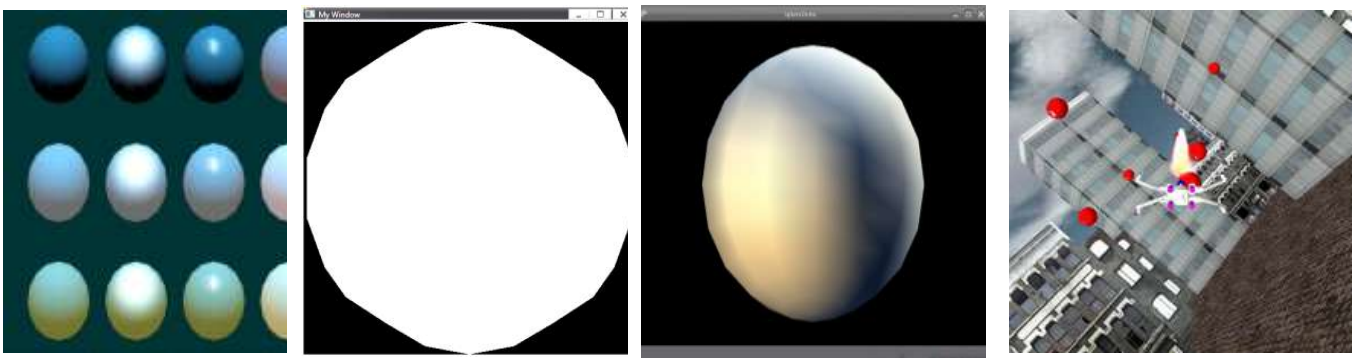
Business at the speed of game production: OpenGL or XNA?

Necemon Yai
Department of Computer Science, Swansea University
******@swansea.ac.uk



(a) Some OpenGL spheres [Silicon Graphics]

(b) An OpenGL sphere

(c) An XNA sphere

(d) Some spheres in an XNA game [Riemers]

Figure 1: Teaser

Abstract

A common start-ups' question in the gaming industry is: Which graphical environment should we use? As time is money, a crucial factor in a choice is time: not only the time it takes to build complete games with each API but also a fluid timing during the gaming experience to make it enjoyable. In this analysis, we will compare the OpenGL industry standard to the Microsoft XNA game framework on the speed factor. The study will include learning curve, programming, modeling and rendering speeds on both sides.

1 Introduction

A graphics API is basically a way to talk to graphics hardware. By using an API, it gets simpler to send drawing commands to the hardware, since human understandable formats can be used. There are different graphics APIs available nowadays and each API has got advantages and disadvantages. There are many factors that can influence the choice of a specific API, namely cross-platform support, performance, ease of use, popularity, licensing, etc. When it comes to speed, we are more concerned with the performance factor, mainly determined the amount of functional layers (abstraction) between the API and the hardware can differ. Performance is often about the amount of code that's being processed, so in many cases more code equals lower performance. A business aiming for production speed would also be concerned with the learning curve and the programming time that should be involved.

OpenGL are both great graphics APIs currently used in the game industry. It can be really a dilemma to go for a particular one. Keeping the speed factor in mind, we will try to come up with the most relevant one according to our needs.

2 Background

Let's introduce the APIs and their features.

2.1 OpenGL

OpenGL (Open Graphics Library) is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives. OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization, and flight simulation. It is also used in video games, where it competes with Direct3D on Microsoft Windows platforms. OpenGL is managed by a non-profit technology consortium, the Khronos Group [Wikipedia 1].

2.2 Microsoft XNA

Microsoft XNA ('XNA's Not Acronymed) is a set of tools with a managed runtime environment provided by Microsoft that facilitates computer game development and management. XNA attempts to free game developers from writing "repetitive boilerplate code" and to bring different aspects of game production into a single system. The XNA toolset was announced March 24, 2004, at the Game Developers Conference in San Jose, California. A first Community Technology Preview of XNA Build was released on March 14, 2006. XNA Game Studio 2.0 was released in December 2007, followed by XNA Game Studio 3.0 on October 30, 2008.

XNA currently encompasses Microsoft's entire Game Development Sections, including the standard Xbox Development Kit and XNA Game Studio [Wikipedia 2].

3 Learning curve and programming VS Performance

Since OpenGL is a native C library, the interface provided does not implement any classes or namespaces.

XNA on the other hand is a .NET platform enabled Framework which means that it cannot be used by native languages, only by managed languages but it implements classes and namespaces. XNA also includes a lot of helper functions and libraries and is fully compatible with the .NET framework. So it is easier to write XNA programs. It is also easier to learn it.

While OpenGL is closer to the hardware, implements fewer levels of abstraction and can be ported to a higher level language, the time it takes to develop an OpenGL application will be higher than that of an XNA application.

The XNA Framework is a .NET library which invokes native Direct3D calls to be used on the virtual machine in which the .NET Framework operates. This high level of abstraction can cause performance issues and it is recommended to use the XNA Framework solely for lower performance applications [Scriptionary].

From this analysis we can deduct that XNA has a faster learning curve and an better ease of use; But OpenGL seems to offer a better performance. Let's check this assumption through a performance test.

4 Performance Test

4.1 Description

The comparative performance test will consist of displaying a huge number of spheres using each API and check which API takes more time. The glutsolidsphere() function is used to draw spheres in OpenGL but there is no straight away mean of drawing primitives in XNA, so we will be using some third party class for our purpose [Tshrove].
The experiment happens in a windows 7 environment. The system is a HP Envy 15 laptop with 4 GB ram and an i7 quadcore processor.
On both sides, we draw spheres with only radius 1, 10 stacks and 10 slices.

4.2 Code

4.2.1 OpenGL Code

We use a loop to create many spheres using the glutSolidSphere() function. We gradually increase the maximum value of the counter

```
for(i=0;i<10000;i++)
                glutSolidSphere(1,10,10);
```

We print the current time twice in the console window: once before we draw the spheres and again just after we draw the sphere:

```
time(&now);
      current = localtime(&now);
printf("the    time    is    %i:%i:%i\n",
current->tm_hour,       current->tm_min,
current->tm_sec);
```

4.2.2 XNA Code

The process is similar. We draw a big number of spheres using a 'for loop' and we increase that number gradually.

```
for (long i = 0; i < 100; i++)
            {
                m = new
Tshrove.Primitives.Sphere(this, 10, 10,
1.0f, Content.Load<Texture2D>("t"),
                Matrix.CreateLookAt(new
Vector3(10.0f, 10.0f, 600.0f),
Vector3.Zero, Vector3.Up),

Matrix.CreatePerspectiveFieldOfView(Mat
hHelper.PiOver4,
GraphicsDevice.Viewport.AspectRatio,
0.0001f, 1000.0f));
                slist.Add(m);
                m.Initialize();
            }

foreach (Tshrove.Primitives.Sphere sp
in slist)
        this.Components.Add(sp);
```

[Aaron Reeds]
We note the current time twice also, once before drawing spheres and once after drawing spheres. This time, we don't print times in a console application. We print that in a text file.

```
file = new
System.IO.StreamWriter("n:\\test.txt",
true);

file.WriteLine(System.DateTime.Now +
"." + System.DateTime.Now.Millisecond);
            file.Close();
```

4.3 Screenshots

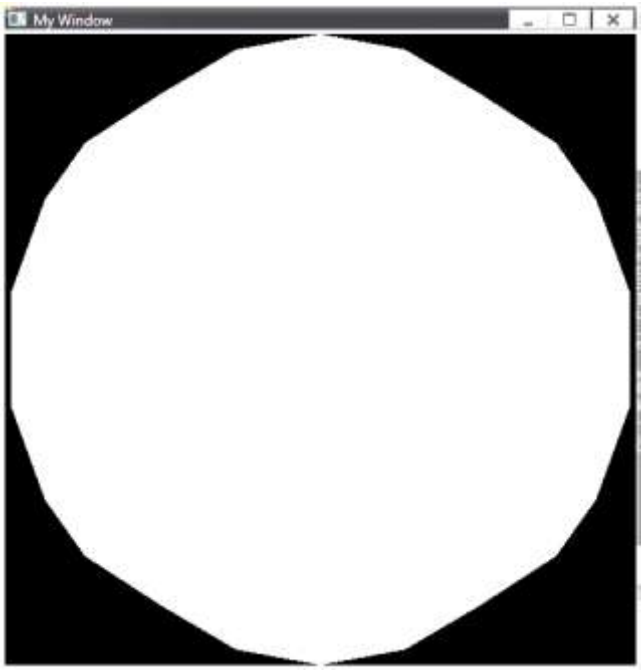


Figure 2: A sphere in OpenGL

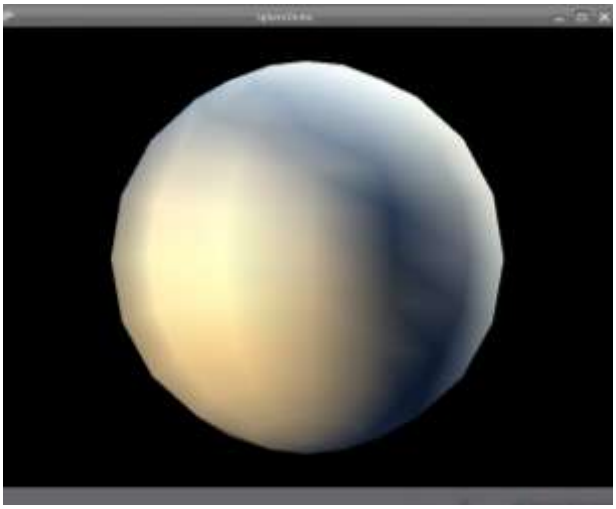


Figure 3: A sphere in XNA

4.4 Results

Number of spheres	OpenGL time	XNA time
1000	< 1 sec	0.191 sec
10000	< 2 sec	6.610 sec
50000	< 10 sec	2min 9.396 sec
70000	< 14 sec	5min 6.342 sec

Table 1: Comparison between OpenGL and XNA drawing times

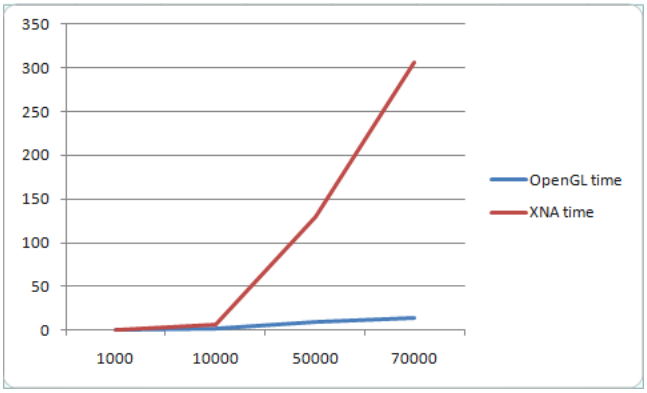


Chart 1: 2 D line chart expressing the results

5 Conclusion

Through the results, we can note that OpenGL provides a much better performance but XNA also brings some advantages to the game business. XNA is much easier and faster to learn. Regarding speed, even though XNA is easier to learn and to code, we can feel that OpenGL can be more interesting on a long term to produce industry games. A team of well trained OpenGL developers definitely got a performance advantage over XNA rivals.

References

Silicon Graphics, Addison Wesley 1994, OpenGL Programming Guide

Riemers XNA tutorials, http://www.riemers.net/eng/Tutorials/XNA/Csharp/Series2/Point_sprites.php

Wikipedia (1), The Free Encyclopaedia, Wikipedia on OpenGL, http://en.wikipedia.org/wiki/OpenGL, accessed in March 2010

Wikipedia (2), The Free Encyclopaedia, Wikipedia on Microsoft XNA, http://en.wikipedia.org/wiki/Microsoft_XNA, accessed in March 2010

Scriptionary, Direct3D, OpenGL and XNA Fieldguide, http://scriptionary.com/Direct3D,_OpenGL_and_XNA_Fieldguide, accessed in march 2010

Tshrove, a place where developers can add code for other developers to use and customize, XNA sphere class. http://code.tshrove.com/?page_id=8, accessed in march 2010

Aaron Reeds, O'Reilly 2009, Learning XNA 3.0