Doomsday Engine - Feature #1318

Creating Bullet Holes

2004-01-03 12:24 - danij

Status:	Closed	**Start date:**	2004-01-03
Priority:	Normal	**% Done:**	100%
Assignee:	skyjake		
Category:			
Target version:			

Description

Use the following definition to create a bullet hole:

1. Bullet Hole
   Generator {
   State = "PUFF1"
   Flags = blendrsub | static | spawn | srcvel
   Speed = 1
   Spawn age = 1
   Max age = 250
   Particles = 1
   Spawn rate = 1
   Stage { Type = "pt_tex02"; Tics = 1;   Flags =
   stagetouch | flat   Radius = 30;   Color { .5 .5 .5 .01 }; };
   Stage { Type = "pt_tex02"; Tics = 200;   Flags =
   flat   Radius = 3;   Resistance = 1;   Color { .5 .5 .5 .8 }; };
   Stage { Type = "pt_tex02"; Tics = 10;   Flags = flat
   Radius = 3;   Color { .5 .5 .5 .8 }; };
   Stage { Type = "pt_tex02"; Tics = 10;   Flags = flat
   Radius = 3;   Color { .1 .1 .1 0 }; };
   }

Unfortunetly the particles are never rendered flat
against the wall/floor as expected.

Not strictly a bug as the reason is because the bullet
puff sprite is not spawned close enough to the wall for
it's associated particles to detect they are touching the
wall.

What can be done to get this working correctly?
Perhaps giving puff 1 a velocity that will force the
associated particle "into" the wall?

Dani J

Labels: Graphics

History

#1 - 2004-01-05 16:52 - skyjake

Logged In: YES
user_id=717323

The logical solution would be that the spawned particle(s) in this
scenario would move (very fast) in the direction of the bullet/missile
trajectory and stick to whatever surface gets in the way. There are
some issues to consider, though.

For one thing, the particle code has a hard time figuring out which way
the particles should be moving because the bullet logic is done in the
Game module and the particles code is in the engine. But supposing
the Game provides a direction vector?

Then there is the problem with the inaccuracy of particle collision physics. Because there can be so many particles at once, the physics have been simplified so that the collision detection does not bother to look for accurate collision points; it just checks whether a move ends up in a wall, and if it does it'll consider it a collision and modify the particle momentum. If the particle is moving at a high speed, it might end up several units away from the surface.

If one were to implement a proper decal system (and it is indeed looking quite doable at the moment) there would need to be a particle flag that would tell the particle code that when a collision occurs, the accurate coordinates should be calculated and a decal should be generated at the point of contact.

Now the question is, what should be decal look like? Duplicating the appearance of the particle would seem convenient.

#2 - 2004-01-05 17:23 - danij

Logged In: YES
user_id=849456

Yes I think for now duplicating the appearence of the particle
is probably the best way forward.

The only thing with decals is that really they should be
fullcolor with blending rather than one color. This way it's
easier to create VERY convincing gore/scorch/bullet effects.

Would a decal system mean that they would recieve proper
edge clipping?

The other consideration is what happens with non hitscan
weapons (or is this not an issue)?

#3 - 2004-02-17 21:23 - danij

Closing as duplicate.

#4 - 2006-10-23 20:24 - skyjake

Logged In: YES
user_id=717323

Yes, a decal system would mean proper edge clipping. The decals would be rendered pretty much like dynamic lights, only with different textures and more suitable blending modes.

Non-hitscan weapons aren't really an issue. The real issue is how to determine which way the decal-generating particles should move from the source. For example, a rocket explosion. The game would have to just tell the engine that there is an associated direction vector: the rocket's former momentum.

#5 - 2009-10-04 21:58 - danij

Logged In: YES
user_id=849456

Making this an RFE.

#6 - 2009-10-04 23:05 - danij

The current objlink and dynlist implementation could easily be modified to handle decals.