iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 2 **Issue:** XI **Month of publication:** November 2014

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

3D development with WebGL

Simran Bhatti[1], Vandana Tayal[2], Pooja Gulia[3]
Students, Computer Science, Maharishi Dayanand University

Abstract: ***The WebGL API gives JavaScript developers the ability to tap directly into the powerful built-in 3D graphics acceleration capabilities of today's PC and mobile-device hardware. Supported transparently in modern browsers, WebGL makes it possible to create high-performance 3D games, applications, and various 3D-enhanced UIs for mainstream web users. This article presents a series of information for JavaScript developers who are new to WebGL. In this, we have gone through a basic explanation that demonstrates WebGL fundamentals and related 3D graphics concepts.***

I. INTRODUCTION

We live in a 3D world, yet almost all of our interactions with computers and computerized devices occur over 2D user interfaces. High-speed, fluid, realistic 3D applications — at one time the exclusive domain of computer animators, scientific users, and gaming enthusiasts — were out of reach for mainstream PC users until relatively recently. Today, all mainstream PC CPUs have built-in 3D graphics acceleration, and gaming PCs have additional dedicated high-performance graphics processing units (GPUs) to handle 3D rendering. This trend is reflected in the reduced instruction set computing (RISC)-based CPUs in phones and tablets. All current mobile CPUs include powerful 3D-capable graphics-acceleration GPUs. The supporting software drivers have also matured, and are now stable and efficient. Advances in modern browser technology bring with them hardware-accelerated WebGL, a 3D JavaScript API that runs alongside feature-rich HTML5. JavaScript developers can now create interactive 3D games, applications, and 3D-enhanced UIs. With WebGL integrated into mainstream browsers, 3D application development is finally accessible to a huge population of developers armed simply with a browser and a text editor.

II. 3D HARDWARE EVOLUTION: A BRIEF HISTORY

In the early days of personal computing (circa early 1980s), the 3D accelerated graphics hardware was available only on expensive specialized workstations from companies such as Silicon Graphics Computer Systems (later called SGI). These machines were mostly available to scientists and researchers in academia, government, and the defense industry and to entertainment industry. The graphics hardware in early conventional PCs had very low-resolution 2D capabilities; any 3D computation and drawing had to be performed offline — slowly — by application software. In the late 1990s, affordable (but still relatively expensive) 3D accelerated graphics adapters from companies such as ATI Technologies Inc. (now AMD) and Nvidia Corp. started to become available to PC enthusiasts. These adapters implemented technological breakthroughs that brought workstation-quality graphics to the PC. But the GPUs in early 3D graphics adapters were limited by low display resolution and low rendering/processing power.
By the middle of the last decade, mainstream CPU manufacturers such as Intel and AMD had started a competitive trend to integrate 3D rendering hardware into the main CPUs, or bundling GPU(s) with CPUs on the same chip. This trend dramatically improved the accessibility of 3D graphics to all users. For the first time in the PC's history, the cost of entry to formerly workstation-grade 3D graphics on the PC was just the cost of downloading a supporting software driver.

III. THE 3D APPLICATION SOFTWARE

For most of early PC history, 3D hardware drivers were bundled with, or compiled in, the application. This configuration optimizes access to the hardware-accelerated features of the hardware, resulting in the best possible performance. Essentially, you code directly to the hardware's capabilities. Well-designed games or computer-assisted design (CAD) applications can squeeze every ounce of juice out of the underlying hardware. Figure 1 shows this software configuration.

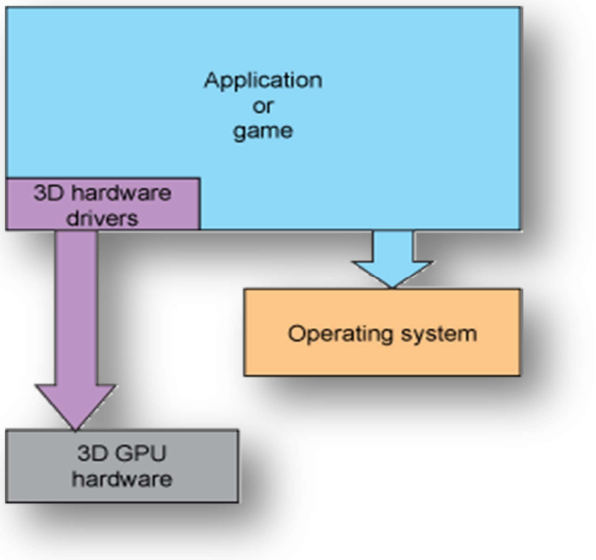
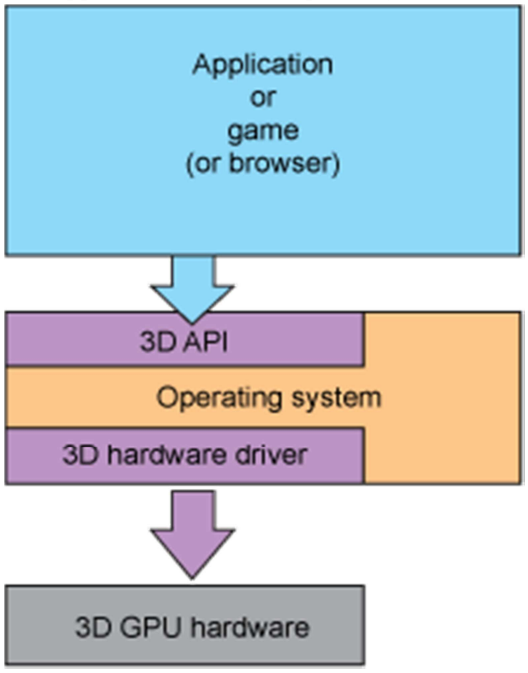


Figure 1 Embedded 3D hardware drivers-Application

As soon as new hardware is introduced (with a new or updated driver), the software vendor must make and distribute new releases, this was a major distribution problem prior to the widespread accessibility of high-speed broadband networks. As a solution to the driver-update problem, the operating system took on the role of hosting the 3D graphics driver(s). The application or game calls an API that the OS provides, and the OS in turn converts the call into the primitives that the native 3D hardware driver accepts. Figure 2 illustrates this arrangement.

Figure 2. Application using the operating system's 3D API



In this way an application can be programmed to the OS's 3D APIs. The application is shielded from changes to the 3D hardware driver and even from the evolution of the 3D hardware itself. For many applications, including all mainstream browsers, this configuration worked adequately for a long time. The OS acted as a middleman that tried valiantly to cater to all types or styles of applications and to graphics hardware from competing vendors. But this one-size-fits-all approach takes its toll in terms of 3D rendering performance. Applications that require the best hardware-acceleration performance still must discover the actual installed graphics hardware, implement tweaks to optimize code for each set of hardware, and often be programmed to vendor-specific

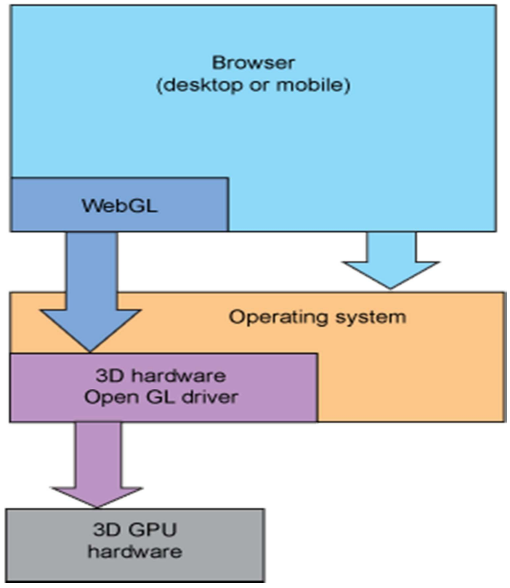
extensions to the OS's API — again making applications hostage to the underlying drivers or the physical hardware.

A. *WebGL Span*

Enter the modern age, with high-performance 3D hardware built into every desktop and mobile device. Applications are increasingly developed with JavaScript to leverage browser capabilities, and web designers and web application developers clamored for faster and better 2D/3D browser support. The result is a wide support of WebGL by mainstream browser vendors. WebGL is based on OpenGL Embedded System (ES), which is a low-level procedural API for accessing 3D hardware. OpenGL — created in the early 1990s by SGI — is now considered a well-understood and mature API. WebGL gives JavaScript developers near-native-speed access to the 3D hardware on a device for the first time in history.

WebGL APIs gets almost direct access to the underlying OpenGL hardware driver, without the penalty of code being translated first through the browser support libraries and then the OS's 3D API libraries. Figure 3 illustrates this new model.

Figure 3. JavaScript application accessing 3D hardware through WebGL



Hardware-accelerated WebGL enables 3D gaming on browsers, real-time 3D data visualization applications, and futuristic interactive 3D UIs — to name just a few possibilities. The standardization of OpenGL ES ensures that new vendor drivers can be installed without affecting existing WebGL-based applications, delivering on the utopian "any 3D hardware on any platform" promise.

B. *Drawing 3D in web*

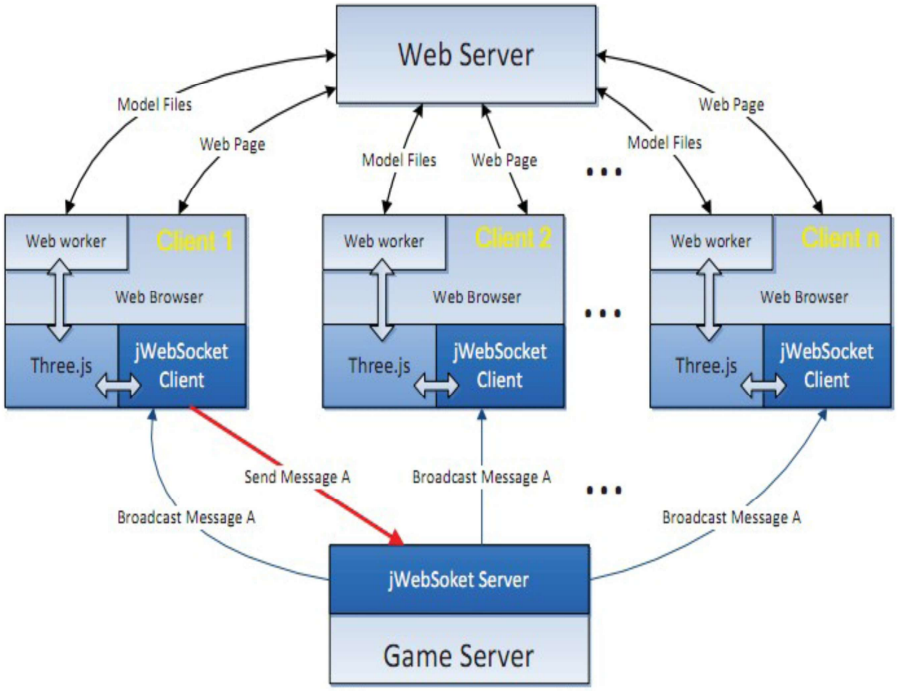
Before WebGL existed, there were two different ways to bring 3D content to the browsers – mainly Flash and Java Applets. As Java Applets are almost non-existent, flash is remaining as the only Known competitor to WebGL. Main disadvantage of Java Applets and Flash is need to have an installed plug-in into the browser. This has always been problematic as it is even impossible in some computers (for example on school computer, where user cannot install his own plugins or upgrade version of those). Mobile devices are also growing problem, as those do not support Flash or Java Applets mostly. Adobe also discontinued the mobile device support, so its future as provider of 3D-content to multiple different devices. So the only relevant comparison at this point is between

C. *WebGL and Flash.*

There is possibility to draw 3D objects for both of the platforms using applications like Blender, Copper Cube etc. Both of them have also its 3D engine – Away3D for flash and Three.js for WebGL. However WebGL needs an GPU support for shader rendering to be supported and Viewable by the user. WebGL have currently quite poor browser support (IE does not support at all, neither do any stable version of Safari and Opera for OS X).

D. Potential WebGL usage

There are three main areas where there might be good spots for WebGL to step in: different kind of information visualization, gaming and 3D modeling. As seen on Performance-chapter, the performance of WebGL is very promising compared to Flash. However the browser support of WebGL is currently quite weak, so if idea of the app is to be available to as much users as possible then it's a good idea to stick to the flash and leave WebGL to the future projects. This is the thing with the games: If you want to write popular game for the Facebook, you don't want to use WebGL to it. But for the future WebGL have a lot of potential in web game industry in future, especially in mobile browser support. Bijin Chen pushed the web gaming into the future by implementing browser based multiplayer online gaming framework using WebGL and Web Sockets for communicating. According to Bijin browser based multiplayer might be the future as it does not need any separate installation, and game could be played from any computer. Although there was not much research papers and articles on the topic available, there is one interesting project called Tinker cad. In Tinker cad user can draw an 3D model inside the web browser and send it to some printing company to create an actual model out of the drawing. The project seems promising, although the functionality is a bit limited at the beta stage of the project. It could also be nice to have some kind of object importer there. Third way to utilize WebGL is to visualize complex data which requires 3D presentation. One such a project is presented by John Congote et al, who applied WebGL to medical imaging application.

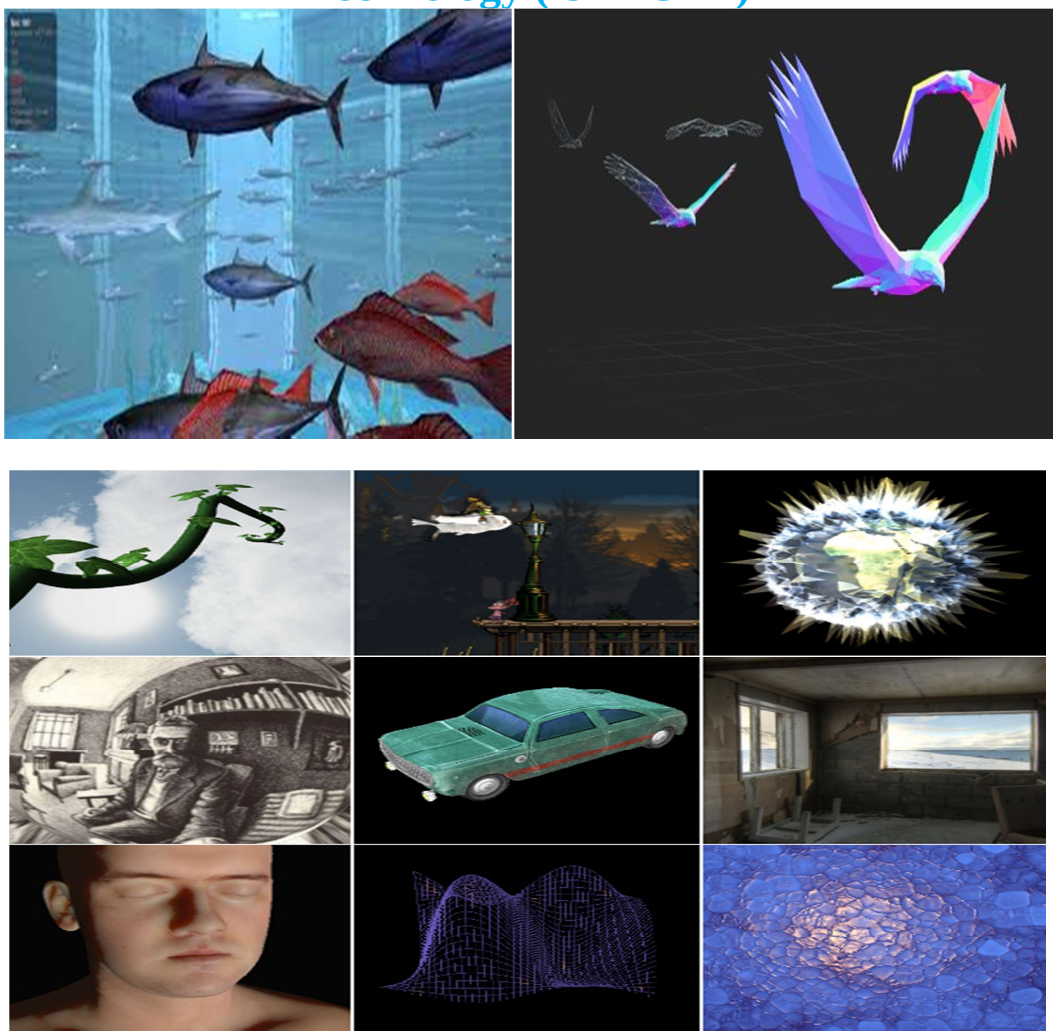


The framework architecture is shown in the following image. Web server is handling the interaction and providing the model files, and jWebSocket is handling the communication between the players. Another potential usage is 3D modeling applications.

E. What is WebGL used for?

WebGL allows developers to put real-time interactive 3D graphics in the browser. WebGL can be applied to interactive music videos, games, data visualization, art, 3D design environments, 3D modeling of space, 3D modeling of objects, plotting mathematical functions, or creating physical simulations.

For Example:-



IV. CONCLUSION

WebGL opens up the raw 3D hardware for JavaScript API access, but the API remains low-level:

A. WebGL has no notion of and does not care about what is being displayed within the 3D scene. The simple 3D pyramid object in this article's example isn't known to the WebGL API layer. Instead, you must painstakingly keep track of the vertices data that makes up the object.

B. Each call to draw a WebGL scene renders only a 2D picture. Animation itself is not part of WebGL and must be implemented by additional code.

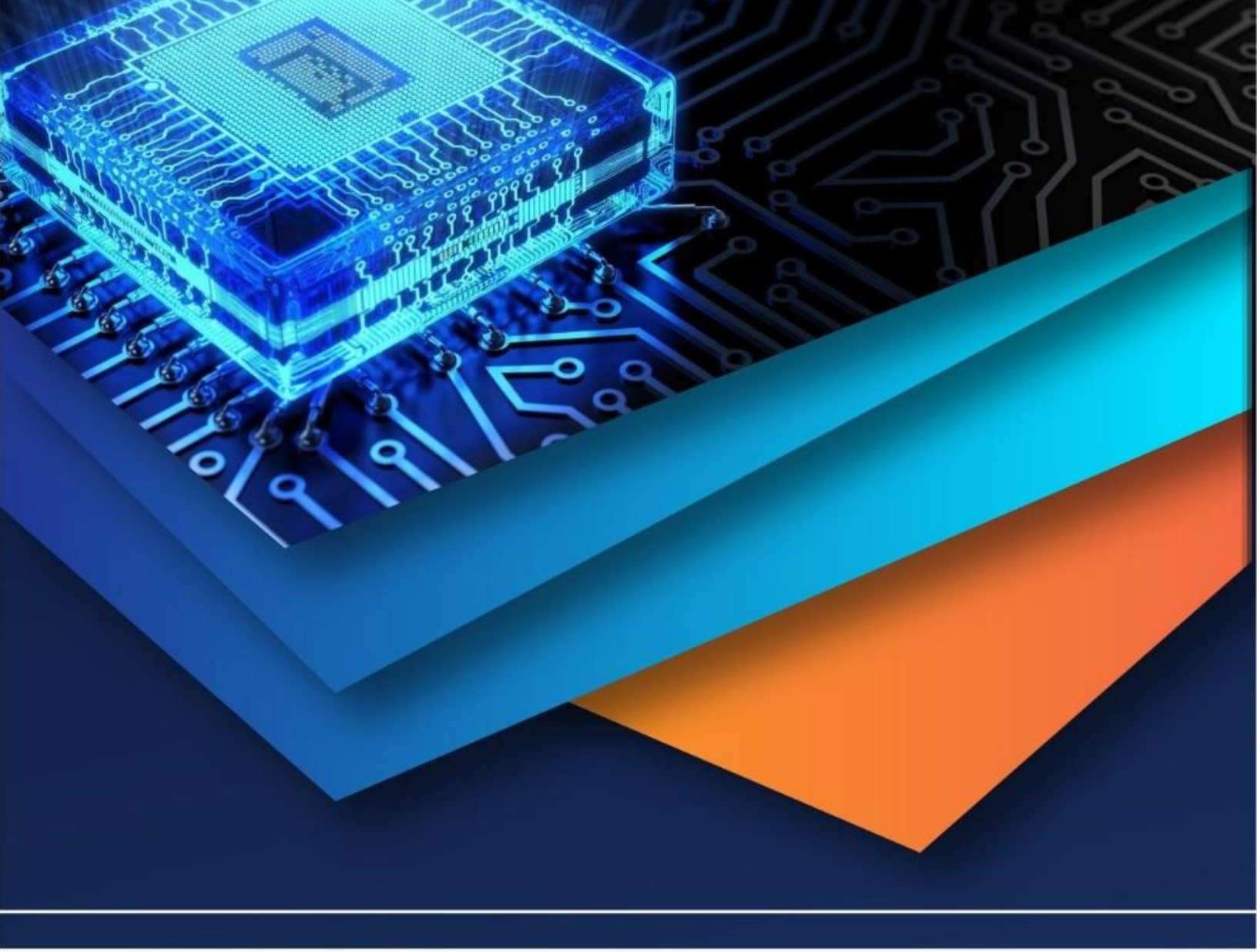
C. Movement and interactions between object and environment, such as light reflections and object physics, must be implemented in additional code.

D. Handling of events, such as user input, object selection, or object collision, must be implemented in higher-level code.

E. WebGL is widely supported in modern browsers. However its availability is dependent on other factors like the GPU supporting it.

REFERENCES

[1] Khornos Group home pages, available at http://www.khronos.org
[2] http://en.wikipedia.org/wiki/WebGL
[3] https://dev.opera.com/articles/introduction-to-webgl-part-1
[4] https://www.khronos.org/**webgl**/wiki/Main_Page



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY