Эффективное использование GPU на примере геймдева

Ростислав Михеев

Elverils

@hacenator

rmikheev@elverils.com

C++ Russia 2023

О нас

Эффективное GPU

Использование ресурсов:

- памяти
- времени исполнения

Архитектура GPU

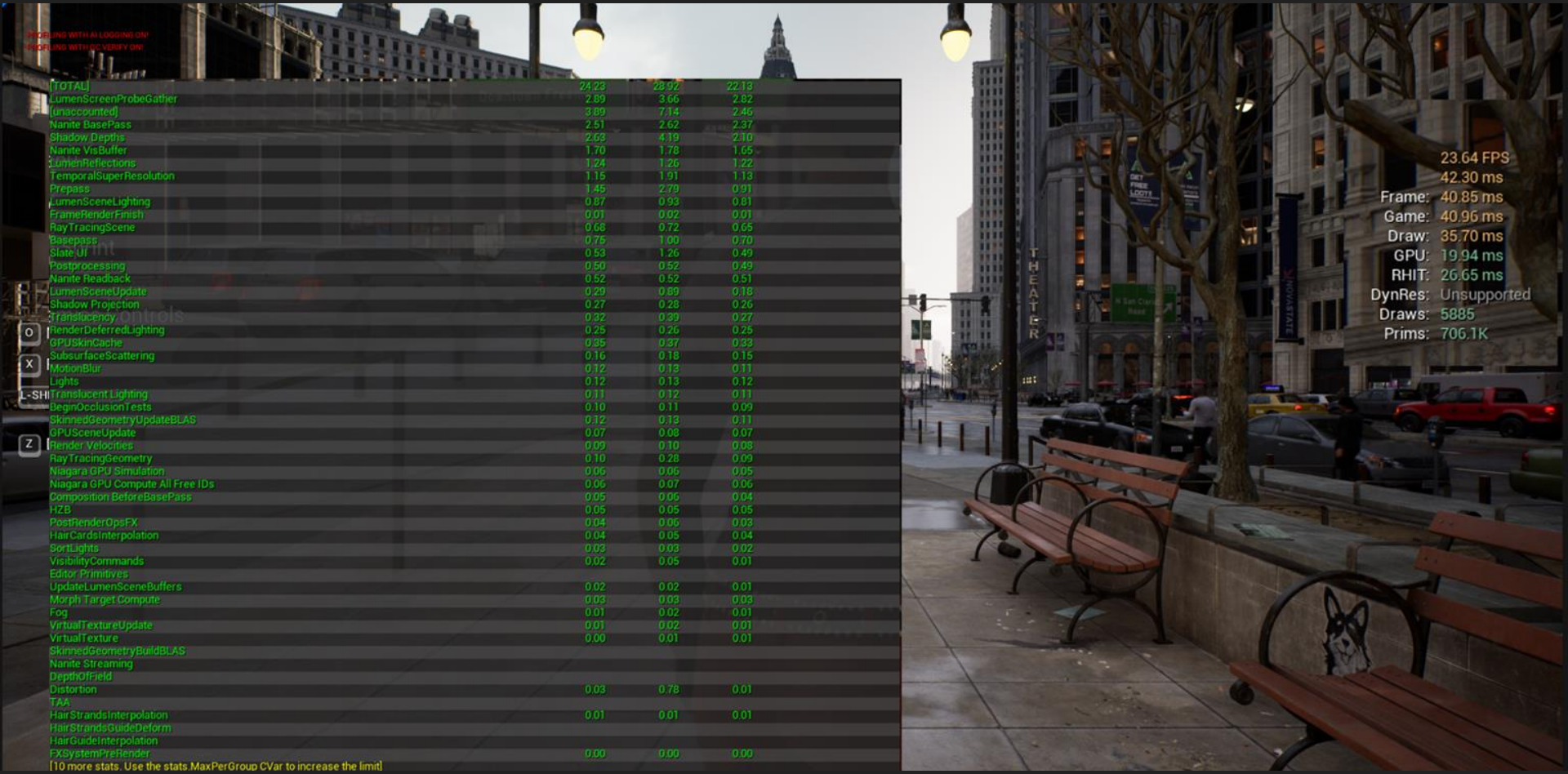
RTX 4090

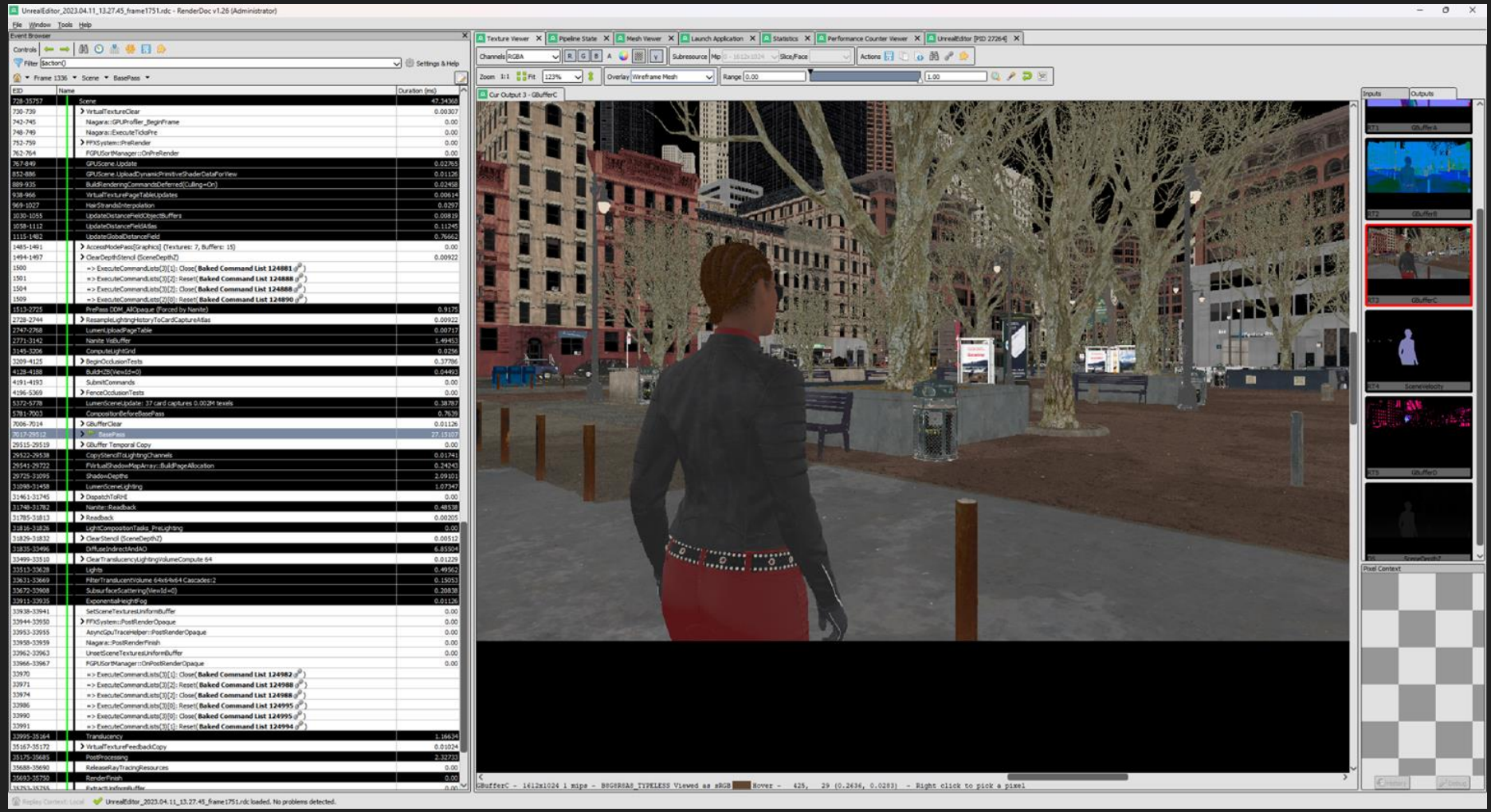
16384 cuda cores

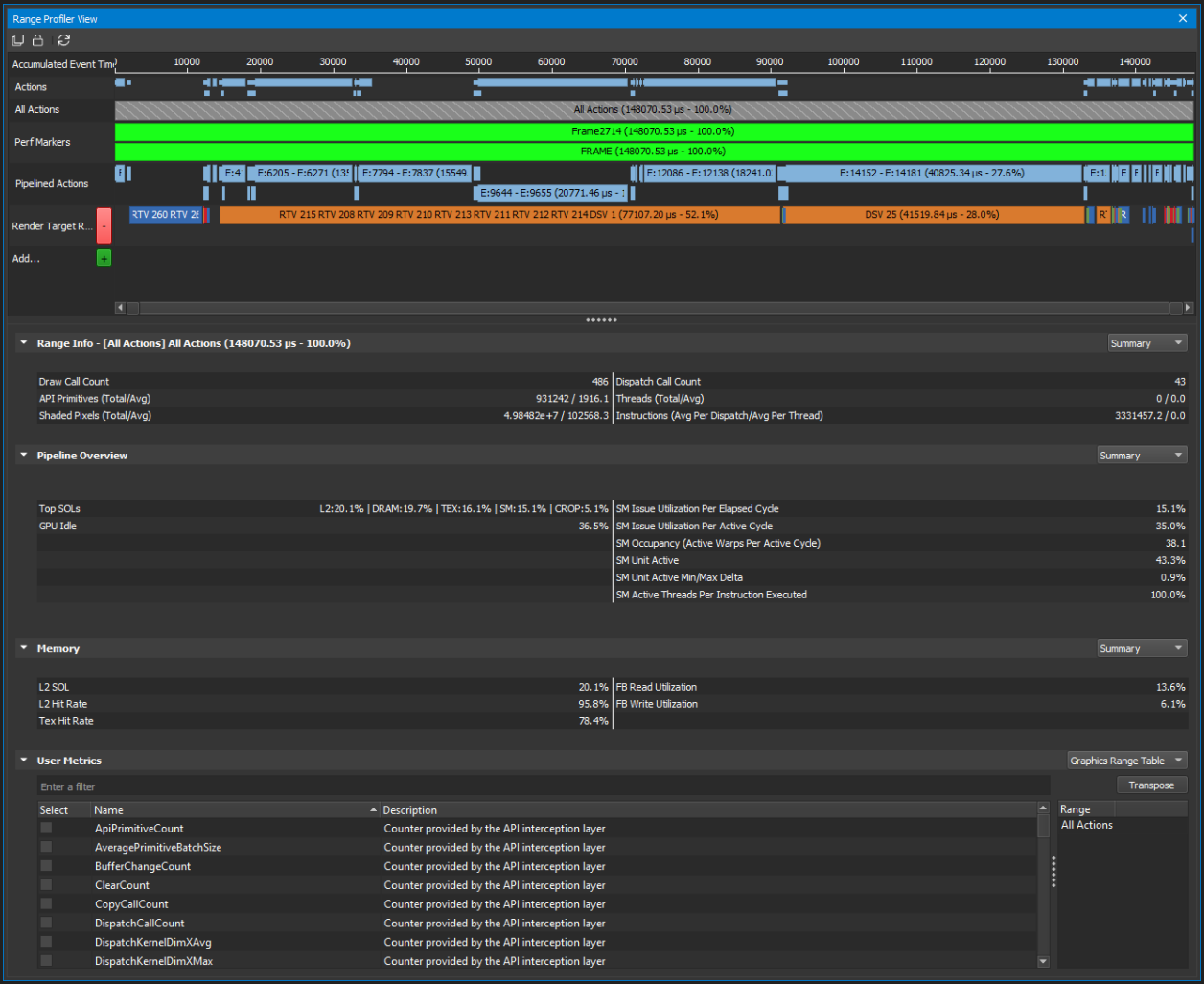
- GPU это монстр с тысячами ядер
- Single Instruction Multiple Threads (SIMT)
- Различные типы памяти (registers, shared memory, local memory, L1 & L2 кеши, global memory)
- Вспомогательные блоки и ядра (Texture Processing, RT cores, Tensor cores, Rasterization)
- GPC <- TPC <- SM <- CUDA cores

Инструменты анализ производительности

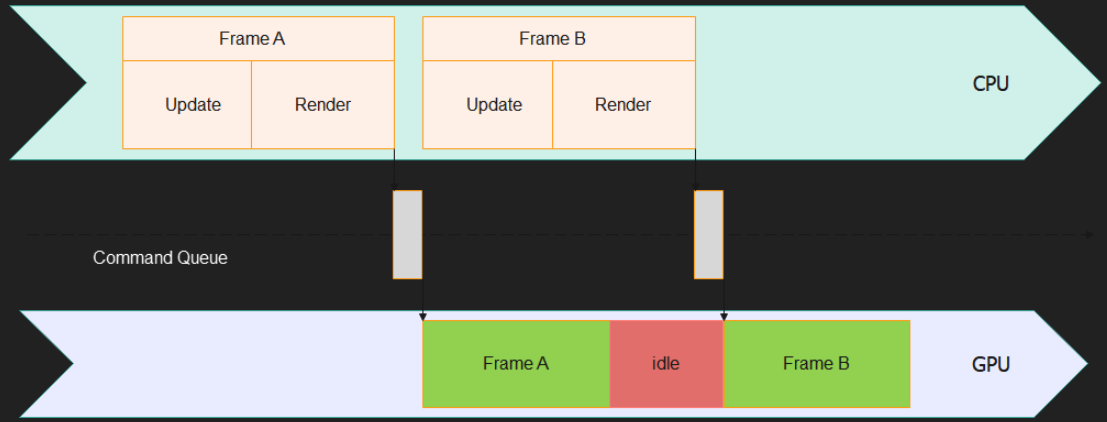
- Рантайм статистика: fps, ms на весь/часть кадра
- Снимок кадра: последовательность команд
- Анализ синхронизации GPU/CPU: выявление простоя (дыр) на GPU вследствие неправильного передачи команд (partial submit, threshold); анализ readback
- Профилирование вызовов отрисовки и шейдеров


PROFILING WITH AI LOGGING ON!
PROFILING WITH GC VERIFY ON!
[TOTAL] 24.23 28.92 22.13
LumenScreenProbeGather 2.89 3.66 2.82
[unaccounted] 3.89 7.14 2.46
Nanite BasePass 2.51 2.62 2.37
Shadow Depths 2.63 4.19 2.10
Nanite VisBuffer 1.70 1.78 1.65
LumenReflections 1.24 1.26 1.22
TemporalSuperResolution 1.15 1.91 1.13
Prepass 1.45 2.79 0.91
LumenSceneLighting 0.87 0.93 0.81
FrameRenderFinish 0.01 0.02 0.01
RayTracingScene 0.68 0.72 0.65
Basepass 0.75 1.00 0.70
Slate UI 0.53 1.26 0.49
Postprocessing 0.50 0.52 0.49
Nanite Readback 0.52 0.52 0.51
LumenSceneUpdate 0.29 0.89 0.18
Shadow Projection 0.27 0.28 0.26
Translucency 0.32 0.39 0.27
RenderDeferredLighting 0.25 0.26 0.25
GPUSkinCache 0.35 0.37 0.33
SubsurfaceScattering 0.16 0.18 0.15
MotionBlur 0.12 0.13 0.11
Lights 0.12 0.13 0.12
Translucent Lighting 0.11 0.12 0.11
BeginOcclusionTests 0.10 0.11 0.09
SkinnedGeometryUpdateBLAS 0.12 0.13 0.11
GPUSceneUpdate 0.07 0.08 0.07
Render Velocities 0.09 0.10 0.08
RayTracingGeometry 0.10 0.28 0.09
Niagara GPU Simulation 0.06 0.06 0.05
Niagara GPU Compute All Free IDs 0.06 0.07 0.06
Composition BeforeBasePass 0.05 0.06 0.04
HZB 0.05 0.05 0.05
PostRenderOpsFX 0.04 0.06 0.03
HairCardsInterpolation 0.04 0.05 0.04
SortLights 0.03 0.03 0.02
VisibilityCommands 0.02 0.05 0.01
Editor Primitives
UpdateLumenSceneBuffers 0.02 0.02 0.01
Morph Target Compute 0.03 0.03 0.03
Fog 0.01 0.02 0.01
VirtualTextureUpdate 0.01 0.02 0.01
VirtualTexture 0.00 0.01 0.01
SkinnedGeometryBuildBLAS
Nanite Streaming
DepthOfField
Distortion 0.03 0.78 0.01
TAA
HairStrandsInterpolation 0.01 0.01 0.01
HairStrandsGuideDeform
HairGuideInterpolation
FXSystemPreRender 0.00 0.00 0.00
[10 more stats. Use the stats.MaxPerGroup CVar to increase the limit]
23.64 FPS
42.30 ms
Frame: 40.85 ms
Game: 40.96 ms
Draw: 35.70 ms
GPU: 19.94 ms
RHIT: 26.65 ms
DynRes: Unsupported
Draws: 5885
Prims: 706.1K


UnrealEditor_2023.04.11_13.27.45_frame1751.rdc - RenderDoc v1.26 (Administrator)
File Window Tools Help
Event Browser
Controls
Filter
Settings & Help
Frame 1336
Scene
BasePass
EID
Name
Duration (ms)
728-35757 Scene 47.54368
730-739 VirtualTextureClear 0.00307
742-745 Niagara::GPUProfiler_BeginFrame 0.00
748-749 Niagara::ExecuteTicksPre 0.00
752-759 FFXSystem::PreRender 0.00
762-764 FGPUSortManager::OnPreRender 0.00
767-849 GPUScene.Update 0.02763
852-886 GPUScene.UploadDynamicPrimitiveShaderDataForView 0.01126
889-935 BuildRenderingCommandsDeferred(Culling=On) 0.02458
938-966 VirtualTexturePageTableUpdates 0.00614
969-1027 HairStrandsInterpolation 0.0297
1030-1055 UpdateDistanceFieldObjectBuffers 0.00819
1058-1112 UpdateDistanceFieldAtlas 0.11245
1115-1482 UpdateGlobalDistanceField 0.76662
1485-1491 AccessModePass[Graphics] (Textures: 7, Buffers: 15) 0.00
1494-1497 ClearDepthStencil (SceneDepthZ) 0.00922
1500 => ExecuteCommandLists(3)[1]: Close(Baked Command List 124881)
1501 => ExecuteCommandLists(3)[2]: Reset(Baked Command List 124888)
1504 => ExecuteCommandLists(3)[2]: Close(Baked Command List 124888)
1509 => ExecuteCommandLists(2)[0]: Reset(Baked Command List 124890)
2513-2725 PrePass DDM_AllOpaque (Forced by Nanite) 0.9175
2728-2744 ResampleLightingHistoryToCardCaptureAtlas 0.00922
2747-2768 LumenUploadPageTable 0.00717
2771-3142 Nanite VisBuffer 1.49453
3145-3206 ComputeLightGrid 0.0256
3209-4125 BeginOcclusionTests 0.37786
4128-4188 Build<ZB(ViewId=0) 0.04493
4191-4193 SubmitCommands 0.00
4196-5369 FenceOcclusionTests 0.00
5372-5778 LumenSceneUpdate: 37 card captures 0.002M texels 0.38787
5781-7003 CompositionBeforeBasePass 0.7639
7006-7014 GBufferClear 0.01126
7017-29512 BasePass 27.15107
29515-29519 GBuffer Temporal Copy 0.00
29522-29538 CopyStencilToLightingChannels 0.01741
29541-29722 FVirtualShadowMapArray::BuildPageAllocation 0.24243
29725-31095 ShadowDepths 2.09101
31098-31458 LumenSceneLighting 1.07347
31461-31745 DispatchToRHI 0.00
31748-31782 Nanite::Readback 0.48538
31785-31813 Readback 0.00205
31816-31826 LightCompositionTasks_PreLighting 0.00
31829-31832 ClearStencil (SceneDepthZ) 0.00512
31835-33496 DiffuseIndirectAndAO 6.85504
33499-33510 ClearTranslucencyLightingVolumeCompute 64 0.01229
33513-33628 Lights 0.49562
33631-33669 FilterTranslucentVolume 64x64x64 Cascades:2 0.15053
33672-33908 SubsurfaceScattering(ViewId=0) 0.20838
33911-33935 ExponentialHeightFog 0.01126
33938-33941 SetSceneTexturesUniformBuffer 0.00
33944-33950 FFXSystem::PostRenderOpaque 0.00
33953-33955 AsyncGpuTraceHelper::PostRenderOpaque 0.00
33958-33959 Niagara::PostRenderFinish 0.00
33962-33963 UnsetSceneTexturesUniformBuffer 0.00
33966-33967 FGPUSortManager::OnPostRenderOpaque 0.00
33970 => ExecuteCommandLists(3)[1]: Close(Baked Command List 124982)
33971 => ExecuteCommandLists(3)[2]: Reset(Baked Command List 124988)
33974 => ExecuteCommandLists(3)[2]: Close(Baked Command List 124988)
33986 => ExecuteCommandLists(3)[0]: Reset(Baked Command List 124995)
33990 => ExecuteCommandLists(3)[0]: Close(Baked Command List 124995)
33991 => ExecuteCommandLists(3)[1]: Reset(Baked Command List 124994)
33995-35264 Translucency 1.16634
35167-35172 VirtualTextureFeedbackCopy 0.01024
35175-35683 PostProcessing 2.32733
35683-35690 ReleaseRayTracingResources 0.00
35693-35755 RenderFinish 0.00
Texture Viewer
Pipeline State
Mesh Viewer
Launch Application
Statistics
Performance Counter Viewer
UnrealEditor [PID 27264]
Channels RGBA
Subresource
Slice/Face
Actions
Zoom 1:1 Fit 123%
Overlay Wireframe Mesh
Range 0.00 1.00
Cur Output 3 - GBufferC
Inputs
Outputs
Pixel Context
GBufferC - 1612x1024 1 mips - B8G8R8A8_TYPELESS Viewed as sRGB Hover - 425, 29 (0.2636, 0.0283) - Right click to pick a pixel
UnrealEditor_2023.04.11_13.27.45_frame1751.rdc loaded. No problems detected.

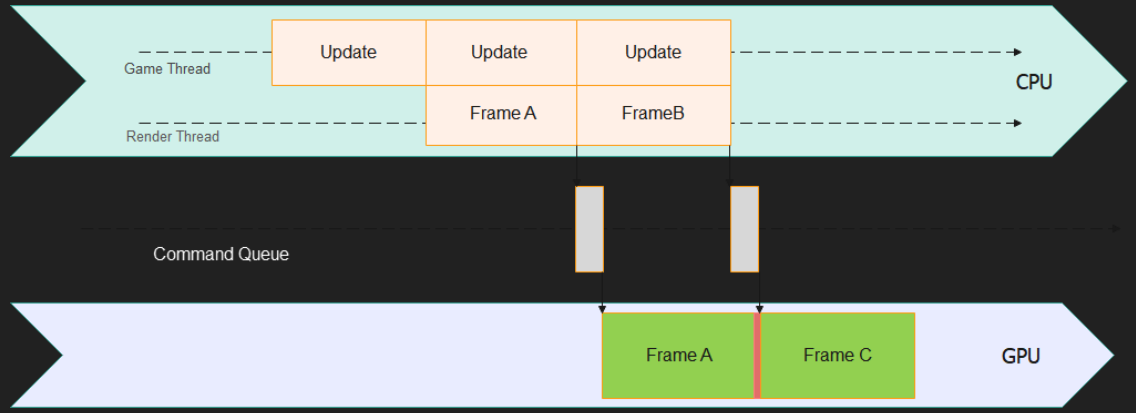

Range Profiler View
Accumulated Event Time
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
110000
120000
130000
140000
Actions
All Actions
All Actions (148070.53 µs - 100.0%)
Perf Markers
Frame2714 (148070.53 µs - 100.0%)
FRAME (148070.53 µs - 100.0%)
Pipelined Actions
E:6205 - E:6271 (13
E:7794 - E:7837 (15549
E:12086 - E:12138 (18241.0
E:14152 - E:14181 (40825.34 µs - 27.6%)
E:9644 - E:9655 (20771.46 µs -
Render Target R...
RTV 260 RTV 2
RTV 215 RTV 208 RTV 209 RTV 210 RTV 213 RTV 211 RTV 212 RTV 214 DSV 1 (77107.20 µs - 52.1%)
DSV 25 (41519.84 µs - 28.0%)
Add...
Range Info - [All Actions] All Actions (148070.53 µs - 100.0%)
Summary
Draw Call Count 486
API Primitives (Total/Avg) 931242 / 1916.1
Shaded Pixels (Total/Avg) 4.98482e+7 / 102568.3
Dispatch Call Count 43
Threads (Total/Avg) 0 / 0.0
Instructions (Avg Per Dispatch/Avg Per Thread) 3331457.2 / 0.0
Pipeline Overview
Summary
Top SOLs L2:20.1% | DRAM:19.7% | TEX:16.1% | SM:15.1% | CROP:5.1%
GPU Idle 36.5%
SM Issue Utilization Per Elapsed Cycle 15.1%
SM Issue Utilization Per Active Cycle 35.0%
SM Occupancy (Active Warps Per Active Cycle) 38.1
SM Unit Active 43.3%
SM Unit Active Min/Max Delta 0.9%
SM Active Threads Per Instruction Executed 100.0%
Memory
Summary
L2 SOL 20.1%
L2 Hit Rate 95.8%
Tex Hit Rate 78.4%
FB Read Utilization 13.6%
FB Write Utilization 6.1%
User Metrics
Graphics Range Table
Enter a filter
Transpose
Select
Name
Description
ApiPrimitiveCount Counter provided by the API interception layer
AveragePrimitiveBatchSize Counter provided by the API interception layer
BufferChangeCount Counter provided by the API interception layer
ClearCount Counter provided by the API interception layer
CopyCallCount Counter provided by the API interception layer
DispatchCallCount Counter provided by the API interception layer
DispatchKernelDimXAvg Counter provided by the API interception layer
DispatchKernelDimXMax Counter provided by the API interception layer
Range
All Actions

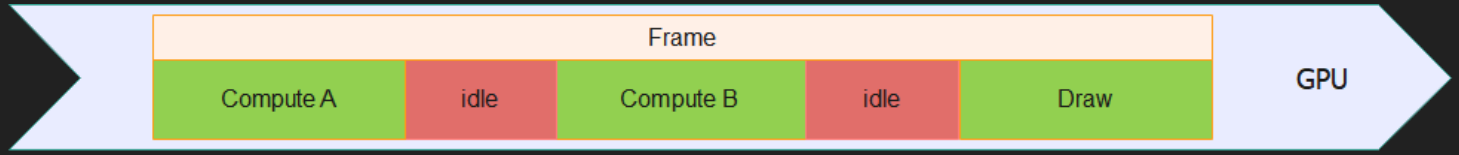
GPU idles



- Render Thread
- Избегать readback в том же кадре

GPU idles

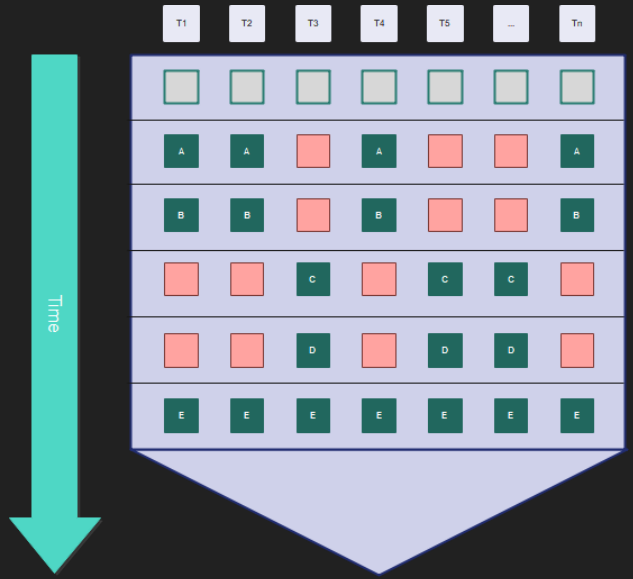

Update
Update
Update
Game Thread
CPU
Frame A
FrameB
Render Thread
Command Queue
Frame A
Frame C
GPU


Frame
Compute A
idle
Compute B
idle
Draw
GPU

Ветвление на GPU

Неправильное или чрезмерное использование условий внутри шейдеров приводит к снижению производительности

```
if (rand() % 2) {
  A();
  B();
} else {
  C();
  D();
}
E();
```



Оптимизация ветвления

- Выносить общие куски за пределы блоков условий
- Заменять ветвление на арифметические операции (умножение, saturate, step)

```
float x = rand() % 2;
A()*x+B()*x+C()*!x+D()*!x;
E();
```

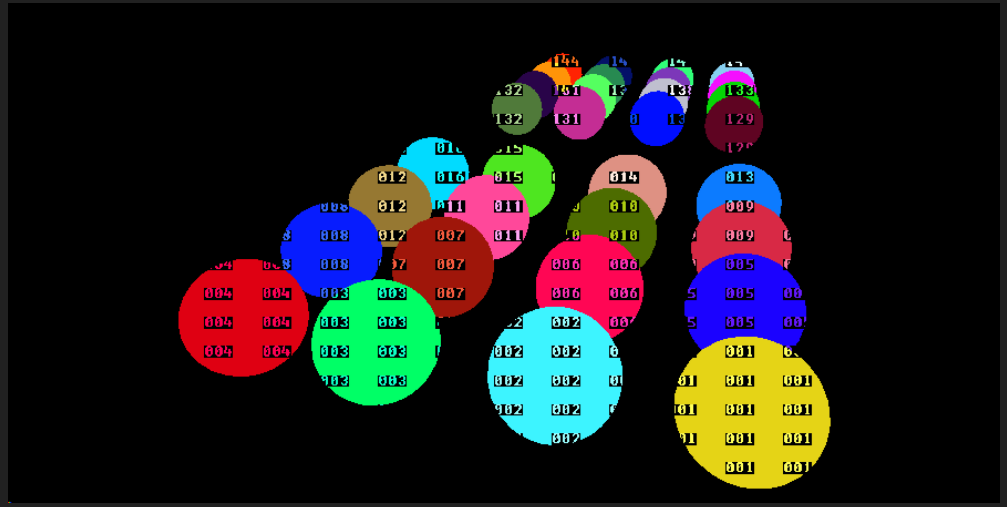
Оптимизация ветвления

- Статическое бранчевание
- Использовать permutation через дефайны

```
#if BRANCH
  A();
  B();
#else
  C();
  D();
#endif
E();
```

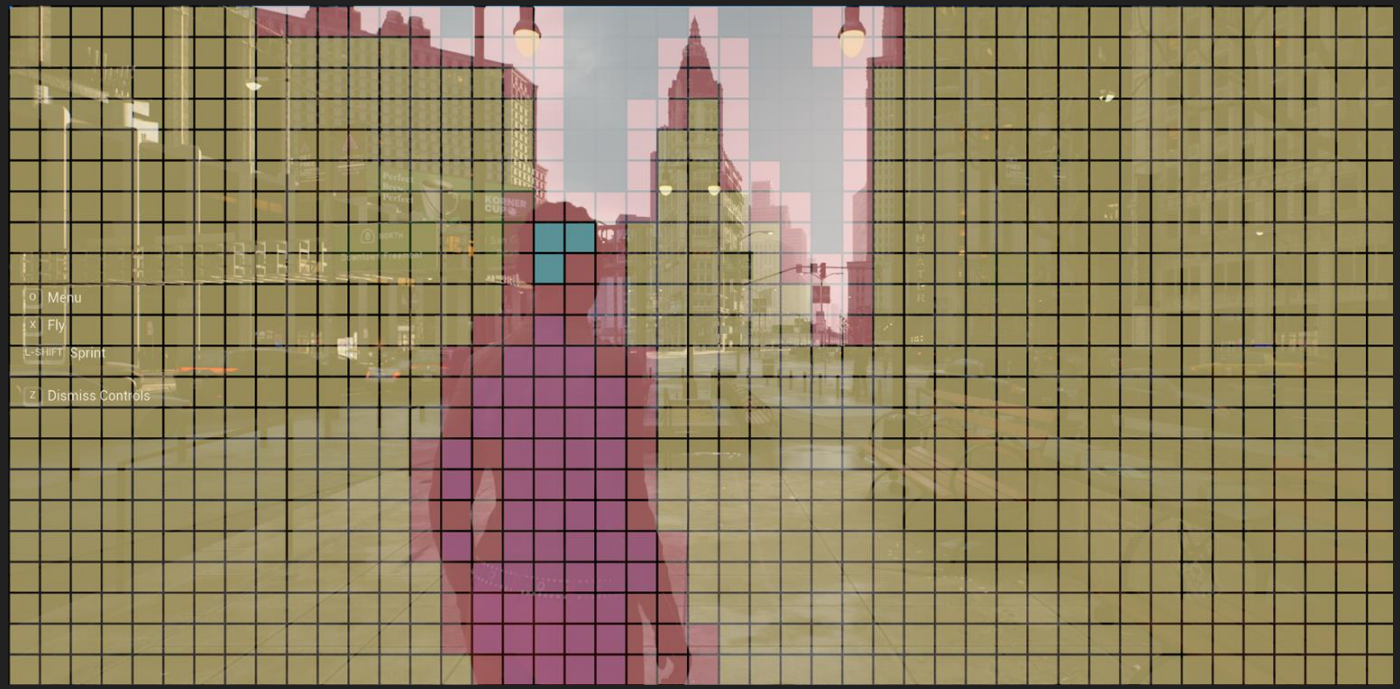
Оптимизация ветвления

Использовать маску (через depth, stencil) для фрагментных шейдеров



Потайловый компьют

Потайловая проходка через ExecuteIndirect() для compute шейдеров



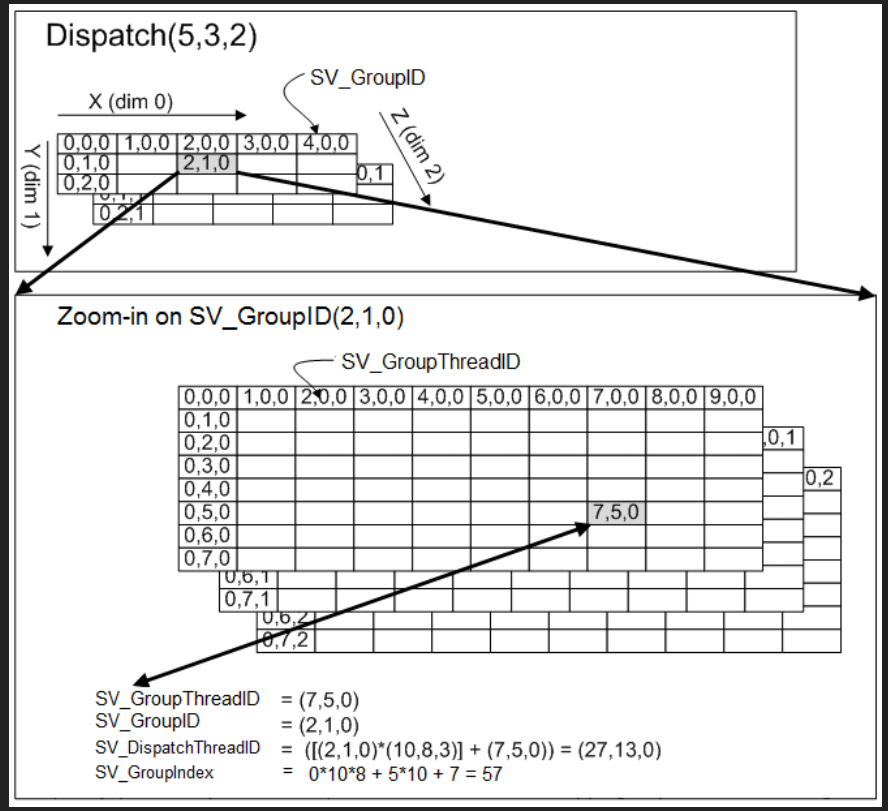
Циклы

[unroll] / [loop]

```
uint count = 4;
float w = 0.0f;

[unroll]
for (uint i = 0; i < count && w <= 1.0f; ++i)
{
    A();
    w += B();
}
```

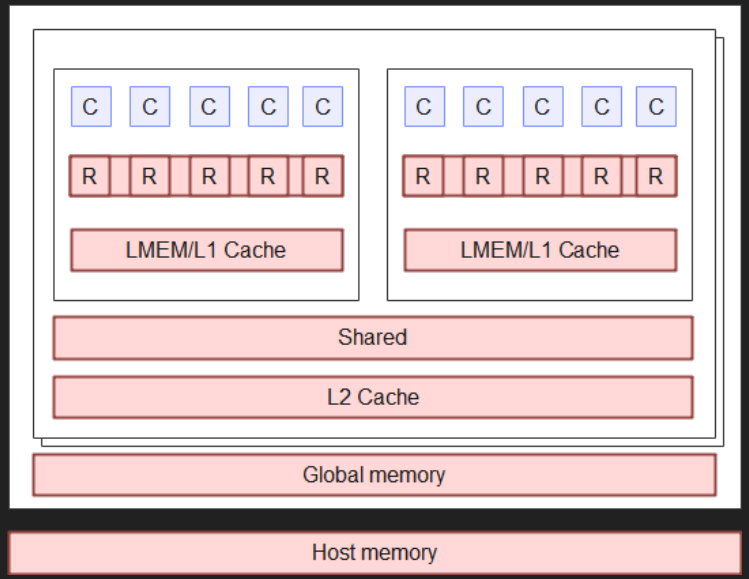
Потоки



Dispatch() вызывает 5*3*2=30 групп потоков в неопределенном порядке

В вычислительном шейдере указано что каждая группа состоит из 10*8*3=240 потоков

Память GPU



Название	Объем	Область видимости	Скорость чтения и записи
Registers	Маленький	тред	высокая или очень высокая
Shared	Маленький	группа тредов	высокая
Local Memory	Средний	тред	средняя
Global memory	Большой	группа тредов	низкая или очень низкая

Функции для Wave

- Wave Query - Общие операции для контроля (IsFirstLane / GetLaneCount / GetLaneIndex)
- Wave Vote - Операции сравнения значений между потоками (ActiveBallot / AllTrue / AnyTrue)
- Wave Broadcast - Получение значения из других тредов (ReadLaneFirst / ReadLaneAt)
- Wave Reduction - Арифметические и логические операции для активных тредов (ActiveMax / ActiveSum / BitOr и т.д.)

Memory barriers

В условиях SIMT архитектуры GPU для правильной организации вычислений нам нужны синхронизации.

Барьеры в основном различаются по области памяти которую они синхронизируют - shared, device, all, none.

В некоторых случаях простая перестановка барьеров внутри кода шейдера может влиять на производительность.

Пример использования

Shared mem:

```
groupshared float SharedMaxRadius[SIZE];

SharedMaxRadius[GroupThreadIndex] = MaxRadius;
GroupMemoryBarrierWithGroupSync();

[unroll]
for (uint i = 0; i < 5; i++)
{
  const uint ReduceSize = 32 >> i;
  if (GroupThreadIndex < ReduceSize)
  {
    MaxRadius = max(MaxRadius, SharedMaxRadius[GroupThreadIndex + ReduceSize]);
    SharedMaxRadius[GroupThreadIndex] = MaxRadius;
  }
}

TileMaxRadius = SharedMaxRadius[0];
```

Пример использования

Atomic:

```
groupshared uint SharedMaxRadius;

uint Unused;

InterlockedMax(SharedMaxRadius, asuint(MaxRadius), Unused);
GroupMemoryBarrierWithGroupSync();
TileMaxRadius = asfloat(SharedMaxRadius);
```

Wave op:

```
TileMaxRadius = WaveActiveMax(MaxRadius);
```

Упаковка

В ряде случаев точности float16 достаточно для проведения расчетов. Поэтому важно помнить о возможности упаковки данных.

- Bit-packing (f16tof32 <-> f32tof16) Можно запихивать в 2 раза больше параметров. Есть векторизация для float16 - т.е. некоторое железо может объединять две float16 операции в одной инструкции
- Текстуры - компоненты, битность. YCoCg color space
- Сферические гармоники - карты окружения в частности для лайт проб
- hardware компрессии - есть при растеризация. Поэтому иногда для полноэкранной отрисовки пиксельный шейдер будет предпочтительнее чем компьют (ROP)

Дополнительно

- fast-math оптимизации
- Избегать overdraw и контролировать blending
- Upscale/Downscale, Dynamic resolution, Checkerboard
- EarlyZ
- Threadgroup swizzeling

Автоматизация

Как и всегда частая проблема, что в активной фазе разработке изменения могут иметь деструктивный эффект, поэтому очень важно собирать и хранить статистику по билдам.

Нет возможности проверять билд целиком с каждой оптимизацией.

- скриншоты выбранных мест;
- сбор статистики в таблицы (DC, primitives count, FPS, ms);
- автоматизация проходок - пролеты камеры (можно использовать navmesh).
- анализ использования памяти

Чтиво

- nvidia-ada-gpu-architecture.pdf
- GDC2017 Wave Operations in DX SM6 and Vulkan (gpuopen.com)
- Optimizing Compute Shaders for L2 Locality using Thread-Group ID Swizzling | NVIDIA Technical Blog