Throttle Documentation

Throttle Controller Formulas

Below is a mathematical breakdown of the **throttle** portion of the racing algorithm, reflecting the logic in the Unity C# code.

We use the following notation:

- $\{\mathbf{p}_i\}$: **list of checkpoint positions** in world coordinates
- v / a_s: **current speed** of the car (scalar)
- $\mathbf{pos}$: **current position** of the car (not given in the final function signature, but used for intermediate geometric calculations)
- $\mathbf{fwd}$: **forward direction** of the car (2D projection, also an intermediate)
- $\mathbf{dir}$: **direction to checkpoint** from the car's position
- θ : **signed angle** between (\mathbf{fwd}) and the direction to the checkpoint
- d / d_c : **distance** to the selected checkpoint
- a_f : **acceleration factor**, controls the expected speed function...
- e_s : **expected speed**
- a_o : **required acceleration**
- t' : **raw throttle** from ([-1,1])
- T_{final} : **validated throttle** (the ultimate output)

1. Computing the Angle and Distance

1. **Compute Lookahead Index** ℓ. this depends on `speedLookAheadSensitivity` multiplied by the current speed, tells us which checkpoint to look at in the list, given by Index ℓ.. Mathematically:

$$\ell = \min\Big(\lfloor \text{speedLookAheadSensitivity} \cdot v \rfloor + 1,\ \text{numberOfCheckpoints} - 1\Big).$$

2. **Direction to the Checkpoint**:

$$\mathbf{dir} = \mathbf{p}_\ell \ - \ \mathbf{pos}, \quad d = \|\mathbf{dir}\|.$$

3. **Signed Angle** θ between the car's forward direction $\mathbf{fwd}$ and $\mathbf{dir}$, in degrees:

$$\theta = \text{Signed Angle}\big(\mathbf{fwd},\ \mathbf{dir}\big).$$

2. Expected Speed

From the code:

$$e_s = \begin{cases} 30.1964649875, & \text{if } \theta = 0, \\ \min\left(\left|\frac{1}{(\text{accelerationFactor})\cdot\theta}\right|,\ 30.1964649875\right), & \text{otherwise.} \end{cases}$$

> - If the angle θ is zero (straight line), the code uses a "full" speed of 30.1964649875.
> - Otherwise, it computes $\frac{1}{(\text{acceleration Factor}\cdot\theta)}$ (absolute value) and clamps it to 30.1964649875.

3. Required Acceleration

From the code:

$$a_o = \frac{e_s - v}{\frac{d}{v}} \;=\; \frac{(e_s - v)\,v}{d}.$$

Occasionally, the code sets a_o to 1 if it is extremely close to zero, ensuring the car moves forward.

4. Raw Throttle Computation

The raw throttle t' is computed as:

$$t' = \text{clamp}\Big(\ \max\!\big(0,\ \tfrac{a_o}{3.4323432343}\big)\ +\ \min\!\big(\tfrac{a_o}{10},\ 0\big),\ -1,\ 1\Big).$$

- $\frac{1}{3.4323432343}$ serves as a scale factor for **positive** (forward) acceleration.
- $\frac{1}{10}$ is applied if a_o is **negative** (braking).
- `clamp(...)` ensures the value stays in ([-1, 1]).

5. Validating And Re-Scale the Throttle

After computing t', the code calls `ValidateThrottle`, which applies:

$$T_{\text{final}}(t') = \max\!\big(0,\ t' \cdot 3.4323432343\big)\ -\ \max\!\big(0,\ -10\,t'\big).$$

This effectively separates positive and negative contributions again, based on different upper limits on acceleration and braking scaling them back to 3.43 and -10 as these are the values the physics model accepts.

NOTE:

You may be wondering why do we compute Raw Throttle, scale it there and then again scale it while Validating. This is because in the general case we want the throttle to be a value between -1 and 1, we may need to scale to differently depending on where we use it; here for the physics model we need it to be between 3.43 and -10 while in other cases it might 100 or -100 etc...

Complete Function $\mathbf{T_c}$

We now unify all these steps into **one** mathematical function,

$$T_c\big(\{\mathbf{p}_i\},\, v\big),$$

that **only** requires the list of checkpoints $\{\mathbf{p}_i\}$ and the current car speed v.

(Internally, one would still need the car's current position $\mathbf{pos}$ and forward direction $\mathbf{fwd}$ to compute the angle θ. For brevity, we show them as part of the formula but treat them as known geometry.)

1. $\ell = \min\Big(\lfloor \text{speedLookAheadSensitivity} \cdot v \rfloor + 1,\ \text{N} - 1\Big),$
2. $\mathbf{dir} = \mathbf{p}_\ell - \mathbf{pos}, \quad d = \|\mathbf{dir}\|,$
3. $\theta = \text{SignedAngle}\big(\mathbf{fwd},\, \mathbf{dir}\big),$
4. $e_s = \begin{cases} 30.1964649875, & \text{if } \theta = 0, \\ \min\Big(\Big|\, \frac{1}{(\text{accelerationFactor})\,\theta}\,\Big|,\ 30.1964649875\Big), & \text{otherwise}, \end{cases}$
5. $a_o = \dfrac{(e_s - v)\,v}{d} \quad \Big[\text{if } a_o \approx 0,\ \text{set } a_o = 1\Big],$
6. $t' = \text{clamp}\Big(\max\Big(0,\ \frac{a_o}{3.4323432343}\Big)\ +\ \min\Big(\frac{a_o}{10},\, 0\Big),\ -1,\ 1\Big),$
7. $T_{\text{final}}(t') = \max\Big(0,\ t' \cdot 3.4323432343\Big)\ -\ \max\Big(0,\ -10\,t'\Big).$

$$\boxed{T_c\big(\{\mathbf{p}_i\},\, v\big)\ =\ T_{\text{final}}\big(t'\big).}$$

Hence, $T_c(\{\mathbf{p}_i\},\, v)$ is the final **throttle value** after validation, which the code then applies to control the car's forward or brake input.

so for any one chosen checkpoint ℓ the formula/function would be...

$$T_c = \min\Big(\ \max\Big(\max\big(0,\ \frac{\Big(\min\big(\big|\frac{1}{a_f\,\theta_\ell}\big|,\,30.2\big) - a_s\Big)\,a_s}{\frac{d_c(\ell)}{3.43}}\cdot\big)\ +\ \min\big(\frac{\Big(\min\big(\big|\frac{1}{a_f\,\theta_\ell}\big|,\,30.2\big) - a_s\Big)\,a_s}{\frac{d_c(\ell)}{10}}\cdot,\,0\big),\,-10\Big),\,3.43\Big),$$

Important Relations

1. **Lookahead Logic**
   - The function picks how far ahead in the checkpoint list to look based on current speed. Higher speed $\implies$ larger index ℓ.
2. **Angle-Based Speed Expectation**
   - The function inversely relates turning angle θ to speed. Large angles reduce e_s.
3. **Acceleration Computation**
   - Dividing by $\frac{d}{v}$ is effectively dividing by “time to reach checkpoint if speed stayed constant.”
4. **Positive/Negative Split**
   - The raw throttle t' is computed by separate positive and negative scaling. Then the “validation” step re-applies additional constraints.
5. **Constants**
   - (30.1964649875) is the maximum “expected speed” (in the code).
   - (3.4323432343) is the maximum *acceleration* scaling factor
   - (-10) is the maximum deceleration factor.