

**BSc (Hons)** **Computer Games Technology**

**A JavaScript Runtime for Hardware Accelerated Applications**

**William Taylor**

**B00235610**

**22/04/2016**

**Supervisor:** **Paul Keir**

# Acknowledgements

Paul Keir

Here is why I am acknowledging his help.

Mark Stansfield

Here is why I am acknowledging his help.

Alastair MacMonnies

Here is why I am acknowledging his help.

A JavaScript Runtime for Hardware Accelerated Applications

**William Taylor**

**April 22, 2016**

To investigate and explore how we can make hardware acceleration experimentation more accessible, a prototype runtime was developed on top of Google’s V8 JavaScript compiler and popular open source frameworks. In tandem, demonstrations were built using the runtime to investigate the advantages of our approach. Reasons for creating the platform are numerous, they include making the technology more accessible and aiding pre-existing efforts to find new ways of leveraging the hardware. The development of the runtime is discussed in detail as are revisions to the JavaScript standard that allow it to be used for general-purpose scripting. The runtime demonstrates that an all in one platform can streamline development and make general purpose computation on graphics hardware easier for both novices and experts.

**Keywords:** GPUs, GPGPU, Google, V8, JavaScript, Platform, Learning, Development, Experimentation

Table of Contents

[Acknowledgements 2](#_Toc472775883)

[1.0 Introduction 5](#_Toc472775884)

[2.0 Background Information 6](#_Toc472775885)

[2.1 GPUs 6](#_Toc472775886)

[2.1.1 Hardware 6](#_Toc472775887)

[2.1.2 Manufacturers 6](#_Toc472775888)

[2.1.3 Software 6](#_Toc472775889)

[2.2 JavaScript 6](#_Toc472775890)

[2.2.1 Design 6](#_Toc472775891)

[2.2.2 Typed Arrays 6](#_Toc472775892)

[2.2.3 ECMAScript 2015 6](#_Toc472775893)

[2.3 V8 6](#_Toc472775894)

[2.3.1 Design 6](#_Toc472775895)

[2.3.2 Concepts 6](#_Toc472775896)

[2.3.3 Usages 6](#_Toc472775897)

[3.0 Software Design 7](#_Toc472775898)

[3.1 Requirements 7](#_Toc472775899)

[3.2 Architecture 7](#_Toc472775900)

[3.3 Interface 7](#_Toc472775901)

[4.0 Development 8](#_Toc472775902)

[4.1 Embedding V8 8](#_Toc472775903)

[4.2 Module System 8](#_Toc472775904)

[4.3 Common Libraries 8](#_Toc472775905)

[4.4 GPU Bindings 8](#_Toc472775906)

[5.0 Testing 9](#_Toc472775907)

[5.1 3D Cube Demo 9](#_Toc472775908)

[5.2 Grayscale Demo 9](#_Toc472775909)

[5.3 Finance Demo 9](#_Toc472775910)

[6.0 Future Work 10](#_Toc472775911)

[7.0 Critical Appraisal 11](#_Toc472775912)

[8.0 Conclusion 12](#_Toc472775913)

# 1.0 Introduction

In recent history, there has been a seismic shift in technology. Processors have stopped getting faster at an exponential rate. Increasing the clock speed of processors has now been abandoned in favour of multicore processors (David Geer, 2005). Due to 3D and high resolution media increasing in popularity we now see Graphics Processing Units (GPUs) integrated into modern computers by default (Intelcom, 2016, Amdcom, 2016). Easily learning and experimenting with this new technology is of great importance if we are to see general purpose computing on graphics processing units (GPGPU) more widely adopted. This is the topic for this honours project where we will explore the possibility of an integrated platform for GPU technique experimentation and development. Specifically, we will look at a dedicated platform that leverages the popular scripting language JavaScript to provide a reliable and flexible tool to those learning how to leverage GPUs for the first time and to those who wish to develop their own GPU techniques in an easier manner.

There are currently several problems we have identified in experimenting with GPUs and writing GPU based applications that limit a programmer’s ability to get stuck into this exciting piece of technology.

One is that at the time of writing this paper there is currently no easy to use integrated environment to experiment with various GPU APIs such as OpenCL and OpenGL. While one could argue that the Web provides an integrated environment through WebCL and WebGL which are web equivalents of OpenCL and OpenGL, I would strongly disagree for several reasons. The first being that due to the requirement of a browser being portable it is unable to provide support for GPU technologies designed for specific hardware such as CUDA or specific APIs locked into a single operating system such as DirectX. The other reason for disagreeing is because the browser has a security model that disables local access to the computer making the loading of data such as complex 3D geometric models overbearing and complicated. So, while the Web may provide a way to write GPU programs it is more for web developers to speed up their applications and not to provide a toolset to make GPU programming as easy as possible.

Another issue identified is the frustration of using the native bindings to the APIs from C++. As C++ doesn’t provide native support through the standard template library (STL) for images, models, input and windows it leads to a lot of extra work with additional libraries and APIs rather than letting you get on with your GPU technique development. The result is a lot of boilerplate before you get to writing what you will be experimenting with and that is the GPU programs themselves whether that is kernels in OpenCL or shaders in OpenGL. This issue is after you install various SDKs and tools to get access to these APIs, making it not only difficult when you start writing your program but difficult to get started in the first place.

The proposed solution is this project where we aim to build an all in one platform suitable for GPGPU experimentation, learning and prototyping. We will develop a JavaScript runtime which aims to provide a bulk of features out the box to reduce the learning curve required and provide native bindings to popular industry standard APIs that are suitable to both novices and experts. The platform should be easy to install and easy to use, skipping lengthy and numerous SDK installations in favour of a onetime install platform that provides everything required out of the box. The development of the platform and research should highlight several key points. The first showing the speed of compilation and execution of JavaScript and how it can be utilized as a generic scripting language for numerous environments. The second showing how leveraging specialised hardware which is more common than ever in today’s world can accelerate traditional applications. Finally, by showing the importance and relevance of both modern JavaScript as a general scripting language and accelerated programming for being the tool the programmers must leverage if we are to see more performant software.

# 2.0 Background Information

## 2.1 GPUs

### 2.1.1 Hardware

### 2.1.2 Manufacturers

### 2.1.3 Software

## 2.2 JavaScript

### 2.2.1 Design

### 2.2.2 Typed Arrays

### 2.2.3 ECMAScript 2015

## 2.3 V8

### 2.3.1 Design

### 2.3.2 Concepts

### 2.3.3 Usages

# 3.0 Software Design

## 3.1 Requirements

## 3.2 Architecture

## 3.3 Interface

# 4.0 Development

## 4.1 Embedding V8

## 4.2 Module System

## 4.3 Common Libraries

## 4.4 GPU Bindings

# 5.0 Testing

## 5.1 3D Cube Demo

## 5.2 Grayscale Demo

## 5.3 Finance Demo

# 6.0 Future Work

# 7.0 Critical Appraisal

# 8.0 Conclusion

References

Appendix 1 – 3D Demo Source Code

Appendix 2– Grayscale Demo Source Code

Appendix 3 – Finance Demo Source Code