**Weekly Tracking Notes**

**Week49: July.15~Jun.19,2024**

July.19

PathTracer.slang :: writeOutput

// default kCandidateSamples = 1 ( struct StaticParams{

uint32\_t candidateSamples = 1;

) in ReSTIRPTPass.h

                if (sampleId == kCandidateSamples - 1) // last sample

July.18

SpatialPathRetrace.cs.slang

          void **ReSTIR(const uint2 pixel)** //Build a reservoir of paths whose suffix can be merged to a pixel's prefix.

                      //traceRandomReplayPathHybridSimple in PathTracer.slang while loop, trace path in full length

                Tp2 = traceHybridShiftRays(params, false, centralPrimaryHitPacked, centralPrimarySd, neighborReservoir, dstRcPrevVertexHit2, dstRcPrevVertexWo2);

                      //  ReconnectionData data[RCDATA\_PATH\_NUM]  ==>  RCDATA\_PATH\_NUM = 6, RCDATA\_PAD\_SIZE = 1  (256 bytes)

**reconnectionDataBuffer**[centralOffset].data[2 \* i + 1] = ReconnectionData(dstRcPrevVertexHit2, dstRcPrevVertexWo2, Tp2);

**PixelReconnectionData reconnectionDataBuffer,** size is 256/512

struct PixelReconnectionData

{

    ReconnectionData data[RCDATA\_PATH\_NUM];// real time: RCDATA\_PATH\_NUM = 6, RCDATA\_PAD\_SIZE = 1  (256 bytes)

    float4 padding[RCDATA\_PAD\_SIZE]; //offline:   RCDATA\_PATH\_NUM = 12, RCDATA\_PAD\_SIZE = 2 (512 bytes)

}

shift.slang:

computeShiftedIntegrandHybrid connect path:

        dstRcPrevVertexHit = rcData.rcPrevHit;

        dstRcPrevVertexWo = rcData.rcPrevWo;

PathTracer.slang:

**float3 traceRandomReplayPath**(HitInfo hit  ⇒   **while loop**

               // function to trace a path using random number replay

      ⇒  **void handleHit**(inout PathState path)

             //Handle the case when a scatter ray hits the scene.After handling the hit, a new scatter ray is generated or the path is terminated.

July.17

locate the place where: **SampleNext1D** in Falcor\Utils\Sampling\SampleGeneratorInterface.slang

return random data, which from Falcor\Utils\Sampling\Pseudorandom\LCG.slang generate random data

**LCG.slang** generate random data  ⇒ **SampleNext1D**

SampleNext1D

SampleNext2D

SampleNext3D

Generates a pair of 32-bit pseudorandom numbers based on a pair of 32-bit values.

blockCipherTEA(uint v0, uint v1, uint iterations = 16)

[ByteAddressBuffer](https://learn.microsoft.com/en-us/windows/win32/direct3dhlsl/sm5-object-byteaddressbuffer):

A read-only buffer that is indexed in bytes.You can use the ByteAddressBuffer object type when you work with raw buffers.

**Method Description**

GetDimensions Gets the resource dimensions.

Load Gets one value.

Load2 Gets two values.

Load3 Gets three values.

Load4 Gets four values.

[AreaReSTIR](https://research.nvidia.com/labs/rtr/publication/zhang2024area/) [Decorrelating ReSTIR](https://research.nvidia.com/labs/rtr/publication/sawhney2022decorrelating/)

During **/// TALBOT RMIS ///** reusing, loop embeds loop, select one sample from neighbor, select another from neighbor, then merge two

1. 1st sample randomly selected from neighbor A, could from pixel itself
2. 2nd sample randomly selected from neighbor B, could from pixel itself
3. merge A&B to a new sample

**/// CONSTANT RMIS ///** doesn’t reuse itself sample, comparing with Talbot