BSc INITIAL REPORT

Submitted for the
BSc Honours in Computer Science with Games Development

April 2009

Particle System for the PSP

by

Jamie Sandell

Contents

1 - Introduction and Project Brief

1.1 - Introduction

The purpose of this initial report is to clearly define and outline my final year project. This report will analyse the context of the project and also identify key tasks. These tasks will then be further analysed to accurately assess how long they should take and pinpoint any risks that are involved. This task information is then taken and presented in the form of a Gantt chart so as to clearly outline the timescale of the project. The initial report also contains a reference list of information and sources that are deemed to be key material to the project, and that have been used in this initial report, or, will most likely be used in the future reports of the project. Finally the initial report closes with the appendix page, which contains any relevant information that doesn't fit in with the main body of text.

1.2 - Project Brief

Project Code: JHP1
Title: Particle Engine for PSP.

Specification:
This project is to produce a particle engine and tool for use on the PSP development kit. This will require the student to use an industry standard PSP development kit and SDK. The project is suitable for a proficient programmer and is ideal for someone hoping to work in the games industry.

2 - Analysis of Context and Identification of Tasks

The ultimate aim and goal of the project is to design and create a fully working particle engine and tool for the PSP development kit. The first piece of analysis that must be done is to actually discover not only just what a particle system is and how a particle system works, but, also, why such a system is required. On-top of this the question must be raised - *how can the project be incorporated onto the PSP and how can it run efficiently on the hardware that the PSP offers*. A broad overview of how this can be achieved must be supplied (See Identification of Tasks). Finally any alternatives to achieve the intended output must be touched upon.

2.1 - What is a Particle System?

A particle system is a "method for modeling fuzzy objects such as fire, clouds, and water. Particle systems model an object as a cloud of primitive particles that define its volume." (Reeves, 1983, p 1).

2.2 - How a Particle System Works

There are many different approaches to a particle system, however, they all work on the same underlying basic principles, i.e.:

- The particle is generated by an emitter (see Appendix II)
- It's new position is calculated
- Render the particle
- If it's at the end of its lifespan it is removed, if appropriate it is reinitialised.

It is how the particle engine does this that makes the effects very different from a different effect, e.g. an explosion effect compared against a rain effect, these effects are completely different but are in fact created using the exact same principles. For instance the emitter can be changed to any emitter type wanted. The calculation of the particle's new position can involve physics based simulation or stochastic ("*The most common choice is to implement some kind of dynamics to the particles so they mimic real-world phenomena.*"(Dalmau)). Then the particle engine renders the particle, this can be anything that the graphics library supports, e.g. a point or a triangle strip.

2.3 - Why a Particle System?

Why would someone need to program a formal algorithm to create such visual effects, why can't conventional tools (e.g. Maya 3D) be used to create such an effect and animate it? Well this is answered and explained perfectly by Sabo in his article, he states, "*Particles making up the system are non-static unlike standard geometry objects. Particles are born, they change over time, and then die off. A key point regarding particle systems is that they are chaotic. Instead of having a completely predetermined path, each particle can have a random element, called a stochastic component, which modifies its behavior. This random element is, in fact, the main reason why particle systems are so good at reproducing realistic effects.*"(Sabo).

2.4 - Porting to the PSP

Porting the project to the PSP will not be too difficult, however, converting some graphical commands (OpenGL) will require converting the commands to their equivalent using a graphical library on the PSP.
The biggest challenge with porting the engine will be getting it to run on the PSP efficiently, it must be kept in mind that the PSP is a handheld device, and as such, when compared to a cutting edge PC, or even an average PC it is lacking in hardware capabilities (See Appendix I). This actually poses a big challenge because a particle system can be displaying literally thousands of particles on the screen at any given timeframe, and not only just drawing them but also calculating their positions.

2.5 - Alternatives to Achieving the Goal

Classically there are two methods of achieving the goal, the first is to code the particle system as a class hierarchy with ParticleSystem as an abstract class and new particles effects (e.g. fire) deriving via inheritance from this parent.
The second alternative is to "*create a deep, complex particle system class, where individual systems are nothing but data-driven objects that parameterize the class in many ways.*" (Dalmau).
The plan of action for the project is to actually utilise both methods to achieve the project's goal. The second method will be utilised first to create the editor (which will be used on a PC); this method is chosen for the editor because once an effect has been edited it will be easy to update the display to show the new particle effect (e.g. a switch statement to select the kind of emitter to use). The editor will then write the effects to a new file which will be in the form of a specified class framework.
The first method will be the one actually used on the PSP to show the particle effects in action. This method has been chosen for this task because when in a game environment, or any big project environment, it is best to use an Object-Orientated approach as it becomes more manageable and easier to understand (easier to abstract and view high-level concepts).

3 – Project Task List and Timescales

#1 Initial Report
A report outlining the ultimate aims and goals of the project accompanied by a plan and timescale.
Duration – 9 days.

#2 Research and Design
Continue with background research and design the system at a high level.
Duration – 3 days.

#3 Prototyping

Produce a prototype of the project software and assess it externally.
Duration – 1 day.

#4 Design
Go back to the initial design document/(s) and alter according to feedback.
Duration – 1 day.

#5 Prototyping
Alter the prototype accordingly.
Duration – 1day.

#6 Unit Implementation
Code the necessary units.
Duration – 42 days.

#7 Unit Testing
Test all units extensively (white box and black box testing).
Duration – 42 days.

#8 Unit Integration
Integrate the necessary units.
Duration – 5 days.

#9 Integration Testing
Test the integration extensively (white box and black box testing).
Duration – 5 days.

#10 Interim Report
The interim report is essentially a progress review including any and all revisions, updates, and modifications that need addressing at this stage of the project.
Duration – 14 days.

#11 Final Report
The final report is and all encompassing document of everything that I have done and achieved during my final year project.
Duration – 60 days.

#12 Project Appraisal
This is a review of how far successfully I have managed my project and my time.
Duration – 14 days.

#13 Presentation
Preparing and executing the presentation of my final year project.
Duration – 28 days.

4 - Project Timeplan

See Appendix III

5 - Risk Analysis

5.1 - Risk Analysis

Risk	Severity	Likelihood	Significance
Equipment Failure	High	Low	Medium
Cognitive Shortfall	Medium	Medium	Medium
Deadline Pressure	High	Low	Medium

5.2 – Risk Strategies

Risk	Evaluation	Action
Equipment Failure	If there was any equipment failure then there is a cost to fix it/replace it and in the worse case a loss of data.	Keep version backups of all work on different types of media (including the internet).
Cognitive Shortfall	Failing to understand the work is always an issue with anything new and if not combated is quite an issue.	Try my hardest to understand everything, if there is something I don't understand then I must seek the relevant help until I do.
Deadline Pressure	Deadline press is a very severe issue that can cause extreme stress, depression, and ultimately project failure.	Be aware of deadlines and steadily work towards them constantly, do not put the work off.

6 - Background References

Lander J, 1998, *The Ocean Spray in Your Face* [online], Available: http://www.darwin3d.com/gamedev/articles/col0798.pdf [Accessed 16th October 2008].

Dalmau DSC, 2003, *Core Techniques and Algorithms in Game Programming*, New Riders Publishing.

Reeves TR, 1983, Particle Systems – *A Technique for Modeling a Class of Fuzzy Objects*, SIGGRAPH [online], 17(3), p359. Available: http://www.lri.fr/~mbl/ENS/IG2/devoir2/files/docs/fuzzyParticles.pdf [Accessed 16th October 2008].

Zhang J, Angel E, Alsing P, Munich D, *An Object-Orientated Particle System for Simulation and Visualization* [online], Available: http://www.cs.unm.edu/~treport/tr/01-06/particle.pdf [Accessed 17th October 2008].

VDB John, 2000, *Building an Advanced Particle System* [online], Game Developer, Available: http://www.mysticgd.com/misc/AdvancedParticleSystems.pdf [Accessed 18th October 2008].

Molofee J, *Lesson: 19* [online], Available: http://nehe.gamedev.net/data/lessons/lesson.asp?lesson=19 [Accessed 17th October 2008].

Allen M, *Particle Systems* [online], Available: http://web.cs.wpi.edu/~matt/courses/cs563/talks/psys.html [Accessed 17th October 2008].

Latta L, 2004, *Building a Million-Particle System* [online], Available: http://www.gamasutra.com/view/feature/2122/building_a_millionparticle_system.php [Accessed 22nd October 2008].

Sabo M, *Improving advanced particle system by adding property milestones to particle life cycle* [online], Available: http://www.cescg.org/CESCG-2004/web/Sabo-Miroslav/ [Accessed 22nd October 2008].

7 - Appendix

7.1 - Appendix I

PSP Hardware Specifications

CPU

- Allegrex CPU
  - MIPS r4000 32-bit core 1-333mhz
  - 16kib I-Cache & D-Cache
    - 64-byte line length
    - 2-way set associative, LRU
  - No TLB
  - 7-stage pipeline
  - 32 32-bit registers
  - FPU (COP1)
    - 32-bit single precision
    - 32 32-bit registers
    - IEEE 754 compliant
    - Sqrt (28 cycles), div(28 cycles), most others 1 cycle
  - VFPU (COP2)
    - Vector FPU “Macromode only”
    - Designed for vector and matrix ops
    - 128 32-bit registers
      - Reconfigurable as scalar, vector or matrix
      - IEEE 754 Single precision float
    - Can also handle 32-bit int, 16-bit int, 8-bit int, half float
    - vmmul.z vd, vs, vt - 4×4 matrix/vector multiply, 22 cycles

- Media Block CPU
  - MIPS r4000-based core
  - 2MB Embedded DRAM
  - VME - Virtual Mobile Engine
    - Reconfigurable processor to decode audio & video
    - ATRAC3plus & MP3 for music
      - ATRAC3plus & ADPCM for games but not MP3 due to licensing issues
  - AVC H.264 engine
    - MPEG-4 Hardware accelerator
    - Up to 720x480x30fps
      - Libraries support 480x272x29.97fps

Bus

- Main bus shared by CPU and Graphics Engine
- CPU only has level-1 cache, recomend minimizing memory usage
- Cache miss ~70 cycles
- VRAM read ~44 cycles

    - contention with GE
- Scratchpad read ~38 cycles

Graphics

- Graphics Engine (GE)
    - 2MB Embedded DRAM
    - Supports Lighting, skinning (8 weights), morphing, subdivision, pixel operations
    - Operates at bus speed (default 111mhz)
    - 3.5GB/s Bus bandwidth
    - 444 Mpixels/sec fill rate
    - 23 Mploys/sec T&L

Storage

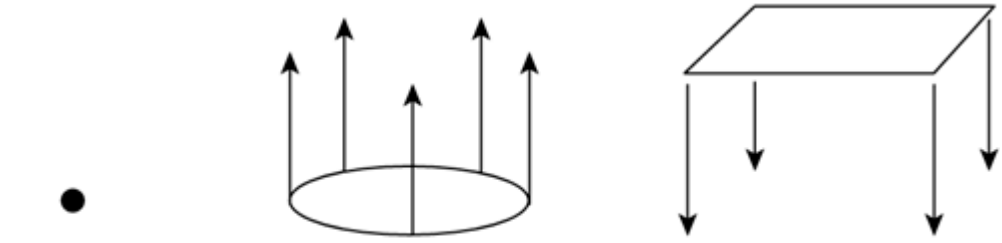
- UMD
    - UMD 1.8 GiB DISC

Ports

- Serial-Port (RS-232)
    - http://mc.pp.se/psp/phones.xhtml

- USB-Port

7.2 - Appendix II



"*Particles are created at some kind of emitter, which initializes their parameters. If we want our system to behave in any interesting way,*
the key is to generate each particle with slightly different initial values, so the behavior rules (which are shared by all particles) make each one look like a unique element." (Dalmau)

7.3 - Figure 1

	Task Name	Duration	Start	Finish	Predecessors
1	Initial Report	9 days	Tue 21/10/08	Fri 31/10/08	
2		0 days	Fri 31/10/08	Fri 31/10/08	1
3	Research and Design	3 days	Sat 01/11/08	Tue 04/11/08	2
4	Prototyping	1 day	Wed 05/11/08	Wed 05/11/08	3
5	Design	1 day	Thu 06/11/08	Thu 06/11/08	4
6	Prototyping	1 day	Fri 07/11/08	Fri 07/11/08	5
7	Unit Implementation	42 days	Sat 08/11/08	Mon 05/01/09	6
8	Unit Testing	42 days	Sat 08/11/08	Mon 05/01/09	
9	Unit Integration	5 days	Tue 06/01/09	Mon 12/01/09	8
10	Integration Testing	5 days	Tue 06/01/09	Mon 12/01/09	7
11	Interim Report	14 days	Tue 13/01/09	Fri 30/01/09	10
12		0 days	Fri 30/01/09	Fri 30/01/09	11
13	Final Report	60 days	Sat 31/01/09	Thu 23/04/09	12
14		0 days	Thu 23/04/09	Thu 23/04/09	13
15	Project Appraisal	14 days	Fri 24/04/09	Wed 13/05/09	14
16	Presentation	28 days	Thu 14/05/09	Mon 22/06/09	15
17		0 days	Mon 22/06/09	Mon 22/06/09	
18					
19					
20					
21					
22					
23					
24	08343 ACW1	15 days	Wed 08/10/08	Tue 28/10/08	
25		0 days	Tue 28/10/08	Tue 28/10/08	
26	08356 ACW1	29 days	Thu 16/10/08	Fri 21/11/08	
27		0 days	Fri 21/11/08	Fri 21/11/08	
28	08343 ACW2	42 days	Mon 27/10/08	Fri 19/12/08	
29		0 days	Fri 19/12/08	Fri 19/12/08	
30	08356 ACW2	42 days	Mon 27/10/08	Fri 19/12/08	
31		0 days	Fri 19/12/08	Fri 19/12/08	
32	Semester1 Exams	9 days	Mon 19/01/09	Thu 29/01/09	
33	Semester 2 Exams	20 days	Thu 28/05/09	Wed 24/06/09