OBSIDIAN

A Vulkan Component based Rendering Library

Fall 2016 Senior Project | Version 1.0

Author: Alain Galvan
agalvan@fiu.edu

Mentor: Francisco Ortega
fortega@fiu.edu

Instructor: Masoud Sadjadi
sadjadi@cs.fiu.edu

TA: Mohsen Taheri
mtahe006@fiu.edu

Supported by HPDRC / OpenHID (Dr. Naphtali Riche / Dr. Francisco Ortega)

Introduction

Obsidian is a C++ Vulkan powered component based rendering library inspired by React Native.

It uses functional composition to build a scene graph and updates it using the Flux architecure.

Benchmarks show it performing >8x faster than OpenGL. It's available for download at github.com/openhid/obsidian.

JavaScript

```
import React, {Component} from 'react';

type HelloProps = {};
type HelloState = {};

class HelloWorld extends Component<HelloProps, HelloState> {
  render() {
    return (
      <Text>
        Hello World
      </Text>
    );
  }
}

import {render} from 'react-dom';

render(<HelloWorld/>);
```

C++

```
#include "obsidian.hpp"

struct HelloProps {};
struct HelloState {};

class HelloWorld : public Component<HelloProps, HelloState>
{
  HelloWorld(HelloProps props = {}) : Component(props)
  {

  }

  render() {
    return (
      CreateElement<Text>({}, "Hello World")
    );
  }
};

int main()
{
  render(CreateElement<HelloWorld>());
}
```

Prior Art

Component based Architecture - Apps are built as tree of function factories, resulting in less overhead than domain specific languages.

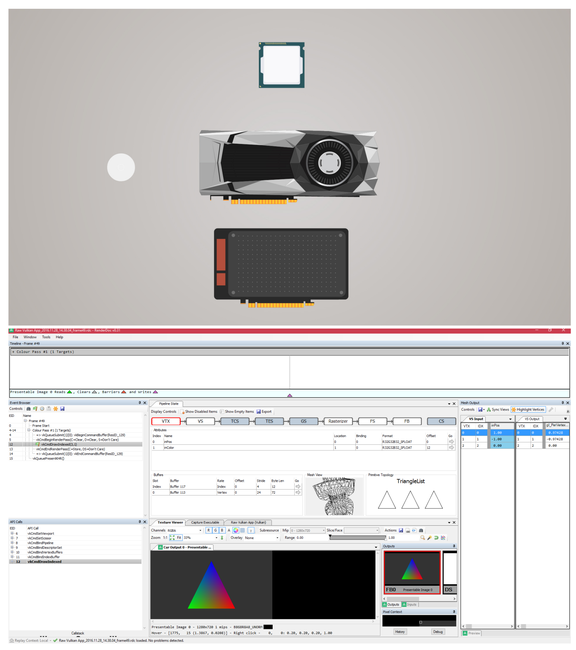
- React
- React Native
- Inferno
- Angular 2
- Unreal Engine 4
- Ember

Graphics API - Low level API designed to communicate directly with GPUs.

- OpenGL
- OpenCL
- WebGL
- DirectX
- Cuda
- Apple Metal

Results

- Obsidian is semantically equivalent to React, however for closer integration a "cppx" compiler would be nesseary, or integration with XAML similar to WinRT.cpp.
- Vulkan constructs are composed to build simple view components similar to HTML.
- Vulkan constructs can also be used to make game engine objects like environments and models.
- RenderDoc debugging proved essential to profiling performance and tracking errors.



Architecture

Package Management - Conan.io manages all our dependencies.

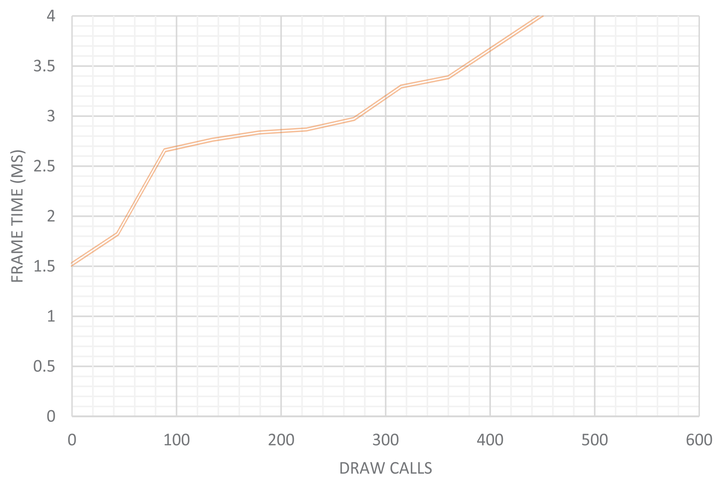
Template programming - For compile-time type safety , zero runtime, and variadic types.

Factory Functions - avoid new, allows custom allocation schemes.

Actor Model - Components are independent actors in an application.

```
|- docs/
|- src/
  |- core/
    |- component.hpp
    |- factory.hpp
    |- utils.hpp
  |- components/
    |- win32.hpp
    |- material.hpp
    |- mesh.hpp
    |- view.hpp
  |- config.hpp
  |- provider.hpp
  |- obsidian.hpp
|- tests/
|- conanfile.txt
|- readme.md
```

Benchmarks



Specs: Win 10 x64 | Nvidia 650M | i7 3610QM @ 2.3 Ghz | 16 GB DDR3 Ram

Future Work

- Shim that bridge Vulkan with other Graphics APIs.
- A C++ 17 `constexpr` shader string compiler as a part of the library.
- Cross platform compilation to all Vulkan OS targets.

SHIRAEF, J. 2016. An Exploratory Study of High Performance Graphics Application Programming Interfaces. University of Tennessee.

FISHER, B. 2015. Flux: A Unidirectional Data Flow Architecture for React Apps. Applicative 2015.