# Reaching for the Limits of PS2 Performance How Far Have We Got?

SCEE Technology Group

SONY

COMPUTER ENTERTAINMENT

# Contents

- **Introduction**
  - Who are we ?
  - What is the Performance Analyser ?
- **Looking at the scans**
  - How do we read the scans ?
  - Live action
- **Statistics**
  - Average numbers
  - What did we learn ?
- **Conclusion**
  - The next move

# Introduction

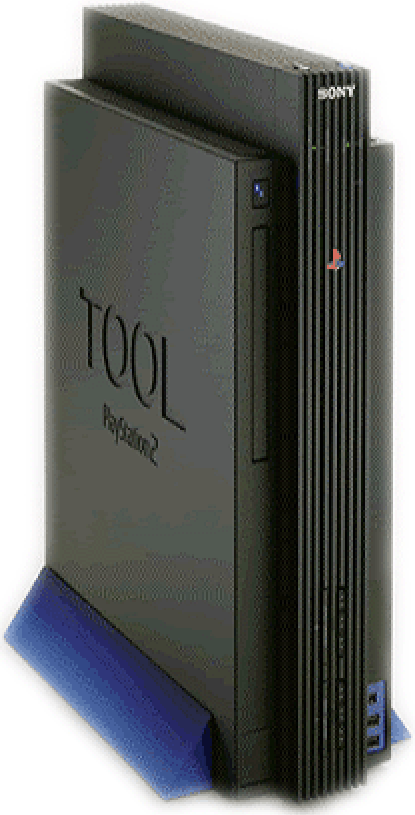
- **Who are we ?**
  - SCEE Technology Group
  - Based in London, UK
- **Who am I ?**
  - Lionel Lemarié
  - Developer Support Team

# Performance Analyser
# The Hardware and
# The Software

# What is the Performance Analyser ?

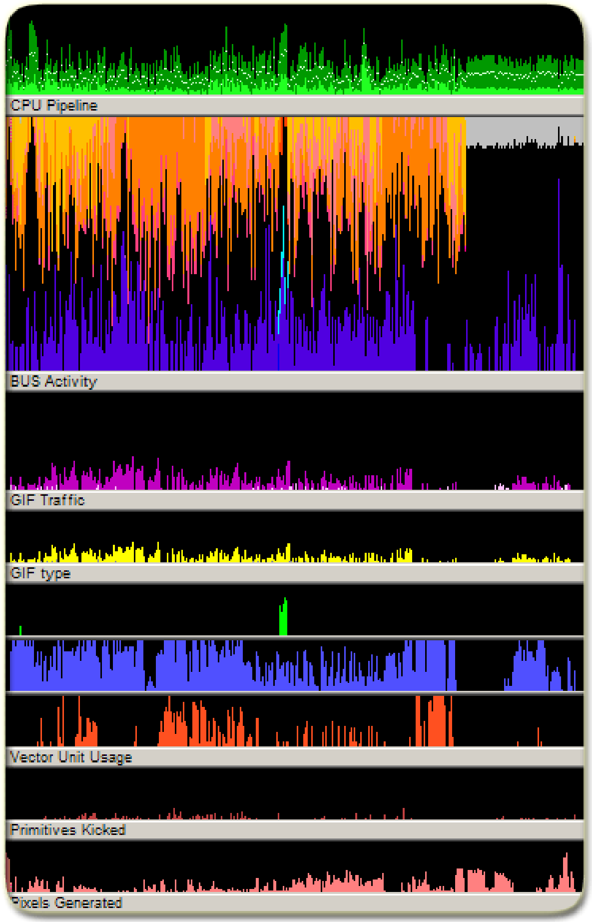
- **DTL-T15000**
  - Like a devkit but even better
  - Captures several frames of data
    - Over 100 signals
    - Cycle accurate

- **PA Software**
  - WinPACon
  - AProbe
  - GIF Packet Viewer



# How Do We Read the Graphs ?

- CPU cycles
- Bus occupation
- GIF traffic
- VU activity
- Primitives kicked
- Pixels output
- And many, many more…



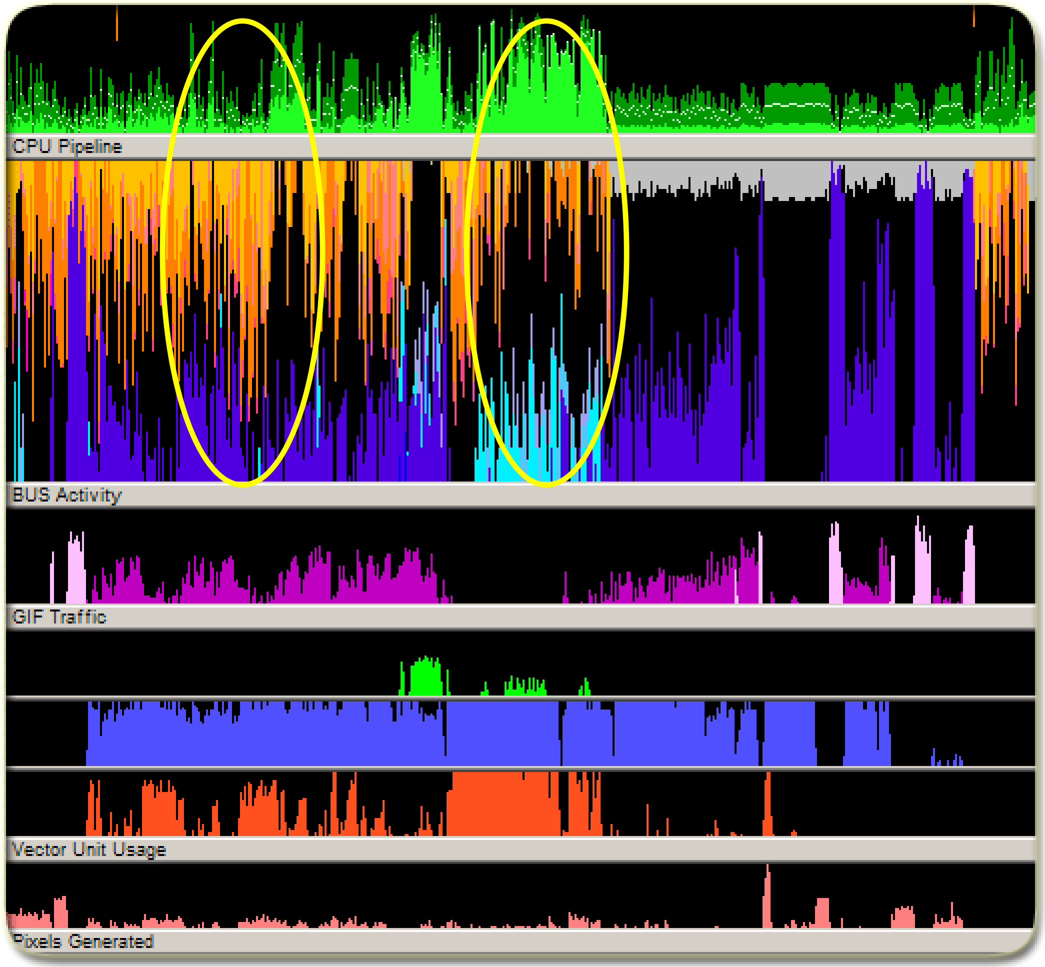
# Plug-in API

- **The GIF Packet Viewer can be used to visualise the scene**
  - The scene is fully 3D
  - Helps finding polygons guilty of not being GS friendly
  - Offers different drawing modes

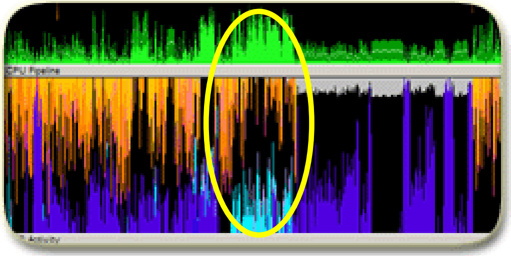
- **The SDK will be made available**
  - Have fun making your own plug-ins !

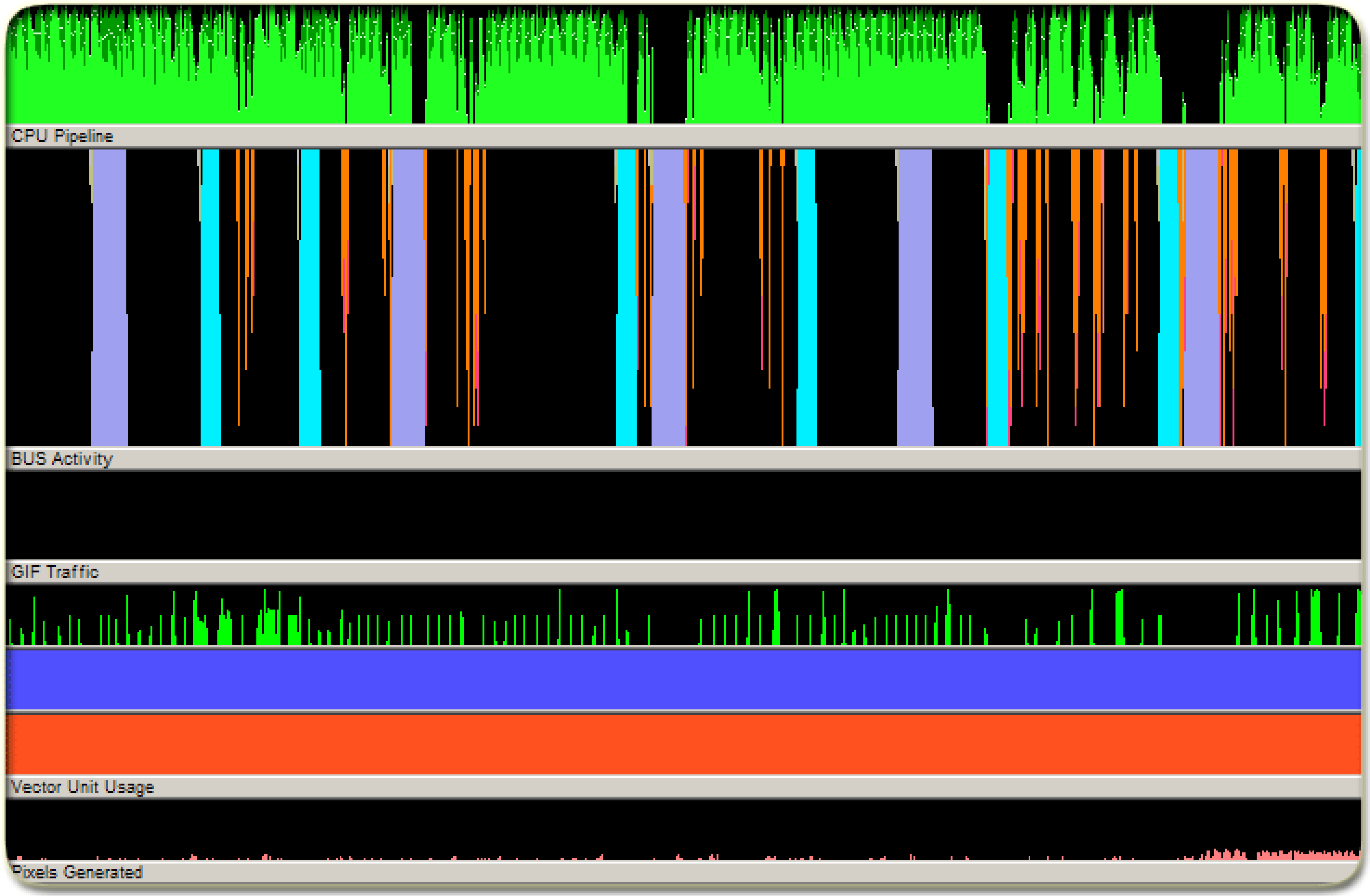
# Performance Analysis

- Good CPU activity
- A lot of cache misses
- Low DMA usage
- A lot of geometry
- A bit of VU0
- Good VU1 activity
- A lot of stalls
- A lot of pixels

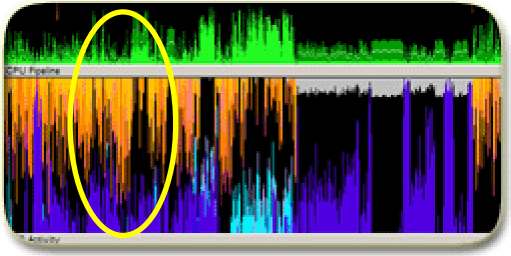


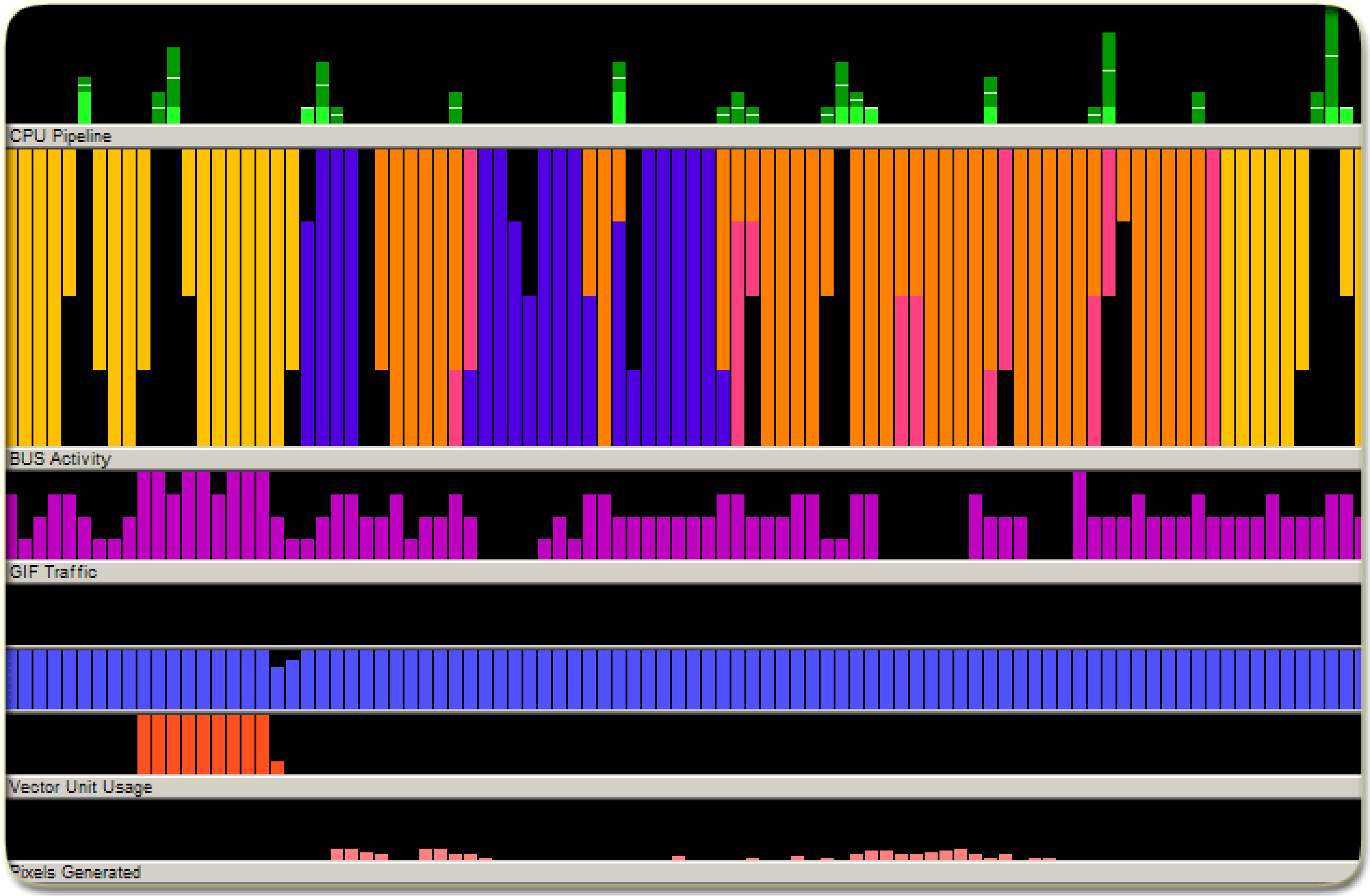
# High Performance



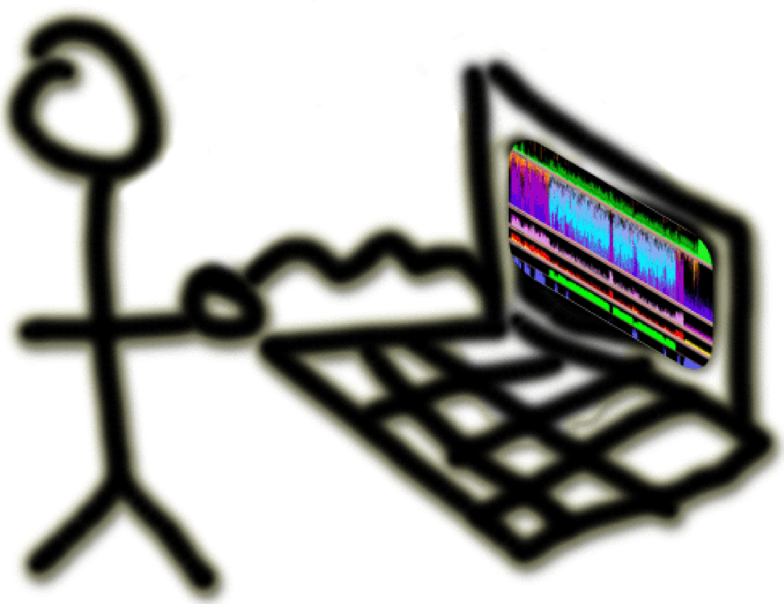
# Low Performance





# Looking at Some Scans

- **Live action !**



(Picture not contractual)

# Statistics From 2 Years of Scanning Games

# Rendering Analysis

- **52,000 polys per frame**
  - Min 10,000 – Max 145,000
- **Framerate: 60% were running at 25/30 or less**
- **95% were using full height buffers**

# Vector Unit Analysis

- **2% VU0 usage**
  - Most games are still not using VU0
  - Best performing games use up to 8% VU0

- **56% VU1 usage**
  - Due to stalls on large polygons and textures
  - Higher numbers don’t always mean better performance

# Data Transfer Analysis

- **2.3MB of data sent to VIF1 for geometry**
  - From 0.8MB to 5.3MB

- **1.5MB of data sent to the GIF for textures**
  - From 0.5MB to 5.5MB

# Processing and Rendering

- **3.6M pixels output**
  - From 0.9M to 12M
  - A full screen worth of pixel is about 0.3M pixels
    - That is 12 full screens worth of pixels on average
    - With a maximum of 40 !
- **120% Processing time**
- **115% Rendering time**

# What Did We Learn ?

# Common Techniques

- **There are common techniques used in most games**
  - There are common problems too
- **Some implementations did not prove to run as efficiently as expected**

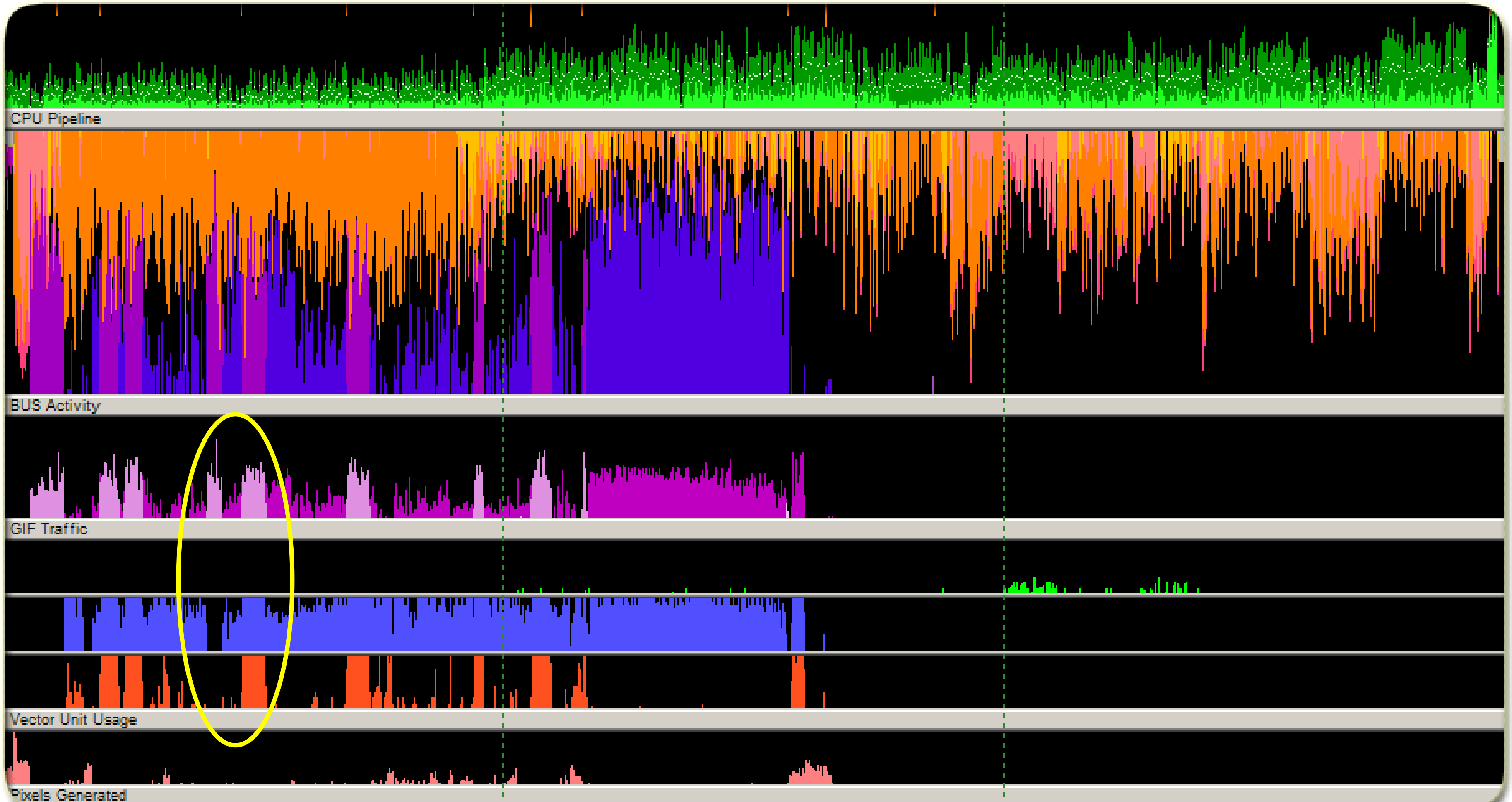
# Texture Syncing

- **Three popular techniques**
  - PATH2
  - Interrupts
  - MSKPATH3
- **All three have the same problem**
  - Texture is needed too soon and is not finished uploading
- **Send textures earlier**
  - Take in consideration upload time and drawing time
  - Try to find balance between geometry and textures
  - End of the frame is a good time for first texture

# VU1 Usage

- **Should run almost 100% of the time**
- **Often stalls on textures**
- **Often stalls on big polygons**
  - Subdivide when possible (e.g. particles)
- **Don’t overdo clipping**

# Texture Issues



# Clipping

- **Clipping is expensive**
- **Options to reduce clipping**
  - Test per object against “guardband”
  - Then test per triangle against guardband
  - If object needs clipping, clip against screen
  - Characters often don’t need full clipping
  - Use culling if possible
  - Use GPV to see if clipping is overdone/underused

# Data Packing

- **It is essential to keep DMA transfers as light as possible**

- **Use palletised textures when possible**
  - More friendly to the DMA and VRAM
  - Good quantitiser is essential
  - Swizzle for optimal performance
  - Don't listen to the artists, convert the textures yourself
  - Don't tell them I said that

# Data Packing

- **Pack the geometry**
  - e.g. V3_16 for vertex data
  - Use vertex compression, i.e. delta compression
- **Pack textures together to limit syncing problems**
  - Double buffer texture area in VRAM
  - Less interrupts
  - Big textures are ok, as long as the texel to pixel ratio is ~1:1

# Limit cache misses

- **Cache misses are the biggest issue**
  - 16KB for instructions, 8KB for data
- **ASM helps a lot**
- **SPR helps too**
- **Easy for me to say**

# Fast code vs. small code

- **Compiler lets you choose between “fast code” and “small code”**
  - Usually defaults to fast code
  - Very tempting to choose fast code
- **Cache issues are main bottleneck**
- **Use “small code” and save up to 10% in a mouse click**
  - Fastest optimisation ever

# Pixels Output

- **Usually not the main bottleneck**
  - Expect for particles and fullscreen effects
- **Keep polygons GS friendly**
  - 32 pixels wide, especially for fullscreen passes
  - Vertex locality is important
- **Check on PA that fillrate is what the geometry deserves**
  - A lot of surprises, e.g. in B->A operations

# Doubles Are Sneaky

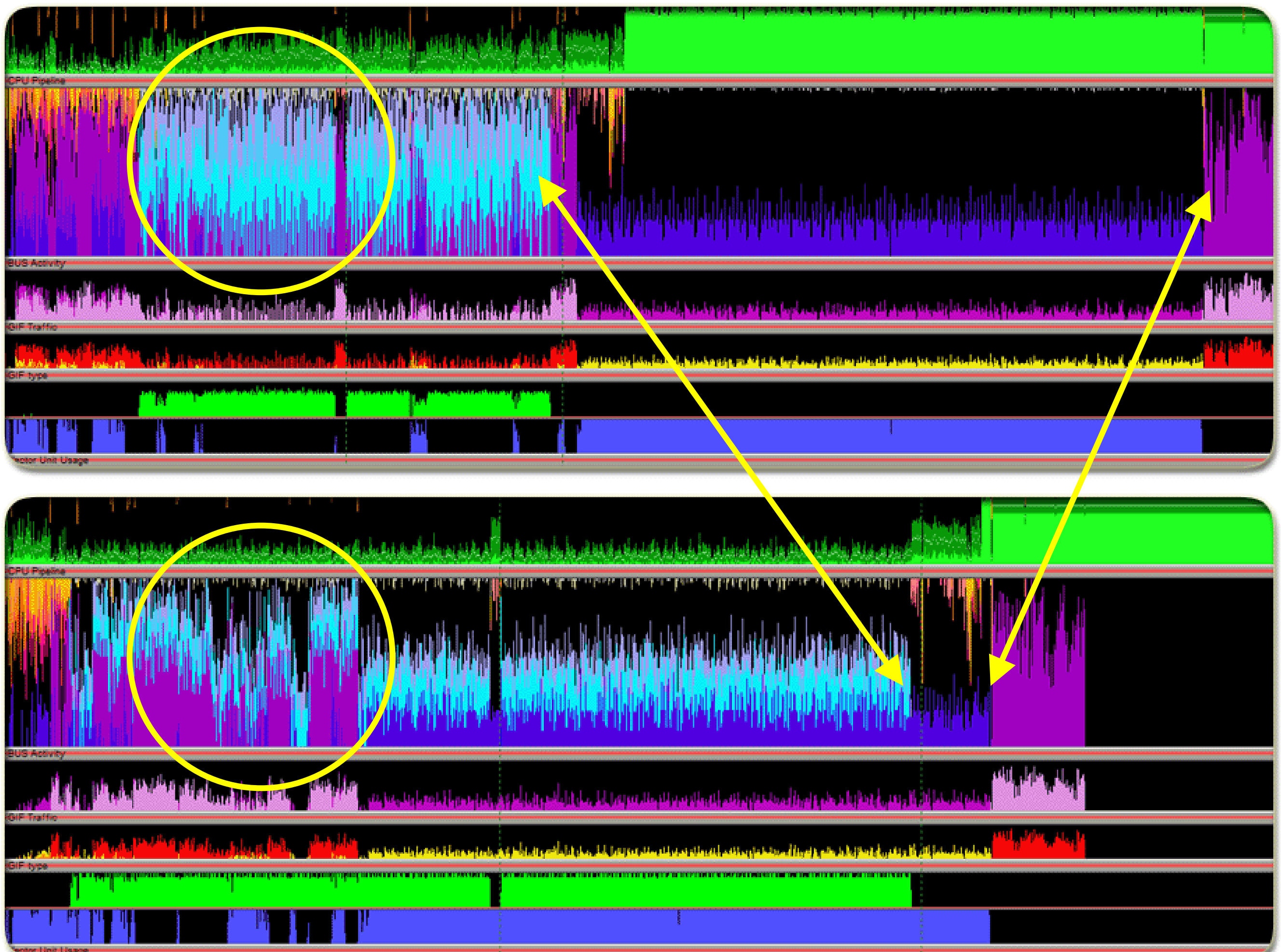
- **Doubles are calculated in software**
  – You wouldn't believe how many time we have found doubles in supposedly double-free code

- **Easy to spot on PA**
  – Shows as big spike of good CPU activity
  – If you've not done it on purpose, it may be a double
  – Shows in symbols table, look for "dp"
  – Don't laugh until you have re-checked your code

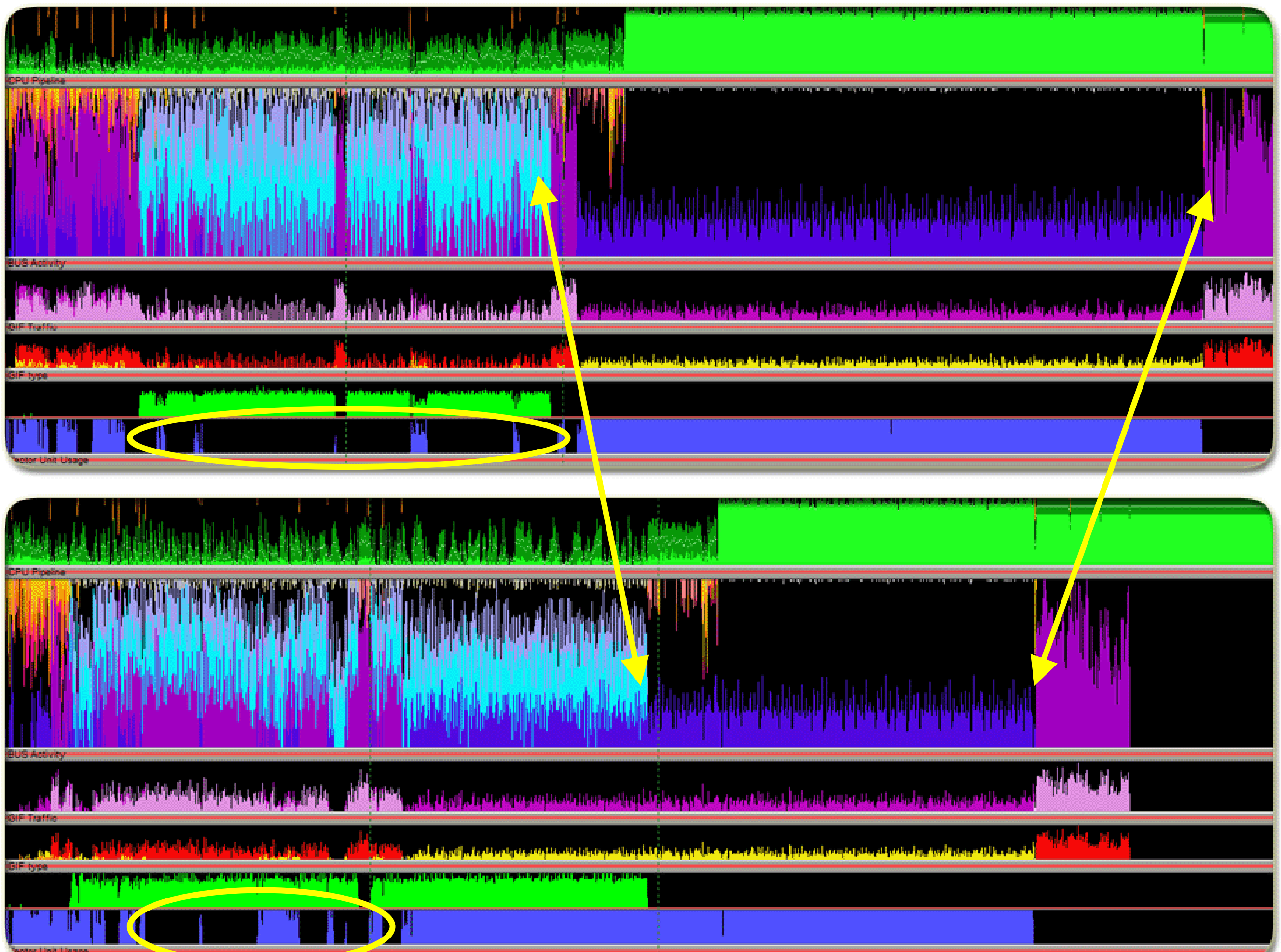
# VU0 Usage

- **Can be used for skinning, testing visibility, physics, particles, (AI?)**
- **Double-buffered SPR gives good results**
  - CPU arbitrates data flow, read/writes VU0 in macro-mode
  - DMA fills SPR with new data
  - Back to back transfer helps prevent large CPU stalls
  - Cycle stealing can help

# Bus is a shared resource



# Experiment with the PA



# Summary

- **More than half the games run at 25/30**
- **Most games still don't use VU0**
  - We've seen more VU0 usage in recent games
  - Best performing titles tend to use VU0 quite a bit
- **Most recent games draw over 50k polys**
  - Fastest so far seems to be 125k polys at 60fps
- **Most games draw between 2 to 5 Mps**
- **Main slowdown is still CPU efficiency**
  - Cache misses

# The Next Move

- **Main technical improvements are design improvements**
  - Streamed data saves on loading time
  - Packed data helps free precious bus time
- **Most of all, CPU efficiency is low**
  - Did I mention the cache misses ?
  - VU0, VU0, VU0 !

# Contact Us

- **SCEE Technology Group**
  - www.technology.scee.net
- **SCEE Corporate Site**
  - www.scee.net
- **PlayStation Global Hub**
  - www.playstation.com