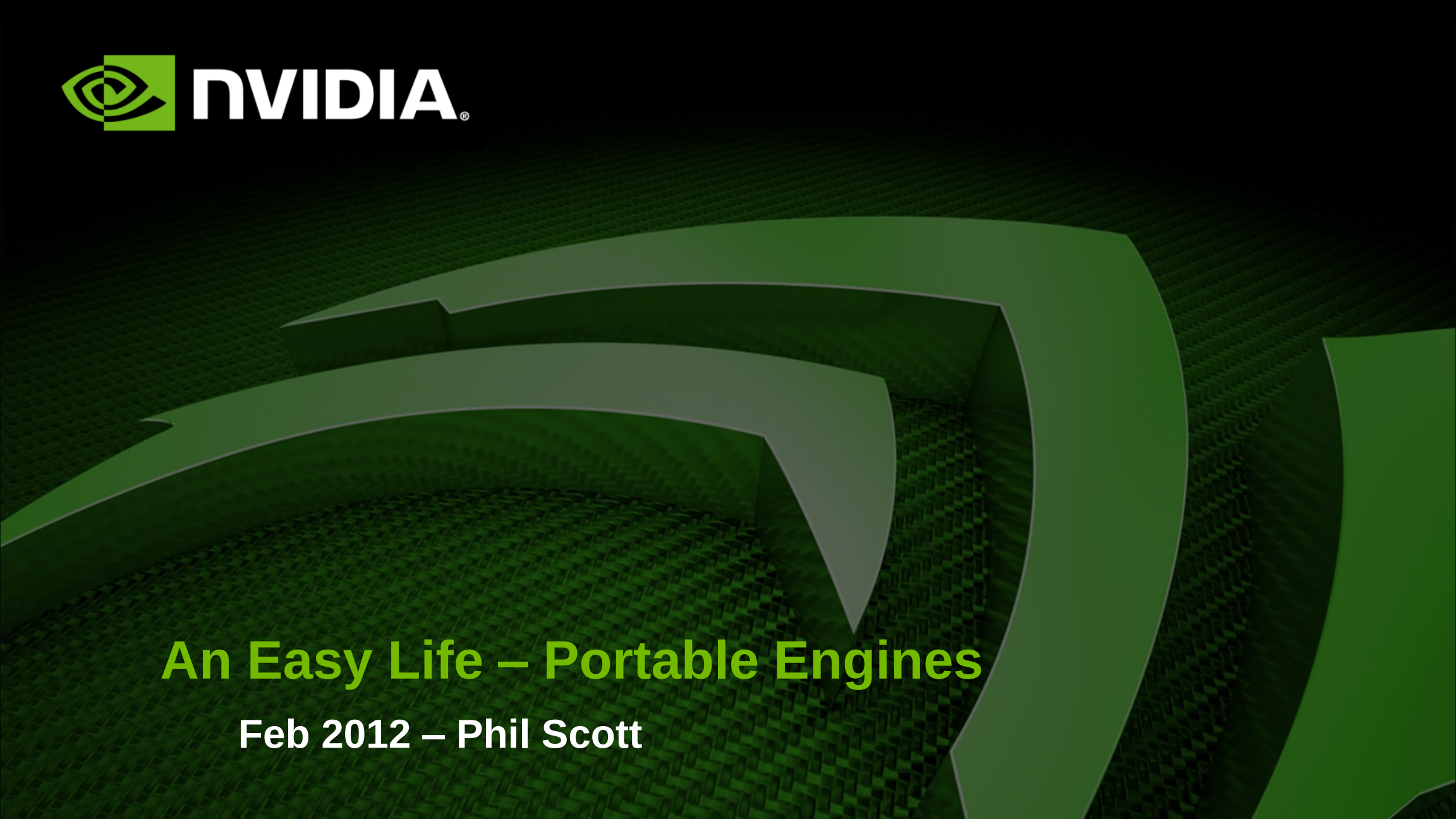


An Easy Life – Portable Engines

Feb 2012 – Phil Scott

A bit about me.

- Been doing this (and getting paid for it) since 1987
  - That makes me feel old.
    - A 8086 was 8mhz
    - A huge 500k of memory that was usable
    - And addressing was in PAGE mode with 2x 16bit registers
- Some reasonable successes
- Major contribution to 50 games+

Life was tough

- We lived in a cardboard box in “t’ middle ‘t road.”
  - Near enough...
- Life back in the stone-age
  - No debugger
  - Assembly times were 5 minutes
  - Tools were scant and you made your own mostly
  - C code was for ‘them other types’
  - But Pacman could be written in Cobol... There was hope yet!
  - SHOCK....

SHOCK...

- No GOOGLE
- No Wikipedia
- No Sourceforge
- No Flipcode
- No developer sites
- No ipad, no laptop...
- No online resources, community, samples or anything.

- Sinclair User was our only saviour

Machines of the time

- My first job
  - C64 – C16 – C+4 – BBC micro - 6502 derived – vastly different hardware
  - ZX Spectrum – Amstrad CPC - Z80 ☺ - No hardware just memory
  - ATARI ST - AMIGA - 68000 – hardware differences
  - PC - x86 no two machines alike, no hardware
- I was the PC guy... Aka the ‘12 bit’ department.

All of those machines

- What do you think we did?

All of those machines

- What do you think we did?
- Platform specialists all creating the same game (sorta)
  - Kevin Blake – BBC
  - Mick Hedley – C64
  - Dave Mann – Amiga
  - Bruce Nesbit / Steve Tall – ST
  - Me... PC
  - We didnt even share graphics!

Our Mantra

- Any game , can go on any platform.... It just may need a bit of attention.
  - This meant...
    - It would be different
    - It would be a total rewrite
    - Near nothing would be shared except verbally
    - One version could be great and another utterly dire (sadly happened a lot)


SUMMER OLYMPIAD
PROGRAM GARY GRAY
CHRIS ROBSON
MUSIC HAL BEBAN
ART PAUL DRUMMOND
TYNESOFT


PROGRAMMED
I. DAVISON
M. HEDLEY
GRAPHICS
M.LANDRETH
MUSIC
IAN.CRABTREE
HAL.BEBEN
TYNESOFT
COPYRIGHT 1988
ALL RIGHTS RESERVED
SUMMER OLYMPIAD

ALL

- All entirely drawn from scratch... Each time.

I was a noob

- Now...
  - Being on a platform that nobody thought was a gaming platform
  - I got NO service
    - You cant ship with programmer art.
- I needed a solution...

Pizza!

- Its a long story!!!
  - PC had 5 different adapter modes
    - CGA – 4 colour – fixed palette
    - EGA – 16 colour – fixed palette
    - VGA – 16 colour – RGB palette
    - MCGA – 256 colour – RGB palette
    - Hercules – 2 colour – Very hires
- Custom conversion software
  - Would take Amiga/ST source art, and retarget it.
  - Did a close enough job to get going...
    - Quite a few games shipped with Pizza-graphics

Then came the beeb

- BBC programmers saw that I could convert graphics
  - And liked it...
  - Pizza replaced 5dotEd (another long story)
  - BBC used low resolution PC assets
  - C64 then rotoscoped these with colour
  - Speccy had dithery patterns
- Our pipeline

WOOT... We developed a pipeline!

- Produce art on ST (Degas Elite & DP2)
  - Push to PC (via a converted floppy drive)
  - All dev systems were on PC for 8bit (PDS)

  - Convert to PC or 8bit target using PIZZA
  - Recolour for C64
  - Use directly on BBC micro

- Reduced art production time vastly / improved output of 5 artists to work on ST only (with touch ups on native)

That sounds awesome!!!

- What happened next?
  - Graphics review scores went up (highlight was 99% in Zzap 64) using converted graphics!
  - You should be minted!!!

Except it came too late

- Tynesoft went bankrupt
- Oh well...

Lesson Learned

- Next company...
  - Please move on!

Enter UBER-ED

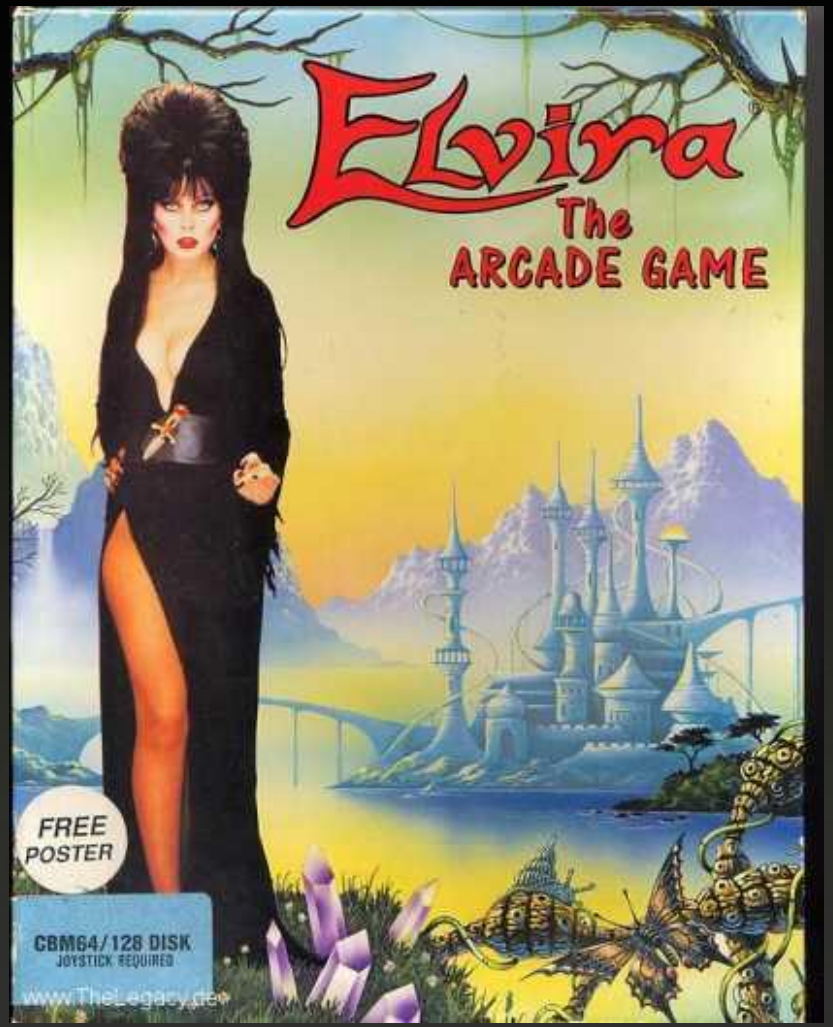
- New company was made up 90% of old team.
- I'd been promoted to lead the games due to Pizza at old place
- We needed a new platform for our games
- UBER-ED
  - Modest but fair ☺

First iteration

- On Atari ST..
  - Was buggy, and crashy
  - Originally art was made on ST, so felt like the natural home
  - Saved to floppy only
- Rewritten on PC
  - Much more stable
  - HDD storage
  - Art moved to PC... Life was good ☺
- It just edited level backdrops

Then the dawning of the future

- 4 platforms
  - Huge game content
  - 5 months
  - OH HELL!

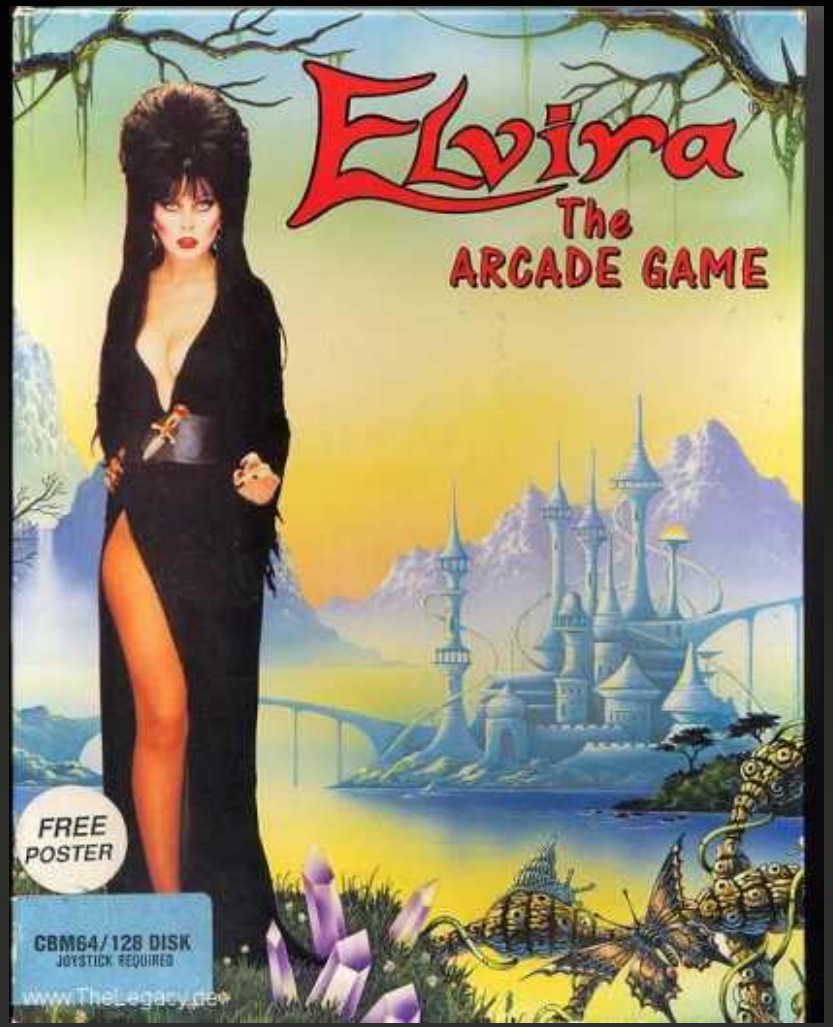


Then the dawning of the future

- The game content
  - Level structure
  - Puzzles
  - Attack waves
  - General game flow

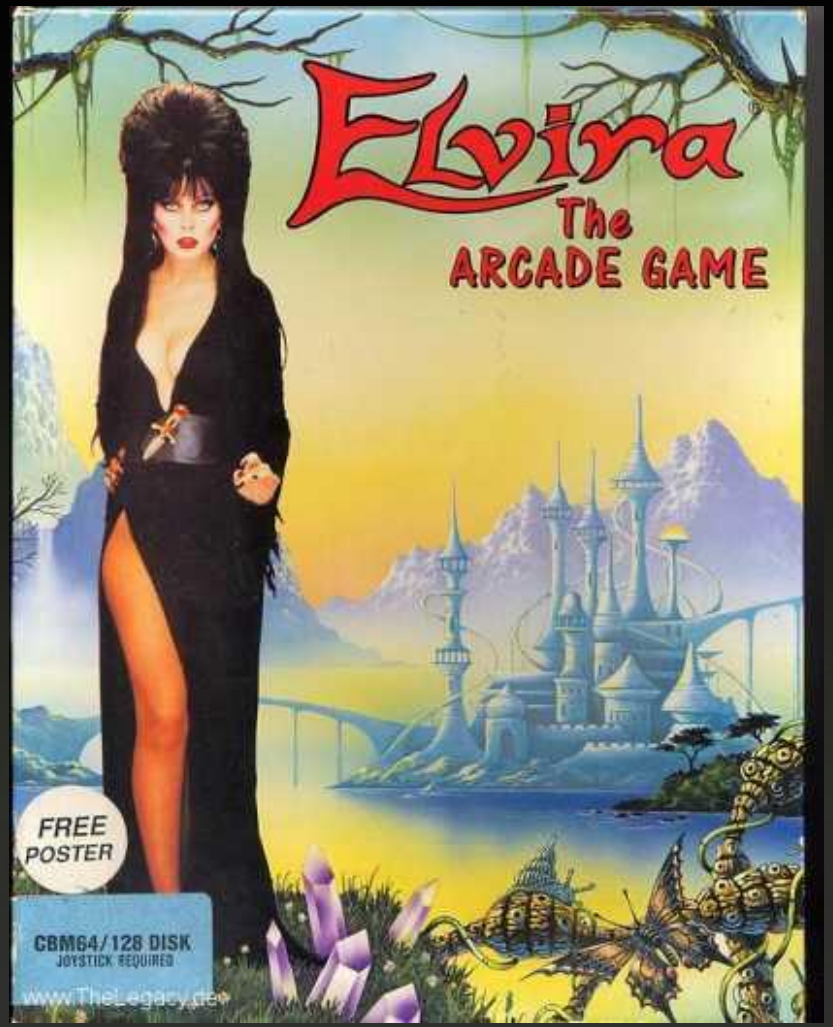
  - ALL created with a VM
    - VM functionality was clunky
    - But it worked

  - PC lead, ST/AMIGA , C64 took VM code
    - Ported by porting the VM
    - Lean and efficient p-code



Then the dawning of the future

- Three more games followed in a year.
- We had accidentally made an ENGINE...
  - We called it IVOR
  - It was DATA driven game mechanics
  - Maybe more could be?
- Rudimentary VM increased our output
- Game code still needed adding though for ‘gameplay’ features and new things to do.



Our VM

- P-Code... A bit like z80.
  - Opcode followed by data
  - Unique opcodes for different subfunctions
  - Why it worked so fast.
    - At tool save time all data was verified...
    - Load data, and generate offset (in place) to interpreted function
    - Grab return address from stack, as the data was sat directly after the call function in memory.
    - Modify stack to adjust return address to next p-code function.

    - NOTHING at runtime was compiled by runtime... Ever...

New engines

- Still in assembly code... Hanging onto that security blanket.
- DOS4GW
  - Dos extender that shipped with Watcom C
  - Doom used it.
    - Changed the world forever in PC game programming
- I guess I need to learn C ... Beuller?

Around this time (1996)

- PSX (playstation 1)
  - All C libraries
  - All samples were C code
  - Enough hardware to hide the shoddy slow C code
- Previously
  - At least 80% of your time was spend rendering / updating the screen with ASM calls... The game was tiny in code
  - On machines with HW (sprites, scrolling), most of the time was spent battling the poor slow processor... C64 had SUB 1mhz

Armed with a C compiler.

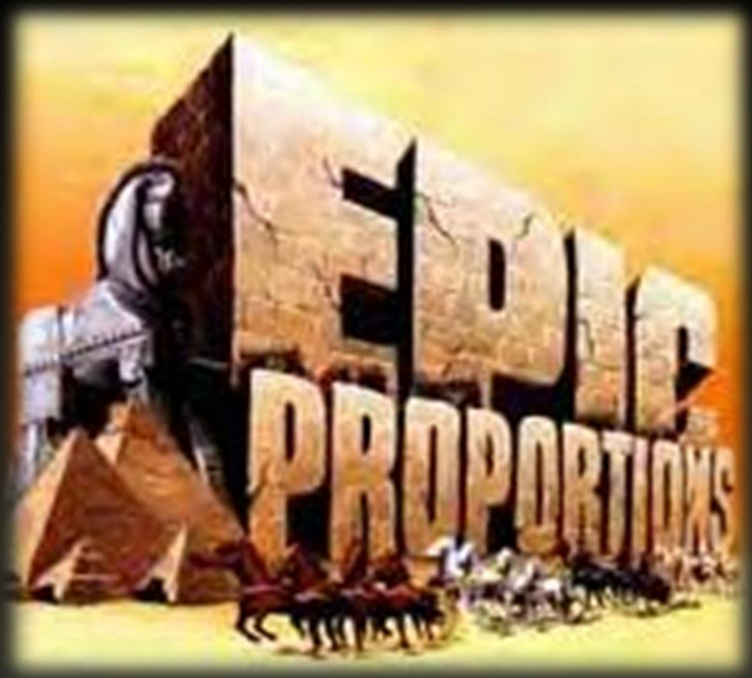
- Set about making a new game (for a new company)
  - Found out quickly a few lessons
- PSX had hardware... PC didn't.
  - I spent a lot of time making rendering libraries that matched PSX
- The core mechanic was there...
- Game was constructed in Editor tools on PC.
- Shared art...
- Shared C code (50%)
- Source level debugging... Nope!

Any questions so far?

- I'll take a breath...

Making of an epic.

- How does a small team make a world class game in 15 months?
  - Core staff was 8.
- No engine, start from scratch...
  - In the space of a year... 3D cards had become way better than we could achieve in software.
- Radical rethink.
  - Time pressure to ship something...



What do we have?

REUSE is good... Be shameless about it. If it ain't broken...

- PSX art tools...
  - Modelling was done in 3DSMax
  - Textured using PSX art tools
  - TM-ED ... A custom tool I'd created for PSX guys 2 years earlier.
- Known workflows if fit for task, are a real home run.

Segregation of work

- I love small teams!
- Easy to manage, and pull together...

- Our division or work was as follows
  - Phil – Anything hardware, Front end,player related, ‘art programming’
  - Kev – Editor, scripting, AI

  - Mick – Working on trying to make our stuff work on PSX

Set lofty targets... Be ambitious, not ambiguous

- Our CEO – Paul Finnegan – "*Shoot for the stars lads*"
  - Err ... Thanks Paul.

- At this stage we had art pipeline, and an idea of work.
  - Whilst I built a renderer
  - Kev would build the editor so we could pull stuff together.
  - Art guys would make a 'look and feel'

  - ...go... 2 months of 3 for the prototype to be up and running.

2 months in

- We could render some stuff.
  - Models
  - Heightmap
  - Basic animation (wobbly stickman)
  - Lighting of sorts
  - Camera logic

  - Crucially... (slight of hand)
    - Score / lives / bullet count
    - Player controls (basic)
    - 2 players on one screen (coop)

By the way guys...

- The approval meeting will be pulled in 2 weeks...
- We had 2 weeks to make stuff work...
  - Time for the pub!
  - Plan put together as to what we would show.
    - Studio head wasn't convinced
- And one late night... When the shoemaker went to bed.
  - The game making elves came...

The all-nighter.

- By elves...
  - 3 of us stopped late one Thursday night (approval meeting was Monday the following week)
  - With a plan to ignore the studio head, and have belief in our own way.
    - Added some explosion code / chain reaction code ... And sound effects. (critically)
    - We made a few monsters move
    - And listened to Eye of the Tiger 55x....

Following day

- *‘Wow’... I knew I was right all along – Studio Head*

- **It looked great.**
  - **The editor had been our savior because we could iterate quickly and test.**

  - **Crucially...**
    - **The game was running INSIDE the editor... Kev pulled a masterstroke.**
    - **We could iterate on content at the drop of a hat.**

    - **This set the tone for the rest of development from here out.**

The big day

- Of course we passed approval ☺
- Now the hard work began!
  - The race was on...

5 months of work... Next target.

- 4 playable (fully) vertical slice levels
- Weapons
- Multiplayer
- Rendering niceness
- AI

- Basically a full game.

The evolution.

- The editor became the tool that EVERYBODY used.
  - Except me.
- The “game” was “entirely” data driven.
  - I would add new features, and they would be wired up to editor functions.
  - Visual scripting...
    - Like sinclair basic. You couldn’t make a typo.
  - Used the same very fast execution logic we previously had used.
- Management thought we were nuts!

5 months pass

- We have a 4 month vertical (which became an OEM version)
- We did a full QA pass at that point.
  - Again, modifying script ‘live’ was such a good thing.
- The game was porting nicely to PSX
  - No rewriting of game code... Only systems... Now we’re cooking!

The run in...

- We ended up shipping a full game 6 months afterwards
- We sold in excess of 3 million copies
- Ported to 14 languages
- On 4 platforms
- We finished with 10 staff members.

- With 4 weeks to go we had ZERO ‘A’ bugs.
  - 393 script bugs... No coder needed (art guys did the scripts)
  - Gameplay tuning... And more tuning.
  - We added the front end at this stage.

Code level...

- How did we do it.

Old adage

- *There is no substitute for hard work...*

Code level...

- How did we do it.

***NEW** adage*

- *There is no substitute for **PLANNING** hard work...*

Firstly... planning

- Learn to ask questions
  - Employers like this... (except ‘can I have more money?’)
- Don’t take a schedule as a bible / talmut
  - Its there to help, not be a rod to hit you with
  - If its unrealistic... Saying at the start, is better than the end.
- Break down the tasks uniformly

Planning

- Understand ownership...
  - Take what you own and make it excellent.
  - Make it easy for them to interact with
  - Think of it in reverse...
- Establish the ground rules for
  - Naming... Hungarian or not?
  - Code style... What is verboten?
  - File organisation...
  - Don’t invent acronyms and jargon if established verbage works

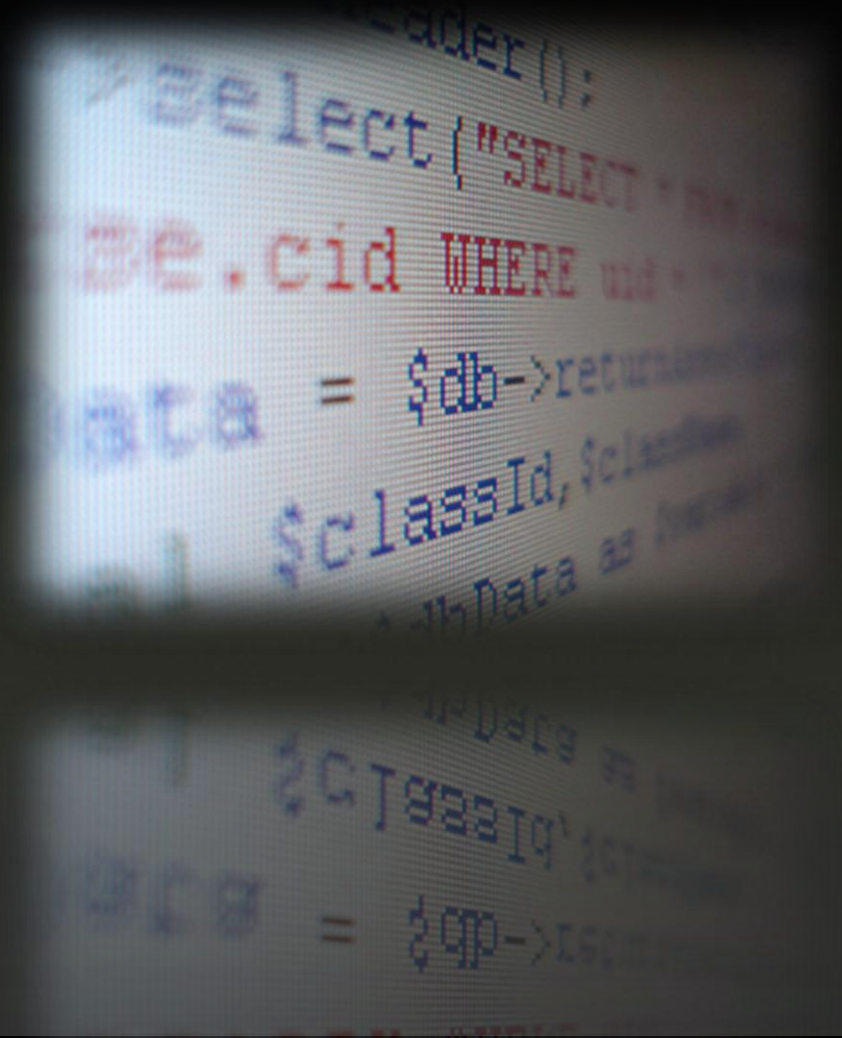
Simple code = good code.

- Basic rules
  - Fancy code = slow code (mostly)
  - No substitute for slight of hand

  Our engine

  - Class::Init()
  - Class::Process()
  - Class::Draw()
  - Class::Destroy()

  - THATS IT!



How deep?

- Derive your classes well
  - You can run anything in the Process function
  - Essentially its just a state machine anyway
  - A well derived base class gives you tons of flexibility
    - Let the data do the work...

Keep it simple

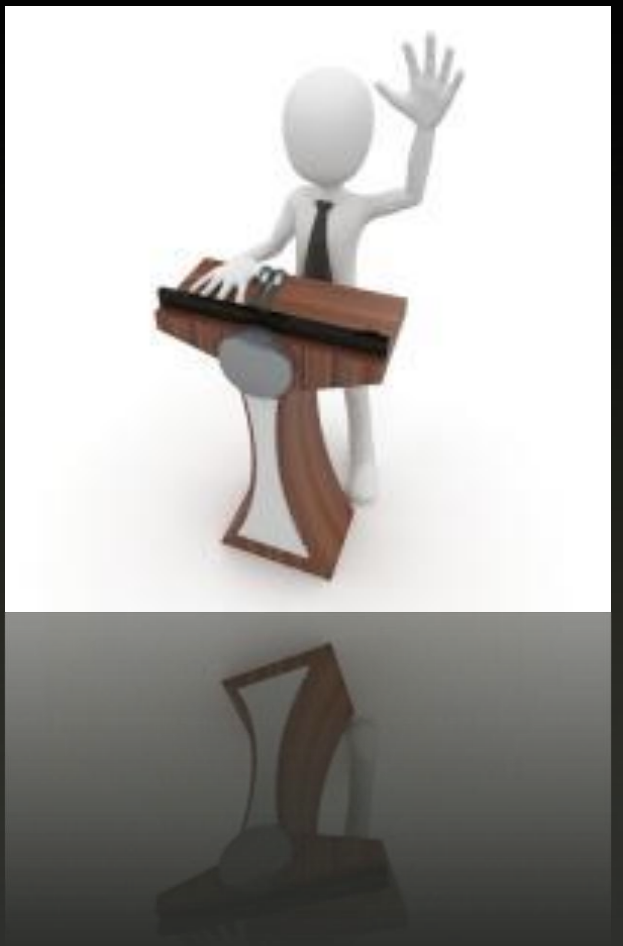
- Eg. Particles were entities within a global Particle system class.

Don’t be fooled by fancy indirection

- Stay as shallow as possible... Each pointer indirection is cycles lost.

The religious argument

- STL looks all fancy
  - Does loads of stuff for you.
  - Does loads of stuff behind your back that you don't know.
  - I'm not saying dont use it.
    - Know what it does first (look at the ASM code)
- Virtual functions ARE ALSO EVIL.
  - Topic of lively debate.
  - No... They arent...
    - They're dangerous, but useful... Like a chainsaw!
- I dislike templates except for INLINE code.
  - Math functions etc.
  - Small chunks of code, easy to debug.

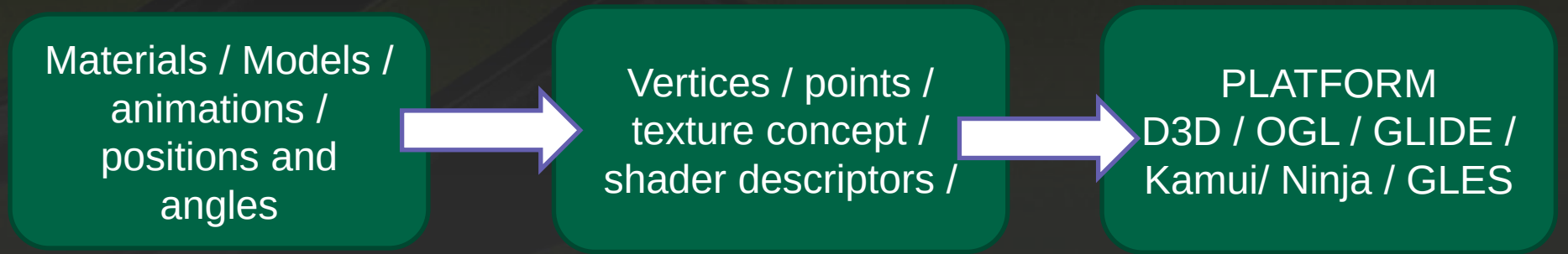


Main loop memory allocation

- A source of much mirth amongst the more experienced ones of us.
  - AS MUCH AS ITS TEMPTING ***DONT DO IT***..
  - YOU WILL FEEL PAIN
    - PAIN LEEDS TO SUFFERING...
    - You know the rest
  - Stable memory behaviour makes code portability workable.
    - Stable memory behaviour gives determinism
    - Use memory pools, that are pre-allocated and find slots (round robin or similar)

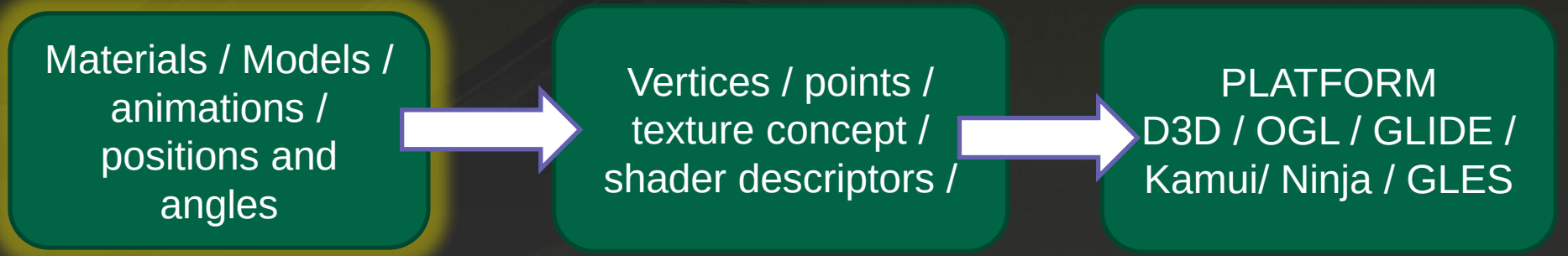
Learn to be abstract – this is true of all our code

- Initial renderer was written in D3D.
- Game code / Script code has to have NO understanding of this.
- Use abstracted terms... Be descriptive... Think ‘artist’ friendly



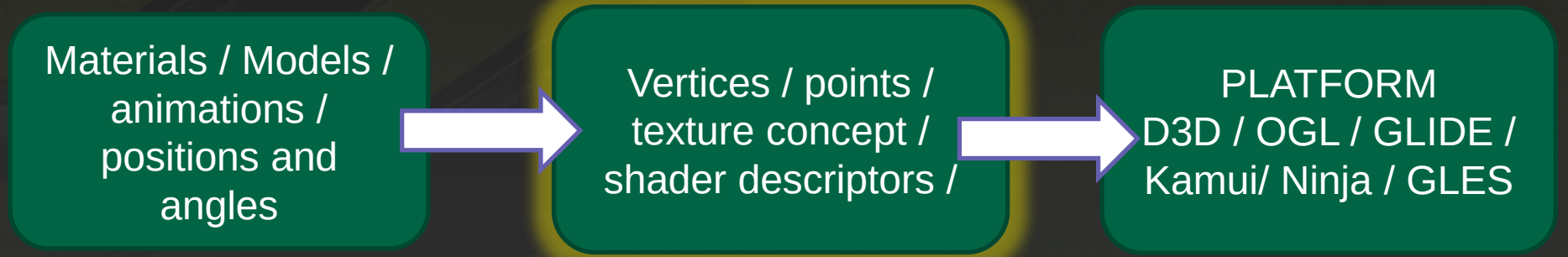
Learn to be abstract

- **Game code level**
  - **Abstraction is critical to success.**
  - **Don’t worry about the cycles... Really...**
  - **Nothing should describe the platform... In code.**
  - **Provide the engine with ‘information’**
  - **‘I want it to look like ‘this’ ‘**



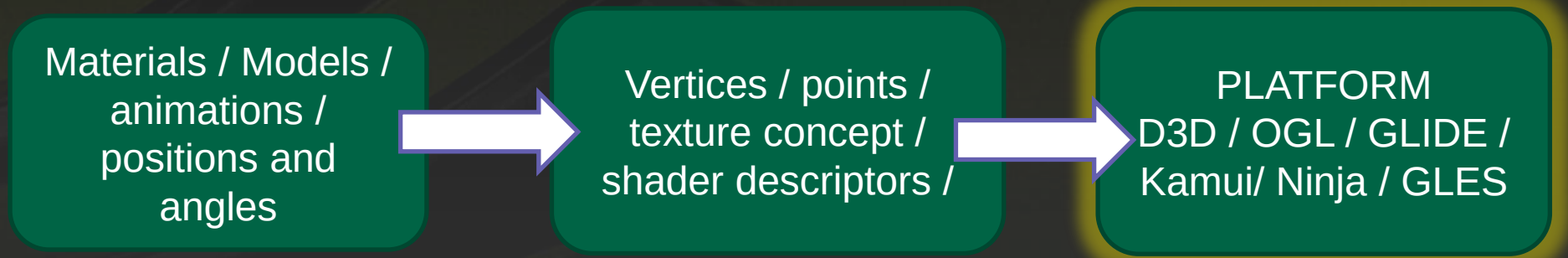
Learn to be abstract

- Critical to get this right. (this is ENGINE code)
  - Its ok to use terms such as ‘ONEONEALPHA’
  - Textures should be referred to as a class, not an object
  - Vertices are still sort of abstract... They’re just data.
  - Its not ok to have anything rendering API in here.



Learn to be abstract

- Gloves off.. Go for the metal
- All the work of your nicely abstracted code comes into here..
- Interpret and run with it.
  - Not all hardware supports all modes. Coping mechanism is here
- A well written renderer should port EASILY.
  - NOTE - I didn't say performantly



How well does this work

- Dreamcast was a new platform at this time
  - We had 1 guy, not in our studio who knew ‘a bit’
  - He ported our game ENTIRELY in 6 weeks (and through QA)
  - We hit JP DC launch and sold 250k copies that day!

Where does this lead us too today.

- Engines
  - UNITY, UDK, UE3.... All work this way
  - We didn’t invent it, but we must have been close.
  - KISS!
    - Keep It Simple Stupid.

What happens when...

- You take a 12 year old D3D game (well written)
  - State of the art in its day
- And port it to OpenGL in your spare time.

- Let me show you... Its far easier.

Questions

- Ask away!