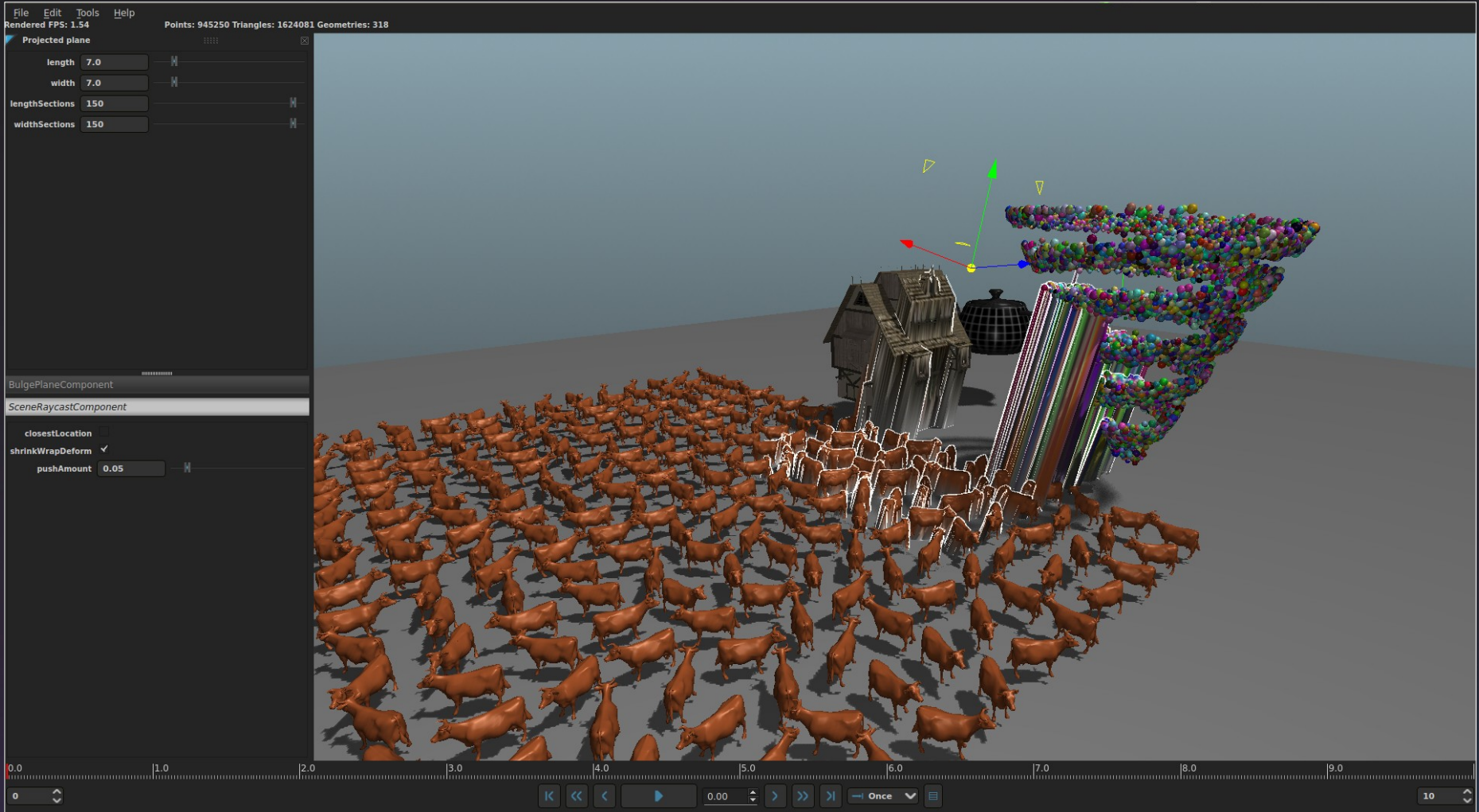


The Plan

- Overview of Fabric Engine
- Uses of LLVM within KL
- Looking forward

What is Fabric Engine?

- Digital Content Creation (DCC) framework
- Standalone applications built on PySide



What is Fabric Engine?

- Digital Content Creation (DCC) framework
- Standalone application built on PySide
- Integration with existing DCC tools (Splice)

File Edit Modify Create Display Window Assets Select Mesh Edit Mesh Proxy Normals Color Create UVs Edit UVs Muscle Pipeline Cache Fabric:Splice Help

Polygons

General Curves Surfaces Polygons Deformation Animation Dynamics Rendering PaintEffects Toon Muscle Fluids Fur nHair nCloth Custom

View Shading Lighting Show Renderer Panels

Channel Box / Layer Editor Attribute Editor Splice Editor Splice Editor

FABRICSPLICE

Available Ports
 in Scalar angleA, in Scalar angleB, in Scalar angleC, io PolygonMesh geo, in Vec3 moveA, in Integer nbSpheres, in Mat44 offsetMat, in Mat44 rootMat, in Scalar scaleFactor, in Integer shells, in Integer spans, in Integer spans2, in Scalar thickness

KL SourceCode
 spheresOp

```

0001 require PolygonMesh;
0002 require Vec3;
0003 require Vec4;
0004 require SInt32;
0005 require Scalar;
0006 require Xfo;
0007 require Quat;
0008
0009 function createSphere(Xfo root, Integer spans, Integer spans2, io PolygonMesh geo)
0010 {
0011   Vec3 profile[];
0012   for(Size i=0;i<spans;i++)
0013   {
0014     Vec3 pos(0, 1, 0.001);
0015     if(i > 0)
0016     {
0017       Quat q;
0018       q.setFromEulerAngles(Vec3(i * angleA / Scalar(spans - 1), 0.0, 0.0));
0019       pos = q.rotateVector(pos);
0020     }
0021     profile.push(pos);
0022   }
0023
0024   Xfo xfos[];
0025   for(Size i=0;i<spans2;i++)
0026   {
0027     Xfo offset;
0028     offset.setIdentity();
0029     if(i > 0)
0030     offset.ori.setFromEulerAngles(Vec3(0.0, 0.0, -Scalar(i) * angleB / Scalar(spans)));
0031     xfos.push(root * offset);
0032   }
0033
0034   sphere.addExtrusion(xfos, profile, false);
0035 }
  
```

Compile Load Save

1.00 1.00 1 24 24.00 48.00 No Anim Layer No Character Set

MEL

Select Tool: select an object

What is Fabric Engine?

- Digital Content Creation (DCC) framework
- Standalone application built on PySide
- Integration with existing DCC tools (Splice)
- In use by production studios

MPC



HYBRIDE
A **UBISOFT**® Division

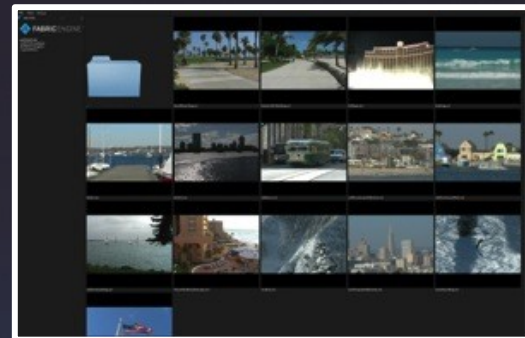
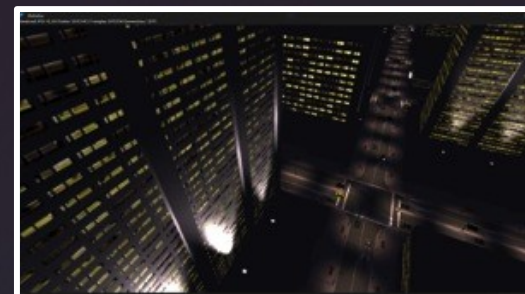
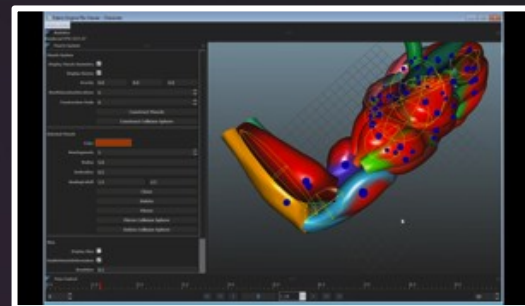
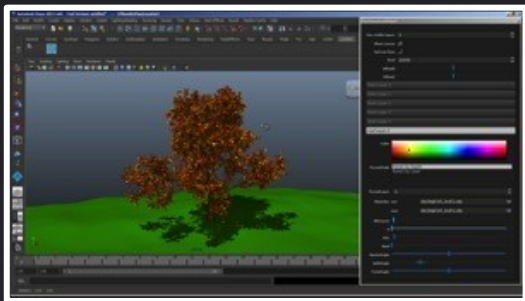
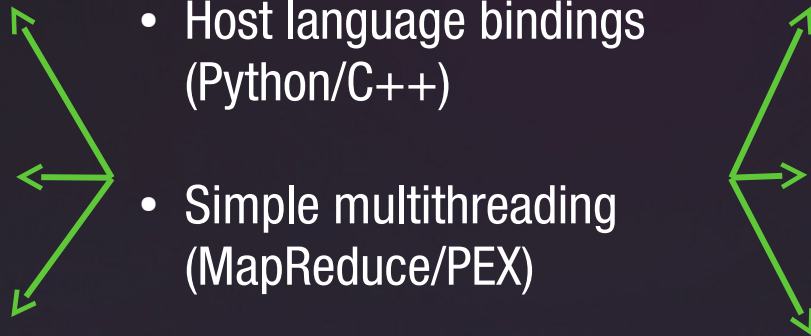
Why does Fabric Engine exist?

- Many DCC tools are old
- Closed and not configurable
- Writing plugins is hard (C++) or slow (Python)
- Development held back by software limitations



FABRICENGINE[®]

- KL language + dependency graph
- Host language bindings (Python/C++)
- Simple multithreading (MapReduce/PEX)
- Fully cross-platform (Windows/Linux/OSX)
- Code portable among other DCC applications



How does KL help?

- Ease of Python with performance of threaded C++
- Write once, use anywhere
- Crash-free and updateable on the fly
- Supports extensions for integration with existing libraries
- Target selection at runtime (CPU or GPU)

```
~/llvm-sample.kl (FabricEngine) - Sublime Text 2
llvm-sample.kl x
1  require Math;|
2
3  operator parallel_init<<index>>>(io Vec3 v[], Float32 m)
4  {
5      v[index] = Vec3(index*m, index, index*2);
6  }
7
8  function initialize(io Vec3 v[])
9  {
10     parallel_init<<v.size()>>>(v, 3.3);
11 }
12
13 operator entry()
14 {
15     Vec3 v[];
16     v.resize(1024);
17     initialize(v);
18     report(v);
19 }
20
```

INSERT MODE, Line 1, Column 14

Tab Size: 2 KL

So how does KL achieve this?

- In short: LLVM!
- MCJIT-backed
- Fabric Core compiler + scheduler
- Let's look at some specifics...

What's important for KL?

- Ease of use
- Fastest possible execution time
- Minimal memory footprint
- No significant startup delay

KL – Compilation Passes

- JIT languages slower to start than interpreted (ex. Python)
- Want maximum performance from LLVM
- Two–pass compilation
 - First unoptimized compilation pass
 - Fully optimized code generated in background

KL – Compilation Passes

- Sample case: CityAtNight.py
- 37k lines KL
- = 1.8M lines IR (pre-opt)

| <i>Method</i> | <i>Startup time</i> |
|-------------------------|---------------------|
| Upfront optimization | 2m56s |
| Background optimization | 0m37s |



KL – Caching

- Using MCJIT ObjectCache since its introduction
- Cache both IR and objects
- Key based on hash of KL AST
- Use of IR “stubs” with cached data


```
~/llvm-sample.kl (FabricEngine) - Sublime Text 2
llvm-sample.kl x
8 function initialize(io Vec3 v[])
9 {
10     parallel_init<<<v.size()>>>(v, 3.3);
11 }
12

llvm-sample.ll x
1 define private void @function.initialize.io_AS0.STVec3_VA(%STVec3_VA.Bits** nocapture %v) {
2 entry:
3     %cpuArgsStructPtr = alloca %internal.parallel_init.stub.cpu.args.00
4     %gpuArgsStructHostPtr = alloca %internal.parallel_init.stub.gpu.args
5     %0 = call i32 @method.size.in_AS0.STVec3_VA(%STVec3_VA.Bits** %v)
6     %cpuArgValuePtr.0 = getelementptr inbounds %internal.parallel_init.stub.cpu.args.00* %cpuArgsStructPtr, i32 0,
7     store %STVec3_VA.Bits** %v, %STVec3_VA.Bits*** %cpuArgValuePtr.0
8     %cpuArgValuePtr.1 = getelementptr inbounds %internal.parallel_init.stub.cpu.args.00* %cpuArgsStructPtr, i32 0,
9     store float 0x400A6666660000000, float* %cpuArgValuePtr.1
10    %cpuArgsStructVoidPtr = bitcast %internal.parallel_init.stub.cpu.args.00* %cpuArgsStructPtr to i8*
11    %nonZeroCountCond = icmp ne i32 %0, 0
12    br i1 %nonZeroCountCond, label %ep.nonZeroCount, label %ep.done
13

llvm-sample.stub.ll x
1 define private void @function.initialize.io_AS0.STVec3_VA(%STVec3_VA.Bits** nocapture %v) {
2 entry:
3     unreachable
4 }
5

INSERT MODE, Line 1, Column 1
Tab Size: 2
KL
```

KL – Caching

- Sample case: CityAtNight.py
- 37k lines KL
- = 1.8M lines IR (pre-opt)

| <i>Method</i> | <i>Startup time</i> |
|-------------------------|---------------------|
| Upfront optimization | 2m56s |
| Background optimization | 0m37s |
| From cache | 0m4s |



KL – Linking

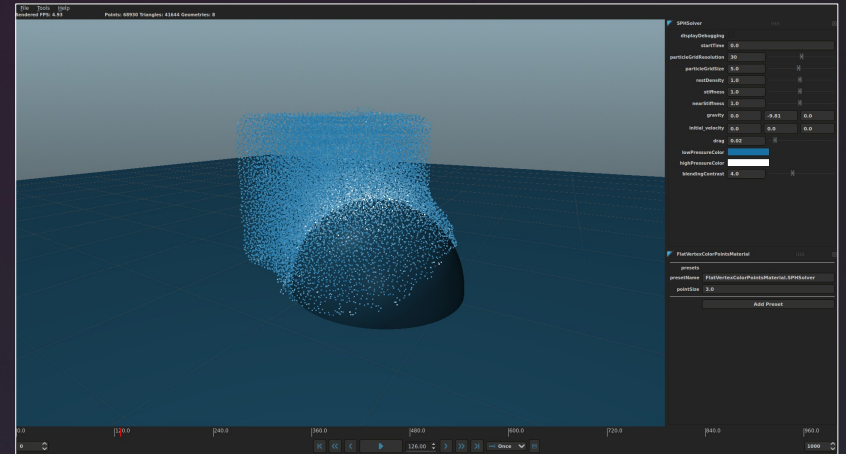
- Extensions export functions and methods
- Core links swappable function pointer into KL
- Same mechanism used in optimization pass
- Allows updating linked runtime code
- Explicit 'inline' modifier for extension functions

```
Vec3.kl
173
174 // equals operator
175 inline Boolean == (Vec3 a, Vec3 b) {
176     return a.x == b.x && a.y == b.y && a.z == b.z;
177 }
178
179 // not equals operator
180 inline Boolean != (Vec3 a, Vec3 b) {
181     return a.x != b.x || a.y != b.y || a.z != b.z;
182 }
183
184 // adds to vectors
185 inline Vec3 + (Vec3 a, Vec3 b) {
186     return vecAdd(a, b);
187 }
188
189 // adds a vector to this one
190 inline Vec3. += (Vec3 other) {
191     this = this + other;
192 }
193
194 // subtracts two vectors
195 inline Vec3 - (Vec3 a, Vec3 b) {
196     return vecSub(a, b);
197 }
198
199 // subtracts a vector from this one
```

KL – Linking

- Sample case: SPHSimulation.py

| <i>Method</i> | <i>Startup time</i> | <i>FPS</i> |
|---------------------------|---------------------|------------|
| Inline everything | 2m11s | 26 |
| Nothing inlined | 0m34s | 22 |
| Selective use of 'inline' | 0m35s | 26 |



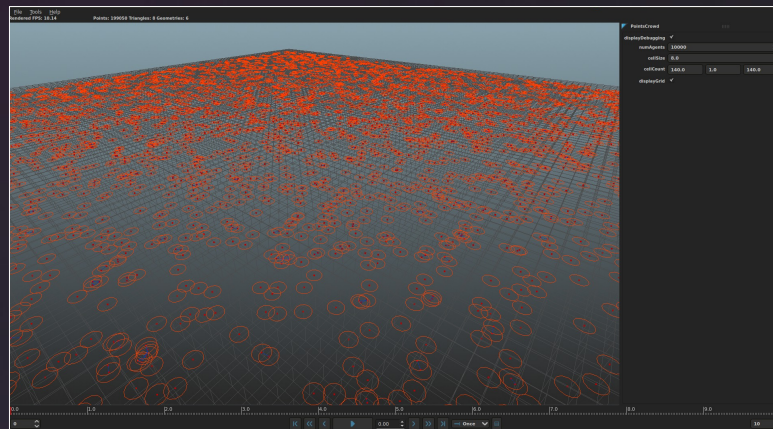
KL – Memory Use

- After compilation want minimal memory use
- LLVM 3.4: delete Module after compile
- Still need multiple ExecutionEngines

KL – Memory Use

- Sample case: Crowd.py
- With ObjectCache

| <i>Method</i> | <i>RSS (MB)</i> |
|---------------------------|-----------------|
| Full IR + no removeModule | 797 |
| Stub IR + no removeModule | 428 |
| Full IR + removeModule | 367 |
| Stub IR + removeModule | 356 |
| Shared ExecutionEngine | 296 |



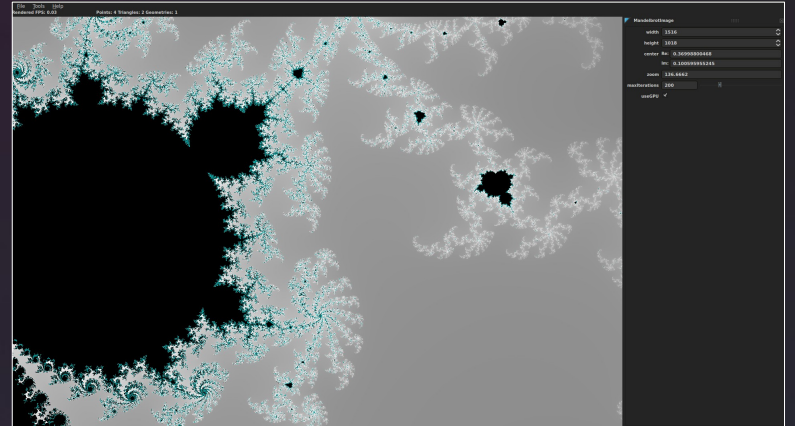
KL – GPU Compute

- KL code run without modification on CPU or GPU
- AMD HSA hardware shared memory
- Nvidia Cuda 6 “shared memory” via driver
- Speedup varies by application and hardware but up to 10x faster
- First release coming in May 2014

KL – GPU Compute

- Sample case: Mandelbrot.py
- Standard desktop hardware

| <i>Target</i> | <i>FPS</i> |
|-------------------------------|------------|
| Intel Core i7–3770k @ 3.50GHz | 3.7 |
| NVIDIA Quadro K5000 | 23.5 |



```
~/Fabric/working/SceneGraph/Python/Apps/Math/Mandelbrot/Mandelbrot.kl (FabricEngine) - Sublime Te
Mandelbrot.kl x
52
53 operator computePixels(
54     Size cols,
55     Size rows,
56     io Image2DRGB image,
57     Complex64 center,
58     Float64 zoom,
59     Size maxIterations,
60     Boolean useGPU
61 ){
62     image.resize(cols, rows);
63     computePixel<<<image.pixels.size()@useGPU>>>( ←
64         cols,
65         rows,
66         image.pixels,
67         center,
68         zoom,
69         maxIterations
70     );
71     image.incrementVersion();
72 }
73
```

INSERT MODE, Line 1, Column 1 Spaces: 2 KL

KL – Debugging

- Dwarf info via LLVM DIBuilder
- LLDB JIT support
- Breakpoints, threads, variable inspection, etc.
- Python + PySide LLDB front–end

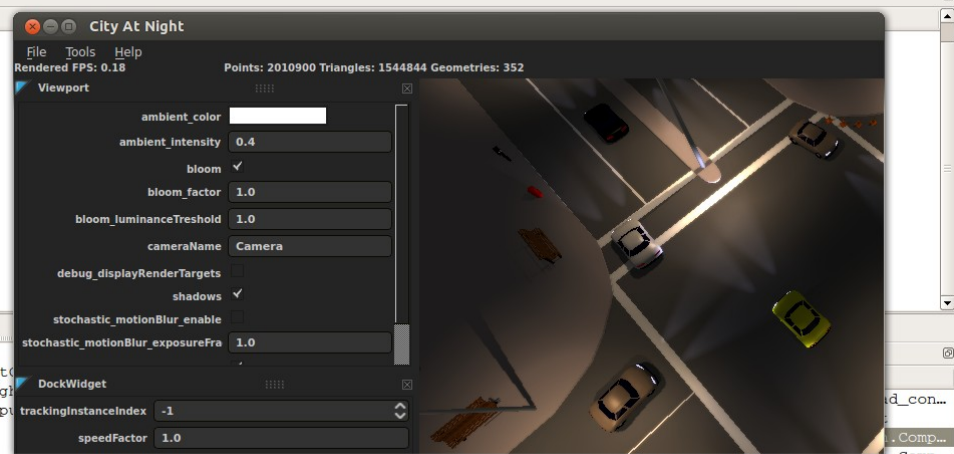
| Name | Value | Details | Type |
|------------|--------------------|---------|------------------------|
| pos | 0x0000000034a05ea0 | | Vec3 * |
| ori | 0x0000000034a05e90 | | Quat * |
| segment | 0x00000000349a1440 | | CarTrajectorySegment * |
| type | 0 | | int |
| startPt | | | Vec3 |
| x | 428.725006 | | float |
| y | 0 | | float |
| z | -206.649994 | | float |
| endPt | | | Vec3 |
| startSpeed | 8 | | float |
| endSpeed | 4 | | float |
| timeLapse | 2 | | float |
| currSpeed | 5.10337448 | | float |
| distance | 9.48889255 | | float |

| # | PC | Location |
|----|----------------|--|
| 0 | 0x7f80598888a1 | function.ComputeTrajectorySegmentPosA... |
| 1 | 0x7f80598887bb | operator.AnimateCars.in.UI32.io_AS0.S... |
| 2 | 0x7f80598884ae | kl.internal.AnimateCars.stub.cpu |
| 3 | 0x7f8063adb776 | Fabric::Util::Delegate4<unsigned int,... |
| 4 | 0x7f8063adb75a | Fabric::RT::KLDelegateCallAndLog<Fabr... |
| 5 | 0x7f8063adb750 | void Fabric::RT::CallAndLogExceptions... |
| 6 | 0x7f8063ae24ec | Fabric::Util::Thread::AltStackBracket... |
| 7 | 0x7f8063ae24da | Fabric::Util::Thread::CallOnAltStack (... |
| 8 | 0x7f8063ae24a2 | CallAndLogExceptions<Fabric::RT::KLDe... |
| 9 | 0x7f8063ae24a2 | CallWithRuntimeServices<Fabric::Util:... |
| 10 | 0x7f8063ae24a2 | void Fabric::KLSymbol::call<unsigned ... |
| 11 | 0x7f8063ae244e | Fabric::Util::ParallelCall::executeStub... |
| 12 | 0x7f8063ae2319 | Fabric::MT::ParallelCall::ExecutePara... |
| 13 | 0x7f8063add924 | Fabric::Util::SpecialDelegate4<void*,... |
| 14 | 0x7f8063add90b | Fabric::RT::KLDelegateCallAndLog<Fabr... |
| 15 | 0x7f8063add901 | void Fabric::RT::CallAndLogExceptions... |
| 16 | 0x7f8063addb93 | Fabric::Util::Thread::AltStackBracket... |
| 17 | 0x7f8063addb85 | Fabric::Util::Thread::CallOnAltStack (... |
| 18 | 0x7f8063addb50 | CallAndLogExceptions<Fabric::RT::KLDe... |
| 19 | 0x7f8063addb50 | CallWithRuntimeServices<Fabric::Util:... |
| 20 | 0x7f8063addb42 | void Fabric::RT::SetRuntimeServicesAn... |
| 21 | 0x7f8063add895 | Fabric::DG::KLRuntimeServices::Execut... |
| 22 | 0x7f806385c130 | Fabric::CallOnAlternateStack |

```

87
88     lightSourceColors.push(lightSourceColors);
89     lightSourceTransfos.push(lightSourceColors);
90     lightSourceMaterialIndices.push(lightSourceColors);
91 }
92 }
93 }
94 }
95 }
96 }
97 function ComputeTrajectorySegmentPosAndOri( CarTrajectorySegment segment, Scalar timeOffset, io Vec3 pos,
98     Scalar currSpeed = segment.startSpeed + (segment.endSpeed - segment.startSpeed) * timeOffset / segment.timeLapse;
99     Scalar distance = 0.5 * (segment.startSpeed + currSpeed) * timeOffset;
100     if (segment.type == 0) { //line
101         Vec3 dir = (segment.endPt - segment.startPt).unit();
102         pos = segment.startPt + distance * dir;
103     }
104     //TODO: there must be a more efficient way!!!
105     Scalar angle = acos(dir.x);
106     if (dir.z > 0)
107         angle = -angle;
108     ori.setFromAxisAndAngle(Vec3(0.0,1.0,0.0), angle);
109 }
110 else { //arc

```



LLDB

```

STOP breakpoint 3.1
STOP breakpoint 3.1
STOP breakpoint 3.1
STOP breakpoint 3.1
STOP breakpoint 3.1

```

lldb:

| ID | Description |
|----|---|
| 3 | SBBreakpoint: id = 3, file = 'CityAtNight.kl', line = 99, locations = 1 |

Output Breakpoints

Looking ahead

- Further reducing MCJIT memory footprint
- Better error handling in out-of-memory scenarios
- LLDB on Windows
- Clang on Windows
- GPU debugging?



<http://fabricengine.com/>

andrew.macpherson@fabricengine.com