, for the moment, pretend that braking traction is independent of cornering traction. In reality, if the car is at $v_{\max}$, all of its traction is used up keeping it from sliding sideways; no traction would be available for braking.

The equations for position and velocity as a function of time under constant acceleration are

$$x(t) = x_0 + v_0 t + \frac{1}{2} a t^2 \tag{21}$$

$$v(t) = v_0 + at \tag{22}$$

Since we're describing braking, the number that we plug in for a will be a negative number. To get $v(x)$ we note that $v^2 = v_0^2 + 2v_0 at + a^2 t^2$ and that $2a(x - x_0) = 2v_0 at + a^2 t$. Substituting and solving gives

$$v(x) = \sqrt{v_0^2 + 2a(x - x_0)} \tag{23}$$

Here, x_0 is the position of the car when braking starts. For the remainder of the discussion we will set $x_0 = 0$ and interpret x as the distance traveled since braking started. The initial speed v_0 is the car's speed when braking started. Our final drag-free braking equation is

$$v(x) = \sqrt{v_0^2 + 2ax} \tag{24}$$

Equation 24 defines the boundary between reachable and unreachable points in the x-v graph for track positions ahead. This boundary curve must not exceed

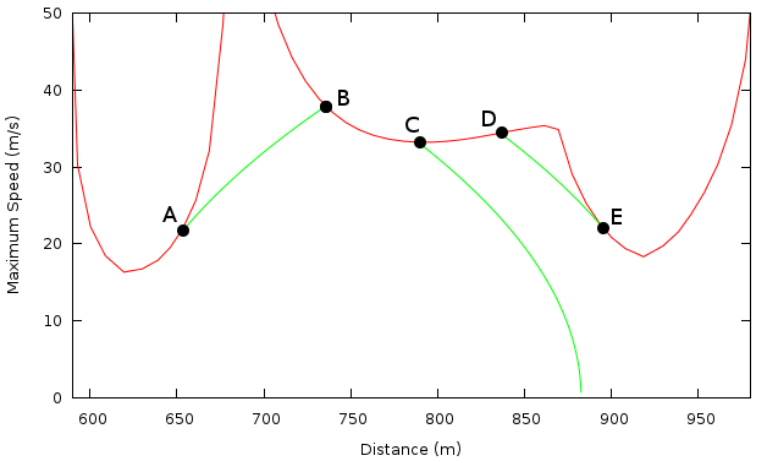

Figure 8: Optimal speed through turns 4, 5, and 6 for a car that can accelerate and brake at 0.6 g. At the exit of turn 4 (point A) the car undergoes maximum acceleration until it reaches $v_{\max}$ in turn 5. At point C, the braking curve stays below $v_{\max}$, so the car maintains $v_{\max}$. Braking begins at point D to avoid exceeding $v_{\max}$ in turn 6. Braking ends at point E. From here the car maintains $v_{\max}$ until it can accelerate at the exit of turn 6.

$v_{\max}$ or the car will slide off the road. We can ensure that it does not by checking the curve during each timestep in the simulation. If the curve touches $v_{\max}$ it's time to brake. Figure 8 illustrates the process.

4.2.2 Other Forces Affecting Deceleration

In general, a number of forces affect deceleration. Aerodynamic drag helps to slow the car regardless of how much traction is available. Gravity may help or hurt, depending on the slope. And, as with cornering, humps and dips affect the normal force, and consequently, the tires' grip level.

We define $\hat{\mathbf{p}}$ as the unit vector tangent to the track in the direction of travel. If the track has a slope such that the gravitational force has a component in this direction, then it will work against the frictional forces that are slowing the car. We can write the total force slowing the car as

$$\mathbf{F}_{\mathrm{b}} = -\mathbf{F}_{\mathrm{g}} \cdot \hat{\mathbf{p}} - F_d \hat{\mathbf{p}} - \mu F_n \hat{\mathbf{p}} \tag{25}$$

Using the expressions for the forces found when calculating cornering speed (equations 16, 17, 18, 19) and expressing the drag force as $F_d = v^2\beta$ we arrive at the expression for acceleration under braking.

$$a_b = g(\hat{\mathbf{z}} \cdot \hat{\mathbf{p}} + \mu \hat{\mathbf{z}} \cdot \hat{\mathbf{n}}) - v^2(\beta/m + \mu(\alpha/m + c\hat{\mathbf{r}} \cdot \hat{\mathbf{n}})) \tag{26}$$

Since this expression depends on the car's position on the track as well as its speed, we can no longer calculate an expression for $v(x)$ as in equation 24.

Instead, we will use equation 26 to predict the car's speed a short distance ahead, and then repeat using the speed and position from the previous iteration. In effect, we run a short braking simulation to see if braking is necessary.

4.2.3 Traction Budget

Return to figure 8. At point C we are on the $v_{\max}$ curve but still decelerating. If we're using all of our traction for cornering then we won't be able use the brakes to stay at $v_{\max}$ as it decreases. (Although aerodynamic drag can still slow us down.) As the car's speed gets closer to $v_{\max}$, a_b must decrease. A linear scaling of a_b works well.

$$a_b \leftarrow a_b(1 - v/v_{\max}) \tag{27}$$

5 Gear Selection

As you accelerate from a standstill, engine power increases with speed. At some point power will reach a maximum and start to decline. Shifting to a higher gear gives a lower engine speed for same wheel speed. Deciding when to shift under acceleration is straightforward. If you can get more power from the engine in a higher gear then shift.

Upshifting can also be useful for saving fuel. If the next higher gear has enough power for the current situation, you can operate at lower revs by shifting to it.

Downshifting is trickier. Shifting to a lower gear under braking can aid the brakes in slowing the car. However, it can also upset the balance and cause the tires on the driven wheels to lose grip. It would make sense to shift to a lower gear when that gear would give more engine power. In practice, that strategy often leads to loss of control. Shifting down one gear when shifting down two gears would give more power, and never shifting to first gear, appears to be reliable.

The current algorithm could be improved upon. A real driver would blip the throttle when downshifting to ease the transition to a lower gear. This has not been implemented for the robot driver.

As anyone who has driven a car with an automatic transmission knows, trying to pick the best gear based on the current conditions can lead to indecisive shifting. A better algorithm might use knowledge of the track to anticipate the need to shift.

Currently the robots use more fuel than they need to. A better shifting algorithm could also improve fuel consumption.