Subject: U++ Allocator & Vulkan
Posted by Xemuth on Sat, 21 Aug 2021 15:47:39 GMT
View Forum Message <> Reply to Message

I installed my Vulkan work on my Debian desktop and tried to compile the most basic code :

```
//#include <Core/Core.h>
#include <iostream>
#include <vulkan/vulkan.h>

int main(int argc, const char *argv[])
//CONSOLE_APP_MAIN
{
 VkInstance instance;
 VkApplicationInfo appInfo{};
    appInfo.sType = VK_STRUCTURE_TYPE_APPLICATION_INFO;
    appInfo.pApplicationName = "Hello Triangle";
    appInfo.applicationVersion = VK_MAKE_VERSION(1, 0, 0);
    appInfo.pEngineName = "No Engine";
    appInfo.engineVersion = VK_MAKE_VERSION(1, 0, 0);
    appInfo.apiVersion = VK_API_VERSION_1_0;

    VkInstanceCreateInfo createInfo{};
    createInfo.sType = VK_STRUCTURE_TYPE_INSTANCE_CREATE_INFO;
    createInfo.pApplicationInfo = &appInfo;
    createInfo.enabledExtensionCount = 0;
    createInfo.ppEnabledExtensionNames = nullptr;
    createInfo.enabledLayerCount = 0;

    if (vkCreateInstance(&createInfo, nullptr, &instance) != VK_SUCCESS) {
        #ifndef CORE_H
   std::cout << "Error\n";
        #else
   Upp::Cout() << "Error\n";
        #endif
    }else{
  vkDestroyInstance(instance, nullptr);
    }
}
```

A simple vulkan instance creation using default allocator. Here is my probblem:

If I include Core in my project without using USEMALLOC flag. This simple code result in memory leak :

with USEMALLOC (or removing Core package) no problem happen.

From my knowledge, Vulkan, in order to work properly must have allocator that's aligned on 4 octet. May it's the reason of this error ?

Subject: Re: U++ Allocator & Vulkan
Posted by mirek on Sun, 22 Aug 2021 07:37:12 GMT

View Forum Message <> Reply to Message

Xemuth wrote on Sat, 21 August 2021 17:47I installed my Vulkan work on my Debian desktop and tried to compile the most basic code :

```
//#include <Core/Core.h>
#include <iostream>
#include <vulkan/vulkan.h>

int main(int argc, const char *argv[])
//CONSOLE_APP_MAIN
{
 VkInstance instance;
 VkApplicationInfo appInfo{};
    appInfo.sType = VK_STRUCTURE_TYPE_APPLICATION_INFO;
    appInfo.pApplicationName = "Hello Triangle";
    appInfo.applicationVersion = VK_MAKE_VERSION(1, 0, 0);
    appInfo.pEngineName = "No Engine";
    appInfo.engineVersion = VK_MAKE_VERSION(1, 0, 0);
    appInfo.apiVersion = VK_API_VERSION_1_0;

    VkInstanceCreateInfo createInfo{};
    createInfo.sType = VK_STRUCTURE_TYPE_INSTANCE_CREATE_INFO;
    createInfo.pApplicationInfo = &appInfo;
    createInfo.enabledExtensionCount = 0;
    createInfo.ppEnabledExtensionNames = nullptr;
    createInfo.enabledLayerCount = 0;

    if (vkCreateInstance(&createInfo, nullptr, &instance) != VK_SUCCESS) {
        #ifndef CORE_H
   std::cout << "Error\n";
        #else
   Upp::Cout() << "Error\n";
        #endif
    }else{
  vkDestroyInstance(instance, nullptr);
    }
}
```

A simple vulkan instance creation using default allocator. Here is my probblem:

If I include Core in my project without using USEMALLOC flag. This simple code result in memory leak :

with USEMALLOC (or removing Core package) no problem happen.

From my knowledge, Vulkan, in order to work properly must have allocator that's aligned on 4 octet. May it's the reason of this error ?

U++ allocator aligns to 16 bytes. More likely Vulkan is calling new/delete and it is "system" new/delete - there is no solution to the problem AFAIK. Same issue we encountered on macos; partial solution is to use U++ allocator where U++ is using it directly (which is still plentiful) and use standard new/delete elsewhere. I think it is flagSTDNEWDELETE - check PLATFORM_MACOS.

Subject: Re: U++ Allocator & Vulkan
Posted by Xemuth on Sun, 22 Aug 2021 14:50:42 GMT
View Forum Message <> Reply to Message

mirek wrote on Sun, 22 August 2021 09:37

U++ allocator aligns to 16 bytes. More likely Vulkan is calling new/delete and it is "system" new/delete - there is no solution to the problem AFAIK. Same issue we encountered on macos; partial solution is to use U++ allocator where U++ is using it directly (which is still plentiful) and use standard new/delete elsewhere. I think it is flagSTDNEWDELETE - check PLATFORM_MACOS.

Hello Mirek, thanks for your time.
To prevent the fact Vulkan would use system new/delete instead of U++ one. I did a Custom memory allocator for Vulkan (Vulkan allow you to set up some callback to execute in order to allocate memory instead of using Vulkan default way.

Here is my 2 memory functions :

```
void* UVkCustomAllocator::DefaultAllocation(size_t size, size_t alignement,
VkSystemAllocationScope allocationScope){
 void* ptr;
 if(alignement < size){
  ptr = malloc(size);
  ASSERT_(ptr, "Error during malloc for size of " +  AsString(size));
 }else{
  int rc = posix_memalign(&ptr, alignement, size);
  ASSERT_(rc == 0,"Error code : "+ AsString(rc) +" during posix_memalign for size of " +
AsString(size) + " and alignement of " + AsString(alignement));
 }
    return ptr;
}

void  UVkCustomAllocator::DefaultFree(void* pMemory){
 if(pMemory){
  return free(pMemory);
 }
}
```

Thoses functions are really simple and is based on malloc and posix_memalign. Right after implementing this (Test with some LOG), all my Vulkan call to allocate some memory use boths functions.
After tracking each malloc and free call, all the memory allocated by vulkan are correctly freed. So I don't understand why I still have some memory leaks when Vulkan don't use U++ allocator ? (or maybe I think he don't use it anymore)

PS: Moreover, when I launch my test program on Windows, (without USEMALLOC flag) I don't have any memory leak (CLANG / MVSC)
PS2: Using USEMALLOC or STD_NEWDELETE work fine even without custom memory allocator