A Rough Guide to Rocket Behaviour in Painkiller

In version 1.35, player model were four spheres, on top of each other; the topmost sphere with a radius equal to 0.5 of radius of the rest of spheres. Plus there was a small ray object below the spheres, for movement.

In 1.5/1.6+, the player model is based on a box. The need for this was based on removing Havok from clientside PK. However the box isn't really a Havok object, it's coded from scratch - the distance and the direction of explosion push are measured to the centre of the player box. Although beneficial for netcode and some other issues, the changes had serious side effects on the original PK gameplay design.

The result of this change in 1.5 was that although rocket behaviour remained reasonable for rocket fights, for rocket jumps, the simplification of the physics with the rocket blast placed at the players feet lead to much impulse being horizontal as well as vertical. This lead to rocket jumps stopping players, or at least heavily reducing their horizontal velocity. This effect is considerably more pronounced on dedicated servers.

In 1.61/2 several things were tried by the developer, but ultimately the 'ugly hack' (within the sourcecode '-- ugly hack for MP Rocket Jump') chosen by PCF was a rocket jump fix that involved disallowing the player to lose speed in the direction of player's movement as a result of an explosion impulse (effectively the part of the push force that works against the speed vector was discarded). There was code put in to allow restricting that to explosions caused by player's rockets only or such,

but this wasn't completed and proved from testing at that point not to be a suitable fix.

This is also combined with a change between 1.5 and 1.6/2 of the explosion strength being increased by 12%. The reasons for this are that with horizontal pushback removed regardless of the angle, very shallow incline and lazy rocket jumping became possible. Normally, a small amount of horizontal pushback as is natural from a shallow incline rocket jump, would force players to be more precise with their rocket jumps ensuring steeper inclines (such as in 1.35). Because of this over-compensation in removing horizontal forces completely, the tweaking-hazard the testers immediately fell into was adjusting the explosion strength for these new lazy rocket jumps (it is possible to see how wrong this is by trying a truly vertical rocket jump on dm_sacred at GA - it is almost possible to get onto the roof with it from ground level - something impossible in 1.35). In rocket fights this leads to excessive 'bouncing', which ultimately leads (along with no horizontal rocket impacts) to a completely different game.

However, the lack of horizontal pushback is still the core problem surrounding the 'ugly hack' in 1.61/2.

The more sensible method is not to change how the rocket explosion affects the player in ALL cases, but only in the cases where the 1.5 blast is no longer suitable given the new player model. The first stage of the fix is to remove the 1.61/2 hack, which is done simply enough.

Secondly rocket blasts need to become based on the direction of the rocket, which is the most intuitive solution. That being that you would expect more horizontal blast from a rocket striking you in the face,

than one fired directly downwards beneath you. This equates to the shape of blast impulses being based on the direction of the rocket.

Thankfully there is a method available to modders to change the shape of the rocket explosion. The easiest way to visualise this change is imagine an unchanged explosion as a sphere, and the ability to change it into a distorted ellipsoid. NOTE: This ellipsoid is NOT the blast range, but a representation of the magnitude of impulse vectors from the centre of the ellipsoid (the source point of the explosion)

Figure 1 shows a range of shapes into which the explosion might be changed; this is purely to help visualise a range of blast distortions.

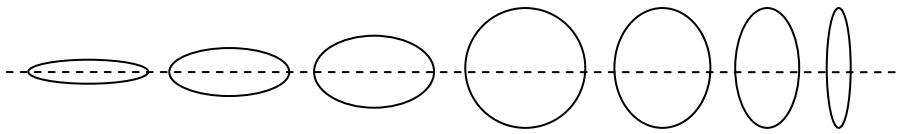

Figure 1. Range of Ellipsoids

The first point to note; is that ultimately, whichever shape is used, the height of the ellipsoid always needs to be constant, so that rocket jump height is always the same.

The width of the ellipsoid then becomes the determining factor in the amount of horizontal push the player receives. In order to verify when and where this factor is applied logically applied, we must look at what we are trying to achieve here again.

Firstly we noted that 1.5 rockets were good in rocket fights, but bad for rocket jumps. Secondly we noted that 1.61/2 rockets were good for

rocket jumps, but bad for rocket fights (no pushback on moving players lead to stale racing car gameplay).

To base the rocket blast on direction, the blast needs to be adjusted based on the velocity of the rocket, NOT on the velocity of the player as PCF assumed.

When basing the rocket blast on direction, the directional rocketfactor shouldn't necessarily create a linearly proportional change to the blast. That is to say that the horizontal force reduction for a 60 degree (50% vertical velocity) rocket should not be half that of a purely vertical rocket. Such a change could cause close rocket fights to be less effective, and for rocket jumping to become lazy in the sense that players wouldn't need to place the rocket too far vertical.

Instead the rocketfactor would need to be changed based on incline to a higher order. The higher the order used, the more pronounced the change becomes as the rocket incline approaches the vertical (see figure 2). This way only pure rocket jumps are most affected by the change in blast shape, while normal rocket play (circa 1.5) remains relatively untouched.

A higher rocketfactor order would increase the amount of differentiation between a rocket jump-based blast, and one which might be expected in a typical rocket fight.

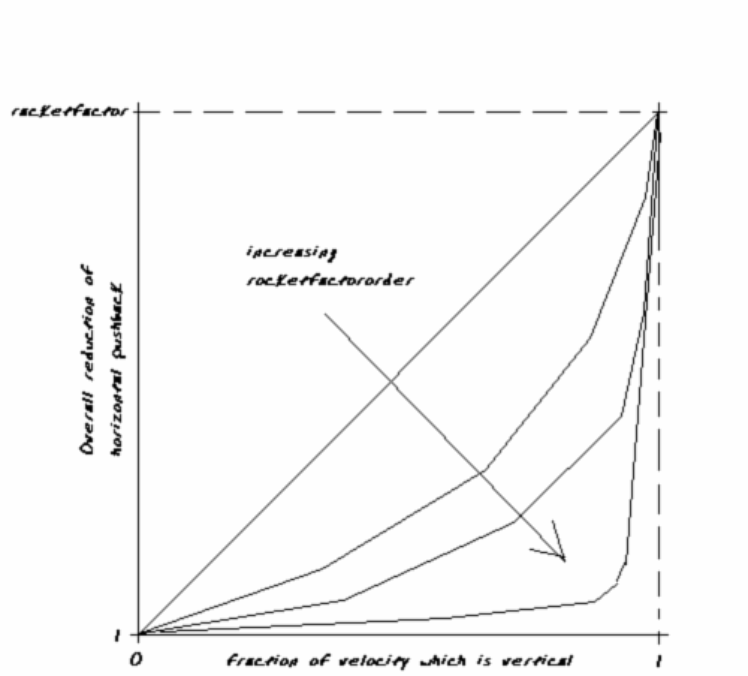

Figure 2. Horizontal Pushback Reduction versus Fraction of Incline

Apart from a visualisation tool, Figure 2 has another important implication for the process here; in allowing a functional curve to be closely correlated to the 1.35 (or ideal) behavioural data-set.

In summary, the method proposed here removes the need for an 'ugly hack' to try to correct rocket behaviour between two very different simulation systems, whilst presenting an intuitively sound systematic approach to solving the problem.