# Rendering Engine

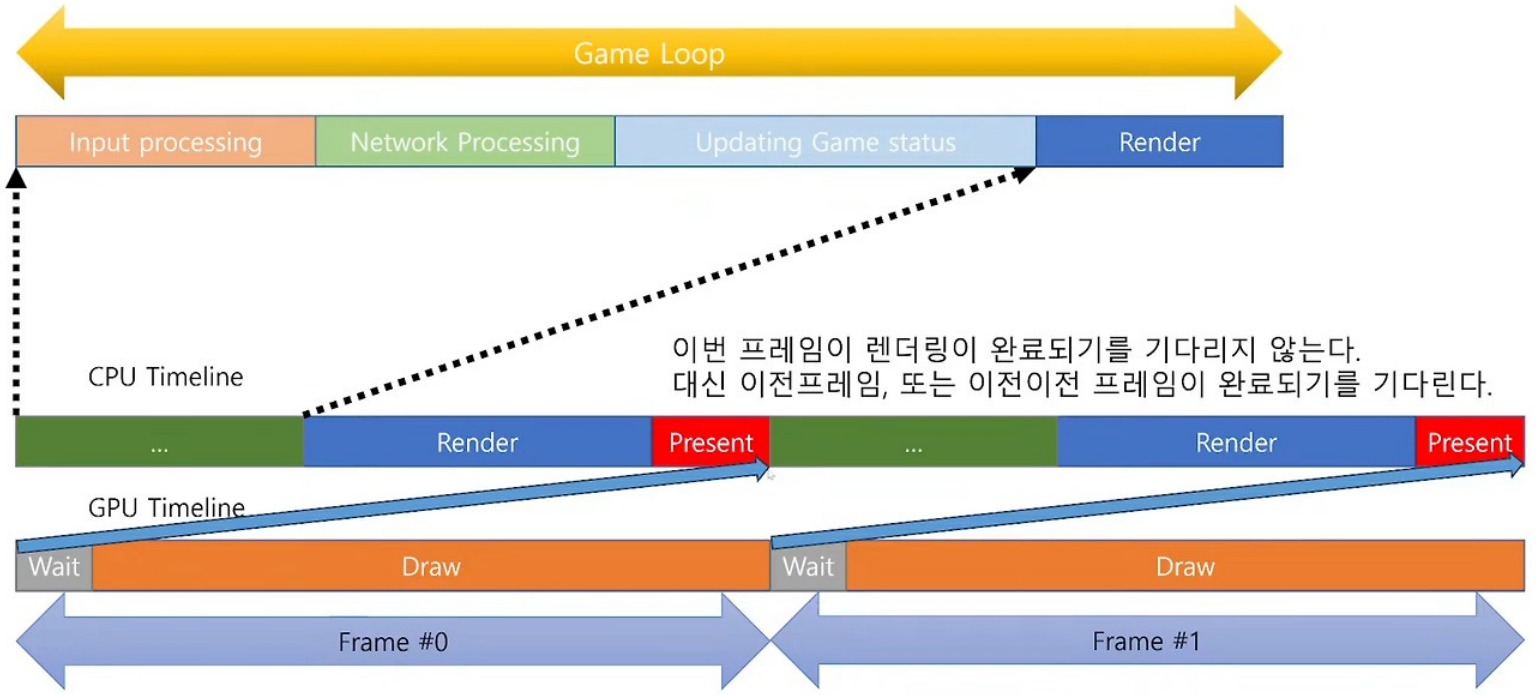
: 24101515 이재훈

목표: 3D Rendering Program

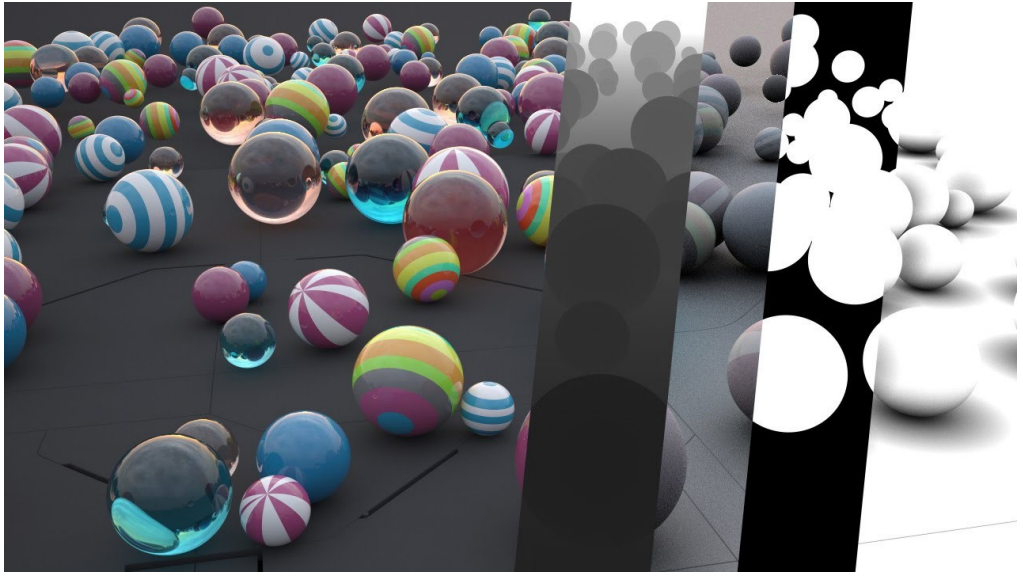
문제의식:

# Modern Asyn Render

게임 루프 내에서의 timeline – 비동기식 중첩 렌더링

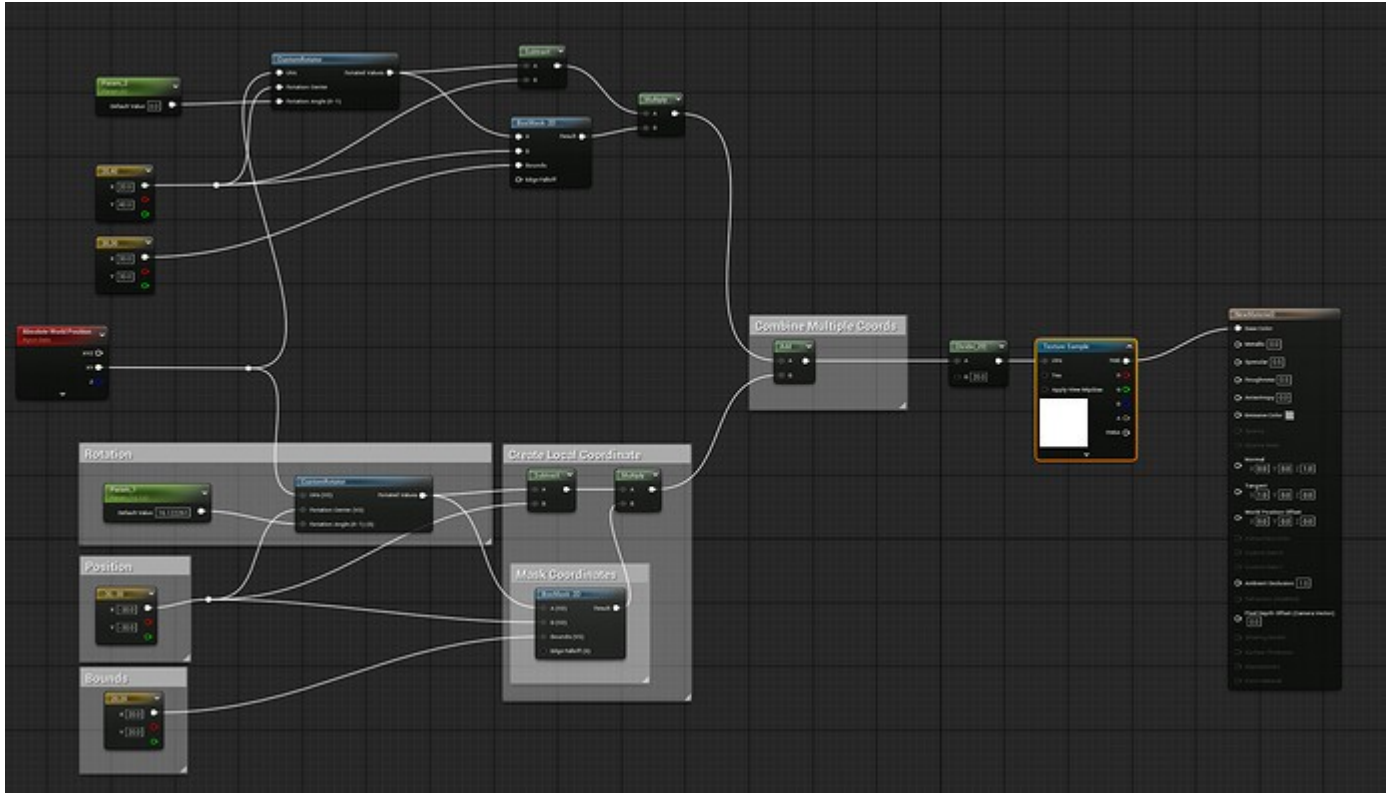


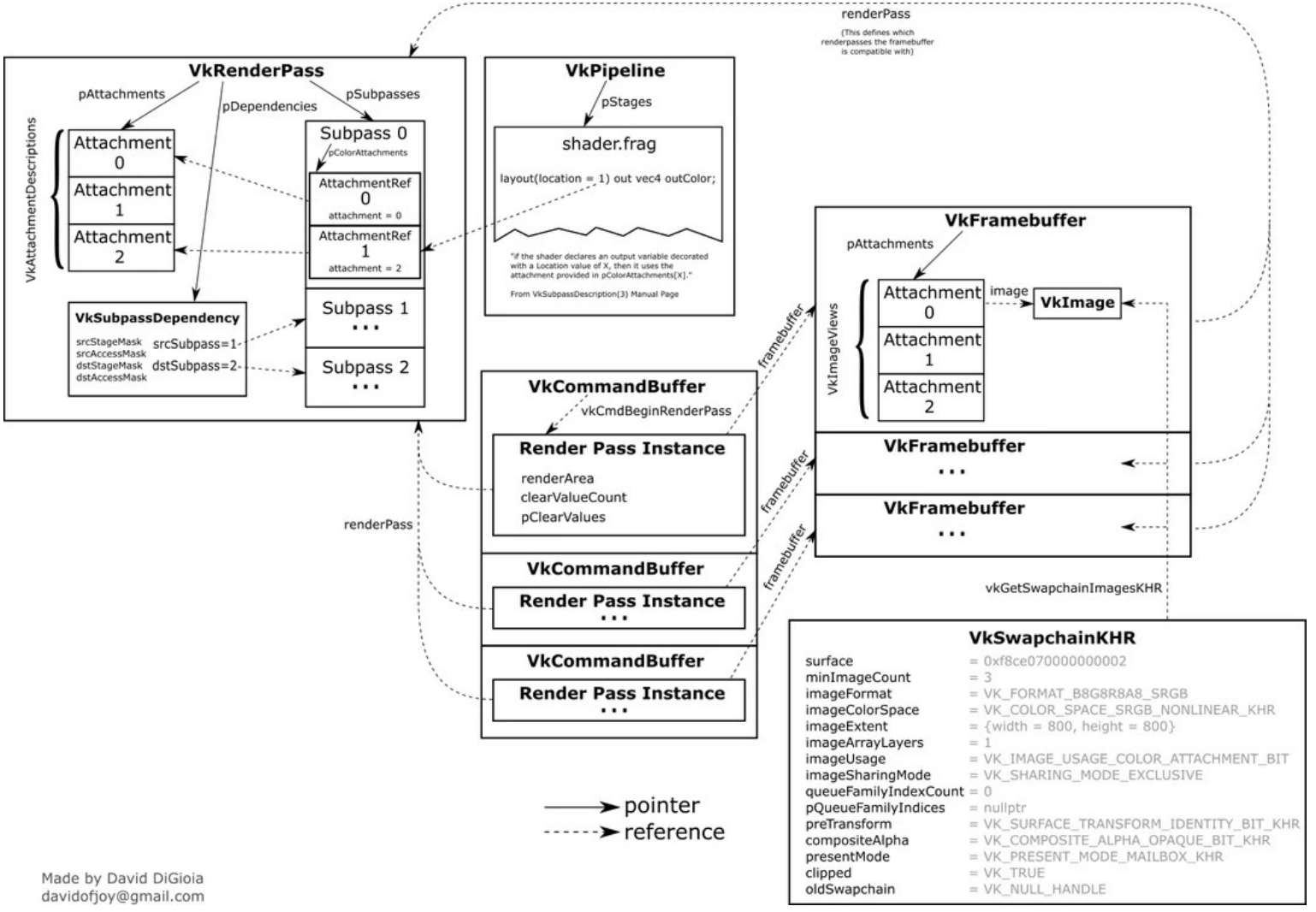
# Multi Pass Rendering



# Deffered Rendering

# Rendering Graph






Made by David DiGioia
davidofjoy@gmail.com

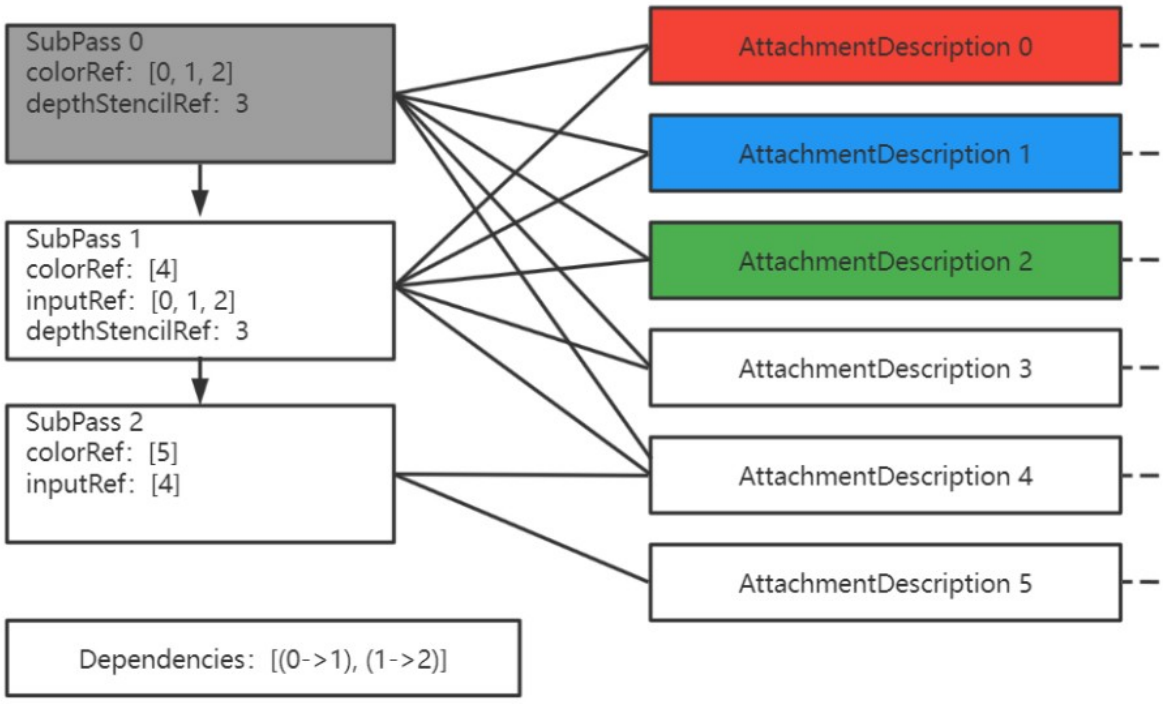
# Dynamic Rendering :

- Dynamic Render:

  Very Simple:

  just Render!

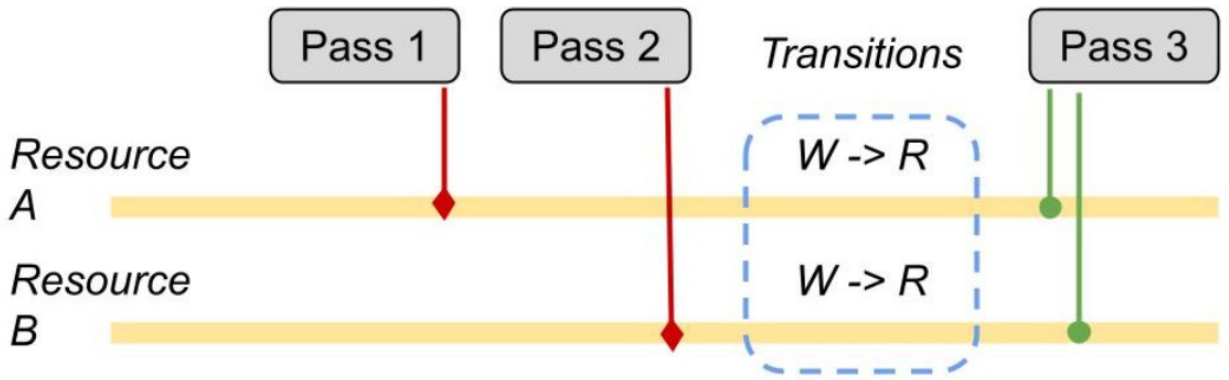
- Sync Need!!



# Last writer Tracking:

- For resource Sync, insert barrer

  - W -> W



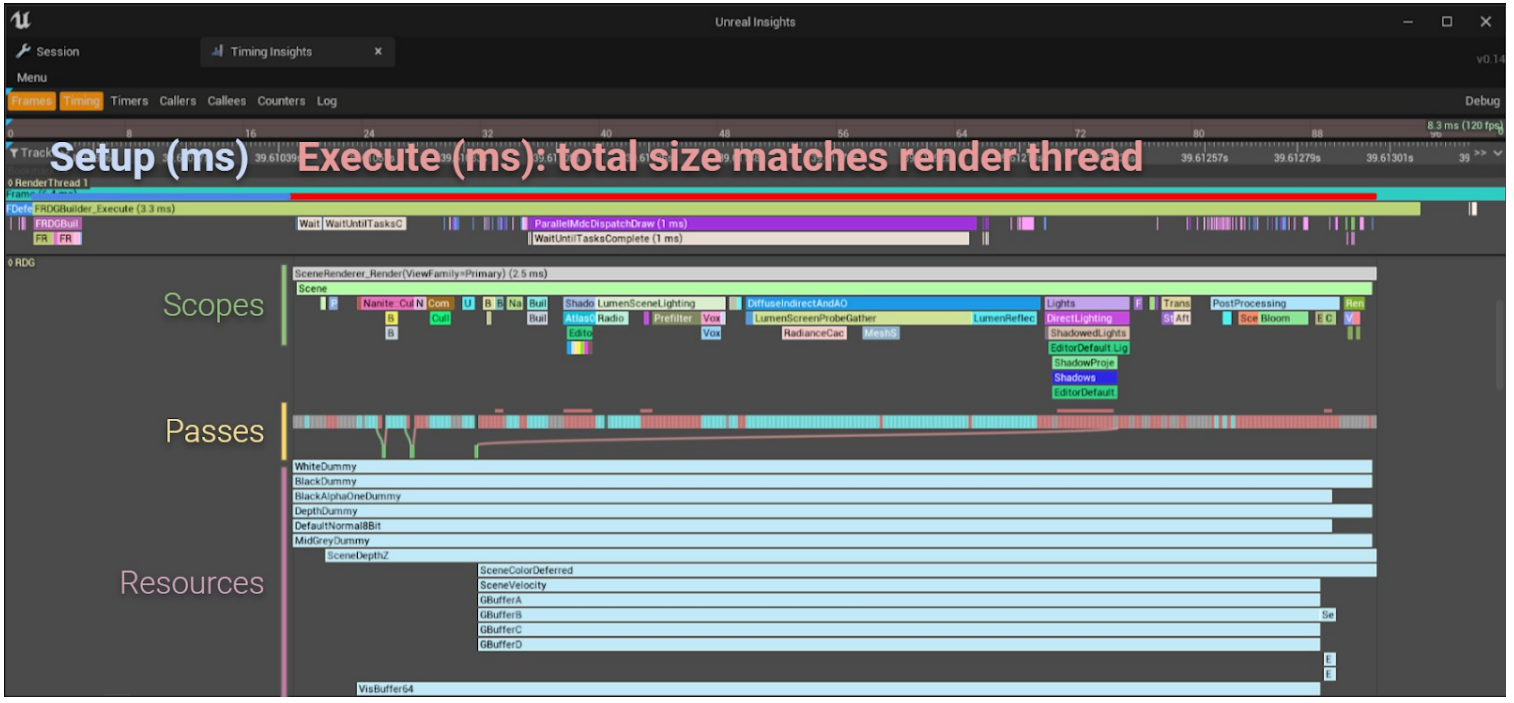
# Resource Tracking

- For W -> R
- cashing resource State :

  insert Barrier

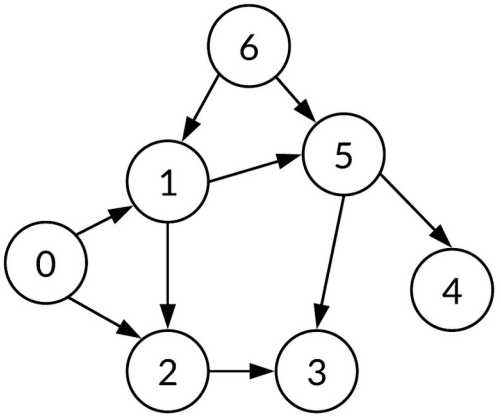
```
uint32_t currentPipeline__        = VK_PIPELINE_STAGE_TRANSFER_BIT;
uint32_t writePipeline__          = VK_PIPELINE_STAGE_TOP_OF_PIPE_BIT;
VkAccessFlags writeAccessMask__   = VK_ACCESS_MEMORY_READ_BIT;
VkAccessFlags currentAccessMask__ = VK_ACCESS_NONE;
```

# Rendering Graph propiling on UE

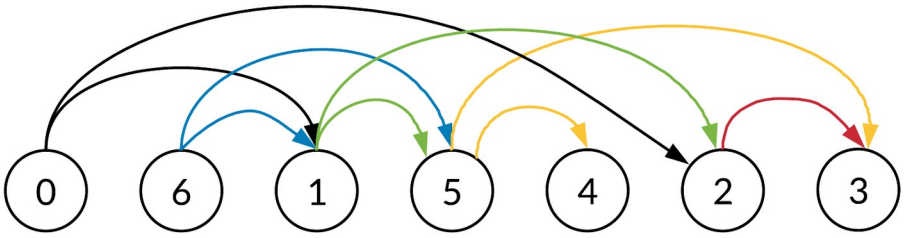


# Multi Pass Rendering

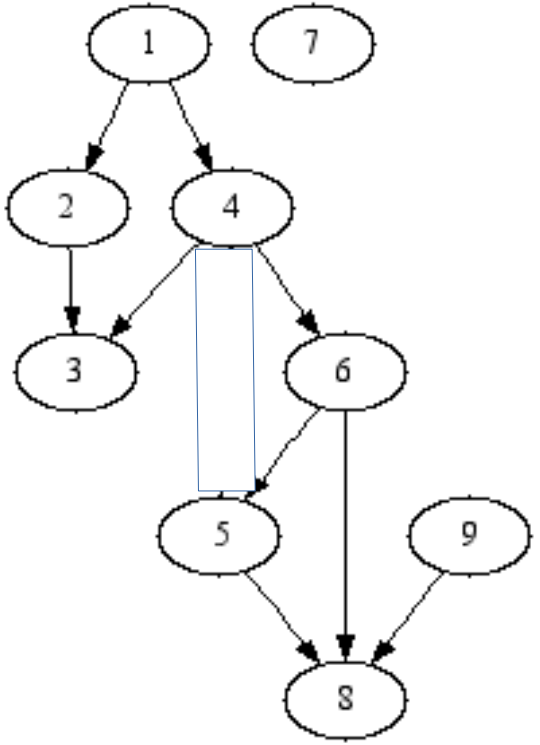
Unsorted graph




Topologically
sorted graph

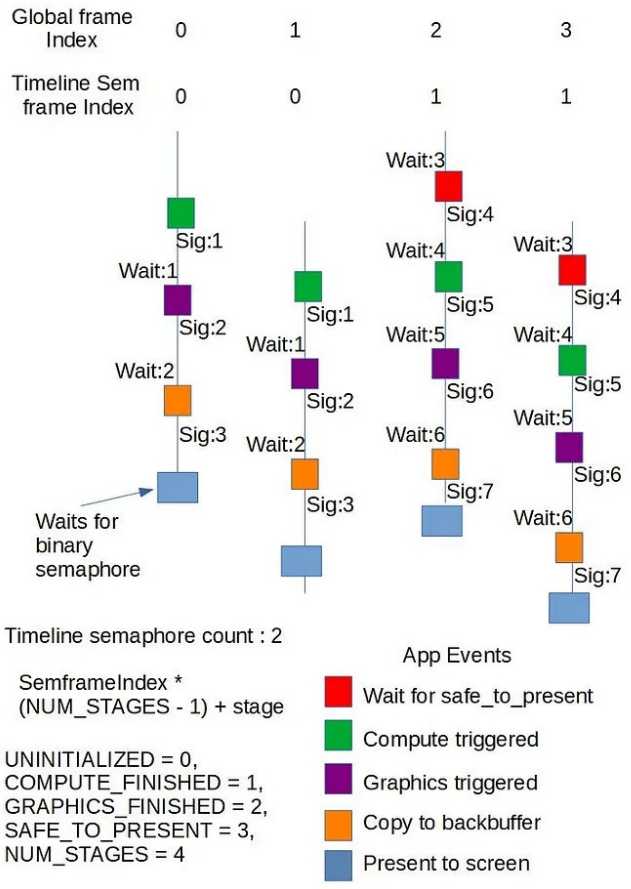


# Pass Level:




PASS level :
ordered
by
dependency

# Timeline semaphore




Sync parrel recored
CB with semaphore

# Memory Optimization

Logical Resource is not Physical Resource

Logical Resource is consisted with

1. Physical Resource
2. Meta data
3. Handler

# Memory Optimization

Find memory :
if exist Desired memory ,
get memory handler

```
Allocation MemoryAllocator::allocate(VkMemoryRequirements requirements,
                                     VkMemoryPropertyFlags desiredFlags,
                                     const std::string &debugName)
{
  uint32_t memoryType = findMemoryType( typeFilter: requirements.memoryTypeBits,  properties: desiredFlags);
  for (auto *pool : MemoryPool * : pools)
  {
    if (pool->getMemoryTypeIndex() == memoryType)
    {
      Allocation result;
      if (pool->allocate(requirements.size, requirements.alignment,  &: result))
      {
        return result;
      }
    }
  }
  // 256MB -> minmum chunk
  VkDeviceSize poolSize = std::max(requirements.size * 8, (VkDeviceSize) 256 * 1024 * 1024);
  MemoryPool *newPool   = new MemoryPool(devic                                    pe,  size: poolSize);
  pools.push_back(newPool);
  Allocation result;
  if (!newPool->allocate(requirements.size, requirements.alignment,  &: result))
  {
    throw std::runtime_error("Failed to allocate memory from new pool");
```

다음에서 선언됨: memory_allocator.cpp
VkMemoryRequirements requirements

# Friend Memory

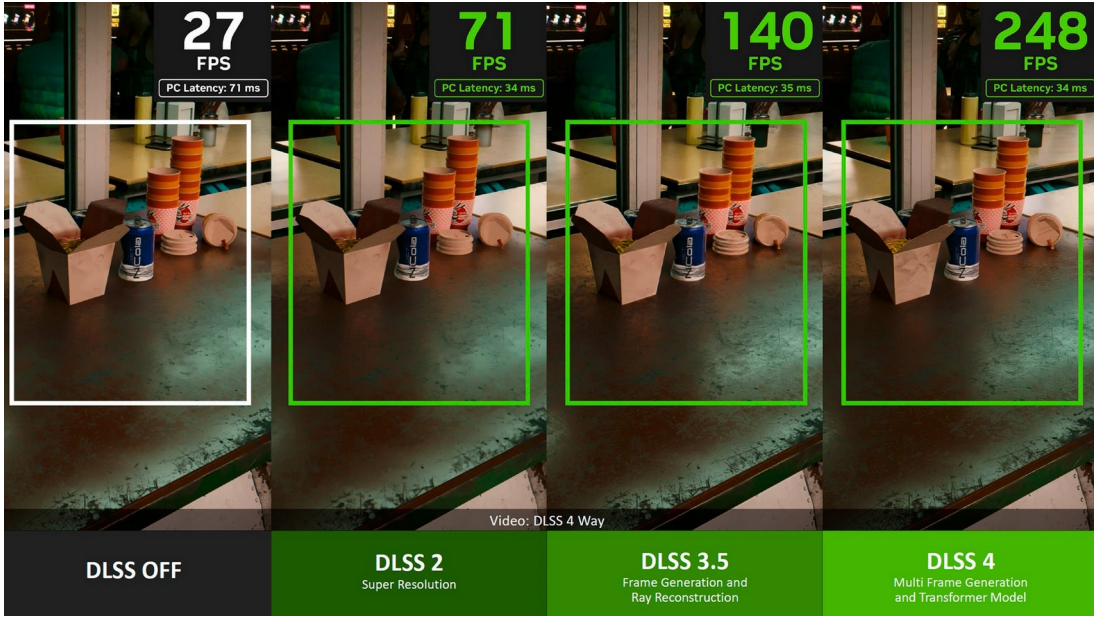
Tracking Frame Resoure lifetime
if Memory can be reuse,
Make Frame A and Frame B Friend!
Resource A and B use
same Allocation

# NV DLSS

- Tensor CORE !
- DLSS use

  AI Inference
- FRAME LOOP DEV UP!

- Using DLSS up scaling
- insert inferenced frame



# 추가할 Feature(해커톤 내 못하는 것)

- sculpting engine
- Tensor graph builder  and compute graph renderer
- Stable renderer
- 2D V-Tuber engine :
- material graph
- auto retopology
- Material Draw
- mesh reconstruction
- Render and Compute Farm with 313 Room
- 감사합니다.