Vulkan C++ SDK

About the SDK
- Open-source project
- Works with standard C api
- Included in LunarG Vulkan SDK
- #include <vulkan/vulkan.hpp>
- Available at https://github.com/KhronosGroup/Vulkan-Hpp

Goal of the API
- Add type-safety to enums and bitfields
- STL container support
- Exceptions
- Simple Enumerations

General Syntax
- Camel Case
- Put in vk namespace with all vk/Vk/VK_prefixes removed
  - vkCommandBindPipeline becomes vk::commandBindPipeline
- e prefix for enums
  - vk::ImageType::e2D
- No more _BIT suffix
  - vk::ImageUsage::eColorAttachment

The ‘e’ Prefix
- Some start with a digit
  - vk::ImageType::2D causes compile error
  - With added prefix vk::ImageType::e2D
- Some conflict with precomplier defines
  - vk::CompositeAlphaFlagBitsKHR::OPAQUE clashes with OPAQUE from win32gdi.h
  - Add prefix for vk::CompositeAlphaFlagBitsKHR::eOpaque

Bit Flag Changes
- No implicit cast to integer types
- vk::Flags template for all flags
  - Operator overload on &, |, &=, |=
- Impossible to set bits not part of the enums used
  - ci.usage = vk::ImageUsage::eColorAttachment | vk::ImageUsage::eStorage;
Struct Constructors

- Added constructors for every struct
  - Default constructor setting everything to 0
  - sType initialized internally
  - Always correct
- Allows for named constructor idiom usage to set values

Example Constructor Usages

```cpp
vk::ImageCreateInfo ci(some flags, vk::ImageType::e2D, vk::Format::eR8G8B8A8Unorm, vk::Extent3D{width, height, 1}, 1, 1, vk::SampleCount::e1, vk::ImageTiling::eOptimal, vk::ImageUsage::eColorAttachment, vk::SharingMode::eExclusive, 0, 0, vk::ImageLayout::eUndefined);
```

std::string and std::vector

- Uses std::string instead of const char* for strings
- Uses std::vector instead of (count, ptr) for sized arrays

Passing Structs to Functions

- Pass temporary struct

```c
// C
ImageSubResource subresource;
subresource.aspectMask = 0;
subresource.mipLevel = 0;
vkGetImageSubresourceLayout(image, subresource);
```

```cpp
auto layout = device.getImageSubResourceLayout(image, { /* flags */, 0 /* miplevel */, 0 /* layout */ });
```

Return Values, Exceptions

- Checks return value of Vk::Result
  - Throws std::runtime_error if failure
- Return is now the Vulkan handle
- Create device
  - Device device = physicalDevice.createDevice(createInfo);
```c
VkImageCreateInfo ci;
    ci.sType = VK_STRUCTURE_TYPE_IMAGE_CREATE_INFO;
    ci.pNext = nullptr;
    ci.flags = ... some flags...;
    ci.imageType = VK_IMAGE_TYPE_2D;
    ci.format = VK_FORMAT_R8G8B8A8_UNORM;
    ci.extent = VkExtent3D { width, height, 1 };
    ci.mipLevels = 1;
    ci.arrayLayers = 1;
    ci.samples = VK_SAMPLE_COUNT_1_BIT;
    ci.tiling = VK_IMAGE_TILING_OPTIMAL;
    ci.usage = VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT;
    ci.sharingMode = VK_SHARING_MODE_EXCLUSIVE;
    ci.queueFamilyIndexCount = 0;
    ci.pQueueFamilyIndices = 0;
    ci.initialLayout = VK_IMAGE_LAYOUT_UNDEFINED;
    vkCreateImage(device, &ci, allocator, &image));
```