17 year old 'Carmageddon' debugging symbols file dumped (self.gamedev)

396 soumis 2 jours de ça * par jeff-1amstudios

In the Carmageddon Splat Pack folder, there is a file called 'DETHRSC.SYM', last modified 13th November 1997.

It has sat there, un-noticed and un-loved for the last 17 years, ignored by the internet. Having made a remake of the Carmageddon engine, and being generally curious about random binary files, I tried to figure out the file format. Immediately by looking at it in a hex editor, it was obviously a debugging symbol file, the question was which type of symbol file? Of course, there are many symbol files with a .sym extension, and after some trial and error, it turned out to be a Watcom symbol file. I grabbed a copy of OpenWatcom and fired up the debugger, wd. It could read the symbols, but I never found any Carmageddon executable that matched up with it. It seems likely it was left there by mistake from a debug build.

Using wd to look at the symbols 1 by 1 in a little DOS window quickly becomes tiring, so then I wanted to dump the symbols out. For that, I needed the source code for the Watcom tools, and a working Open Watcom development environment.

…Fast forward a while getting the environment up and finding where the code for handling symbol files lives…

In the Watcom world, symbol file support is provided by various DIPs. To use a DIP dll requires the calling program to implement various client-side methods to allow the DIP to alloc memory etc, and then to provide callback functions for the DIP to call when walking the symbol list. It's all pretty complex, but luckily there are a couple of utilities which illustrate generally how it should be done. I based jsonsymdump off dipdump (which is advertised as dumping symbol files to text format, but crashes on DETHRSRC.SYM). I did the minimal amount of work in C required to generate a valid json file, then wrote a node.js script to take that json file and generate some semi-valid-ish c files.

- Browsable symbol output here: https://github.com/jeff-1amstudios/carmageddon1-symbol-dump
- More details here: http://1amstudios.com/2014/12/02/carma1-symbols-dumped/

61 commentaires **partager**

l'intégralité des 61 commentaires

triés par: **apprécié**

[–] **FoxyKit** 130 points 2 jours de ça

MakeFlagWavingBastardWaveHisFlagWhichIsTheProbablyTheLastThingHeWillEverDo()

Holy shit I laughed heartily at that.

perma-lien

[–] **Boracho1121** 74 points 2 jours de ça

how about this one;

DrawSceneyMappyInfoVieweyThing()

Its nice to be reminded that I'm not the only who makes up amusing names for things.

perma-lien **parent**

[–] **DragoonDM** 53 points 2 jours de ça*

void* GetLargestPacketSizeOoErBetterInsertLinfordChristieJokeHere();

void* PDGrabLargestMammaryWeCanPlayWith();

void* AmIGettingBoredWatchingCameraSpin();

void* CalcOpponentConspicuousnessWithAViewToCheatingLikeFuck()

Few of my favorites.

perma-lien **parent**

[–] **HostisHumaniGeneris** 32 points 2 jours de ça

void* CancelPendingCunningStunt();

perma-lien **parent**

[–] **akagetsu01** @akagetsu01 11 points 2 jours de ça

oh man, these sound so cool...I always get told off by my manager when I try and give variables slightly amusing names... >.>

perma-lien **parent**

[–] **philbgarner** 11 points 2 jours de ça

hahah I would insert amusing names myself without hesitation, but the thought of other people that I'm responsible for doing the same thing is frankly scary. Just imagining code-reviews with like my boss' boss or something and there's a method called FuckThisFuckery(); or something. :P

perma-lien **parent**

[–] **st33d** @st33d 10 points 2 jours de ça

Those humorous names appear in the stack trace when your game crashes.

As tempting as it is to put some comedy bullshit in your code, there is the chance you will forget about it and then you will be a story on Daily WTF:

http://thedailywtf.com/articles/We-Burned-the-Poop

perma-lien **parent**

[–] **badsectoracula** 4 points 2 jours de ça

But those particular humorous names also give you an idea about what the code is doing, they aren't totally random. In fact i found this the most interesting part about them - they are both funny and make sense.

perma-lien **parent**

[–] **suspiciously_calm** 3 points 2 jours de ça

Amusing variable/method names should be easter eggs, not all over the place like here.

perma-lien **parent**

[–] **pslayer89** 2 points 1 jour de ça

I don't care what my professors think, I am making up names like these in my code from now on!

perma-lien **parent**

[–] **MrGoodGlow** 2 points 2 jours de ça

Every single one of my subs that goes and cleans up something is always Mrclean

perma-lien **parent**

[–] **GinjaNinja32** 3 points 2 jours de ça

OoerrIveGotTextInMeBoxMissus()

perma-lien **parent**

[–] **mr_sharpoblunto** 38 points 2 jours de ça

Nice! This kind of programmer archeology is super interesting and also important in the long term to make sure that we don't lose all the early gaming culture that is locked up in archaic binary formats.

perma-lien

[–] **foofly** 7 points 2 jours de ça

Exactly. There really needs to be some sort of museum for this sort of stuff.

perma-lien **parent**

[–] **RenaKunisaki** glHF(); 5 points 2 jours de ça

TCRF is all about this kind of thing.

perma-lien **parent**

[–] **jeff-1amstudios** [S] 58 points 2 jours de ça

Just found opponent::CalcOpponentConspicuousnessWithAViewToCheatingLikeFuck() which really makes sense to anyone who has played the original game. The minute you turn your back on a cpu opponent car, they teleport away (definitely cheating like fuck)

perma-lien

[–] **planetworthofbugs** 8 points 2 jours de ça

Fantastic stuff - this really takes me back to many great times :)

perma-lien **parent**

[–] **dazzawazza** http://www.indiedb.com/games/smith-winston 19 points 2 jours de ça

If anyone cares the BRSRC13 files are (I think) from the BRender render engine that Carmageddon used.

http://en.wikipedia.org/wiki/Argonaut_Games#BRender

ps:

```
Opponent._gWanky_arse_tit_fuck
```

looks like an interesting property :)

perma-lien

[–] **jeff-1amstudios** [S] 15 points 2 jours de ça

you're right about the BRender files - I mentioned them near the end of the post. When I was building a remake of the Carma engine (the OpenC1 project), I had to implement parsers for all the weirdo BRender file formats.

perma-lien **parent**

[–] **samlittlewood** 39 points 2 jours de ça

Err - sorry.

In my defense, they were never meant to be hand parsed - there was a common serialization layer that knew how to deal with various composite data structures.

perma-lien **parent**

[–] **jeff-1amstudios** [S] 16 points 2 jours de ça

hi Sam!

Wow, I never thought we'd have a BRender expert commenting on this thread. :)

perma-lien **parent**

[–] **samlittlewood** 14 points 2 jours de ça

Ha - It is great to see you and other people still having fun with stuff and making it work on modern platforms - alongside the technical skills, there was this insane, gleeful joy about the way that Stainless went about their work that deserves remembering.

perma-lien **parent**

[–] **registered-user** 13 points 2 jours de ça

Aha! Now you've exposed yourself, Sam, any chance you're up for sharing anything else about the BRender engine? I'm sure we have a couple of interesting puzzles we've not cracked over the years :)

perma-lien **parent**

[–] **samlittlewood** 45 points 2 jours de ça

Some random notes:

BRender started as a cooldown having finished a real-mode PC flight simulator (Atac) that was mostly written in x86 assembler. That had followed the common practice at the time of expensive offline compilation of 3d models into highly specialized bytecode - each model was a separate procedure that knew how to draw, animate, and sort itself.

As a reaction to this, I took my shiny new Pentium and wrote, in C, a renderer that would quickly extract the front-most surface of a general 3d scene in scanline order. It did work, and a couple of games were started on it - however complexity crept back in and the FX Fighter team were hitting limits. A simple software Z buffer renderer was written as a way of letting those guys get on with their work. Embarassingly, this far outperformed the previous renderer with it's complex data structures and cache hostile patterns, and as a bonus had a nice predictable performance no weird limits.

Following that, and some rather overdue reading up, BRender was born as a pretty standard z buffered triangle rendering pipeline following published SGI, SUN etc. research. A departure from this was a projected texture mechanism by Dan Piponi (sigfpe) that walked lines of constant Z. The triangle rasterizer was also curious in that it arranged to always draw scanlines starting from the major edge of the triangle, moving left or right as appropriate.

A line of development that sadly never saw the light of day was Dan's exploration of using 64k LUTs to encode interesting functions - char = fn(char, char). Eg: by quantizing vectors down to <256 directions and using paletted textures, he got sensible looking bump mapping. He also had a prototype mechanism for composing all these effects at run time.

These, and other ideas fell under the wheels of the hardware rendering card that arrived - even the good ones were limited to gouraud + texture. Per pixel opportunities only really came back with Geforce 3. The development work then turned into a parade of device driver ports for spectacularly awful rendering hardware. (Notable exceptions were renditon and 3dfx)

Carma was probably the highlight of working on BRender - they were a great team - I used to drive down to the Isle of Wight to do support, every visit would see more craziness. They were unashamedly making something that they themselves wanted, with really very few concessions.

perma-lien **parent**

[–] **jeff-1amstudios** [S] 12 points 2 jours de ça

> every visit would see more craziness

Confirmed: http://www.carmageddon.com/blog/oh-christ-mas-party

Those the kinds of office christmas parties that I've always wanted but haven't had (so far)

perma-lien **parent**

[–] **nobby_SG** 11 points 2 jours de ça

Hi Sam, how are you doing mate? Thanks for the kind words - I feel like we ought to quote your last paragraph in our promo material!

perma-lien **parent**

[–] **samlittlewood** 4 points 1 jour de ça

Neil! really good to see you guys bringing it back - and with the right attitude. Quote away!

perma-lien **parent**

[–] **nobby_SG** 4 points 1 jour de ça

Thanks Sam, and yeah, will do!

perma-lien **parent**

[–] **mysticreddit** ProfessionalGameDevTurnedIndie 4 points 2 jours de ça

> each model was a separate procedure that knew how to draw, animate, and sort itself.

So basically the 3d version of *compiled sprites*? :-)

perma-lien **parent**

[–] **invadrzim** C++/DirectX/C#/XNA/ Getting feet wet in Unity 2 points 2 jours de ça

From the perspective of someone who thinks of models as a bunch of coordinate data one hands off to the graphics card to do its thing and it makes things magically appear on screen, could you (or anyone else) elaborate on what each model being a separate procedure that sorts and draws itself means? I try to understand it but I cant quite wrap my head around what that means exactly.

perma-lien **parent**

[–] **samlittlewood** 4 points 1 jour de ça

badsectoracula has covered this already, but I'll add some more detail about the extremes to which this was taken.

(NB: in the context of this thread, this is emphatically NOT what BRender was about - it was the reaction to it).

The reasons for the compiled models were: reduced transform time, compressed models, parameterization/animation, and visibility sorting.

Transforms & Compression:

In those days, multiply were expensive, and divides had to be signed off by management. Anything to reduce these was good, so we exploited symmetries. The opcodes would include things like 'n vertices reflected in YZ plane' - each input vertex would generate 2 on the output from only 1 transformed vector. Another example is 'N evenly spaced points between starting at p, spaced by v' Only need the transform p &v once, then repeatedly accumulate. Continue for points around an arc, etc.

The intermediate transformed vertices could become inputs to further opcodes, so if you did something like considering the 3 vectors of the model->view matrix as the first three transformed points (ie: the unit cube), you could generate a wide range of simple models by simple power-of-2 scales and sums.

Paramerization & Animation:

As well as the model->view matrix, the compiled model had access to the rest of the per instance structure (effectively a 'this' pointer), and also a few global vars. This meant that instance or global vars could become arguments to opcodes - a very common use was having a 0..1 (0 - 0xffff) var drive interpolation along a piecewise linear curve (and then become input to the above vertex gen opcodes). This would allow things like folding undercarriage (+ all doors etc.) spinning radar antenna, opening doors etc. Use a clock from global var to drive nav lights flashing .....

Visibility Sorting:

The final piece of the puzzle is that there was no Z buffer or any other visibility determination - the primitives had to be drawn from back to front to view order to achieve a painters algorithm. We used a BSP tree within models - the decision tree was hard coded in the model with plane test and branch opcodes. It was possible optimize this for cases like loads of coplanar prims (single test), or stuff only visible through a hole eg: window (if window plane not visible, jump over all prims within - assumes camera is never going to be inside model), or a convex cluster (no tests at all).

All this was often hand coded by technical competent artists - and the 3d model generation was a significant undertaking.

perma-lien **parent**

[–] **badsectoracula** 3 points 1 jour de ça

At the time if you wanted to do 3D you had to write your own rasterizer, z sorting (or zbuffer implementation in the rasterizer), transformation, clipping, etc. Your code may look something like

```
if (model_in_frustum(model, camera_frustum)) {
    for (i=0; i < vertices; i++) transform_vertex(vertex + i);
    sort_triangles_by_depth(vertex, triangle, camera_pos);
    for (i=0; i < triangles; i++) draw_triangle(
        vertex + triangle[i].a, vertex + triangle[i].b, vertex + triangle[i].c,
```

```
        texture);
}
```

All the above functions are generic - they read data from a model struct, triangle struct, etc to work, they work over loops that can handle a variable amount of triangles, etc. A common approach (especially for 2D graphics) was to take those model structs and generate machine code that basically did the above but with loops unrolled, triangle/index/etc counts hardcoded, unnecessary ifs removed etc - basically "baking" the code to the model. So you'd essentially have something like

```
void (*airplane)(vec3 pos, int rotation) = compile_model("airplane");
```

and then to render the airplane you'd just call

```
airplane(pos, rotation);
```

Now, in modern times (and by modern i mean anything since the pentium 1 days :-P) this not only would kill your CPU's cache but since the last decade or so, the CPUs are more optimized for the former case than the latter case. But at a time when even an extra if or multiplication could have a massive impact in performance, such things were necessary.

perma-lien **parent**

[–] **invadrzim** C++/DirectX/C#/XNA/ Getting feet wet in Unity 2 points 1 jour de ça

I think I understand it now, thanks!

perma-lien **parent**

[–] **fwork** 9 points 2 jours de ça*

Oh hi! So you're the sam I see in these embedded strings:

```
$Id: surface.c 1.11 1995/03/29 16:41:24 sam Exp $
```

Can I just say, BRender is an awesome renderer? It's amazing the level of performance and effects you can get on such limited hardware. Doing all this in software... I have a real appreciation for how much optimization must have gone into this code, since I've been starting at disassemblies of it for so long.

I'm trying to reverse engineer a program that uses BRender (Microsoft 3D Movie Maker, from 95/96). I've made some good progress and I currently have a proof-of-concept which extracts the models/textures out of RAM and renders them through OpenGL. I did some deep-web spelunking (over several years!) and eventually found a copy of the headers and some libraries for a similar revision (The SDK included on some OS/2 development discs!), which was incredibly helpful.

Is there any chance you'd happen to have any old copies of the BRender SDK? Having a version that more closely matched the one I'm reverse engineering (Windows, compiled with Visual C++ 2.1, fixed point, zbuffer, mid-90s vintage?) would really help. (If it'd help, I can extract all those version strings to give you a better idea of what version it is I'm looking for)

Once again, thanks for building such an awesome engine!

perma-lien **parent**

[–] **samlittlewood** 4 points 2 jours de ça

Thanks! 3DMM was good. I seem to recall it had some roots in Microsoft's acquisition of SoftImage.

I don't have an SDK to hand, but there might be some disks in storage - I'll have a look.

VC2.1 - wow that brings it back!

perma-lien **parent**

[–] **mavispuford** 1 point 2 jours de ça

Wow, I had a blast with 3DMM when I was growing up. Best of luck with your work!

"I wanna be lean, AND mean!"

perma-lien **parent**

[–] **ShadyPotat0** 36 points 2 jours de ça

```
void* _gWanky_arse_tit_fuck;
void* CalcOpponentConspicuousnessWithAViewToCheatingLikeFuck();
void* _gPalette_fuck_prevention;
void* DontLetFlicFuckWithPalettes();
void* LetFlicFuckWithPalettes();
void* AmIGettingBoredWatchingCameraSpin();
void* OoerrIveGotTextInMeBoxMissus();
void* FindAHeadupHoleWoofBarkSoundsABitRude();
void* SaySorryYouLittleBastard();
void* MakeFlagWavingBastardWaveHisFlagWhichIsTheProbablyTheLastThingHeWillEverDo();
void* PaletteFuckedUpByDrugs();
void* CarWorldOffFallenCheckThingy();
```

Ahaha, this is beautiful.

perma-lien

[–] **maximusawesomus** 1 point 1 jour de ça

The Isle of Wight for you lol

perma-lien **parent**

[–] **kiddico** 21 points 2 jours de ça

heehee "void* KnackerThisCar();"

perma-lien

[–] **not_mad_just_upset** Insurgence: Shadow Operation dev. 11 points 2 jours de ça

Fantastic work. This is the kind of posts I like to see here!

perma-lien

[–] **sli** Unity 3D+2D 8 points 2 jours de ça*

```
void* _gDoing_physics;
```

Don't mind me. Just doing physics.

EDIT: `void* FindAHeadupHoleWoofBarkSoundsABitRude();`

perma-lien

[–] **nobby_SG** 8 points 2 jours de ça

Heh, nice one Jeff... Having been in charge of The Art Stuff on Carma, a lot of this internal code stuff passed me by. Much of it would appear to have the stamp of Batwick (Patrick Buckland, Stainless CEO and coder) upon it!

perma-lien

[–] **jeff-1amstudios** [S] 1 point 1 jour de ça

Hey nobby! Thanks! Glad you enjoyed it, I had a good laugh reading through them too.

/r/gamedev is all the better for your presence :)

perma-lien **parent**

[–] **glitchdetector** 7 points 2 jours de ça

```
SaySorryYouLittleBastard()
FindAHeadupHoleWoofBarkSoundsABitRude()
DontLetFlicFuckWithPalettes()
WakeUpOpponentsToTheFactThatTheStartHasBeenJumped()
```

perma-lien

[–] **PWG_myk** 16 points 2 jours de ça

A xbox360 game I worked back in 2007ish on had the following line in it's codebase;

bool g_bPopeGregoryThirteenthAndAllHisRupturedCatamites = false;

I can't recall what it was used for now, but it was understood that it was a bandaid over a nasty and elusive bug and no-one was to attempt to remove it

perma-lien

[–] **FoxyKit** 16 points 2 jours de ça

$50,000 a pop to get through microsoft's xbox certification, you bet your ass nobody fucked with that var.

perma-lien **parent**

[–] **jeff-1amstudios** [S] 5 points 2 jours de ça

You don't want to be *that* guy

perma-lien **parent**

[–] **PWG_myk** 2 points 2 jours de ça

Aye, and it had been around a long time as well, the engine has been in use in various forms/evolutions for roughly 15 years

perma-lien **parent**

[–] **KSKaleido** 2 points 2 jours de ça

It's only 10k if you fail the first time, though! Hwat a deal!

perma-lien **parent**

[–] **FoxyKit** 3 points 2 jours de ça

I'd be tempted to put in a sleep(5000) in the logo loading routine just to take advantage of those savings!

perma-lien **parent**

[–] **Bratmon** 1 point 1 jour de ça

Or do sleep(4750) and leave an air of mystery as to whether it will pass or not.

perma-lien **parent**

[–] **fwork** 2 points 2 jours de ça

Oh, awesome, it's BRender. I'm also reverse engineering a BRender-engine program (Microsoft 3D Movie Maker).

I wonder how the version used in Carmageddon differs? it should be around the same age... maybe slightly newer, actually.

perma-lien

[–] **fwork** 3 points 2 jours de ça

Oh, I don't know if it'll help with what you're doing, but I've found a copy of the .lib files for a version of the BRender SDK (the OS/2 version, floating point enabled), as well as the header files.

Knowing the layout of the in-memory structures was REALLY handy, but the .libs are really only useful for matching functions through disassembly, but if you have debugging symbols you probably already have that.

perma-lien **parent**

[–] **kayamon** 3 points 2 jours de ça

There's a whole bunch of PlayStation1/2-era games that accidentally leave symbol files on the shipped disc.

e.g. Jak & Daxter, Ico...

One day I keep meaning to do some deep source-code investigation on these.

perma-lien

[–] **mysticreddit** ProfessionalGameDevTurnedIndie 1 point 2 jours de ça

The original Master of Orion 2 left Borland C++ debugging symbols in too :-)

perma-lien

[–] **RenaKunisaki** glHF(); 1 point 2 jours de ça

Nice. A few Gamecube games have symbol files left on disc as well. I've been interested in Mario Kart's but apparently it also is for a debug version and doesn't match up with the retail executable.

perma-lien

[–] **sokaroka** 1 point 2 jours de ça*

Interesting thread OP AND fucking Sam Littlewood showed up, pretty cool, have a read guys if you missed the convo OP had with him!

perma-lien

[–] **Blinks-ap** 1 point 18 heures de ça

Mother of god these variables are hillarious!

perma-lien

[–] **firakasha** 1 point 2 jours de ça

TIL there is actually a possibility of someone else some day laughing at all those funny names I use in my code. All those superfluous key strokes are not wastes, hurrah!

perma-lien

> [–] **DragoonDM** 3 points 2 jours de ça
>
> Well, if you accidentally leave in a debugging symbol file, anyway, or use a language that doesn't obliterate all of your function and variable names at compile time.
>
> **perma-lien** **parent**
>
> > [–] **firakasha** 3 points 2 jours de ça
> >
> > "accidentally"
> >
> > kekekekeke....
> >
> > **perma-lien** **parent**

?