Organización del Computador II

Departamento de Computación,
Facultad de Ciencias Exactas y Naturales,
Universidad de Buenos Aires

Trabajo Práctico 1: "Oportuncrisis" (Segunda Entrega)

Primer Cuatrimestre de 2009

Grupo "UNPCKHPD"

Pablo Antonio	290/08	pabloa@gmail.com
Pablo Herrero	332/07	pablodherrero@gmail.com
Estefanía Porta	451/04	estef.barbara@gmail.com

Índice

1. Introducción

La segunda entrega de este trabajo práctico consiste, básicamente, en:

- la *optimización* de las funciones que formaron parte de la primera entrega, y
- la inclusión de *dos nuevas funciones*: `smooth()` y `negative()`

La intención principal de este segundo trabajo es sacarle provecho a las distintas facilidades que proveen las extensiones SIMD[1] de la arquitectura x86 de Intel, así como aplicar distintas técnicas para la optimización del código que se ejecuta en dicha arquitectura.

1.1. ¿Qué se muestra ahora por pantalla?

Recordemos cuál era el ciclo que se repetía todo el tiempo en el juego:

```
1  procesar eventos de entrada
2  actualizar posiciones, estados, etc.
3  chequear IA, colisiones, física, etc.
4  mostrar resultados por pantalla
5  ira a 1.
```

Ahora, en la etapa 4 ("mostrar resultados por pantalla"), tiene lugar la siguiente secuencia (extendida):

```
1  generar el fondo actual (generarFondo)
2  generar el plasma, usando como color-off el color del cielo
       del fondo (generarPlasma)
3  para cada sprite de cada personaje:
4      recortar la instancia que se quiere del personaje
5      aplicar blit (cambiar el color-off del personaje por lo
           que haya en la pantalla)
6  si la tecla 1 o 3 se encuentra oprimida, comenzar la
       transición a un nuevo escenario
7  si la tecla 1 o 3 se encuentra oprimida, aplicar el efecto
       negative
8  si se esta realizando la transición a un nuevo escenario:
9      aplicar el efecto smooth
10     si hay más pixels negros que de otro color:
11         finalizar el cambio de escenario
```

[1] *Single Instruction, Multiple Data*

1.2. El efecto *negative*

1.3. El efecto *smooth*

2. Desarrollo

2.1. Funciones desarrolladas

2.1.1. Función `negative`

```
void negative();
```

Parámetros:

- Ninguno.

Archivo en el que se halla la función: `src/asm/negative.asm`

Pseudocódigo:

```
void negative():
    para cada componente RGB c de cada pixel de la pantalla:
        sumatoria = superior(c) + inferior(c) + anterior(c) + posterior(c) + 1
        c = f(sumatoria)
```

Donde la función f es:

$$f(x) = \frac{1}{\sqrt{sumatoria}} \times 255$$

2.1.2. Función `smooth`

```
bool smooth();
```

Parámetros:

- Ninguno.

Archivo en el que se halla la función: `src/asm/smooth.asm`

Pseudocódigo:

```
bool smooth():
    para cada componente RGB c de cada pixel de la pantalla:
        sumatoria = superior(c) + inferior(c) + anterior(c) + posterior(c)
        c = sumatoria/4
        si sumatoria == 0:
            contador_negros += 1
        si no:
            contador_blancos +=1
    retornar evaluar(contador_negros > contador_blancos)
```

2.2. Funciones optimizadas

2.3. Optimización

2.3.1. Realizando operaciones en paralelo (SIMD)

Desde la aparición del procesador *Pentium II* de *Intel*, se agregaron a la arquitectura varias extensiones que proveen soporte para instrucciones SIMD. Entre ellas se encuentran: MMX, SSE, SSE2, SSE3 y SSE4.

Las instrucciones SIMD se utilizan para realizar operaciones en paralelo; se suele trabajar con vectores de datos en lugar de datos individuales. Las mejoras en *performance* son generalmente apreciables, y muchas veces determinan la utilidad de ciertas aplicaciones.

2.3.2. Evitando saltos

2.3.3. Acceder a memoria no es tan lento (gracias a la memoria caché)

2.3.4. Operaciones aritméticas con `lea` y *shifts*

3. Resultados

4. Conclusiones

Referencias

- Intel R . 64 and IA-32 Architectures Software 1: Basic Architecture
- Intel R. 64 and IA-32 Architectures Software 2A: Instruction Set Reference, A-M
- Intel R. 64 and IA-32 Architectures Software 2B: Instruction Set Reference, N-Z
- Información sobre bitmaps: http://en.wikipedia.org/wiki/BMP_file_format
- Documentación del NASM: http://www.nasm.us/doc/