Doomsday Engine - Bug #1868

Feature # 1648 (Progressed): Complete vanilla DOOM emulation

[Doom] Revenant missiles randomly switch from non-homing to homing

2014-09-27 00:18 - vermil

Status:	Closed	**Start date:**	2014-09-27
Priority:	High	**% Done:**	100%
Assignee:	skyjake		
Category:	Defect		
Target version:	1.15		

Description

The attached save game from BTSXe2 Map25 shows a non-homing Revenant missile become homing if the player performs no action a split second after loading the game.

Performing an action immediately appears to cause the missile to remain un-homing.

I've encountered this bug a few times over the Dday 1.9 series when playing (i.e it had nothing to do with saving/loading, just while playing), but I've not been able to catch it in a save game until now.

Associated revisions

Revision 9d412204 - 2015-01-04 16:54 - skyjake

Fixed|Doom: Behavior of Revenant homing missiles

mapTime is saved and restored when the game is saved and loaded while
GAMETIC is not.

mapTime seems to be the de facto solution for this kind of behavior;
A_Tracer seemed an exception (reason unknown?).

IssueID #1868

History

#1 - 2014-09-27 00:21 - vermil

To elaborate, the action I performed immediately upon the save game loading, was moving forward by constantly tapping the forward key.

#2 - 2014-10-05 01:17 - danij

Initial testing reveals that this is indeed not a savegame issue. After instrumenting and comparing interpretation of the tracer pointers between a session in which the target is tracked correctly vs one where the target is lost shows identical output. Therefore the target must be lost some time after loading has completed.

This has nothing to do with player input. Repeatedly loading this savegame shows that the tracer will seemingly randomly lose its target once every 3-4 times the session is loaded.

#3 - 2014-10-05 01:21 - danij

- Tags set to Doom, PlaySim

- Category set to Defect

#4 - 2014-10-05 02:37 - danij

Further testing shows that the tracer's target is in fact not being lost at all. The problem is that the deterministic behavior inherent in the original logic has been subverted as a result of changes to how game time is managed.

The A_Tracer action will short-circuit the turn-to-face-player logic according to the value of (gametic & 3).

In Doomsday, gametic (int) has been replaced with a conversion from the timing loop's gameTime (timespan_t). This gameTime is sometimes reset to zero when the current map changes, meaning that upon loading a saved game, **all** tracers will occasionally revert to non-homing behavior immediately. (Note that gameTime is also changed in various other places/times.)

However, the gametic is **not** remembered in saved games, either by Doomsday or the original game. Therefore, this bug must also affect vanilla to the extent that tracers switch homing/non-homing behavior seemingly randomly when loading a savegame. (Note that due to the fact targets are not saved correctly in vanilla in any case, this behavior is doubly broken in vanilla).

Open question: Why is it that tracers do not revert to homing behavior once (((int)gameTime) & 3) != true?

#5 - 2014-10-05 02:39 - danij

- Category changed from Defect to Vanilla emulation

#6 - 2014-10-05 02:40 - danij

- Parent task set to #1648

#7 - 2014-10-05 03:58 - danij

- File DoomSav2.save added

#8 - 2014-10-05 03:59 - danij

I have attached a DOOM II savegame, DoomSav2.save which makes debugging this issue easier. In this savegame I have orchestrated a tracer in a stable orbit around the player. When loading the savegame the tracer should continue to orbit the player ad infinitum.

#9 - 2014-10-05 06:09 - danij

Logging the values of GAMETIC and (((int) GAMETIC) & 3) in A_Tracer shows that this bug occurs when the step value for a tic results in a predictable integral sequence of 1, 3, 5, 7, 9. If the step causes the integral to jump out of that sequence then the tracer will resume its homing behavior.

This suggests to me that the conversion from fractional time to game tic is fundamentally flawed when the game tic is assumed to step uniformly 0, 1, 2, 3...

#10 - 2014-10-05 06:24 - danij

- Category changed from Vanilla emulation to Defect

- Priority changed from Normal to High

- Target version set to 1.15

#11 - 2014-12-20 17:55 - skyjake

- Status changed from New to In Progress

- Assignee set to skyjake

#12 - 2014-12-20 21:09 - skyjake

Here's my analysis of the situation. The GAMETIC & 3 is indeed at the root of the problem. However, this function (A_Tracer) is only called on sharp ticks, so there's nothing wrong with the tick counting.

The condition is supposed to mean that the function is only executed once every four ticks. Furthermore, the mobj state has a tick count of 2, meaning the function is only called every two ticks. The intended end result is that the logic is run on every second time it gets called.

The nondeterministic nature of the glitch comes from GAMETIC not being reset when the game is loaded. Therefore, if the restored mobj happens to begin its cycle on an odd tick (bit 1 set), it never gets called on a tick where bit 1 isn't set (2 tick state cycle)— therefore, it never gets executed.

How to fix this, though? It seems to me the safest solution would be to save and restore the game tick counter. Alternatively, one could use a per-mobj counter that ensures the logic gets executed only on every second time the state cycle calls it, although that would not ensure that all A_Tracer's in the map get executed on the same ticks (vanilla behavior risk).

#13 - 2015-01-04 16:29 - skyjake

Looking at this closer, maps in a Hexen-style hub should probably not restore the gametic when loading maps from the current hub. This would make things easier since one wouldn't have to keep the gametic of each hub map separately in the metadata.

#14 - 2015-01-04 16:45 - skyjake

Studying the code more, another solution comes to mind. A_Tracer is pretty much the only place where GAMETIC is used like this — elsewhere in the code mapTime is used for this purpose. I can't think of any reason why A_Tracer couldn't use mapTime, too, and since that is already restored when saving a game, it should fix the problem. Also, this would already handle the Hexen case appropriately.

#15 - 2015-01-04 16:54 - skyjake

- Status changed from In Progress to Resolved

- % Done changed from 0 to 100

#16 - 2015-01-07 13:00 - skyjake

With this fix, the attached *DoomSav2.save* now always seems to restore the missile onto stable orbit. This save file was very helpful for debugging the

issue, btw!

#17 - 2015-01-15 18:02 - skyjake

- Status changed from Resolved to Closed

Files

DoomSav0.save	170 KB	2014-09-27	vermil
DoomSav2.save	16 KB	2014-10-05	danij