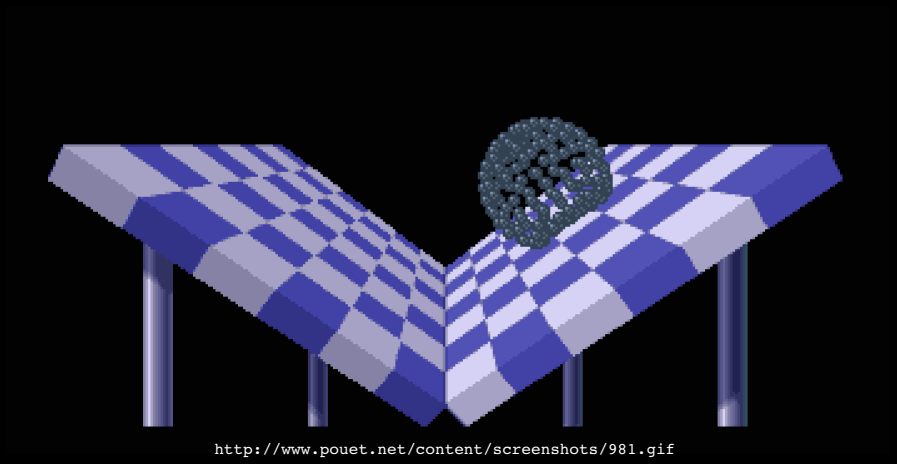
# Beyond the Limits: The Usage of C++ in the Demoscene

Eivind Liland, David Geier
C++ User-Group Meeting
C-Base, 20 May 2014

# What is the Demoscene?

- Broke out of the cracking scene in early 80's
- Exists almost as long as home computers!
- Used to be the coolest stuff you could show on computers! (Still is!)

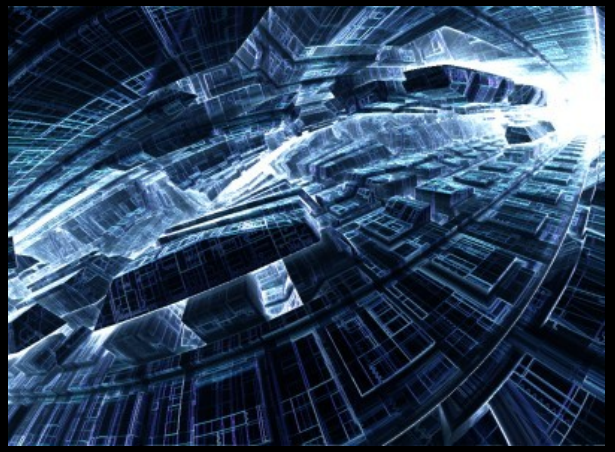


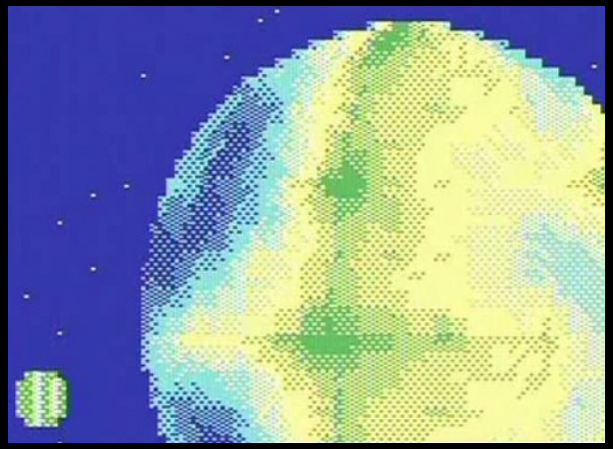
http://www.pouet.net/content/screenshots/981.gif

- What's special about the demoscene?
- Made for the love of it - without commercial interest
- Demos represent a rare combination of great technical and artistic accomplishments

# What is the Demoscene?

Demomaking is teamwork. The range of required skills is too much for one person to master. Demogroups ideally consist of people with complementing interests and skills.





Some coders prefer to write effects from start to finish, while others prefer to write tools and effects that can be tweaked by artists.

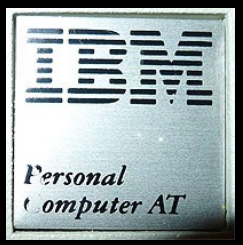
# Competitions

- Different platforms: PC, C64, Amiga, Atari, Console
- Size restriction: 256 bytes, 4k, 64k
  - Artificial limitation
  - Needs special programming skills
  - Creates a point of comparison especially on PC
  - Typically called “intros”

[1]

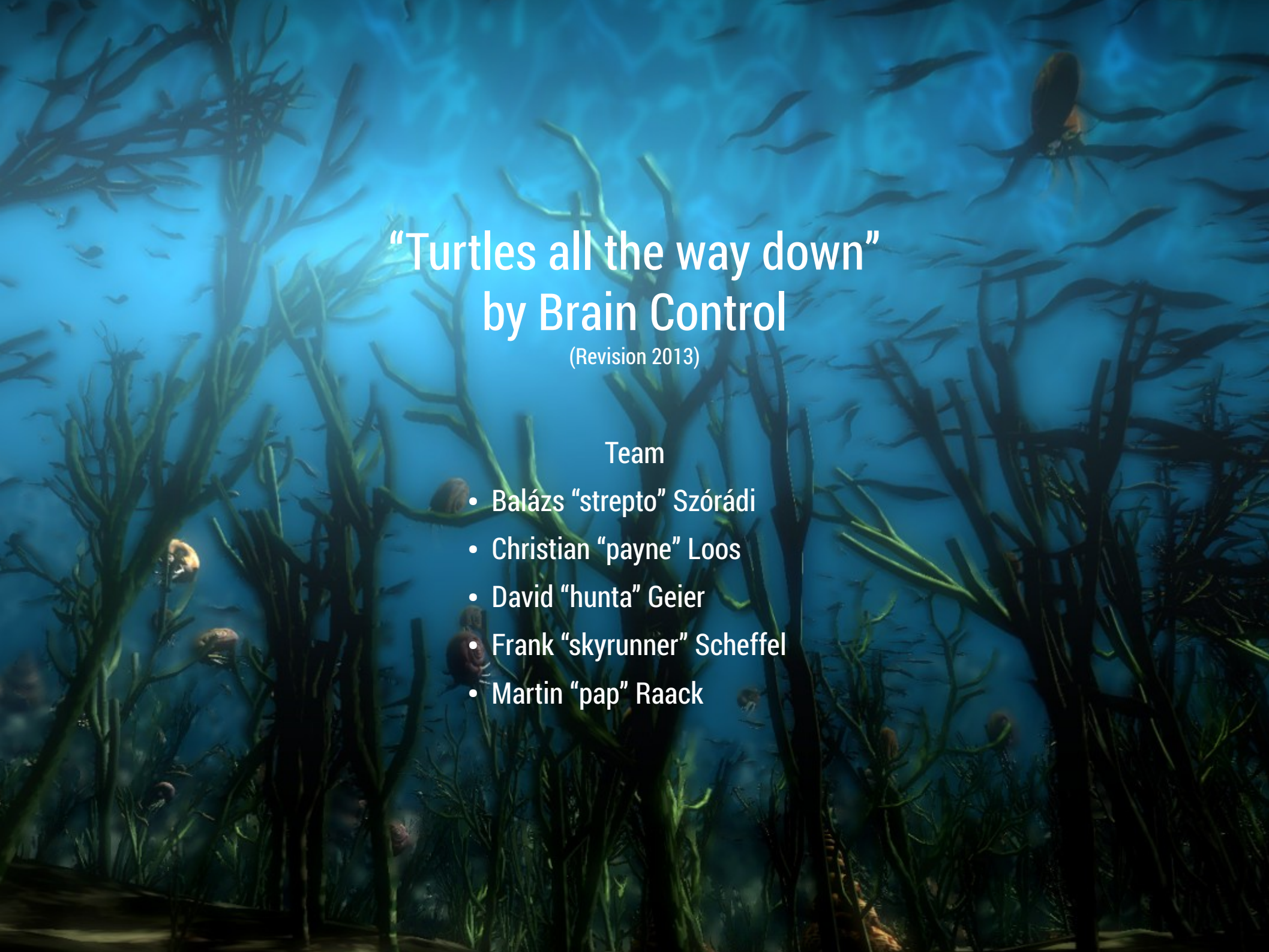
[2]

[3]



[4]

[1] http://logoblink.com/wp-content/uploads/2011/12/atari-logo.jpg
[2] http://upload.wikimedia.org/wikipedia/commons/4/48/C64_startup_animiert.gif
[3] http://fc02.deviantart.net/fs71/f/2012/029/2/d/amiga_logo___dark_blue_background_by_pixeloza-d4o0vdx.jpg
[4] http://www.computermuseumshop.com/computer/hardware/ibm_international_business_machines/ibm_pc_at_5170/ibm_pc_at_logo.jpg



# “Turtles all the way down” by Brain Control

(Revision 2013)

Team

- Balázs “strepto” Szórádi
- Christian “payne” Loos
- David “hunta” Geier
- Frank “skyrunner” Scheffel
- Martin “pap” Raack

# Some facts

- 4 minutes 56 seconds
- One single 65.536 bytes executable
- That's ~7.4 bytes per frame assuming 30 FPS
- Runs on a fresh Windows installation + drivers

- Sound track (wave data): 75.7 MiB
- 49 textures: 24.5 MiB (450 immediate textures: 112.5 MiB)
- Geometry (static index+vertex buffers): 13.4 + 28 = 41.4 MiB
- Total: 229.6 MiB: compression ratio of more than 3500:1

# How did we do it?

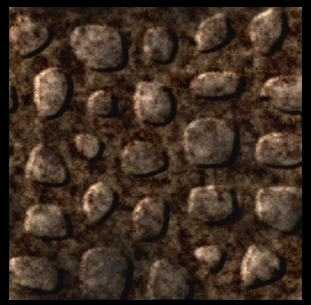
- Not a single JPG, PNG, MP3, 3DS, ... file is used
- Everything is generated when the intro starts
- Code, operations and parameters are stored
- Combine many simple operations to create complex stuff

```
    eOP_PAR_FLOAT(eWaveOp, radius, "Radius", 0.1f, eF32_MAX, 100.0f,
    eOP_PAR_FLOAT(eWaveOp, time, "Time", 0.0f, eF32_MAX, 0.0f,
    eOP_PAR_FLAGS(eWaveOp, affect, "Affect", "X-axis|Y-axis|Z-axis", 1 | 2 | 4,
    eOP_PAR_END)))))))
{
    _copyFirstInputMesh();

    const eBool affectX = eGetBit(affect, 0);
    const eBool affectY = eGetBit(affect, 1);
    const eBool affectZ = eGetBit(affect, 2);
    const eF32 sqrRadius = radius*radius;
```

+

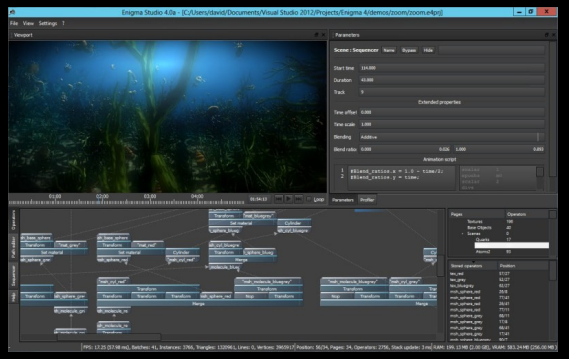

=



- Assumption
  - Code is smaller than data as long as it can be described algorithmically

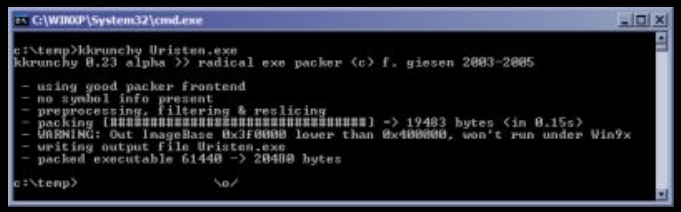
# Used tools

- Demo tool “Enigma Studio” for
  - Creating content
  - Sequencing
  - Doing post FX

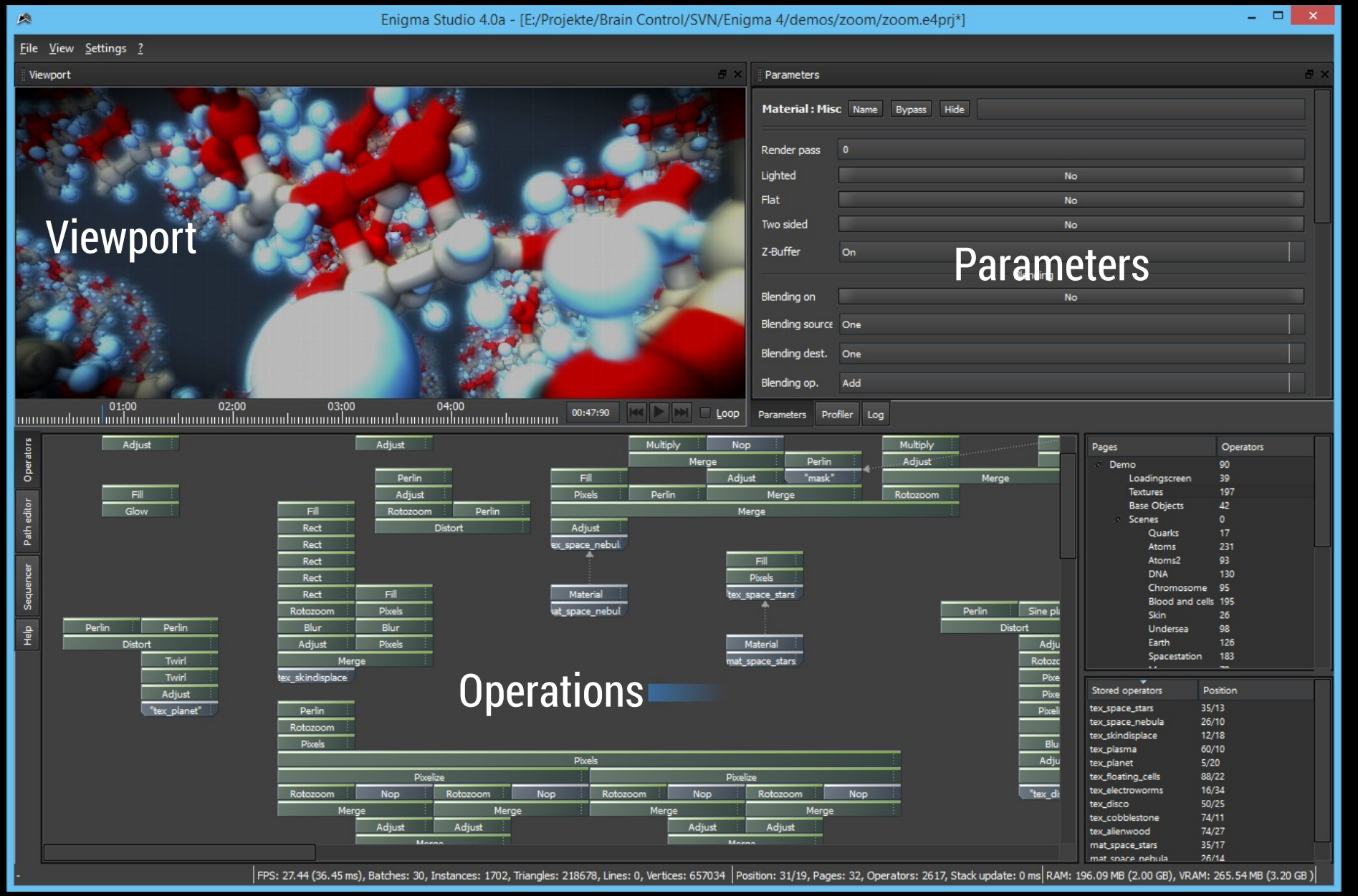


- Synthesizer “Tunefish”
  - Real-time synthesizer
  - VSTi Plugin used with Renoise

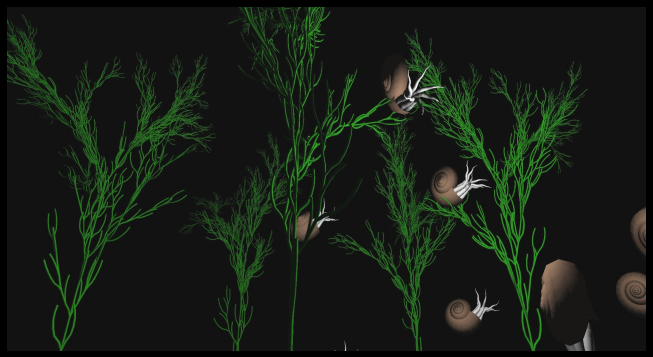
- Exe-packer “kkrunchy” (by ryg/Farbrausch)
  - Packs significantly better than e.g. UPX
  - We saved 22.5 KiB compared to using UPX!

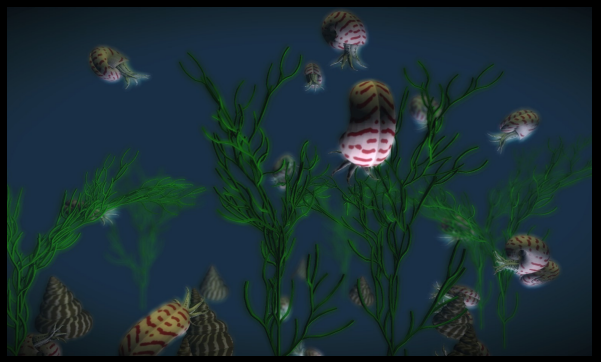


# Demo tool: Enigma Studio 5

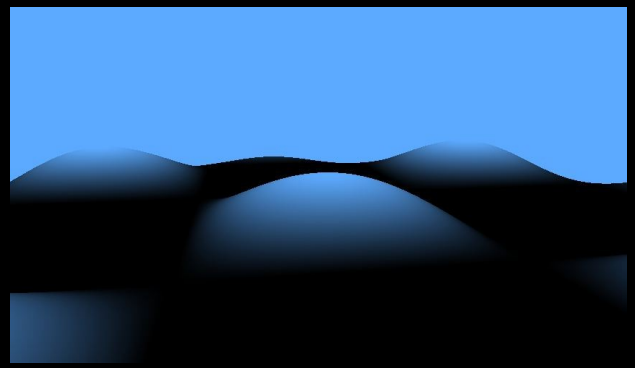


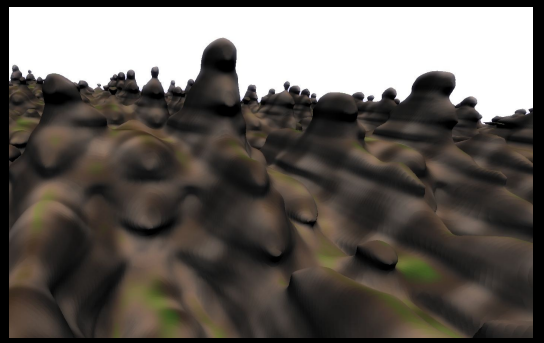
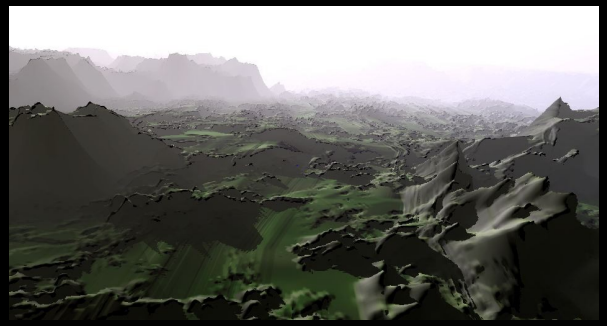
# Waterlife

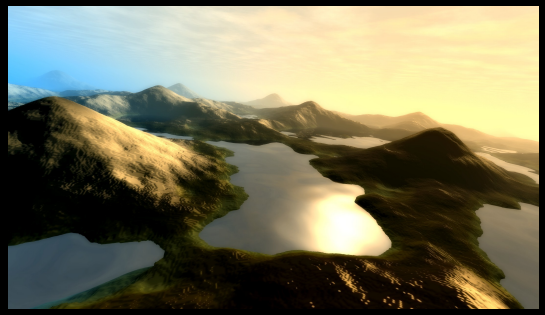
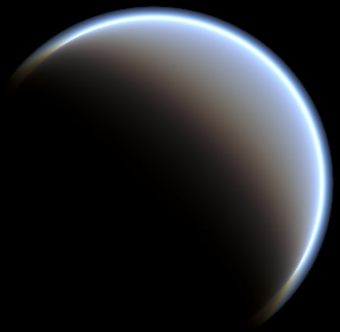




# Planet



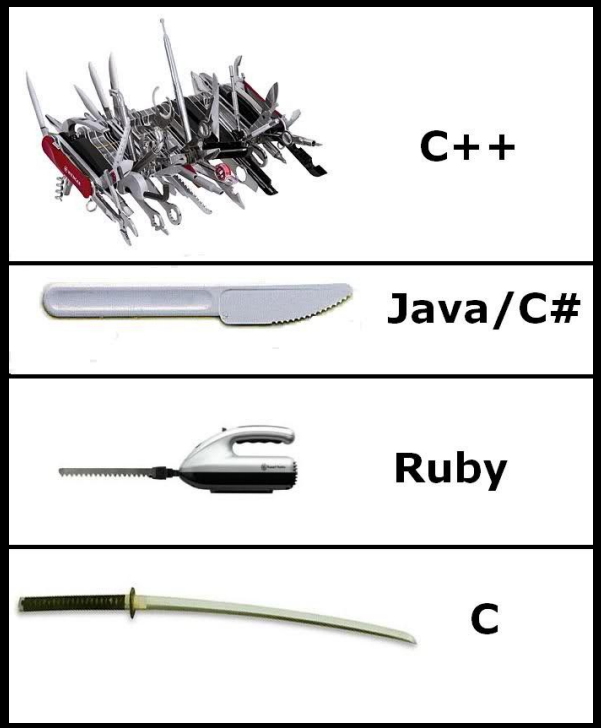



# Why and how did we use C++?

# C++? Perfect!

- Abstract as much as you can afford

  =>
- No templates (no STL)
- No standard C-Lib
- No inline functions
- No virtual functions
- No exceptions (`try` … `catch`)
- Take care about your constructors
- Write compression friendly code (know you packer)




http://murmolka.com/img/1/i42.tinypic.com/140x543.jpg

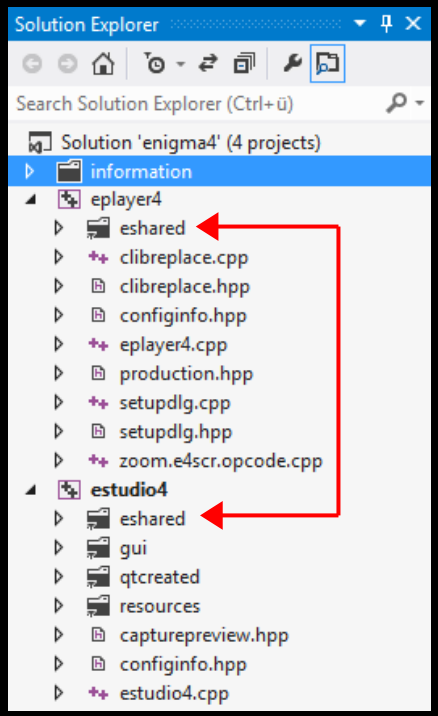
# Demo tool and demo player are one big code base

## Demo tool

- Modern C++11: STL, lambdas, ...
- User-interface based on Qt5

## Demo player

- Contains no UI
- As small as possible




Conditional compilation

```
eIOperator::~eIOperator()
{
    _clearParameters();
#ifdef eEDITOR
    m_memMgr.remove(this);
#endif
}
```

## Demo engine

- Code used by tool and player
- Maintainable, compact, fast

# Crafted our own C-Lib

- Linking against standard C-Lib not feasible
  - Dynamic linking => forbidden dependencies (msvcr120.dll)
  - Static linking => file size
- Mostly straightforward (1-5 LOC per function)
  - Examples: `strlen`, `memcpy`, `sin`, `new`, `delete`, …
  - Intrinsics are your friend
- A few tougher ones
  - Inserted by compiler: `_alloca_probe_16`, `_chkstk`, ...
  - Getting static/global initializers to work

# Virtual functions

- Runtime polymorphism induces size overhead
  - V-Table for each virtual class
  - Indexed dereference per virtual function call
  - Sometimes additional `this` pointer fixup
- For small classes with rarely called functions acceptable
- We initially used it for the render system (D3D/OGL)
  - Very bad idea
  - Interface consists of 63 functions
  - There are > 250 calls to these functions
  - Saved around 0.5 KiB (compressed)

# Templates: avoid code bloat

- We didn't want to live without them!
- Linker map can be used to check for code bloat
- Exclusively used for tiny functions (e.g. `abs`, `min`, ...)
- And our beloved `std::vector` replacement `eArray`

```
template<class T> class eArray
{
public:
    eFORCEINLINE eArray(eU32 size=0)
    {
        eArrayInit((eArray<void *> *)this, sizeof(T), size);
    }

    eFORCEINLINE void resize(eU32 size)
    {
        eArrayResize((eArray<void *> *)this, size);
    }

    ...

public:
    T *     m_data;
    eU32    m_size;
    eU32    m_capacity;
    eU32    m_typeSize;
};
```

```
void eArrayInit(eArray<void *> *a, eU32 typeSize, eU32 size);
void eArrayCopy(eArray<void *> *a, const eArray<void *> *ta);
void eArrayClear(eArray<void *> *a);
void eArrayFree(eArray<void *> *a);
void eArrayReserve(eArray<void *> *a, eU32 capacity);
void eArrayResize(eArray<void *> *a, eU32 size);
ePtr eArrayAppend(eArray<void *> *a);
void eArrayInsert(eArray<void *> *a, eU32 index, const ePtr data);
void eArrayRemoveAt(eArray<void *> *a, eU32 index);
void eArrayRemoveSwap(eArray<void *> *a, eU32 index);
eInt eArrayFind(const eArray<void *> *a, const void *data);
eBool eArrayEqual(const eArray<void *> *a0, const eArray<void *> *a1);
```

# Constructors

- Usually, the default constructor
  - zeros out `Vector`
  - sets `Matrix`, `Quaternion` to identity
- Can induce unnecessary code

Bad!

```
eVector3 a(1, 2, 3), b(4, 5, 6);
eVector3 res;
res = a+b;
```

Good!

```
eVector3 a(1, 2, 3), b(4, 5, 6);
eVector3 res(eNoInit());
res = a+b;
```

```
class eNoInit
{
};

class eVector3
{
public:
    eVector3() : x(0.0f), y(0.0f), z(0.0f)
    { }

    eVector3(const eNoInit &)
    { }

public:
    float x, y, z;
};
```

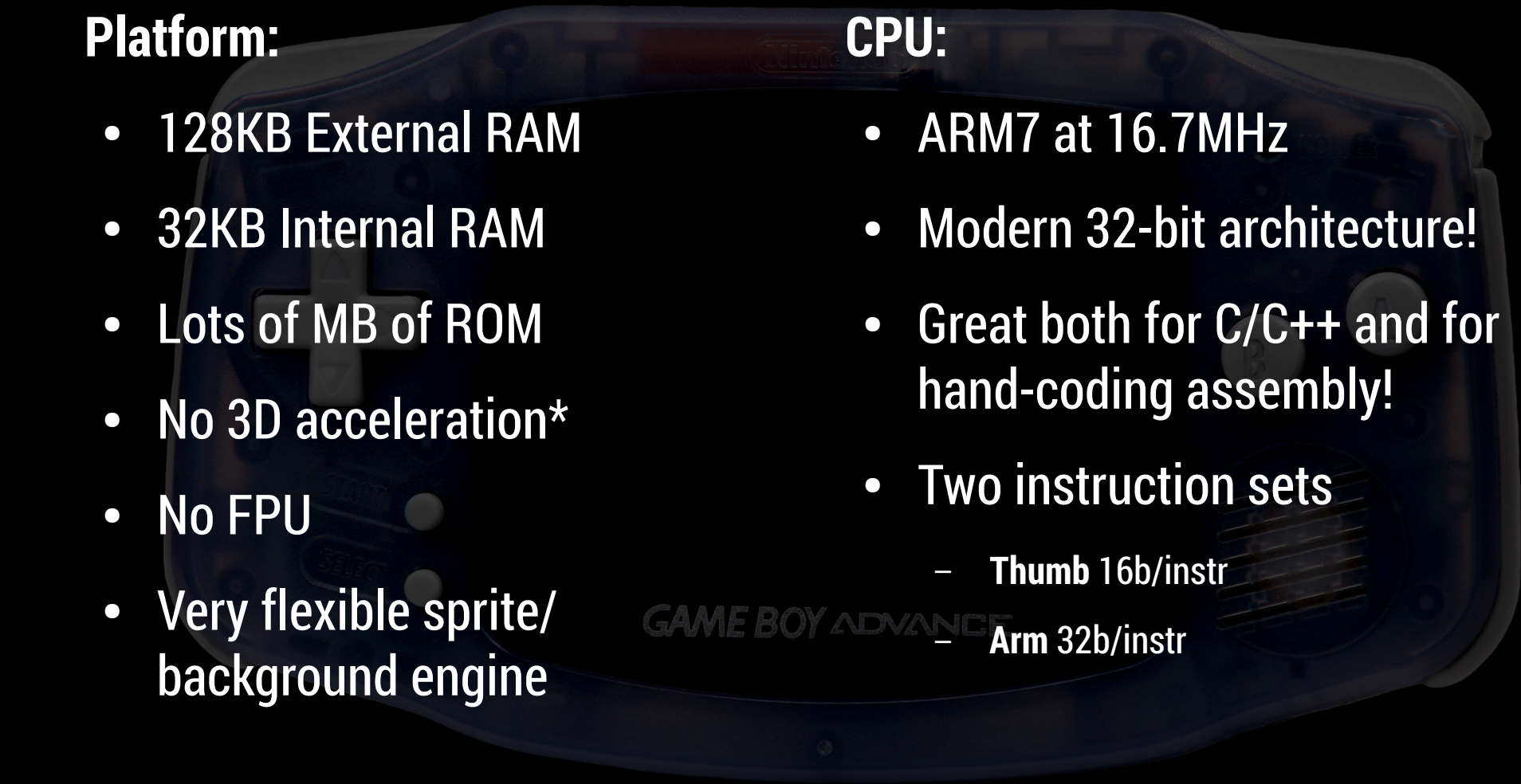
# Gameboy Advance (2001)


http://upload.wikimedia.org/wikipedia/commons/1/1d/Game-Boy-Advance-1stGen.png
http://www.mobygames.com/images/shots/l/97274-the-legend-of-zelda-the-minish-cap-game-boy-advance-screenshot.png

“Matt Current”
by


BREAKPOINT 2007
GAME BOY ADVANCE

# Gameboy Advance (2001)




**Platform:**

- 128KB External RAM
- 32KB Internal RAM
- Lots of MB of ROM
- No 3D acceleration*
- No FPU
- Very flexible sprite/ background engine

**CPU:**

- ARM7 at 16.7MHz
- Modern 32-bit architecture!
- Great both for C/C++ and for hand-coding assembly!
- Two instruction sets
  - **Thumb** 16b/instr
  - **Arm** 32b/instr

* Kusma and I were at the time working at a Norwegian start-up, developing a low-power GPU, later to become the ARM Mali.

# Shitfaced Clowns

**Group members:**

- Dixan – Music
- Lug00ber – Music
- Synteesi – Music
- Fred – 2D GFX
- Snarling – 3D GFX
- Kusma – Code, GFX
- Spooky – Code

**Why a GBA group?**

- Fun sprites,
  fun restrictions,
  yet somehow comfy
- Console demos were just becoming popular
- Only a couple of active demo groups
- We thought we could do better

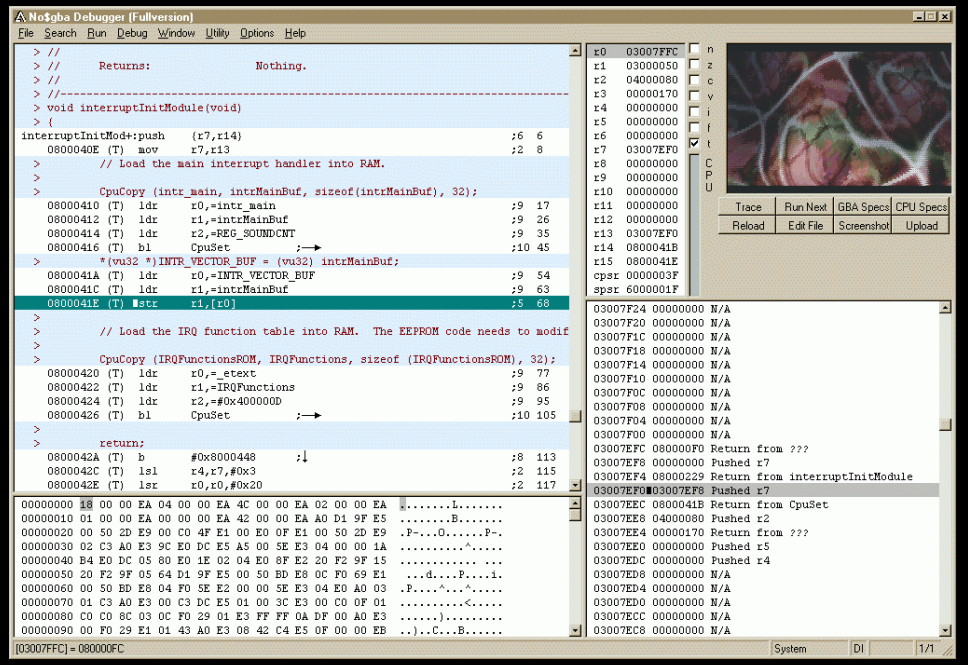
Q&A

Q: It's been 7 years. When's the next demo coming out?

A: Before you know it, bitch!

# GBA Toolchain

- Hardware fully reverse-engineered. Documentation and emulators available on-line.
- Straight-forward to control through memory-mapped registers (no drivers needed!)
- DevkitARM: Homebrew tool-chain based on GNU C++, just install and go!




http://nocash.emubase.de/gbapics.htm

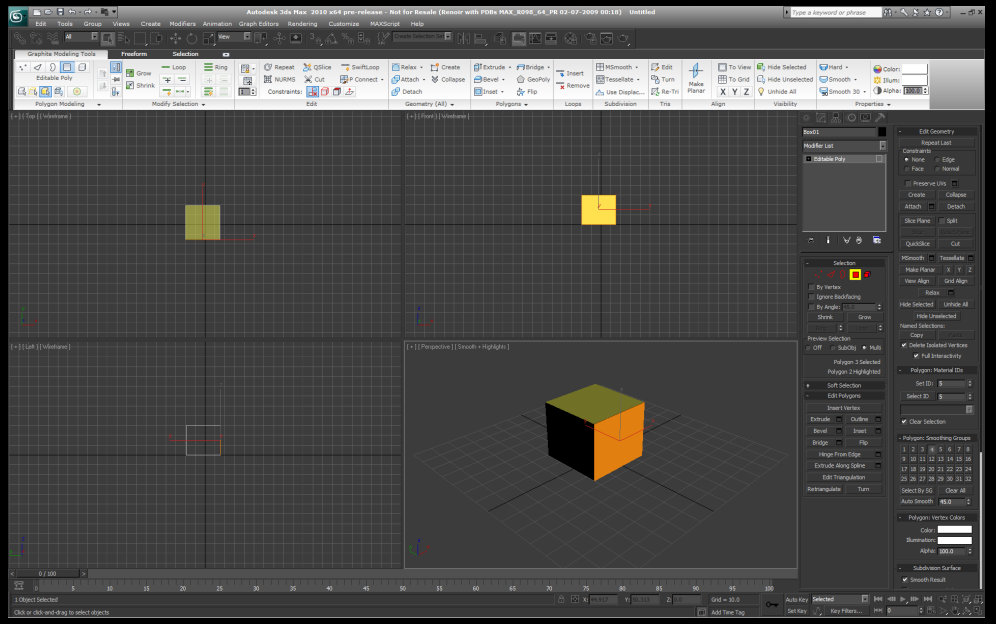
# C++? Perfect!

BUT:

- Avoid standard libraries and bloated runtime
- Avoid dynamic memory allocation
- Inner-loops:
  - Think about how you would write them in assembly
  - Count the cycles
  - Code it in C++ with the desired output in mind
  - Check what the compiler produced!

# Shitfaced GBA Tricks

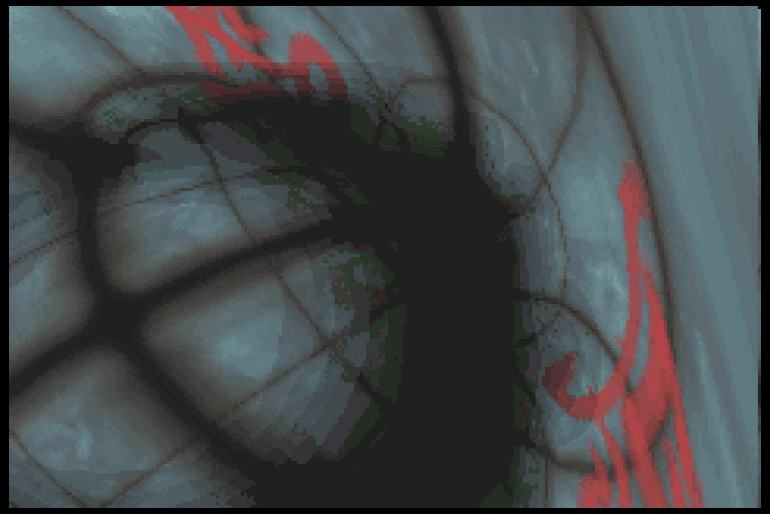
- We developed a largely automated tool-chain. In particular:
- Integration of our 3d-engine and 3D Studio Max made life much easier for the 3d graphics artist
- Due to automatic tool support, we were working in true-color despite the demo running mostly in paletted graphics modes.
- Miscellaneous tools such as an overlay-editor using sprites of different resolution



http://cdn.altrn.tv/s/16eda490-dec4-4cf7-a6dc-0a52a9fb4120_5_full.png

# Shitfaced GBA Tricks

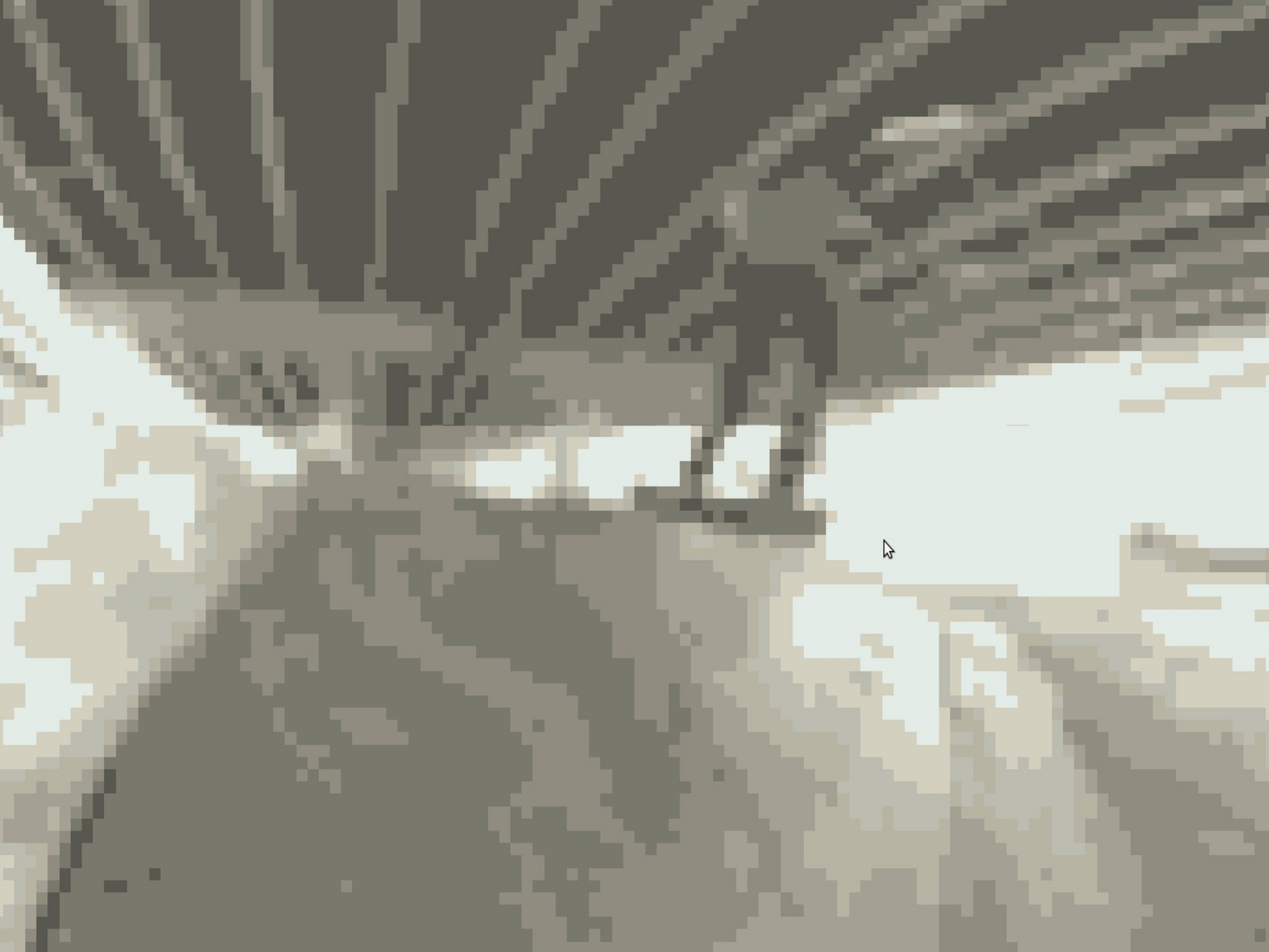
- Trick for paletted graphics modes: Palettes generated such that right-shift tints towards background color. This is *much* cheaper than “shade-tables”, etc.
- Instead of typical sample-based music: Real-time sound decompression with <10% CPU time (ADPCM 4 bits/sample).
- RAM is precious, so store resources such that they can be used directly from ROM.



# Effects breakdown

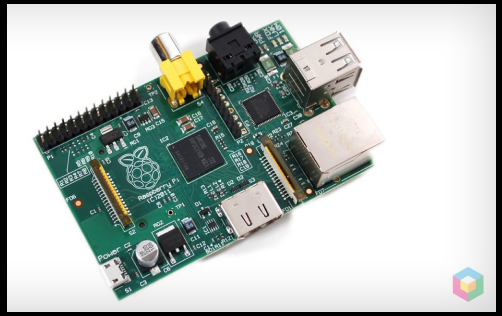

Stagg


Enjoy the Big Size
GAME BOY



# Does it have any practical use?

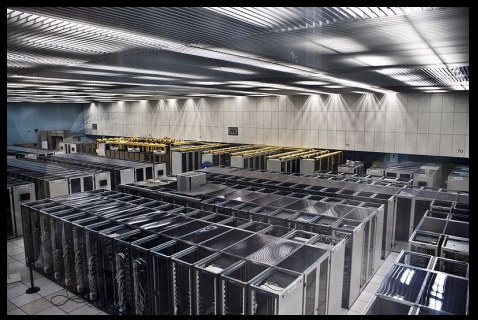
- Embedded platforms



- Mobile platforms

[1]

- Performance critical code



[2]

[1] http://www.mobileapplicationservices.com/images/page-img/cross-platform-development.png
[2] http://upload.wikimedia.org/wikipedia/commons/d/d7/CERN_Server_03.jpg

# Why do we do that?

3

4

# Nothing but fun!

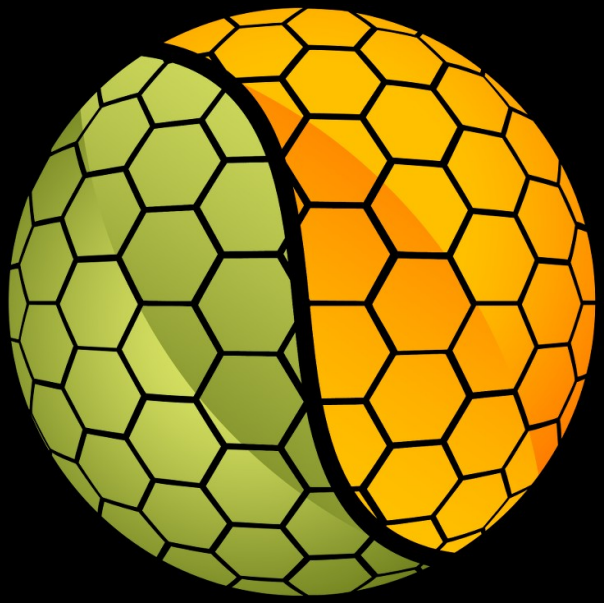
1


2

[1] http://www.slengpung.com/?id=25357&eventid=623
[2] http://www.slengpung.com/?id=25477&eventid=623
[3] http://www.slengpung.com/?id=25521&eventid=623
[4] http://www.slengpung.com/?id=25564&eventid=623

# Getting your feet wet

- You can find more demos on www.pouet.net
- Source code of our demos is publicly available
  - http://www.braincontrol.org/demos/enigma4_turtles_edition.zip
  - https://github.com/kusma/newton
- Go to a demo party: e.g. in Germany
  - Revision in Saarbrücken
  - Evoke in Cologne
  - the Ultimate Meeting near Darmstadt



# swarm64

(shameless ad)

www.swarm64.com

# We're hiring

**Send us a quick message at**

**jobs@swarm64.com**

Thank you for your attention!