If this is your first visit, be sure to check out the **FAQ** by clicking the link above. You may have to **register** before you can post: click the register link above to proceed. To start viewing messages, select the forum that you want to visit from the selection below.

If you'd like to receive bonuses in the Official Wiremod Gmod Server, Link your Steam account to you forum account **here!**

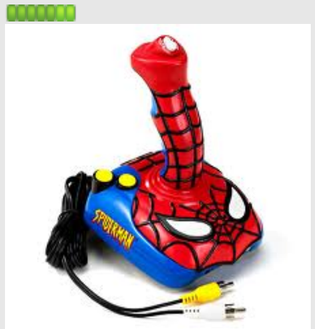
Thread: Comprehensive Gmod physics guide

02-03-2010 #1

Tolyzor

hurrr physics

WIREMOD WM HELPER



Join Date: Aug 2008
Location: England
Posts: 1,192
Blog Entries: 2

Comprehensive Gmod physics guide (your help needed)

Newly discovered mechanisms are now linked from the list below;

Introduction

This page attempts to catalogue all mechanisms in the gmod physics engine[1], so if you know how a particular aspect works, please help by editing it. It also includes popular physics addons like Wiremod and fin tool.

Everything in this guide has been derived using a combination of three techniques; in-game experimentation, looking over game code[2], and using existing real world equations.

Icon-tick.png Means the section is 100% understood, defined with an equation and extensively tested in-game.

Icon-question.png Means the section is mostly understood but still requires additional testing or refinement. For most purposes, this will still give you a good enough definition.

Icon-cross.png Means the section is at most partially understood, and there are no equations defining it. Expect to find speculation or samples of raw of data here.

Please contribute what you can towards the red cross and orange question mark sections, as well as adding any new sections you think are needed!

Most of the time, most physics mechanisms are inactive for whatever reason, these icons will give you an idea of what might be effecting your prop/contraption and decide whether it is negligible/inactive or not.

Icon-fflight.png Free flight - the section can always apply to props.

Icon-entity.png Entity specific - the section only applies when specific entities are involved.

Icon-constraint.png Constraint specific - the section only applies to specific contraint(s).

Icon-contact.png Contact - the section only applies when the prop is contacting another or the world.

Definitions

- Gmod server tickrates vary; by default the tickrate is 66, however some servers use tickrates of 33 or 100, or other values.

To confuse matters further, there is a discrepancy (most likely a bug in the engine, the reason/mechanism is as yet unknown) between the server tickrate (ie 66) and the actual physics engine tickrate (ie 66.566669). These are outlined in the constants section further down.

This guide is concerned only with physics tickrate, not server tickrate, from hereafter.

- A physics tick is 1/t seconds long; there are t ticks every second

This fact gives us a number of definitions involving units;

applyForce is in units of [Kg*glu/tick^2]

Force (for example force due to gravity) is in units of [Kg*glu/s^2]

Therefore;

A force of F = applyForce(F / t)

applyForce(F) = A force of F * t

Because of these definitions, we need to be especially careful to keep quantities like force, torque, velocity, acceleration, etc in the same unit of time. For the purposes of clarity, I will define them in terms of seconds for the rest of this guide.

All equations other than gravity have been derived on a tickrate 66 server, untested on other tickrates to see if it effects them. However as far as I've found, tickrate only effects applyForce/torque/etc and things that use it (like the forcer)

- 'Air' in gmod is ubiquitous. This means air drag and air buoyancy effects still occur when a prop is submerged, in space, or clipping through another prop.

- The 'up' direction in gmod is vec(0, 0, 1).
Constants and notation used in later equations

Unless otherwise stated, vectors are extrinsic (local to world) and are denoted bold.

Symbol Units Description

t ticks/s physics tickrate, for server tickrate 33 t = 32.985, for server tickrate 66 t = 66.566669, for server tickrate 100 t = 99.85. Change server tickrate by adding -tickrate # to end of gmod shortcut target[4].
μd dry friction coefficient. It is dependent on the surface properties of the two entities in contact. (For two phx tiles sliding over each other) = 0.64199. Possibly clamped between phys_minfrictionmass (Default: 10) and phys_maxfrictionmass (Default: 2500) in console
|v| = speed = v:length()

a x b = cross product[12] of a with b = a:cross(b)

a = function(b, c) means a is somehow proportional to b and c

w.r.t - with respect to
Force and translation

Useful mechanics principles for this section are the equations of motion and static equilibrium.

Force acts upon prop centre of mass unless otherwise stated.

Force due to applyForce

Icon-tick.png Icon-fflight.png

E:applyForce(F_input)

F = F_input * t

Force output is clamped as follows; clamp(F, -E:mass() * SpeedLimit() * t, E:mass() * SpeedLimit() * t)

applyOffsetForce is the same, but force is applied to the local vector specified
Force due to gravity

Icon-tick.png Icon-fflight.png

F = vec(0, 0, m * -gravity())

Where gravity() has a default value of 600 and is given by physenv.GetGravity()
Force due to buoyancy in water

Icon-question.png Icon-fflight.png

See: [13]

F = vec(0, 0, m * gravity() * (prop_density / water_density) * (submerged_prop_volume / total_prop_volume))

Where gravity() has a default value of 600 and is given by physenv.GetGravity()

prop_density is given in C:\Program Files (x86)\Steam\steamapps\User_name\garrysmod\garrysmod\scripts\surfaceproperties. prop_density can be changed with the physical properties tool[14], set non default materials with the convar: physprop_material

water_density is given somewhere? and has a defualt value of 1000

The ratio of submerged and total prop volume is for example: 1 when fully submerged, 0.5 when floating midway, and 0 when out of the water.
Force due to drag in air

Icon-question.png Icon-fflight.png

F = 8.18205e-6 * A * ρ * |v|^2 * -v:normalized()

Finding 'A' is explained here.
At very high input applyForce, prop stops, although E:vel() still returns a high velocity force at which effect begins is dependant on function(m, ρ & A)
Force due to drag in water

Icon-question.png Icon-fflight.png assumes entire prop is underwater (Approx) may be affected by ρ

F = 3.2737e-5 * A * |v|^2 * -v:normalized()

Finding 'A' is explained here.

Force due to velocity damping

Icon-cross.png Icon-fflight.png

See: [15]

F = function(|v|, $damping, -v:normalized())

For most props this is 0, either proportional to ρ or non functional. Add to drag eqn if it is dependant on ρ?
Force due to static dry friction

Icon-question.png Icon-contact.png

No relative contact planar velocity between objects

F < μd * Fperpendicular

Where Fperpendicular is perpendicular to the contact plane and pointing towards it. F acts upon the centre of the contact area and opposes force parallel to the contact plane between the two surfaces.

μ is given in C:\Program Files (x86)\Steam\steamapps\User_name\garrysmod\garrysmo d\scripts\surfaceproperties
Force due to dynamic dry friction

Icon-question.png Icon-contact.png

Relative contact planar velocity between objects

F = μd * Fperpendicular * -v:normalized()

Where Fperpendicular is perpendicular to the contact plane and pointing towards it. F acts upon the centre of the contact area and v is the relative velocity between the two objects parallel to the contact plane.

μ is given in C:\Program Files (x86)\Steam\steamapps\User_name\garrysmod\garrysmo d\scripts\surfaceproperties
Force due to contact dampening

Icon-cross.png Icon-contact.png

function(|v|)

Only present in the mud and slime surface property types. Given in C:\Program Files (x86)\Steam\steamapps\User_name\garrysmod\garrysmo d\scripts\surfaceproperties
Force due to elastic constraint

Icon-tick.png Icon-constraint.png

See: [16]

F = -constant_slider * S - damping_slider * v

Where S is in the direction of the constraint. v is the relative velocity between the two constrained objects. F acts upon constraint point.

Relative damping constant: the amount of energy the spring loses proportional to the relative velocity of the two objects the spring is attached to.

The constant_slider value is clamped to 50,000 and below. If a negative value is used, spazz results.

Force to break forcewelds

Icon-tick.png Icon-constraint.png

F = force_max * 30.325 * sqrt(m)

Where m is the lowest mass of the constrained pair of props or if one is frozen, the mass of the unfrozen prop. note: torque has no effect on forcewelds. Multiple welds do not share force.

Force due to explosive

Icon-cross.png Icon-entity.png

Force acts for the duration of 1 tick = 1/t seconds (Acting away from the centre of the explosive) force seems to be highest on props with a mass of 1 (although the force drop off might be due to inertial effects if the mass is greater than 1)

F = function(random variable, props in the way, distance^2, mass)

Effect of a blocking prop varies with it's mass

Force due to bullet impact

Icon-tick.png Icon-fflight.png

Force acts for the duration of 1 tick = 1/t seconds

F = 44464.7 * B * (unit direction vector of bullet)

F acts upon impact point, B is turret bullet force, for other weapons B = ...

If two bullets impact on the same tick, only the force from one is counted.

Force due to thruster

Icon-question.png Icon-entity.png

See: [17]

F = function(Thruster_geometry) * input_A * m

Where F acts upon thruster centre of mass, perpendicular to rear surface plane. function(Thruster_geometry) for speedometer thruster = 68.57. function(Thruster_geometry) is not just a constant multiplied by volume, length, width, box vol, radius, original mass or box xsa. It may be a function of them though. The effect of function(Thruster_geometry) also appears to be clamped at high input_A values

Force due to wire thruster

Icon-question.png Icon-entity.png

F = function(Thruster_geometry) * clamp(input_A, force_minimum_slider, force_max_slider) * m * force_mul_slider * sign(input_A)

Where F acts upon thruster centre of mass, perpendicular to rear surface plane. function(Thruster_geometry) for speedometer thruster = 68.57. function(Thruster_geometry) is not just a constant multiplied by volume, length, width, box vol, radius, original mass or box xsa. It may be a function of them though. The effect of function(Thruster_geometry) also appears to be clamped at high input_A values

Force due to vector thruster

Icon-question.png Icon-entity.png

F = function(Thruster_geometry) * input_vector:normalized() * clamp(input_A, force_minimum_slider, force_max_slider) * m * force_mul_slider * sign(input_A)

Where F acts upon thruster centre of mass, perpendicular to rear surface plane. function(Thruster_geometry) for speedometer thruster = 68.57. function(Thruster_geometry) is not just a constant multiplied by volume, length, width, box vol, radius, original mass or box xsa. It may be a function of them though. - untested, assumed from wire thruster. The effect of function(Thruster_geometry) also appears to be clamped at high input_A values

Force due to balloon air buoyancy

Icon-tick.png Icon-entity.png

F = vec(0, 0, balloon_force_slider * 75)

Force and velocity due to forcer

Icon-tick.png Icon-fflight.png

F = Fmul * force_input * t
F = Fmul * offset_force_input * t

Where F is applied to the point where the beam touches the prop and is in the direction of the forcer beam.

v = velocity_input

Where v is in the direction of the forcer beam. (v can also be applied to players)

Force due to hydraulic

Icon-cross.png Icon-constraint.png

F = function(S, overall length, masses, v (seems to be critically damped))

F acts upon constraint point.

Force due to wire hydraulic

Icon-cross.png Icon-constraint.png

F = function(S, overall length, masses, constant_input, damping input * v (seems to be critically damped))

F acts upon constraint point.

Force due to hoverball

Icon-cross.png Icon-entity.png

F = function(distance from input height, v, m)

Force due to wire hoverball

Icon-cross.png Icon-entity.png

F = function(distance from input height, v, m)

Force due to muscle

Icon-cross.png Icon-constraint.png

F = function(S, time, overall length, masses, v (seems to be critically damped))

F acts upon constraint point.

Force to break magnetize tool

Icon-question.png Icon-entity.png

Acts like a force weld, but with a different break force equation.

Applied when props are in contact. With multiple magnets, there is some attraction at small ranges, if they have previously touched?

F = ~100 * strength_slider

where strength_slider is the highest value of the pair.

Mass has no effect on breaking strength.

Magnetize tool

Note: use of the physical properties tool on a magnet removes its functionality.

Force due to fin tool

Icon-question.png Icon-entity.png

See: [18]

With flat surface dynamics:

F = m * angle_of_attack * |v| * E:up()

, where angle_of_attack is in degrees and is given by;

angle_of_attack = cos^-1(E:up():dot(E:vel():normalized()) - 90

It is a scripted entity, so everything is nicely defined in C:\Program Files (x86)\Steam\steamapps\user_account\garrysmod\garry smod\addons\Fin2\lua\entities\fin_2\init.lua

Lift by Plane Normal testing 'Limited testing seems to say that the size/shape of the fin has little or no effect on the force of the fin when air_density is equal to 0 and the fin itself is level. When air_density was set to its default 2, the fin seemed to slow down and lose a considerable amount of net upward force due to a loss in velocity(v. air_density 0).'

Dependent on air_density?

Effect of lift by plane normal?

Effect of Bernoulli effect by plane normal?

Effect of wind?

Effect of thermal cline?

Force due to inertia

Icon-tick.png Icon-fflight.png

See: [19]

F = m * a

Mass is clamped between 0.001 and 50000 The centre of mass for a contraption is given by R = (Σ(m * r)) / (Σm) where R is the centre of mass of contraption, m is the mass of each prop and r is the location of the prop's centre of gravity (E:massCenter()). When a prop is held by the physgun and m < 45678, prop mass temporarily changes to m = 45678

Coefficient of restitution

Icon-tick.png Icon-contact.png

This is the ratio of speeds after impact / before impact with another object or the world. This velocity change is applied to the dot product of the hitNormal of the surface and the velocity. The velocity component also obviously

switches signs. Referred to as "elasticity" here; C:\Program Files (x86)\Steam\steamapps\User_name\garrysmod\garrysmo d\scripts\surfaceproperties

Translational speed limit

Icon-question.png Icon-fflight.png

Prop speed is clamped (default 4000 [glu/s]). However, this only applies to the centre of mass of the prop, so the outer reaches of a rotating prop can exceed this limit. I can only get a prop to 3960 in sp at the moment... There is a critical speed effect, after reaching the speed limit sometimes props start skipping ticks, and their speed jumps between 2000 and 4000...didn't see this effect when i tested in sp... A similar effect happens at very low speeds?

Force due to collision with water

Icon-cross.png Icon-fflight.png

Force last for one tick

Force opposes direction of velocity

Dependant on frontal area of prop

Dependant on speed of prop

Clamped to F = m * |v| or below. (It will never change the sign of the velocity).

Torque and rotation

Useful mechanics principles for this section are torque and the equations of motion. A few others are described below.

This guide defines all angular quantities in vector form (axis_of rotation_unit_vector * magnitude) not e2's "angular vector" form (pitch, yaw, roll). These definitions would usually mean they are equal to each other, but thanks to gmod's 'up' direction being vec(0, 0, 1), they are not quite the same.

In scalar form, Torque = radial displacement from axis of revolution [glu] * perpendicular force [Kg*glu/s^2] τ = Sr * F, where Sr is radial displacement from axis of revolution

Or, in vector form; Torque = displacement vector from the point where torque is measured to anywhere on the line of force [glu] x force [Kg*glu/s^2] τ = r x F

The vector form is far more useful as the maths means you don't have to first calculate the force's minimum scalar displacement from the axis or the perpendicular force component like you do for the scalar form. It also obviously works out all three components of the torque at the same time.

Torque is expressed in vector form as being positive in the anti clockwise direction about the specified axis (right hand rule[20])

So, τ = vec(1, 0, 0) would be a torque of magnitude 1, about the x axis. (roll) τ = vec(0, 1, 0) would be a torque of magnitude 1, about the y axis. (pitch) τ = vec(0, 0, 1) would be a torque of magnitude 1, about the z axis. (yaw)

An angular vector is ang(pitch, yaw, roll) Wiremod's speedometer and e2 angVel functions return angular vectors(pitch, yaw, roll). Use the angvelVector function to work with my equations, or convert to the correct form in the following way;

So, to convert from a torque vector to an angular vector;

Torque = shiftL(vec(angular_vector)) angular_vector = shiftR(ang(Torque))

Converting between applyTorque and torque is the same as with force; A torque of τ = applyTorque(τ / t) applyTorque(τ) = A torque of τ * t

Note; When their input force magnitudes are equal, applyAngularForce = applyTorque = a couple with 0 resultant force of two applyOffsetForces

Torque is useful because it can be easily converted into a usable force using F = τ x r (this job is done by the wheels on a car for example).

Knowing the torque also allows you to determine angular acceleration with the equation α = Στ / I. This is an extension of Newton's second law[21]; F = m * a, except it deals with rotation. Think of I as "rotational mass". τ = I * α note: entity:inertia() currently returns a value 1550.1 times too small.

Just as torque can be converted to force, the angular quantities ϑ, ω and α can be turned into translational ones at the location r;

S = ϑ x r

v = ω x r

a = α x r

Torque due to applyTorque

Icon-tick.png Icon-fflight.png

E:applyTorque(τ_input)

τ = τ_input * t

Where τ and τ_input are local to prop axis
Torque output is clamped as follows; clamp(τ, -I * toRad(AngSpeedLimit()) * t, I * toRad(AngSpeedLimit()) * t)

Torque due to applyAngForce

Icon-tick.png Icon-fflight.png

E:applyAngForce(Ang_input)

τ = shiftL(vec(Ang_input)) * t

Where input is an angle vector and τ and input are local to prop axis

Torque due to applyOffsetForce

Icon-tick.png Icon-fflight.png

E:applyOffsetForce(F_input, r_input)

τ = F_input x r_input * t

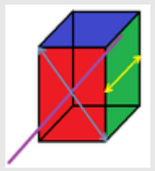
Torque output is clamped as follows; clamp(τ, -I * toRad(AngSpeedLimit()) * t, I * toRad(AngSpeedLimit()) * t)

Torque due to rotational drag in air

Icon-tick.png Icon-fflight.png
Explained in the code here: E2 Ballistic trajectory calculator WITH DRAG

τ = (ρ * h * |ω|^2 * 8.1205e-6 * S^4 * -ω:normalized()) / 32

Derivation explained below;



Cuboid is 'equivalent cuboid'; the dimensions of a cuboid constructed using the translational drag area vector and solving for each dimension. Similar to E:aabbSize().
purple arrow is the axis of rotation
yellow arrow is "h"
blue arrow is "s"

Consider an infinitesimal element d(s/2) on the moment arm s/2 . Linear drag force can be calculated given the translational velocity of the element. Total drag force across the length of s/2 is the integral of this wrt s/2 * |ω|^2.
Total torque is the integral of this wrt s/2.

I checked, and this *is* how the havok engine does it (double integration wrt s of the translational force equation)!

Also effected by the high torque input bug like applyForce/drag is.
Torque due to rotational drag in water

Icon-cross.png Icon-fflight.png

τ = (ρ * h * |ω|^2 * 3.2737e-5 * S^4 * -ω:normalized()) / 32

Where S & h are same as in air case. Derived from integrating translational case - not tested.
Torque due to rotational velocity damping

Icon-cross.png Icon-fflight.png

τ = function(|ω| * $rotdamping * m)

For most props this is 0, proportional to ρ or non functional effects phx tesla wheels
Torque due to rotational dry static friction

Icon-cross.png Icon-contact.png

τ = function(geometry of contact area, perpendicular force, μ)

Where τ acts about centre of contact region, opposing angular velocity, there is a reaction torque.

Likely can be described using the first moment of area on the contact area.
Torque due to rotational dry dynamic friction

Icon-cross.png Icon-contact.png

τ = function(geometry of contact area, perpendicular force, μ)

Where τ acts about centre of contact region, opposing angular velocity, there is a reaction torque.

Likely can be described using the first moment of area on the contact area.
Torque due to static friction of axis constraint

Icon-tick.png Icon-constraint.png

No relative angular velocity about the axis between objects

τ < 27012.2 * friction_slider

Where τ acts about constraint point and opposes torque between the two objects about the axis. There is a reaction torque.

Torque due to dynamic friction of axis constraint

Icon-tick.png Icon-constraint.png

Relative angular velocity about the axis between objects

τ = 27012.1 * friction_slider * -ω:normalized()

Where τ acts about constraint point and ω is the relative angular velocity between the two objects about the axis. There is a reaction torque.

Torque due to static & Dynamic friction of wire clutch

Icon-question.png Icon-constraint.png

τ <= 18900 * friction_input

Value approximate
Where τ acts about...err(can't sum vector representations of angle) the tensor/quaternion sum of the two entities angular velocities? There is a reaction torque?

Torque due to dynamic friction of axis constraint
Torque due to motor

Icon-tick.png Icon-constraint.png

τ = torque_slider * 0.017426 * abs(I * Axis)

Where I is motor inertia and Axis is the local axis about which the motor rotates. Clockwise direction is default "forward". Friction input is the same as axis constraint friction. "Time" slider adjusts how many seconds torque is applied for. No reaction torque. Force limit appears to have no function

Torque due to wheel

Icon-tick.png Icon-constraint.png

τ = torque_slider * 0.017426 * abs(I * Axis)

Where I is motor inertia and Axis is the local axis about which the wheel rotates. Clockwise direction is default "forward". Friction input is the same as axis constraint friction. No reaction torque. Force limit appears to have no function

Torque due to wire wheel

Icon-tick.png Icon-constraint.png

τ = (SpeedMod / 100 + 1) * torque_slider * 0.017426 * abs(I * Axis)

Where I is rotor inertia and Axis is the local axis about which the wheel rotates. When SpeedMod = -100, torque is a little stronger than it should be. No reaction torque. Friction input is the same as axis constraint friction.

Torque due to Keepupright constraint

Icon-cross.png Icon-constraint.png

See: [23]

This either uses setAngle, or a lua pd controller, so will therefore be:

function(ϑ, ω, m)

Torque due to rotational inertia
All rules of real world rigid body dynamics apply.
Icon-tick.png Icon-fflight.png

See: [24]

τ = I * α
I = function(prop_geometry) * m
I = entity:inertia() * 1550.1

For rotational inertia w.r.t another axis;

I = entity:inertia() * 1550.1 + m * S^2

Where S is the distance from the axis of rotation to a parallel axis[25] through the centre of mass. For group of multiple props, the rotational inertia is found by calculating rotational inertia w.r.t the contraption centre of gravity, and then simply summing all values. When a prop is held by the physgun and m < 45678, new_I = I_original * 45678 / m_original

Rules for creation of inertia tensors, reduction to scalar and the parallel axis theorem all apply. (I can provide e2 code for them). Prop's can also undergo torque-free gyroscopic precession.

Force/torque component(s) introduced by rotation in water

Icon-cross.png Icon-fflight.png

ω is somehow related to a drag force which causes acceleration perpendicular to linear velocity

Water speed induced wobble

Icon-cross.png Icon-fflight.png

Prop oscillates angularly, function(v, time) if the prop angle is close to optimum streamlining, the prop will settle at optimum.
Rotational speed limit

Icon-question.png Icon-fflight.png

Prop angular speed is clamped around any axis (default 61.0865 [Radians/s], given by sv_maxvelocity in console in units of [degrees/s]). When at the speed limit, the speed appears to jump around. This is due to the engine clamping the speed each tick. The engine sees the acceleration and therefore predicts a speed greater than is allowed (overestimates) and then underestimates the next tick as the new velocity is registered.
Misc phenomena

Prop interactions use the basic equation for inelastic collisions.

Explosion blast effect

Icon-question.png Icon-entity.png

See: [26]

Includes player damage, prop damage, incendiary effect

Wired explosive damage =~ Damage_slider * (1 - (range / blast_radius))

Where range is the distance from the explosive to the closest point on a line going from the explosive to the prop mass centre. If range > blast_radius, damage = 0. In e2 code this is;

```
rangerFilter(Explosive)
range = rangerOffset(Explosive:pos(), Prop:massCentre()):distance()
```

if a prop is in between the explosive and the prop you are investigating;

Wired explosive damage =~ (Damage_slider * (1 - (range / blast_radius)) - C) * (1 - (clamp(m, 0, 350) / 350))

where m is the mass of the first prop the trace hits. C is approx 80+/-20 and increases roughly with range.

Fire damage

Icon-cross.png Icon-fflight.png

Note differences in singleplayer and multiplayer

function(time, prop health)

Breakable prop damage

Icon-cross.png Icon-entity.png

Only mechanisms are; weapon, prop impact, fire and blast damage Given by C:\Program Files (x86)\Steam\steamapps\user_name\garrysmod\garrysmo d\scripts\propdata\base.txt

depends on sharpness of colliding prop.
The minimum to detroy a phx potato launcher explosive is:
mass = 20 kg, velocity = 784+/-1 glu/s
Acheived using phx cone as impacter
Therefore, ignition energy = 6146560 gmod energy units (Ke = 0.5 mv^2)

square glass plate fired corner first into flat object:
m breakage |v|
1 1709
2 1662
3 1656
50 1591
94 to 50000 1589

Terminal ballistics of props breaking through breakable props

Icon-cross.png Icon-contact.png

The prop doing the breaking stops for 1/10 second or so at collision, then continues on with initial velocity - a little

Moving prop Player and npc damage

Icon-cross.png Icon-contact.png

if |v| < 63.14, prop does no damage.
with m = 12832;
if 0.5 * m * |v|^2 >= 2.5578e7, damage = 5hp
if 0.5 * m * |v|^2 >= 7.1547e7, damage = 10hp

Player and npc crush damage

Icon-cross.png Icon-contact.png

Player crush damage between two objects side on or above and below requires velocity, can't be done with force alone

Player physics

Icon-question.png Icon-entity.png

See: [27]

Governed by q_physics, rather than v_physics like everything else

Gravity constant used is sv_gravity convar, rather than physenv.GetGravity()

speed limit is different - 3500 glu/s

no drag
Gravgun

Icon-question.png Icon-entity.png

See: [28]

The gravgun can pick up 250Kg maximum
Bouncy ball entity physics

Icon-tick.png Icon-entity.png

It is a scripted entity perfect sphere

No translational drag

Disappears when welded/crushed sometimes

Also disappear when people eat them by pressing e
Ragdoll physics

Icon-cross.png Icon-entity.png

complex stuff...most likely needs to be split into bones in order to predict most behaviour

Each bone has to be no gravved separately
Parented prop physics

Icon-cross.png Icon-constraint.png

No collisions. Do not respond to gravity unless welded to their parent. Do not retain constraints to other props unless welded to their parent.

Still effected by bullet force?
Engine peculiarities

These are approximations by definition, as the engine doesn't officially assign a force, but rather tries to set prop position or velocity. Because of this, the causes of any 'give' in these properties is often counter intuitive.

Phx potato launcher tube gravity

Icon-cross.png Icon-fflight.png

Seems to be impossible to balance applyForce with
Force due to rope constraint

Icon-cross.png Icon-constraint.png

Force due to weld constraint

Icon-cross.png Icon-constraint.png

(prop welded to prop welded to prop welded to world)

F = function(m, v, exp(S), duped?)

F acts upon constraint point. v is the relative velocity between the two constrained objects
Force due to slider constraint

Icon-cross.png Icon-constraint.png

See: [29]

Max physics operations

Icon-question.png Icon-fflight.png

(prop stops moving after a while)

phys_collisions_object_timestep or phys_collisions_timestep in console or Physenv.GetPerformanceSettings

props moving at less than 1.333 glu/s are liable to sleep. the time it takes for them to sleep seems to be dependant on speed. this is almost certainly related to server tickrate. A prop will only "sleep" and freeze if it moving slowly under no external forces (ie collision and constraint forces)

Resolving torque from two sources on the same prop
For example; Two pistons in phase and inline on a crankshaft contribute the same torque as one.
see: http://www.youtube.com/watch?v=oS4QESE7geo&t=3m4s

Incorrect speed reporting
E:vel() and E:angVel() report incorrect speeds (different from the real change in position with time), when a prop is low mass and under high load.

Clipping of props

Icon-question.png Icon-contact.png

(high forces/velocities/rotational speeds)

Two props with opposing velocity's of 4000 and 0 ω don't clip through each other when they collide Wheels clip through the map, the higher the poly count, the slower they need to be going before they do.

Given enough applyForce, props can be made to go through the map. |v| is reported as higher than the speed limit, although in reality it is not.
Prop spaz/blackhole effect

Icon-tick.png Icon-constraint.png

Caused by over-constraining. It is a result of the physics engine constraint solver attempting to solve an impossible scenario. Solution - remove some constraints.

Effect of water on weapons

Icon-question.png Icon-entity.png

Rocket launcher rocket is slowed down when entering or leaving water.
Useful links

The original location this guide was published[30]

Collection of physics convars[31][32]

Interesting lua PD controller[33]

Credits

Guide author: Tolyzor

Also thanks to: Black Phoenix, zoombahh, Josef, robowurmz, XXXmags, Wenli, dlb1

*Last edited by Tolyzor; 03-10-2013 at 01:48 PM. **Reason:** added more info to breakable prop damage section, added speed bug*

Share

Fistas likes this.

Reply With Quote

02-03-2010 #2

Drunkie

Ursus maritimus

WIREMOD SUPER MOD

Drunkie's Avatar

Join Date: Feb 2009
Location: Canada
Posts: 6,636
Blog Entries: 1

Re: Comprehensive Gmod physics guide

Looks well written, although I dont know if I'd ever use this guide when I'm building.

Share

Reply With Quote

02-03-2010 #3

Tolyzor

hurrr physics

WIREMOD WM HELPER

Join Date: Aug 2008
Location: England
Posts: 1,192
Blog Entries: 2

Re: Comprehensive Gmod physics guide

Well its also useful for wire or physical control systems of course.

Share

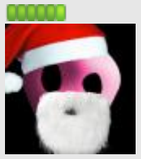
Reply With Quote

02-03-2010 #4

feha

Wire Sofaking

WIREMOD MEMBER



Re: Comprehensive Gmod physics guide

1. Doesnt different server have different so-caleld tickrates? And doesnt that mean your definition of a tickrate only work on some servers?

2. Why use inches? I strongly reccomend using gmod units, otherwise people will whine (like me) that you should use SI units, and if you do that, I bet ppl will ask you to explain how many units = 1 cm.

3. Why use kg instead of gmod weight unit, and how many gmod weight units is 1 kg?

4. I would love some friction and bouncy data about the different options in prop properties tool.

Join Date: Sep 2009
Location: Here
Posts: 1,281

5. What exactly is the max speed limit?

6. Add formulas for physgunning props (as in the force and such).

Share

Holo Combat System!
Line-Plane intersection!
Tracing System!

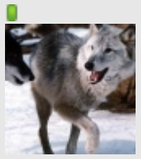
Feha

Reply With Quote

02-03-2010 #5

Nikita

Lifetime Supporter



Join Date: May 2009
Posts: 784

Re: Comprehensive Gmod physics guide

I wanted this for a long time, and now that it's been written, I don't know what to do with it

Oh well. Maybe one day I'll indeed do calculations before trying.

Good job making this guide

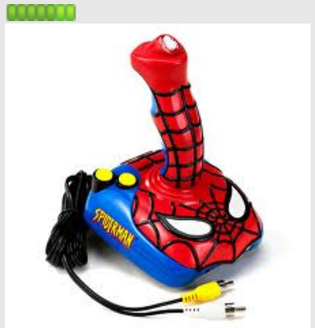
Share

Reply With Quote

02-03-2010 #6

Tolyzor

hurrr physics



Join Date: Aug 2008
Location: England
Posts: 1,192
Blog Entries: 2

Re: Comprehensive Gmod physics guide

I have called gmod units inches, and gmod mass units Kg, as this is what they are most commonly called by players and the wiremod system. I don't have the time to add everything, I posted this now in the hope that i will get some help.

Servers do have tickrates but there's not much I can do about that, and it doesn't seem to have much (if any) effect on any of my calculations. I am of course open to suggestions regarding this.

Speed limits added.

Last edited by Tolyzor; 02-03-2010 at 04:31 PM.

Share

Reply With Quote

02-03-2010 #7

Hitman271

Wire Sofaking

WIREMOD MEMBER

Hitman271's Avatar

Join Date: Feb 2008

Location: Why? You looking for somebody?

Posts: 738

Re: Comprehensive Gmod physics guide

It was nice but I thought there would be more to it. Kinda short for a "comprehensive" guide

Share

> Originally Posted by **Anticept**
>
> *This is not some place where you can toss your dick around and expect people to suck it.*

Community Gpu Thread. Post Yours!

Bouncy Ball

Reply With Quote

02-03-2010 #8

Grocel

Wire Sofaking

WIREMOD MEMBER

Join Date: Mar 2008

Location: Germany, NRW, Remscheid

Posts: 854

Re: Comprehensive Gmod physics guide

> Originally Posted by **feha**
>
> *2. Why use inches? I strongly reccomend using gmod units, otherwise people will whine (like me) that you should use SI units, and if you do that, I bet ppl will ask you to explain how many units = 1 cm.*

The formula is:

Code:

```
Real Inches = gmod/hammer/map units * 0.75
1 cm =  1 gmod units / 0.75 * 2.54
1 feet = 16 gmod units
```

The e2 units converting system uses a wrong formula: :mellow:

Code:

```
Real Inches = gmod/hammer/map units
1 cm =  1 gmod units * 2.54
1 feet = 12 gmod units
```

I found it out, while I was making a real size map of a real place...

Last edited by Grocel; 02-03-2010 at 05:55 PM.

Share

The secret phrases of gmod are: *Rusty bullet hole* and *Walsall England*
Mappers take a look at the *Wire Map Interface (WMI)*.
Im a molecule!

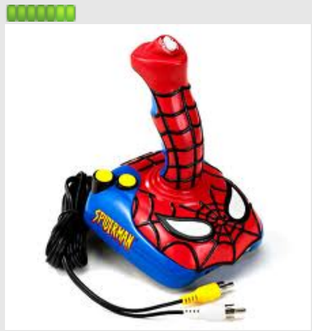
Reply With Quote

02-03-2010 #9

Tolyzor

hurrr physics

WIREMOD HELPER



Join Date: Aug 2008
Location: England
Posts: 1,192
Blog Entries: 2

Re: Comprehensive Gmod physics guide

> Originally Posted by **Hitman271**
> *It was nice but I thought there would be more to it. Kinda short for a "comprehensive" guide*

like I said, I can't see myself finishing it alone, as there is so much to do.

> Originally Posted by **Grocel**
> *The e2 units converting system uses a wrong formula: :mellow:*
> *Code:*

```
Real Inches = gmod/hammer/map units
1 cm =  1 gmod units * 2.54
1 feet = 12 gmod units
```

> *I found it out, while I was making a real size map of a real place...*

Yeah, that's why I use inches and Kg, because wiremod does. It doesn't make any sense using anything else, because all useful equations need to interface with wiremod anyway.

Share

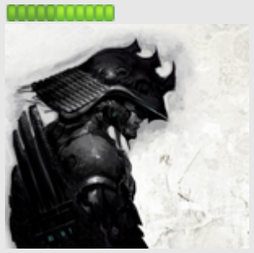
Reply With Quote

02-03-2010 #10

Jat Goodwin

Official Bastard of Wire

WIREMOD SUPER MOD



Join Date: Aug 2008
Location: Colorado Springs
Posts: 2,768

Re: Comprehensive Gmod physics guide

in other words, source physics suck.

Share

Off icia l Bast ard of Wir e
I Require More Minions! Join us on the IRC !
List of Reasons to idle on the IRC: Wire QDB

Reply With Quote

Page 1 of 11 1 2 3 ... Last

« Previous Thread | Next Thread »

LinkBacks (?)

arkstower.com - View topic - GMod rube goldberg contraption 03-10-2010, 09:47 AM
Refback This thread

Tags for this Thread

force, gmod, guide, physics, torque
View Tag Cloud

Bookmarks

Digg
del.icio.us
StumbleUpon
Google
Facebook

Posting Permissions

You may not post new threads
You may not post replies
You may not post attachments
You may not edit your posts

BB code is On
Smilies are On
[IMG] code is On
[VIDEO] code is On
HTML code is Off
Trackbacks are On
Pingbacks are On
Refbacks are On

Forum Rules