TEACHING CONSOLE GAMES PROGRAMMING WITH THE SONY PLAYSTATION2 LINUX KIT

Henry S Fortuna
School of Computing and Creative Technologies
University of Abertay Dundee
Bell Street, Dundee DD1 1HG,
Scotland, UK
E-mail: h.s.fortuna@abertay.ac.uk

KEYWORDS
Games, Console, Programming, PlayStation2.

ABSTRACT

In May 2002, Sony Computers Entertainment Europe released the PlayStation2 Linux Development Kit providing educational establishments with a means for teaching native code development on this current generation of console. Games console programming is fundamentally different from programming games on PC based platforms and is a valuable skill to be mastered by any graduate seeking employment as a programmer in the Computer Games Industry. In order to obtain optimal performance from a console, detailed knowledge and understanding of the system hardware is required. The PlayStation2 contains several processors which operate in parallel, and both the programming and synchronisation of these processors is essential. Program code is developed for the PlayStation2 Linux Kit using a range of generic and proprietary tools. The GNU C++ compiler is used to create high level game code, an assembler and/or inline assembly is used to create custom high-speed routines, a Vector Command Line preprocessor is used to develop low level code for the Vector Unit processors and the Graphics Synthesizer is configured to render to a television or monitor. These techniques and tools are introduced to students to provide them with a realistic insight into modern console game development.

INTRODUCTION

The PlayStation2 Linux kit (Linuxplay Web Site, 2004) is added to a standard PlayStation2 (PS2) transforming it into a Linux workstation which can be used for many purposes including the development of native console game code. The kit consists of a hard disk drive, a keyboard and mouse, an Ethernet network adapter, cables, Linux operating system and development software.

The kit was originally released with two main methods of code development for graphics/games applications: an implementation of OpenGL called PS2GL, and a low level development library called libps2dev (Playstation2-linux Web Site, 2004). PS2GL did not utilise any of the advanced hardware within the PS2 and provided a development experience very similar to using OpenGL on a standard PC. Using PS2GL did not reflect the development methods being used by professional PS2 developers and this method was not pursued by the author.

Development under libps2dev provided some access to the console hardware including the vector units (VUs). However, the major disadvantage of libps2dev is that it did not provide access to the Direct Memory Access Controller (DMAC) which is a key component, central to providing high performance from the PS2. Under libps2dev, the function of the DMAC was emulated, leading to non-optimal performance from the console.

In November 2002, shortly after the release of the kit, the SPS2 Direct Access Development Environment (Osman, 2002) was released. SPS2 is a low level development library providing direct access to the PS2 hardware and unlike the other development libraries, it has been updated several times since its initial release. A significant feature of SPS2 is that it provides direct access to the DMAC, thus allowing the programmer to utilise the full power and performance of the PS2. SPS2 also provides a development infrastructure similar to that of the professional development kit, with code being developed under SPS2 only requiring minor modification to run under the professional development kit. It was for the reasons of similarity of experience with the professional development kit, and superior performance, that SPS2 was adopted as the development environment/library for games development courses at the University of Abertay Dundee.

PS2 ARCHITECTURE

Figure 1 shows the main internal components and data pathways that exist within the PS2. The main "Emotion Engine" (EE) core is a MIPS IV custom processor operating at 300MHz. A 32-bit floating-point unit (FPU) is connected to the EE Core and acts as a coprocessor. Two 128-bit vector processing units are present, VU0 and VU1. VU1 contains 16k of program memory and 16k of data memory and operates in Micro-mode, independently from the main CPU core. VU1 is connected directly to the graphics interface (GIF) which is used to unpack data and send it directly to the graphics synthesiser for rendering. VU1 is mainly used for vertex transformation, lighting and clipping. VU0 contains 4k of program memory and 4k of data memory and can be used in either Micro- or Macro-mode. In Macro-mode, VU0 acts as a second co-processor for the main CPU. In micro-mode VU0 executes its own micro-program independently from the main CPU and can be used in this mode for physics and other intensive in-game calculations. The vector units are connected to the data bus via their associated vector unit interface (VIF). The VIFs are intelligent microcontrollers which interpret the data sent to them using special instructions embedded in the data called VIFCodes.

The Direct Memory Access Controller (DMAC) is responsible for transferring data between main memory and the various processors and scratchpad memory. Correct utilisation of the DMAC is fundamental to obtaining high performance from the PS2. Data transfer is over a 128-bit bus which operates at a maximum transfer speed of 2.4 Gbytes per second.

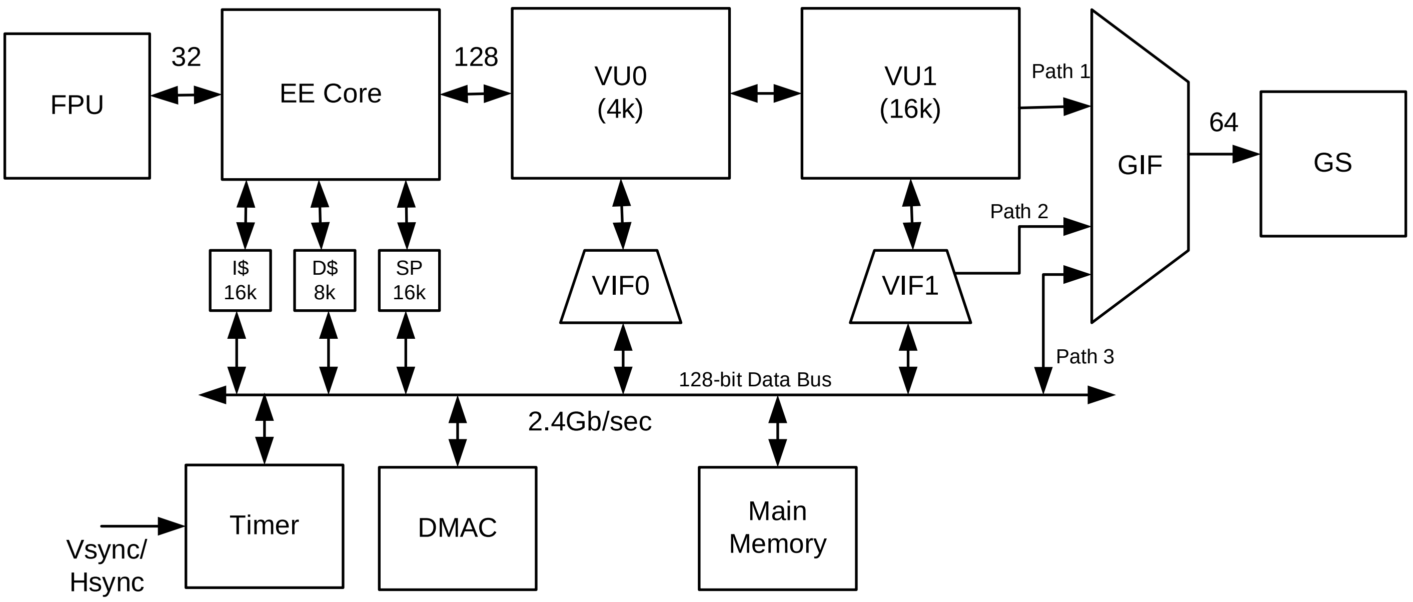

Figure 1

Three paths exist through the GIF to the graphics synthesiser. Path 1 is from VU1 micro-memory, Path 2 is from VIF1 and Path 3 is from the main data bus. Although there is flexibility in the use of each data path, the recommended function of the data paths is as follows (Sony Computer Entertainment Europe, 2001a). Path 3 is for loading image data into texture memory within the GS. Path 2 can also be used to upload texture data and for the setting of configuration registers within the GS. Path 2 transfers are convenient, in that they provide inherent synchronisation between texture data and vertex data. Path 1 is the main geometry path for transferring transformed vertex data to the GS for rendering.

A typical rendering process involves uploading texture data from main memory to the GS via path 2/3 and untransformed vertex data to VU1 micro memory via VIF1. A VU1 micro program transforms, lights and clips the vertex data then sends it over Path 1 to the GS for rendering. Many of these operations are carried out in parallel and are synchronized with the use of appropriate VIFCodes embedded within the vertex and texture data.

DEVELOPMENT PROCESS AND TOOLS

Several methods for games application development are possible with the PS2 Linux kit, with the arrangements adopted by the author being illustrated in figure 2

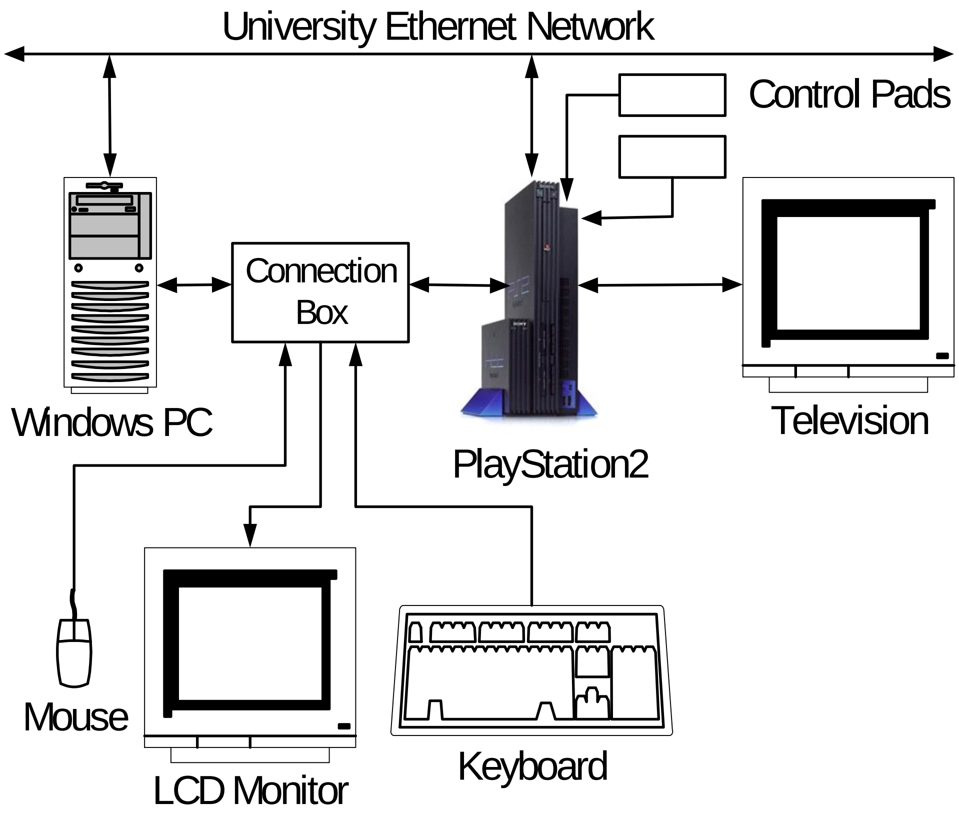

Figure 2

The development station consists of a PlayStation2 Linux kit, Windows PC, two PS2 controllers, dual input LCD monitor, keyboard, mouse, television and peripheral connection box. The dual input LCD monitor is used to display the video output from either the PC or the PlayStation2. The single keyboard and mouse are switchable between the PC and the PS2 using an interconnection box and suitable leads. Both the PC and the PS2 are connected to the University Ethernet network and communicate with each other via the TCP/IP and Server Message Block protocols. Graphics output from the PlayStation2 can be directed to the television for prototyping games applications at the correct resolution and size. This arrangement significantly reduces the amount of equipment needed per development station and maximises the utilisation of the student laboratory.

Students are free to select a development method that is comfortable to them, but an arrangement found to be successful is as follows. The Linux file system is made available to the PC via a Samba server running on the PS2. Files on the PS2 can be created and edited using a suitable text editor (such as UltraEdit or Visual Studio) running on the PC. From either a Telnet or SSH session from the PC to the PS2, programs can be compiled and executed on the PS2, with graphics output being directed to either the LCD monitor or television. Debug output from the program is sent via the Telnet/SSH session to a console window on the PC. Students store their project code on the main University file servers making this arrangement a robust, effective and secure development environment.

Several tools are used in order to develop games applications under PS2 Linux. Core game code is written in either C or C++ and GNU C and C++ compilers are shipped with the kit. The vector units are proprietary chips (Sony Computer Entertainment Europe, 2001b) which are programmed at assembly level. Both vector units have two execution units (upper and lower) which operate in parallel, leading to assembly code which is written in two parallel streams. This assembly code is compiled to native VU object code with a VU assembler (ee-dvp-as) shipped with the kit. The pairing and scheduling of VU assembly code is relatively complex for students with limited experience of assembly language, but a Vector Command Line (VCL) preprocessor which is shipped with the kit is available to help generate VU code. VCL takes a more traditional single stream of assembly language instructions and from that generates the dual stream of VU assembly code with correct scheduling, pairing and optimisation of instructions. The output from VCL is then compiled with ee-dvp-as to produce the vector unit object code.

The operation of the VUs can be remotely monitored and debugged using a visual debugger (Osman, 2003) running on either a Windows or Linux PC. A server program runs on the PS2 kit monitoring the execution of the VU micro program under control of the debugger client running on the PC. Full control of the execution of the micro program is obtained together with access to both program and data memory. Using the debugger it is possible to single step the execution of the VU code and observe output on the television/monitor on a frame-by-frame basis.

TEACHING METHODS

Students studying on the BSc Computer Games Technology course gain access to the PlayStation2 Linux kits for the first time in their second year where they study a full module in Console Game Programming. The module introduces topics such as the internal structure and organisation of a games console, the structure and organisation of a games program, and the tools necessary to create and import media content for games. By the end of this module students will understand how consoles are structured and organised and the methods and techniques that are necessary in order to program consoles effectively

Students entering the third year have a solid grounding in console architecture and programming and it is from this background that the PlayStation2 Linux kit is used to introduce the design and construction of console based 3D games engines. Students develop and use the mathematical routines that are necessary for implementing a 3D engine and generate code that interacts directly with the 3D console hardware such as the vector units. By the end of this module students will have created a small prototype 3D game engine and understand the structure, organisation, development and use of modern 3D games engines.

Students undertake a Group project in their third year and an individual honours project in their fourth and final year of study. The PlayStation2 Linux kit is available as a platform to undertake these projects. It provides access to a modern console for testing and evaluating algorithms, techniques and ideas.

A further theme that the PlayStation2 Linux kit is well suited to exposing is that of network programming and gaming. The Playstation2 Linux kit is supplied with a 10/100 Base-T Ethernet interface network adaptor and a Linux operating system with full network support. Under Linux, the Berkeley Sockets API provides access to the TCP/IP protocol suite that is the backbone of the Internet. The PlayStation2 Linux kit can therefore be used in the teaching of network theory and practice and more specifically in the design and implementation of network computer games. Using the kit it is possible for students to design and create network enabled computer games which have global access through the Internet

CONCLUSIONS

This article has reviewed the internal structure and organisation of the PlayStation2 Linux development kit and has demonstrated how the kit can be applied to teaching and learning on a wide range of topics within Computer Games Technology courses. In practice, the kit has been found to be highly motivational for students and is an invaluable tool for the in-context teaching of Computer Games Technology.

REFERENCES

Linuxplay WebSite, 2004
http://linuxplay.com
"Information on the PlayStation2 Linux Kit."

Osman S, 2002. "A Development Library for Linux (for Playstations2)"
http://playstation2-linux.com/projects/sps2

Osman S, 2003. "Sauce's Visual VU Debugger",
http://playstation2-linux.com/projects/sps2

Playstation2-linux Web Site, 2004
http://playstation2-linux.com/coding-on-playstation2.php
"Console game development options for the PlayStation2 Linux Kit."

Sony Computer Entertainment Europe, 2001a. EE Core User's Manual, 5th Edition

Sony Computer Entertainment Europe, 2001b. VU User's Manual, 5th Edition.

TEACHING CONSOLE GAMES PROGRAMMING WITH THE SONY PLAYSTATION2 LINUX KIT

Henry S Fortuna
School of Computing and Creative Technologies
University of Abertay Dundee
Bell Street, Dundee DD1 1HG,
Scotland, UK
E-mail: h.s.fortuna@abertay.ac.uk

KEYWORDS
Console, Programming, PlayStation2.

ABSTRACT

In May 2002, Sony Computers Entertainment Europe released the PlayStation2 Linux Development Kit providing educational establishments with a means for teaching native code development on this current generation of console. Games console programming is fundamentally different from programming games on PC based platforms and is a valuable skill to be mastered by any graduate seeking employment as a programmer in the Computer Games Industry. In order to obtain optimal performance from a console, detailed knowledge and understanding of the system hardware is required. The PlayStation2 contains several processors which operate in parallel, and both the programming and synchronisation of these processors is essential. Program code is developed for the PlayStation2 Linux Kit using a range of generic and proprietary tools. The GNU C++ compiler is used to create high level game code, an assembler and/or inline assembly is used to create custom high-speed routines, a Vector Command Line preprocessor is used to develop low level code for the Vector Unit processors and the Graphics Synthesiser is configured to render to a television or monitor. These techniques and tools are introduced to students to provide them with a realistic insight into modern console game development.

BIOGRAPHY

Henry Fortuna has a 1st Class Honours Degree in Electrical and Electronic Engineering and a PhD in Solid State Semiconductor Physics. He is a member of the Institution of Electrical Engineers and a Chartered Engineer. He currently teaches on both the BSc and MSc courses in Computer Games Technology at The University of Abertay Dundee. His teaching and development interests are mainly associated with console game programming and DirectX graphics programming. He is actively involved in the PlaySation2 Linux development community and has published tutorials which are shipped with the SPS2 PlayStation2 development environment. He currently maintains a web site (www.hsfortuna.pwp.blueyonder.co.uk) which presents information and tutorials pertinent to games and graphics programming using the PlayStation2 Linux kit.