
DOOM3
DOOM3.COM
© 2003 id Software, Inc. All rights reserved.
ACTIVISION

DOOM III

INTRODUZIONE

- Rilasciato nel 2004, terzo capitolo di Doom
- Primo gioco sviluppato su motore grafico idTech4
- Motore grafico innovativo: unified lighting and shadowing, GUI surfaces, animazione e scripting complessi
- Source code disponibile sul ftp della id, ma anche in svariati git!
- In particolare: https://github.com/TTimo/doom3.gpl
  Questa dovrebbe essere la prima release ufficiale.

- Sviluppato in Visual Studio .NET (ma non c'è una singola linea di codice C#)
- E' possibile compilarlo con Visual Studio 2010

INTRODUZIONE

- Ovviamente più complesso dei motori grafici di Wolfenstein, Doom e Quake

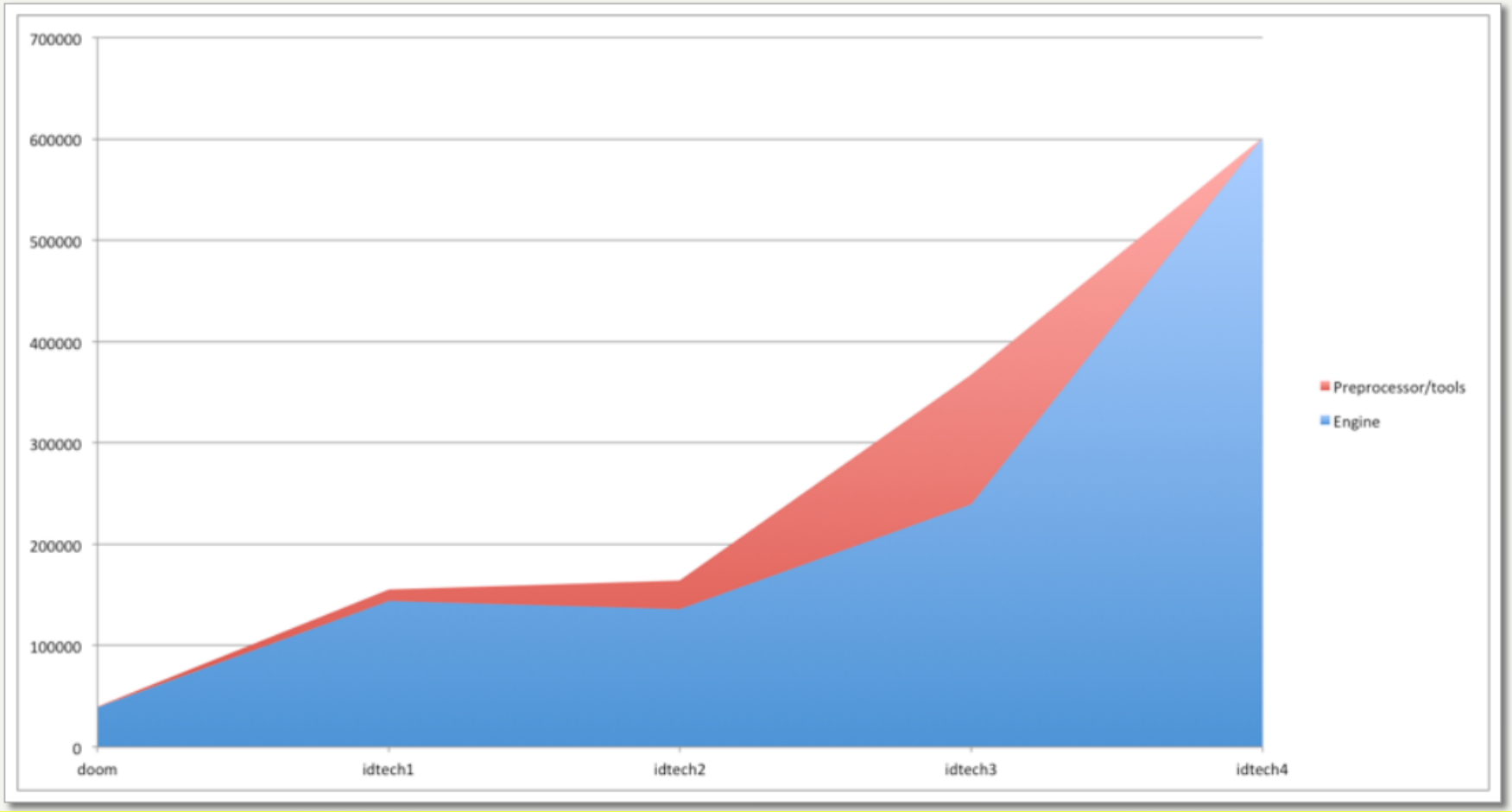
```
2180 text files.
2002 unique files.
626 files ignored.

http://cloc.sourceforge.net v 1.56  T=19.0 s (77.9 files/s, 47576.6 lines/s)

-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
C++                            517          87078         113107         366433
C/C++ Header                   617          29833          27176         111105
C                              171          11408          15566          53540
Bourne Shell                    29           5399           6516          39966
make                            43           1196            874           9121
m4                              10           1079            232           9025
HTML                            55            391             76           4142
Objective C++                    6            709            656           2606
Perl                            10            523            411           2380
yacc                             1             95             97            912
Python                          10            108            182            895
Objective C                      1            145             20            768
DOS Batch                        5              0              0             61
Teamcenter def                   4              3              0             51
Lisp                             1              5             20             25
awk                              1              2              1             17
-------------------------------------------------------------------------------
SUM:                          1481         137974         164934         601047
-------------------------------------------------------------------------------
```

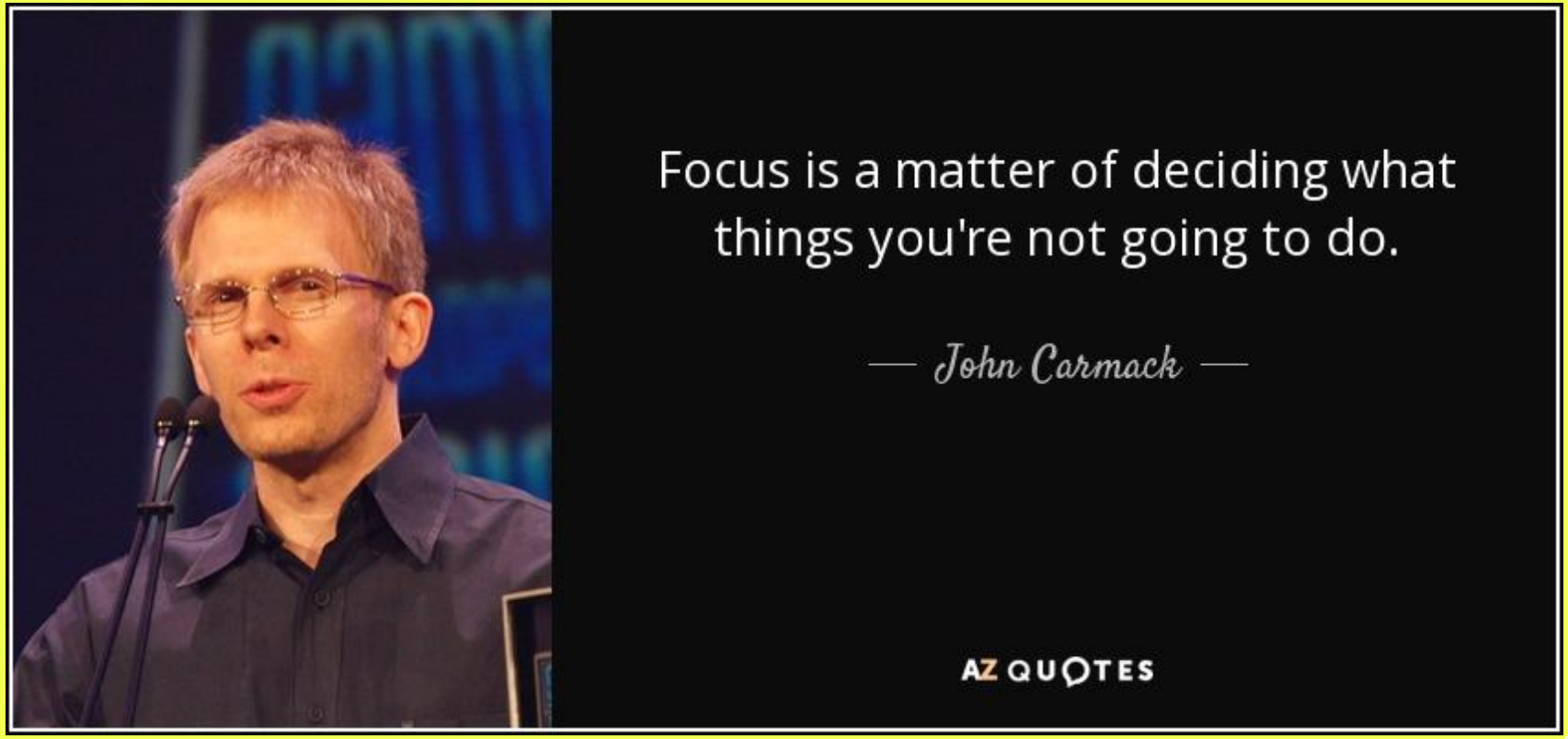
INTRODUZIONE

#Lines of code	Doom	idTech1	idTech2	idTech3	idTech4
Engine	39079	143855	135788	239398	601032
Tools	341	11155	28140	128417	-
Total	39420	155010	163928	367815	601032



INTRODUZIONE

- Oggi non parliamo dell'engine in sé, ma verranno illustrati una serie di commenti da parte del creatore dell'engine, John Carmack



- Analisi del engine: http://fabiensanglard.net/doom3/index.php

COMMENTI SUL CODICE

- Sul linguaggio:
- idTech4 è stato il primo motore della idTech scritto in C++:
- *DOOM is our first game programmed in C++. I actually did the original renderer in straight C working inside the QUAKE III Arena framework, but most of it has been objectified since then, and all of our new code is set up that way.*
- *[…] I sort of meandered into C++ with Doom 3 – I was an experienced C programmer with OOP background from NeXT's Objective-C, so I just started writing C++ without any proper study of usage and idiom. In retrospect, I very much wish I had read Effective C++ and some other material.*
- *[…] Today, I do firmly believe that C++ is the right language for large, multi-developer projects with critical performance requirements, and Tech 5 is a lot better off for the Doom 3 experience.*

COMMENTI SUL CODICE

- Sul linguaggio:
- Uso di astrazione e polimorfismo
- Tutti gli asset del gioco sono memorizzati in human-readable format e interpretati da un parser
  In hindsight, this was a mistake. There are benefits during development for text based formats, but it isn't worth the load time costs. It might have been justified for the animation system, which went through a significant development process during D3 and had to interact with an exporter from Maya, but it certainly wasn't for general static models.
- Template usate solo nelle classi di utilità di basso livello (idLib)
- Un sacco di commenti!!! Circa il 30% del codebase
- Documento coding style conciso e minimale

COMMENTI SUL CODICE

- Sulle scelte di programmazione: const
- Il codice è particolarmente rigido nei confronti della sintassi realtiva al passaggio di parametri alle funzioni
- Regola del "no in-out", ovvero, un parametro di una funzione può essere solo di input o di output, mai entrambi
- Parametri di input sono tutti const

```
int idSurface::Split ( const idPlane &  plane,
                       const float      epsilon,
                       idSurface **     front,
                       idSurface **     back,
                       int *            frontOnPlaneEdges = NULL,
                       int *            backOnPlaneEdges = NULL
                     )                  const
```

COMMENTI SUL CODICE

- Sulle scelte di programmazione: const

- *I am a full const nazi nowadays, and I chide any programmer that doesn't const every variable and parameter that can be.*

COMMENTI SUL CODICE

- Sulle scelte di programmazione: commenti
- Ci sono dove servono: non troveremo parti di codice simile a questa
-
```
/* ***********************
*** restituisce il valore
************************/
int laMiaClasse::getValue() const {
    return value;     // restituisce il valore
}
```

COMMENTI SUL CODICE

- Sulle scelte di programmazione: commenti
- Ci sono dove servono: non troveremo parti di codice simile a questa
- ```
  /* **********************
  *** restituisce il valore
  ************************/
  int laMiaClasse::getValue() const {
      return value;      // restituisce il valore
  }
  ```
- Come è commentata la funzione Split che abbiamo visto prima?
- ```
  // splits the surface into a front and back surface, the surface
  itself stays unchanged
  // frontOnPlaneEdges and backOnPlaneEdges optionally store the
  indexes to the edges that lay on the split plane
  // returns a SIDE_?
  ```

COMMENTI SUL CODICE

- Sulle scelte di programmazione: spazi verticali e orizzontali
- Non si sprecano spazi verticali!
- Tutto è ben incolonnato!

COMM

- Sulle scelt
- Non si spr

```
            index[n++] = UpdateVertexIndex( vertexIndexNum[s], vertexRemap[s], vertexCopyIndex[s], v2 );
            index[n++] = edgeSplitVertex[e0];
            index[n++] = UpdateVertexIndex( vertexIndexNum[s], vertexRemap[s], vertexCopyIndex[s], v1 );
            indexNum[s] = n;
            break;
    }
    case 2: {       // second edge split
            s = sides[v1] & SIDE_BACK;
            n = indexNum[s];
            onPlaneEdges[s][numOnPlaneEdges[s]++] = n;
            index = indexPtr[s];
            index[n++] = edgeSplitVertex[e1];
            index[n++] = UpdateVertexIndex( vertexIndexNum[s], vertexRemap[s], vertexCopyIndex[s], v0 );
            index[n++] = UpdateVertexIndex( vertexIndexNum[s], vertexRemap[s], vertexCopyIndex[s], v1 );
            indexNum[s] = n;
            s ^= 1;
            n = indexNum[s];
            onPlaneEdges[s][numOnPlaneEdges[s]++] = n;
            index = indexPtr[s];
            index[n++] = UpdateVertexIndex( vertexIndexNum[s], vertexRemap[s], vertexCopyIndex[s], v0 );
            index[n++] = edgeSplitVertex[e1];
            index[n++] = UpdateVertexIndex( vertexIndexNum[s], vertexRemap[s], vertexCopyIndex[s], v2 );
            indexNum[s] = n;
            break;
    }
    case 3: {       // first and second edge split
            s = sides[v1] & SIDE_BACK;
            n = indexNum[s];
            onPlaneEdges[s][numOnPlaneEdges[s]++] = n;
            index = indexPtr[s];
            index[n++] = edgeSplitVertex[e1];
            index[n++] = edgeSplitVertex[e0];
            index[n++] = UpdateVertexIndex( vertexIndexNum[s], vertexRemap[s], vertexCopyIndex[s], v1 );
            indexNum[s] = n;
            s ^= 1;
            n = indexNum[s];
            onPlaneEdges[s][numOnPlaneEdges[s]++] = n;
            index = indexPtr[s];
            index[n++] = edgeSplitVertex[e0];
            index[n++] = edgeSplitVertex[e1];
            index[n++] = UpdateVertexIndex( vertexIndexNum[s], vertexRemap[s], vertexCopyIndex[s], v0 );
            index[n++] = edgeSplitVertex[e1];
            index[n++] = UpdateVertexIndex( vertexIndexNum[s], vertexRemap[s], vertexCopyIndex[s], v2 );
            index[n++] = UpdateVertexIndex( vertexIndexNum[s], vertexRemap[s], vertexCopyIndex[s], v0 );
            indexNum[s] = n;
            break;
    }
    case 4: {       // third edge split
```

COM

- Sulle sc
- Non si

```
class idJointMat {
public:

        void                    SetRotation( const idMat3 &m );
        idMat3                  GetRotation() const;
        void                    SetTranslation( const idVec3 &t );
        idVec3                  GetTranslation() const;

        idVec3                  operator*( const idVec3 &v ) const;
        idVec3                  operator*( const idVec4 &v ) const;

        idJointMat &    operator*=( const idJointMat &a );                                  // tran
        idJointMat &    operator/=( const idJointMat &a );                                  // untr

        bool                    Compare( const idJointMat &a ) const;                       // exac
        bool                    Compare( const idJointMat &a, const float epsilon ) const;  // compare with epsilon
        bool                    operator==(     const idJointMat &a ) const;                // exac
        bool                    operator!=(     const idJointMat &a ) const;                // exac

        void                    Identity();
        void                    Invert();

        void                    FromMat4( const idMat4 & m );

        idMat3                  ToMat3() const;
        idMat4                  ToMat4() const;
        idVec3                  ToVec3() const;
        const float *   ToFloatPtr() const { return mat; }
        float *                 ToFloatPtr() { return mat; }
        idJointQuat             ToJointQuat() const;
```

COMMENTI SUL CODICE

- Sulle scelte di programmazione: brackets { }
- Usate sempre, anche quando non obbligatorie
- Non vedremo codice confusionario come:
-
```
while (a)
    if (b > c)
        d = c;
    else if (c > d)
        e = f;
    else
        if (mm)
            a = 0;
        else
            n = 1;
```

COMMENTI SUL CODICE

- Sulle scelte di programmazione: STL
- Id non ha usato la STL; piuttosto, implementazione da zero di container, tipi composti ed algoritmi.
- Scarso uso delle template: programmazione strongly typed e uso dei generics solo quando serve

- *I mistrusted templates for many years, and still use them with restraint, but I eventually decided I liked strong typing more than I disliked weird code in headers. The debate on STL is still ongoing here at Id, and gets a little spirited. Back when Doom 3 was started, using STL was almost certainly not a good call, but reasonable arguments can be made for it today, even in games.*

COMMENTI SUL CODICE

- Sulle scelte di programmazione: get/set
- Funzioni membro get / set implementate solo quando servono veramente
- ```
  class LaMiaClasse {
  public:
      int getVar() const { return var; }
      void setVar( const int v ) { var = v; }
  private:
      int var;
  }
  ```
- Per incrementare la variabile di n?
- ```
  LaMiaClasse lmc;
  lmc.setVar( lmc.getVar() + n );
  ```

SUI TOOL

- Sui tool di sviluppo: static/dynamic code analyzer
- *The most important thing I have done as a programmer in recent years is to aggressively pursue static code analysis. Even more valuable than the hundreds of serious bugs I have prevented with it is the change in mindset about the way I view software reliability and code quality.*
- Code analyzer è un tool che esamina il codice, cercando possibili bug.
- Static Code Analysis prevede analisi del codice senza eseguirlo (source code o bytecode)
- Dynamic Code Analysis prevede analisi del codice durante l'esecuzione
- Che tipo di errori è possibile trovare?
- PVS-Studio test ☺ http://q.viva64.com/ <- non funziona più…

SUI TOOL

- Sui tool di sviluppo: static/dynamic code analyzer
- *NULL pointers are the biggest problem in C/C++, at least in our code. The dual use of a single value as both a flag and an address causes an incredible number of fatal issues. C++ references should be favored over pointers whenever possible; while a reference is "really" just a pointer, it has the implicit contract of being not-NULL. Perform NULL checks when pointers are turned into references, then you can ignore the issue thereafter.*
- *Printf format string errors were the second biggest issue in our codebase, heightened by the fact that passing an idStr instead of idStr::c_str() almost always results in a crash*
- *A lot of the serious reported errors are due to modifications of code long after it was written. […] Examined in isolation, this is a comment on code path complexity, but when you look back at the history, it is clear that it was more a failure to communicate preconditions clearly to the programmer modifying the code.*

SUI TOOL

- Sui tool di sviluppo: static/dynamic code analyzer
- Valgrind: esamina utilizzo di memoria e individual eventuali memory leak. Inoltre, controllo race condition in ambito multithread
- Microsoft analyze: compreso in Visual Studio, anche nella versione Community!!!
- Intel Inspector / Advisor / Trace Analizer / Vtune Amplifier: suite (a pagamento) per complete analisi online e offline e profilazione del codice

- *The takeaway action should be: If your version of Visual Studio has /analyze available, turn it on and give it a try. If I had to pick one tool, I would choose the Microsoft option. Everyone else working in Visual Studio, at least give the PVS-Studio demo a try. If you are developing commercial software, buying static analysis tools is money well spent.*

DOOM III

- Bibliografia:

- http://kotaku.com/5975610/the-exceptional-beauty-of-doom-3s-source-code
- ftp://ftp.idsoftware.com/idstuff/doom3/source/CodeStyleConventions.doc
- http://fabiensanglard.net/doom3/index.php
- http://fabiensanglard.net/doom3_bfg/index.php
- http://fabiensanglard.net/doom3_documentation/index.php
- http://fabiensanglard.net/doom3_documentation/DOOM-3-BFG-Technical-Note.pdf
- http://www.gamasutra.com/view/news/128836/InDepth_Static_Code_Analysis.php
- https://dev.visucore.com/doom3/doxygen/_surface_8h_source.html