



Performance analysis

Profiling a CG application



Profiling

- Goals:
 - To collect and process information about resource and time usage in an application.
- Issues:
 - Where is time spent?
 - Which functions are called more often?
 - Where are the bottlenecks?
- A profiling tool allows to determine the influence of each task ...
- ... And identify potential bottlenecks



Profiling

- Visual Studio – Function profiling

Func Time	%	Func+Child Time	Hit %	Count	Function
19828.853	84.1	19828.853	84.1	179716	_glutSolidSphere@16 (glut32.dll)
1521.176	6.5	1521.176	6.5	1	_glutCreateWindowWithExit@8 (glut32.dll)
891.893	3.8	891.893	3.8	44929	_glutSolidCone@24 (glut32.dll)
703.682	3.0	21424.428	90.9	44929	drawSnowMan(void) (glutsnowman.obj)
214.521	0.9	21955.474	93.2	1248	renderScene(void) (glutsnowman.obj)
134.222	0.6	134.222	0.6	12481	GetExactTime(void) (custom_time.obj)
79.197	0.3	79.197	0.3	1248	_glutSwapBuffers@0 (glut32.dll)
75.232	0.3	22040.395	93.5	1	_glutMainLoop@0 (glut32.dll)
47.964	0.2	47.964	0.2	28	glutSetWindowTitle@4 (glut32.dll)
13.768	0.1	13.768	0.1	1248	MarkTimeThisTick(void) (custom_time.obj)

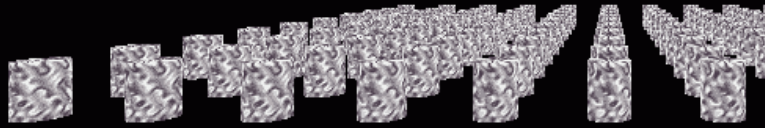


Profiling

Building a CG Profiler

- Features:
 - Block concept (may match a frame)
 - Real time OSD
- Only four functions are required:
 - Init
 - Begin Block
 - End Block
 - Report

Name	#c	ms	wt
Render	1	33.807	0.000
Draw	1	33.257	0.000
immediate	100	33.203	0.006
Stats Dump	1	1.166	0.000
dummy	0	0.000	0.000
Swap	0	0.097	0.000





Profiling

Usage:

```
void renderScene(void) {  
  
    {  
        PROFILE("Render");  
        ...  
  
        {  
            PROFILE("Draw");  
            drawObjects();  
        }  
  
        {  
            PROFILE("Stats Dump");  
            ...  
        }  
  
        { PROFILE("dummy");  
          ... // non graphical tasks ☺  
        }  
  
        {  
            PROFILE("Swap");  
            glutSwapBuffers();  
        }  
    }  
}
```



Profiling

- Analysis:

Block name	Name	# calls per frame	Total time per frame	Time spent in the profiler itself
		#c	ms	wt
Indentation replicates hierarchy	Render	1	40.569	0.000
	Draw	1	33.383	0.000
	immediate	100	33.330	0.008
	Stats Dump	1	1.175	0.000
	dummy	1	5.823	0.000
	Swap	1	0.102	0.000



Profiling

- Analysis :

Immediate mode

w/ vertex buffers

Name	#c	ms	wt	Name	#c	ms	wt
Render	1	40.569	0.000	Render	1	15.204	0.000
Draw	1	33.383	0.000	Draw	1	0.060	0.000
immediate	100	33.330	0.008	VB	100	0.033	0.004
Stats Dump	1	1.175	0.000	Stats Dump	1	1.158	0.000
dummy	1	5.823	0.000	dummy	1	5.704	0.000
Swap	1	0.102	0.000	Swap	1	8.227	0.000



Profiling

- Analysis :
 - Non graphical tasks impact in immediate mode

Name	#c	ms	wt	Name	#c	ms	wt
Render	1	40.569	0.000	Render	1	34.605	0.000
Draw	1	33.383	0.000	Draw	1	33.218	0.000
immediate	100	33.330	0.008	immediate	100	33.165	0.007
Stats Dump	1	1.175	0.000	Stats Dump	1	1.197	0.000
dummy	1	5.823	0.000	dummy	1	0.000	0.000
Swap	1	0.102	0.000	Swap	1	0.112	0.000



Profiling

- Analysis :
 - Non graphical tasks impact with vertex buffers

Name	#c	ms	wt	Name	#c	ms	wt
Render	1	9.526	0.000	Render	1	9.491	0.000
Draw	1	0.056	0.000	Draw	1	0.058	0.000
VB	100	0.033	0.004	VB	100	0.034	0.004
Stats Dump	1	1.331	0.000	Stats Dump	1	1.341	0.000
dummy	1	5.797	0.000	dummy	1	0.000	0.000
Swap	1	2.296	0.000	Swap	1	8.050	0.000



References

- Game Programming Gems, Vol I
- Real Time Rendering, Moller and Haines