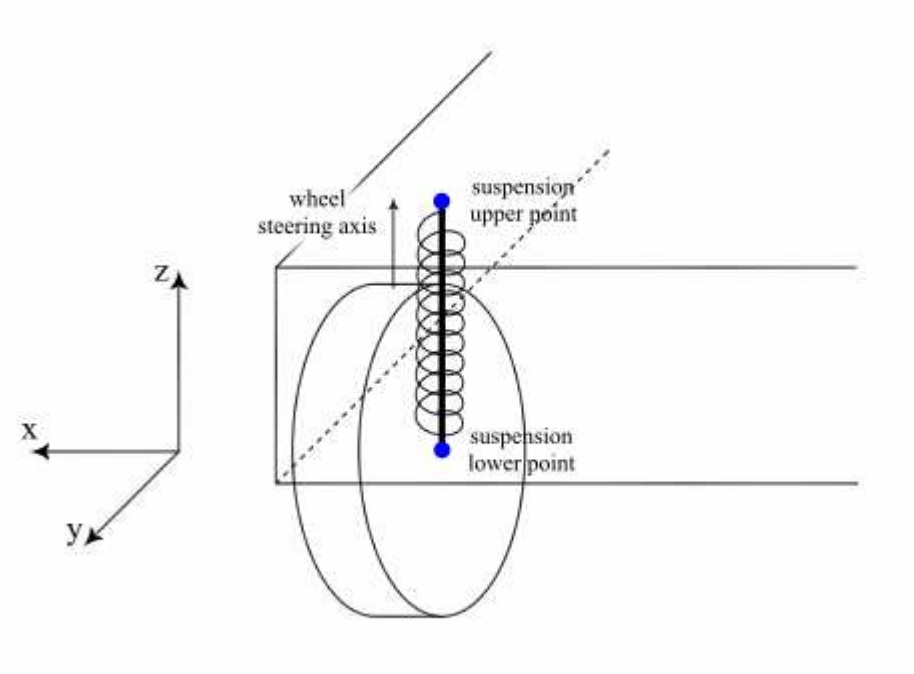
Guidelines for Vehicle Creation

2004-09-08

In local coordinate system vehicle should be oriented along y axis (i.e., when steering is set to 0, vehicle is supposed to move along y axis). Z axis should point up.

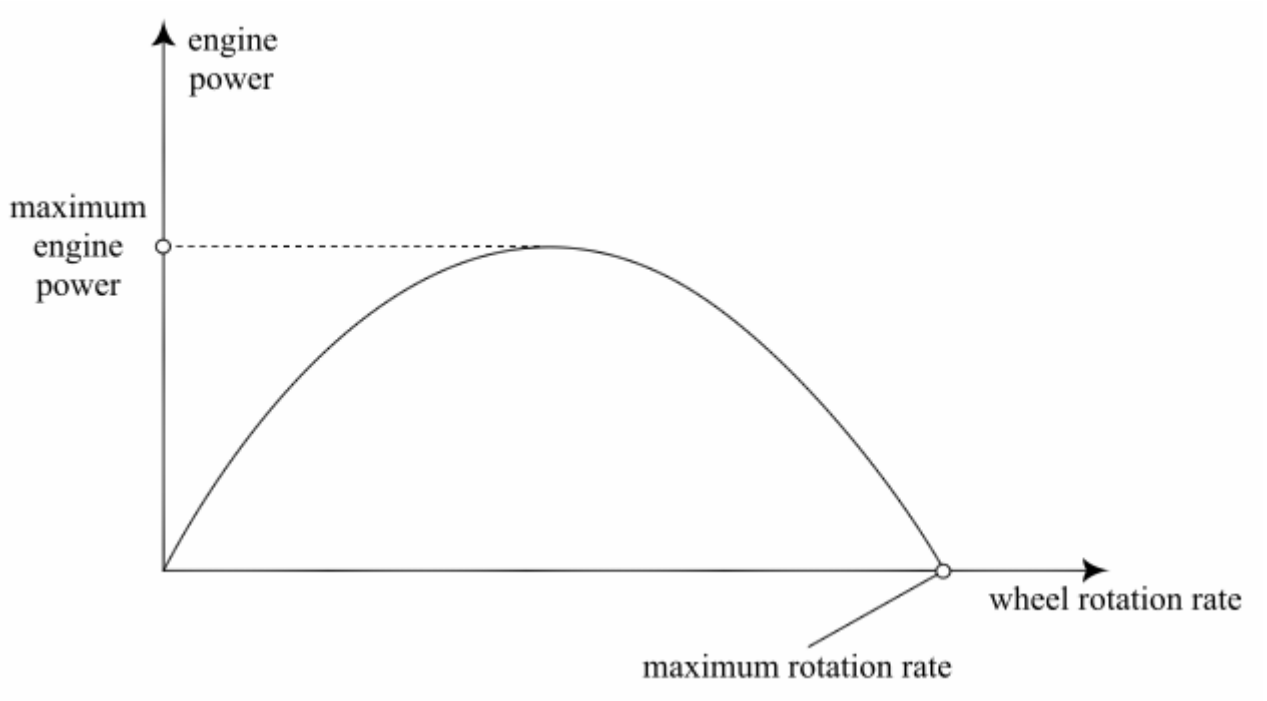
As a physical entity, vehicle should have hull geometries and wheel geometries. There can be several hull geometries, but there can be only one geometry per wheel. Hull geometries can be added via normal pe_geomparams structure, while wheels should use pe_cargeomparams structure with bDriving set to nonnegative value. Some parts of vehicle can be excluded from collision checks, and some may have zero mass, thus having no effect of vehicle dynamics. A common technique is to set all vehicle parts except one have zero mass, and create a specific part that is neither rendered nor participates in collisions, but rather specifies vehicle mass distribution. Typically this latter part will be placed quite low in order to increase vehicle stability.

All wheels are supposed to have independent suspension, i.e. suspension is a spring that has its upper end attached to some point on vehicle hull, and its lower end attached to wheel. The upper point stays attached to the hull, and the lower point is allowed to move only along z axis in vehicle coordinate system. During steering wheels that lie after vehicle's center of mass along y axis are rotated around their own vertical axes.



Wheels are created automatically when a corresponding geometry is added. During creation pivot (suspension upper point) is specified, and lower point is obtained automatically by moving down z axis by len_initial. Vehicle is supposed to be created in a resting position, thus len_initial corresponds to a suspension spring compressed under weight of the hull. Len_max is the relaxed suspension length, and if stiffness of the spring is not specified explicitly, it is calculated basing on len_max and len_initial. Damping can be specified either explicitly, or but setting it to a non-positive value. In the latter case damping value that is required to get completely damped (w/o any oscillations) spring is calculated and multiplied by the value provided (with + sign). Note that zero-oscillation damping is calculated approximately, thus this value might require some minor tweaking.

When vehicle engine is active (pedal is positive), it applies torque to wheels. Torque is calculated as effective power divided by current wheel rotation rate and multiplied by the current gear ratio. Effective power depends on current engine rotation rate as a sine function, with 0 at 0 and maximum rotation rate (vehicle parameter), and maximum power in the middle. This results in torque being maximum (0.5*max. power/max. rotations per second) at 0 rotation rate, and 0 at maximum rotation rate. This torque is multiplied by pedal value and clutch value before applying. Engine revolution rate is usually equal to the driving wheels revolution rate (some driving wheels can have slightly different velocities, so the maximum is used) multiplied by the current gear ratio, but during clutch disengagement-engagement periods it's blended between the current rate and wheel rate multiplied by gear ratio.



LoadVehicle function loads objects from cgf file and constructs physical entity from them. It loads objects in this order:

- hull%d, where %d is an integer number starting from 1. It continues increasing this number until successive object cannot be found. For each object, it also tries to load hullproxy%d object, which, if found, serves as a substitution of hull%d object for physics (normally hullproxy should be a low-poly representation of hull)
- wheel%d, %d – integer starting from 1. Physics system tries to compute cylinder parameters basing on this geometry (inside physics wheels are stored as cylinders anyway). For each wheel wheel%d_upper and wheel%d_lower helpers should be present. Difference in z between the two is used as suspension initial length, and pivot (suspension upper point) is set to wheel_%d_lower position offset up by this length (which might not correspond to wheel_%d_upper position)
- susp%d, %d – integer starting from 1. This creates an object that is attached with its ends to two other vehicle parts. The ends should be marked with susp%d_*target* helpers, where susp%d is the same as the suspension object name, and *target* can be either "hull", or, "wheel%d", or "susp%d" (thus corresponding to either hull, or arbitrary wheel, or another suspension). The suspension geometry is then stretched and rotated so that its ends stay rigidly attached to the moving vehicle parts. This is just a visual effect and it does not affect physical simulation.

Summary

Vehicle model consists of hull parts, wheels, and suspensions. All hull parts should be named hull%d, where %d is an integer number starting from 1, wheels – wheel_%d, suspensions – sus_%d. Hull parts can have low-poly substitutes for collisions with objects (precise models will still be used for raytracing during shooting), named hullproxy%d. If a hull part has a hullproxy, all subsequent parts without hullproxy will not participate in collisions, only in raytracing. Collision checks occur more often and they are more expensive than raytracing, thus as many hull parts as possible should be excluded from collision checks.

Vehicles have gearboxes with automatic gear switching. Gears switch whenever engine revolution rate threshold is reached (either upper or lower). After a gear switch clutch is disengaged and then returns to fully engaged state with a specified speed. During this period torque applied to the wheels is multiplied by the clutch value (0..1), and engine revolution rate gradually levels itself with wheels revolution rate * gear ratio.

Both handbrake and footbrake are supported. Handbrake unconditionally locks the wheels that have 'can_brake' property when 'jump' button is pressed, and footbrake applies a fixed torque to all wheels when 'back' is pressed while the gear is forward (and vice versa). Note that when a wheel is locked with a handbrake, friction is not anisotropic and is computed independently on pull and lateral directions. In order to somehow counter the potential maximum friction increase by 1.41, friction value is multiplied by 0.85 for handbraked wheels (this is implementation-specific, and can possibly change).

In a vehicle script file there should be a record for each hull part present in the cgf file, and a record for each wheel (first records for all hull parts, then records for all wheels). Hull part record contains the following fields:

- *flags* – 0 if part is invisible but physicalized, 1 if part is both visible and physicalized, and 2 if part is visible but not physicalized;
- *mass* – mass of the part in kilograms. It's a normal practice to have just one invisible part with nonzero mass, since it's easier to tweak vehicle behavior in this case. It's important that parts that have nonzero mass have "correct" meshes, i.e. meshes that represent closed surfaces, without degenerate triangles and triangles with non-connected edges (if a part has hullproxy, this restriction applies only to hullproxy);
- *zoffset* – additional offset of the part along vertical axis (0 if not specified). Normally this should be used to tweak position of the part that specifies mass distribution.

Wheel record contains the following fields:

- *driving* – 1 if the wheel is driving;
- *axle* – currently is used snap the wheels that belong to one axis and are almost opposite (snapping means setting them to be exactly opposite) and to apply stabilizers (see below) for the wheels on the same axis;
- *len_max* – length of relaxed suspension spring in meters;
- *stiffness* – suspension spring stiffness in N/m. If 0 is specified this will be computed automatically from assumption that in the initial pose the vehicle stands on the ground and suspensions are compressed with its weight;
- *damping* – damping in N/(m*s). If zero or negative value is specified, damping value that is required to get completely damped (w/o any oscillations) spring is calculated and multiplied by the value provided (with + sign). Note that zero-oscillation damping is calculated approximately, thus this value should be slightly tweaked;
- *surface_id* – surface identifier for the wheel (it will be used to set tire friction value);
- *can_brake* – (0/1) whether handbrake affects this wheel;

- *min_friction* – lower limit for wheel friction value obtained from material properties ((tire_friction+ground_friction)/2); the value is clamped to this limit; all friction modifiers (for dynamic or braking friction, see below) are applied afterwards;
- *max_friction* – upper limit for wheel friction value, works similar to *min_friction*.

For each wheel wheel_%d_upper and wheel_%d_lower helpers should be present. Difference in z between the two is used as suspension initial length, and pivot (suspension upper point) is set to wheel_%d_lower position offset up by this length (which might not correspond to wheel_%d_upper position, as in jeep model). Suspension is an object that is attached with its ends to other vehicle parts (hull, wheels, or suspensions). This is just a visual effect and it does not affect physical simulation.

One can also specify the following parameters in a vehicle script:

- *engine_power* – self explanatory. Use the number 100000 for a 3 tonn vehicle as a starting point;
- *engine_power_back* – same when driving backwards (no longer used);
- *gears* – an array of gear ratios (max. 8); the 1st one should be backward gear (and thus negative), the 2nd – neutral (0), and the rest – forward gears;
- *engine_maxRPM* (or *engine_maxrpm*) – maximum engine flywheel rotation rate (see detailed explanations above); effective engine power is 0 when this rate is reached;
- *engine_minRPM* – if engine RPM drops below this value, clutch is automatically disengaged to prevent engine stalling (thus engine stalling itself is not actually implemented);
- *engine_idleRPM* – this RPM is set whenever the engine is idle (as a result of clutch being disengaged when *engine_minRPM* is reached, for instance);
- *engine_startRPM* – the engine RPM is instantly set to this value when gear is switched from neutral position;
- engine_shiftupRPM – whenever engine RPM becomes higher than this value, gear is switched up (not used for backward gears, since there's only one backward gear);
- engine_shiftdownRPM – whenever engine RPM becomes lower than this value, gear is switched down (until the neutral gear is reached);
- *clutch_speed* – specifies how fast the clutch returns to fully engaged state after disengagement (time = 1/*clutch_speed*);
- *gear_dir_switch_RPM* – switching gear direction (backward while moving forward, or vice versa) is allowed only when wheel (not engine) RPM is below this limit;
- *axle_friction* – internal friction at axles, is always applied as a torque opposing the current velocity;
- *brake_torque* – footbrake torque;
- *slip_threshold* – a wheel is treated as slipping when instantaneous velocity at its contact point differs from the ground velocity by this value;
- *dyn_friction_ratio* – specifies how much the friction is reduced when a wheel is slipping (in fact, it can even increase, if the value is greater than 1, but in reality it should be reduced in such cases);
- *max_braking_friction* – limits the absolute friction value on handbraked wheels (useful for implementing spinbrakes and generally is't better to have more realistic friction values – around 1 – when leaving a vehicle handbraked in bumpy environments);
- *max_steer* – maximum steering angle (in degrees); it is superceded by the following parameters;
- *max_steer_v0, max_steer_kv* – used to calculate maximum steering angle by using a formula max_steer_v0 + max_steer_kv*(current vehicle speed in m/s);
- *steer_speed* – steering speed in degrees per second;
- *steer_speed_valScale* – the steering speed varies from *steer_speed* at 0 steering to *steer_speed*steer_speed_valScale* at maximum steering (in a non-linear fashion);
- *steer_speed_min* – the lower bound for steering speed; makes sense only if *steer_speed_valScale* is negative;

- *steer_relaxation_v0, steer_relaxation_kv* – specifies linear dependency of "relaxation of steering" on vehicle velocity (steering relaxation is the process of returning wheels to 0 steering angle when no steering buttons are pressed);
- *brake_vel_threshold, brake_axle_friction* – obsolete;
- *max_steering_pedal* – whenever vehicle speed is above *pedal_limit_speed*, pedal is capped; at maximum steering the cap is *max_steering_pedal*, at 0 steering – 1; this is supposed to reduce the risk of spinning on sharp turns;
- *pedal_speed* – specifies pedal speed (when acceleration key is pressed, pedal grows from 0 to 1 with this velocity);
- *max_time_step_vehicle* – maximum time step cap for the 'driving' state (i.e. when there are no contacts but at wheels); for explanations of time step cap see 'Physical Properties.doc'
- *sleep_speed_vehicle* – sleep speed for the 'driving' state;
- *damping_vehicle* – overall vehicle damping for the 'driving' state; damping limits the maximum vehicle velocity in addition to maxRPM and other parameters;
- *statibilizer* – specifies an additional force that tries to stabilize the vehicle when wheels on one axis have different suspension lengths; it's expressed as a fraction of stiffness (so its range should be around 0..2.5) and is applied in a similar way;
- *integration_type* – selects between explicit Euler (0) and semi-implicit Euler (1) integration methods; the latter is more stable when the suspension springs are very stiff, but it requires more things to be synchronized over network, so in multiplayer it's currently forced to be 0;
- *cam_stiffness_positive, cam_stiffness_negative* – camera spring stiffness for positive and negative x,y,z coordinates (in vehicle coordinate frame); a camera is attached to the vehicle at eye_pos helper with a 3-dimensional spring; if a stiffness along some direction is 0, the camera is treated as firmly fixed in this direction;
- *cam_limits_positive,cam_limits_negative* – limits the camera offset along all axes;
- *cam_damping* – camera spring damping (it's a uniform damping, not just a damping along the camera spring);
- *cam_max_timestep* – limits the camera step to handle stiff camera springs;
- *cam_snap_dist,cam_snap_vel* – whenever the camera is at most at *cam_snap_dist* from the pivot and its velocity differs from that of the pivot at most by *cam_snap_vel*, the camera is snapped to the pivot and inherits its velocity

