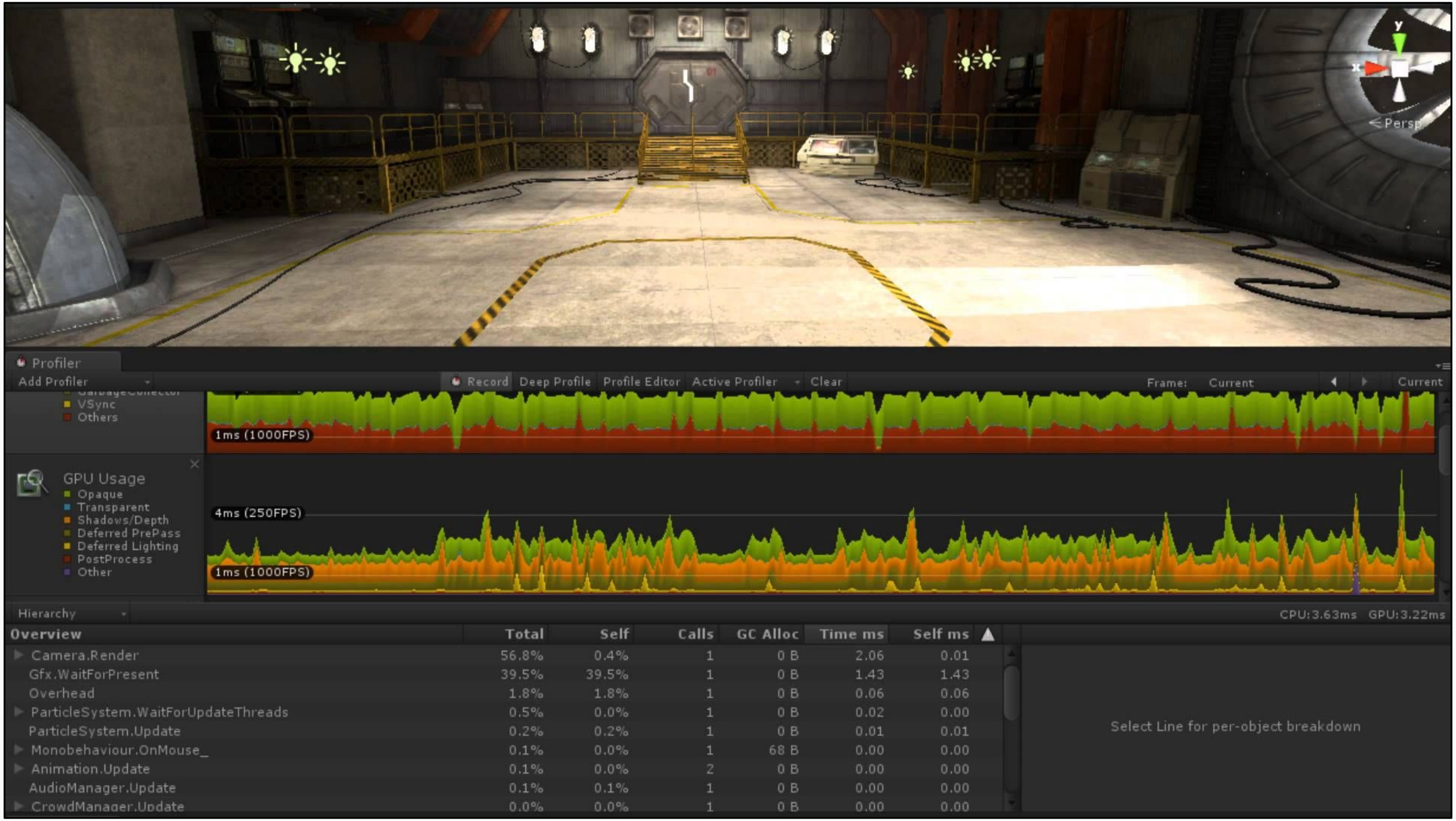
Game Dev: Profiling

Ricard Pillosu - UPC

Profiling in Video Games

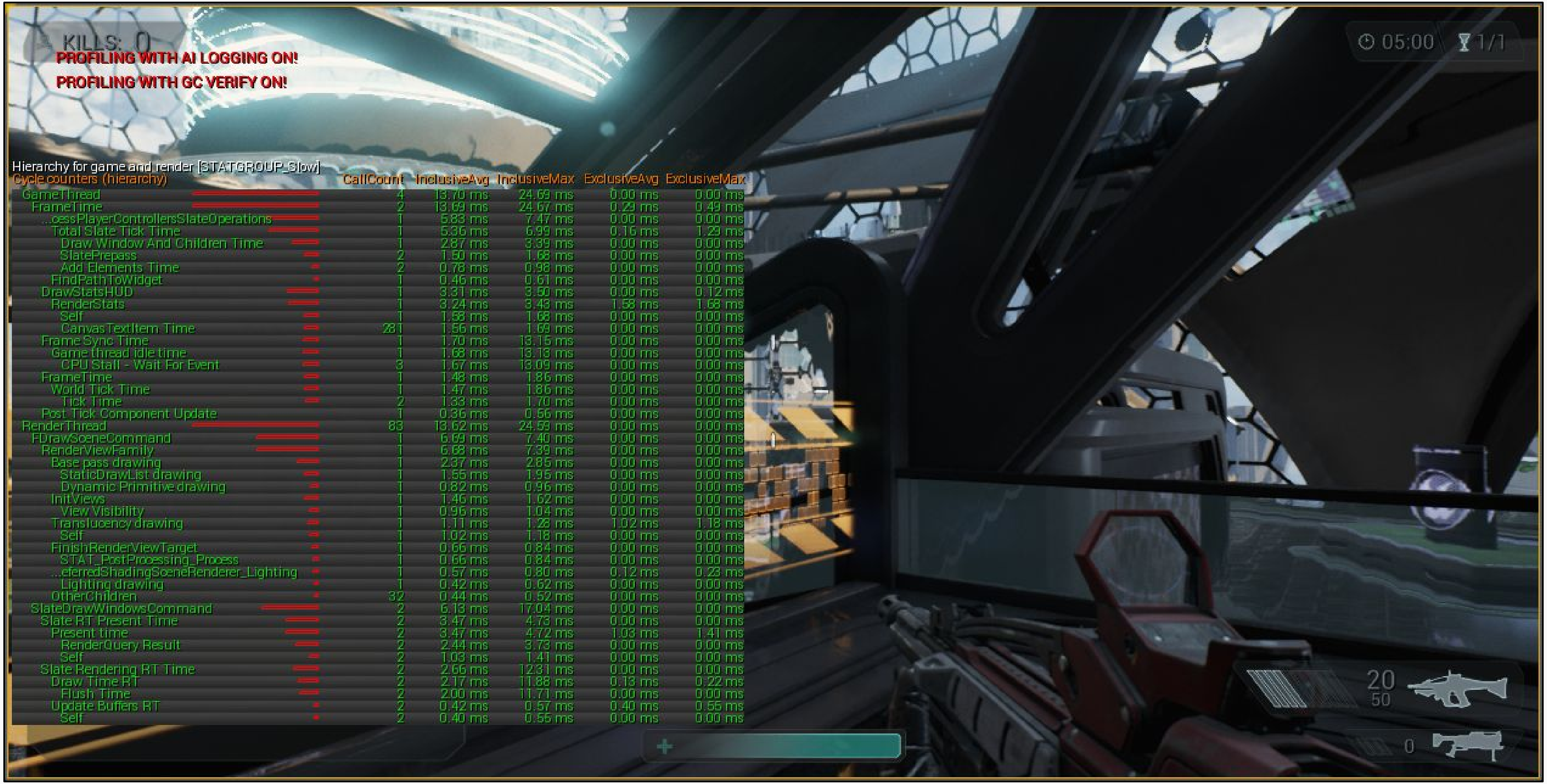
- Complexity and variability of game code makes it unpredictable
- And we want to use 100% of the hardware resources efficiently
- Coders / Artist / Designers use **profiling** to understand the consequences in **performance** of their choices for content of the video game
- We track many things like memory, cpu load, gpu load, etc ...

Profiling in Unity



Overview	Total	Self	Calls	GC Alloc	Time ms	Self ms ▲
Camera.Render	56.8%	0.4%	1	0 B	2.06	0.01
Gfx.WaitForPresent	39.5%	39.5%	1	0 B	1.43	1.43
Overhead	1.8%	1.8%	1	0 B	0.06	0.06
ParticleSystem.WaitForUpdateThreads	0.5%	0.0%	1	0 B	0.02	0.00
ParticleSystem.Update	0.2%	0.2%	1	0 B	0.01	0.01
Monobehaviour.OnMouse_	0.1%	0.0%	1	68 B	0.00	0.00
Animation.Update	0.1%	0.0%	2	0 B	0.00	0.00
AudioManager.Update	0.1%	0.1%	1	0 B	0.00	0.00
CrowdManager.Update	0.0%	0.0%	1	0 B	0.00	0.00

Profiling in Unreal Engine

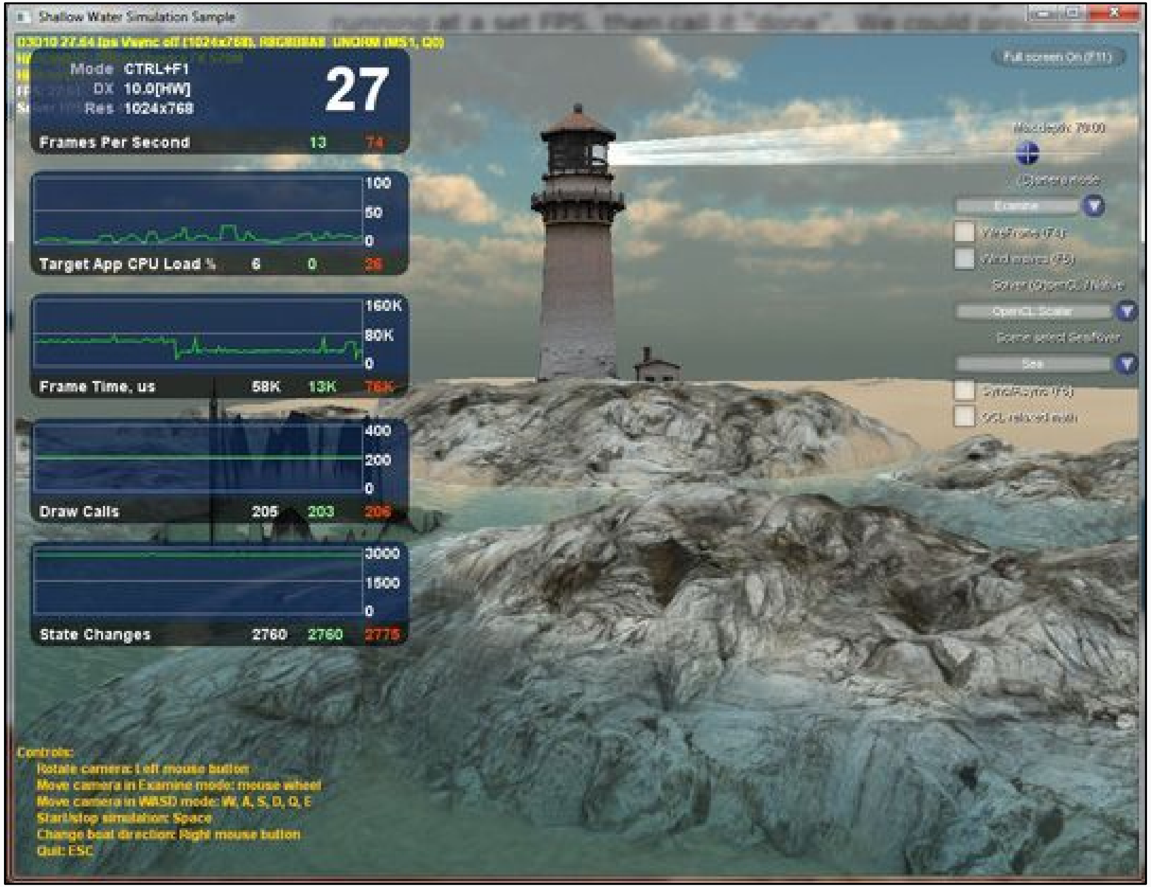


PROFILING WITH AI LOGGING ON!

PROFILING WITH GC VERIFY ON!

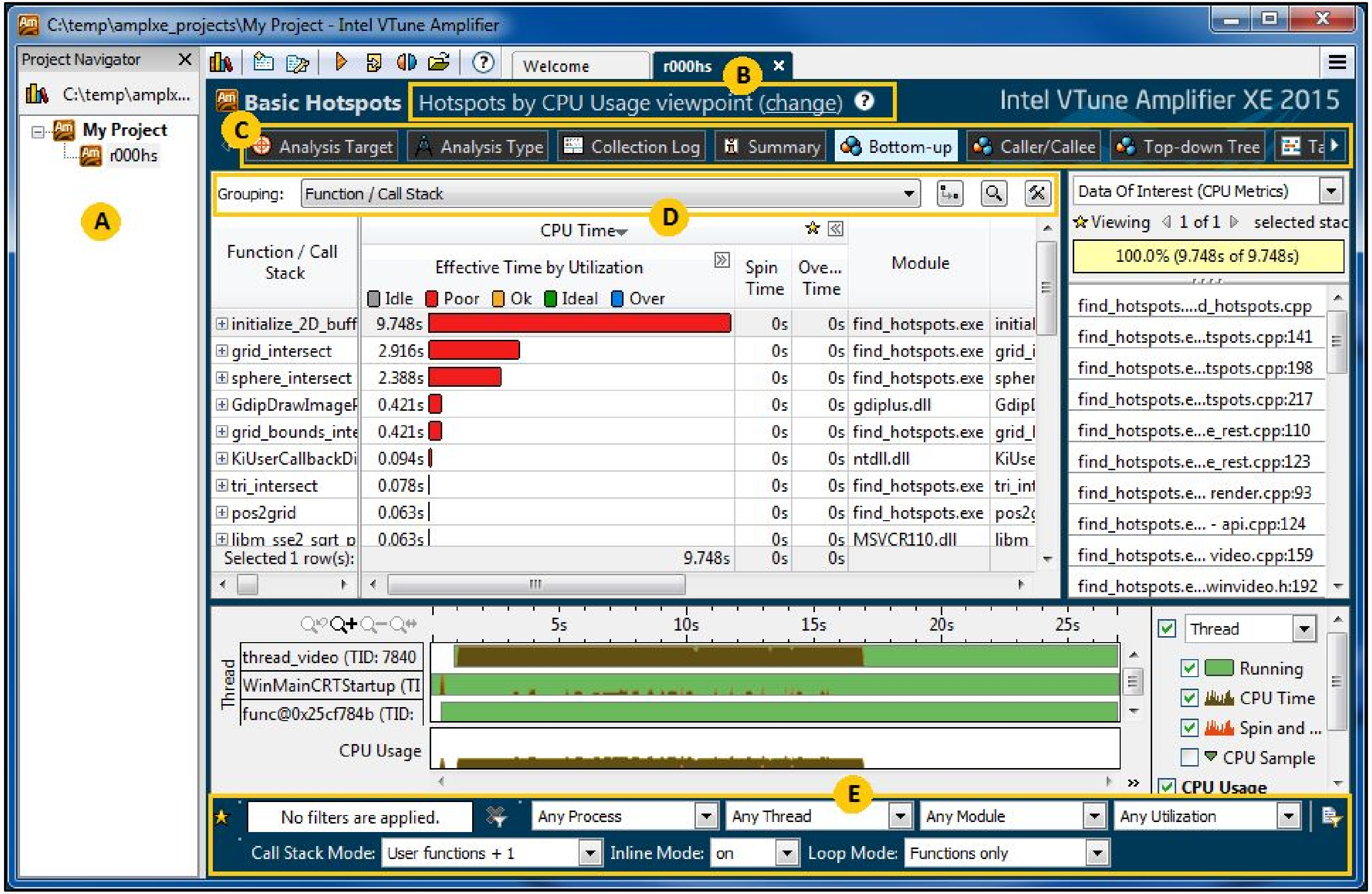
```
Hierarchy for game and render [STATGROUP_Slow]
Cycle counters (hierarchy)                          CallCount  InclusiveAvg  InclusiveMax  ExclusiveAvg  ExclusiveMax
 GameThread                                                 4     13.70 ms      24.69 ms       0.00 ms       0.00 ms
  FrameTime                                                 2     13.69 ms      24.67 ms       0.29 ms       0.49 ms
   ...cessPlayerControllersSlateOperations                  1      5.83 ms       7.47 ms       0.00 ms       0.00 ms
    Total Slate Tick Time                                   1      5.36 ms       6.99 ms       0.16 ms       1.29 ms
     Draw Window And Children Time                          1      2.87 ms       3.39 ms       0.00 ms       0.00 ms
     SlatePrepass                                           2      1.50 ms       1.68 ms       0.00 ms       0.00 ms
     Add Elements Time                                      2      0.78 ms       0.98 ms       0.00 ms       0.00 ms
    FindPathToWidget                                        1      0.46 ms       0.61 ms       0.00 ms       0.00 ms
   DrawStatsHUD                                             1      3.31 ms       3.50 ms       0.00 ms       0.12 ms
    RenderStats                                             1      3.24 ms       3.43 ms       1.58 ms       1.68 ms
     Self                                                   1      1.58 ms       1.68 ms       0.00 ms       0.00 ms
     CanvasTextItem Time                                  281      1.56 ms       1.69 ms       0.00 ms       0.00 ms
   Frame Sync Time                                          1      1.70 ms      13.15 ms       0.00 ms       0.00 ms
    Game thread idle time                                   1      1.68 ms      13.13 ms       0.00 ms       0.00 ms
     CPU Stall - Wait For Event                             3      1.67 ms      13.09 ms       0.00 ms       0.00 ms
   FrameTime                                                1      1.48 ms       1.86 ms       0.00 ms       0.00 ms
    World Tick Time                                         1      1.47 ms       1.86 ms       0.00 ms       0.00 ms
     Tick Time                                              2      1.33 ms       1.70 ms       0.00 ms       0.00 ms
   Post Tick Component Update                               1      0.36 ms       0.56 ms       0.00 ms       0.00 ms
 RenderThread                                              83     13.62 ms      24.59 ms       0.00 ms       0.00 ms
  FDrawSceneCommand                                         1      6.69 ms       7.40 ms       0.00 ms       0.00 ms
   RenderViewFamily                                         1      6.68 ms       7.39 ms       0.00 ms       0.00 ms
    Base pass drawing                                       1      2.37 ms       2.85 ms       0.00 ms       0.00 ms
     StaticDrawList drawing                                 1      1.55 ms       1.95 ms       0.00 ms       0.00 ms
     Dynamic Primitive drawing                              1      0.82 ms       0.96 ms       0.00 ms       0.00 ms
    InitViews                                               1      1.46 ms       1.62 ms       0.00 ms       0.00 ms
     View Visibility                                        1      0.96 ms       1.04 ms       0.00 ms       0.00 ms
    Translucency drawing                                    1      1.11 ms       1.28 ms       1.02 ms       1.18 ms
     Self                                                   1      1.02 ms       1.18 ms       0.00 ms       0.00 ms
    FinishRenderViewTarget                                  1      0.66 ms       0.84 ms       0.00 ms       0.00 ms
     STAT_PostProcessing_Process                            1      0.66 ms       0.84 ms       0.00 ms       0.00 ms
    ...eferredShadingSceneRenderer_Lighting                 1      0.57 ms       0.80 ms       0.12 ms       0.23 ms
     Lighting drawing                                       1      0.42 ms       0.62 ms       0.00 ms       0.00 ms
    OtherChildren                                          32      0.44 ms       0.52 ms       0.00 ms       0.00 ms
  SlateDrawWindowsCommand                                   2      6.13 ms      17.04 ms       0.00 ms       0.00 ms
   Slate RT Present Time                                    2      3.47 ms       4.73 ms       0.00 ms       0.00 ms
    Present time                                            2      3.47 ms       4.72 ms       1.03 ms       1.41 ms
     RenderQuery Result                                     2      2.44 ms       3.73 ms       0.00 ms       0.00 ms
     Self                                                   2      1.03 ms       1.41 ms       0.00 ms       0.00 ms
   Slate Rendering RT Time                                  2      2.66 ms      12.31 ms       0.00 ms       0.00 ms
    Draw Time RT                                            2      2.17 ms      11.88 ms       0.13 ms       0.22 ms
     Flush Time                                             2      2.00 ms      11.71 ms       0.00 ms       0.00 ms
    Update Buffers RT                                       2      0.42 ms       0.57 ms       0.40 ms       0.55 ms
     Self                                                   2      0.40 ms       0.55 ms       0.00 ms       0.00 ms
```

Profiling your own code



Controls:
Rotate camera: Left mouse button
Move camera in Examine mode: mouse wheel
Move camera in WASD mode: W, A, S, D, Q, E
Start/stop simulation: Space
Change boat direction: Right mouse button
Quit: ESC

Profiling with Intel’s VTune



C:\temp\amplxe_projects\My Project - Intel VTune Amplifier

C:\temp\amplx...
My Project
r000hs

Basic Hotspots Hotspots by CPU Usage viewpoint (change) ?

Intel VTune Amplifier XE 2015

Grouping: Function / Call Stack

Function / Call Stack	CPU Time: Effective Time by Utilization (Idle, Poor, Ok, Ideal, Over)	Spin Time	Ove... Time	Module	
initialize_2D_buff	9.748s	0s	0s	find_hotspots.exe	initial
grid_intersect	2.916s	0s	0s	find_hotspots.exe	grid_i
sphere_intersect	2.388s	0s	0s	find_hotspots.exe	spher
GdipDrawImageF	0.421s	0s	0s	gdiplus.dll	GdipI
grid_bounds_inte	0.421s	0s	0s	find_hotspots.exe	grid_I
KiUserCallbackDi	0.094s	0s	0s	ntdll.dll	KiUse
tri_intersect	0.078s	0s	0s	find_hotspots.exe	tri_int
pos2grid	0.063s	0s	0s	find_hotspots.exe	pos2g
libm_sse2_sqrt_p	0.063s	0s	0s	MSVCR110.dll	libm
Selected 1 row(s):	9.748s	0s	0s		

Data Of Interest (CPU Metrics)

Viewing 1 of 1 selected stac

100.0% (9.748s of 9.748s)

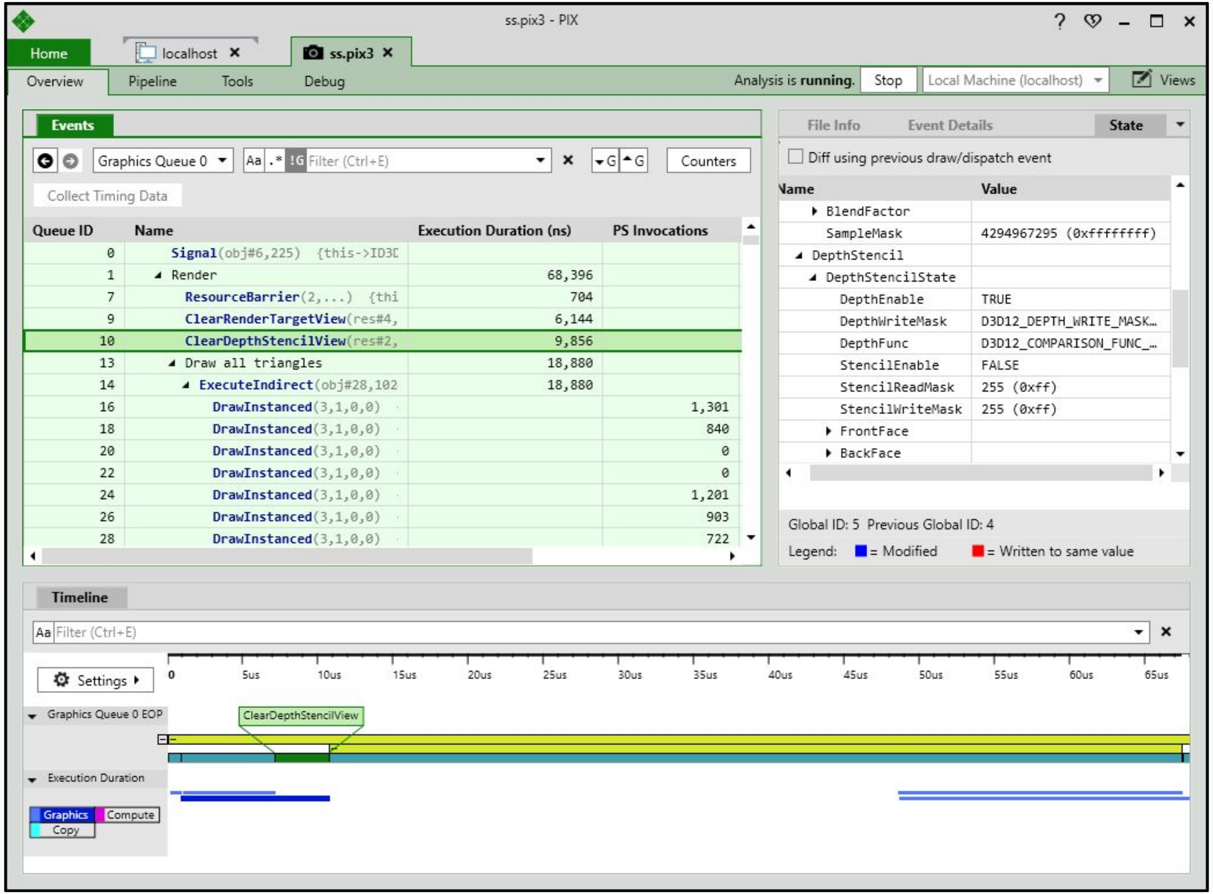
find_hotspots....d_hotspots.cpp
find_hotspots.e...tspots.cpp:141
find_hotspots.e...tspots.cpp:198
find_hotspots.e...tspots.cpp:217
find_hotspots.e...e_rest.cpp:110
find_hotspots.e...e_rest.cpp:123
find_hotspots.e... render.cpp:93
find_hotspots.e... - api.cpp:124
find_hotspots.e... video.cpp:159
find_hotspots.e...winvideo.h:192

Thread
thread_video (TID: 7840
WinMainCRTStartup (TI
func@0x25cf784b (TID:
CPU Usage

No filters are applied. Any Process Any Thread Any Module Any Utilization

Call Stack Mode: User functions + 1 Inline Mode: on Loop Mode: Functions only

Profiling with Microsoft’s PIX



Queue ID	Name	Execution Duration (ns)	PS Invocations
0	Signal(obj#6,225) {this->ID3[		
1	◢ Render	68,396	
7	ResourceBarrier(2,...) {thi	704	
9	ClearRenderTargetView(res#4,	6,144	
10	ClearDepthStencilView(res#2,	9,856	
13	◢ Draw all triangles	18,880	
14	◢ ExecuteIndirect(obj#28,102	18,880	
16	DrawInstanced(3,1,0,0)		1,301
18	DrawInstanced(3,1,0,0)		840
20	DrawInstanced(3,1,0,0)		0
22	DrawInstanced(3,1,0,0)		0
24	DrawInstanced(3,1,0,0)		1,201
26	DrawInstanced(3,1,0,0)		903
28	DrawInstanced(3,1,0,0)		722

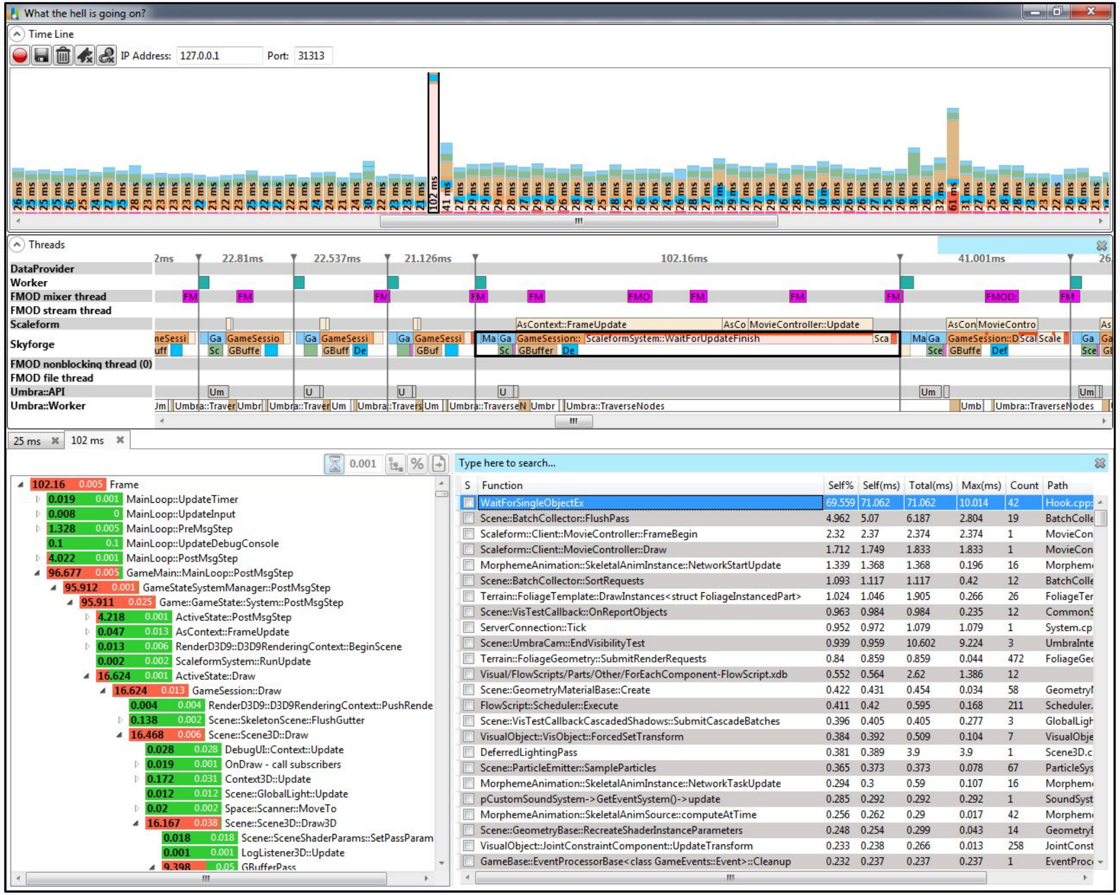
☐ Diff using previous draw/dispatch event

Name	Value
▸ BlendFactor	
SampleMask	4294967295 (0xffffffff)
◢ DepthStencil	
◢ DepthStencilState	
DepthEnable	TRUE
DepthWriteMask	D3D12_DEPTH_WRITE_MASK…
DepthFunc	D3D12_COMPARISON_FUNC_…
StencilEnable	FALSE
StencilReadMask	255 (0xff)
StencilWriteMask	255 (0xff)
▸ FrontFace	
▸ BackFace	

Global ID: 5 Previous Global ID: 4

Legend: 🟦 = Modified 🟥 = Written to same value

Profiling with Brofiler

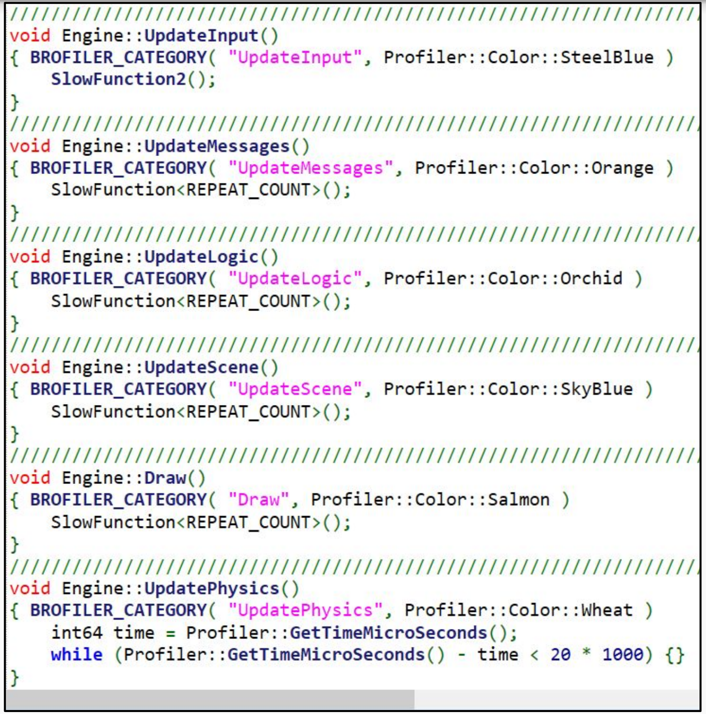


```
◢ 102.16   0.005  Frame
   ▷ 0.019    0.001  MainLoop::UpdateTimer
   ▷ 0.008    0      MainLoop::UpdateInput
   ▷ 1.328    0.005  MainLoop::PreMsgStep
     0.1      0.1    MainLoop::UpdateDebugConsole
   ▷ 4.022    0.001  MainLoop::PostMsgStep
   ◢ 96.677   0.005  GameMain::MainLoop::PostMsgStep
      ◢ 95.912   0.001  GameStateSystemManager::PostMsgStep
         ◢ 95.911   0.025  Game::GameState::System::PostMsgStep
            ▷ 4.218    0.001  ActiveState::PostMsgStep
            ▷ 0.047    0.013  AsContext::FrameUpdate
            ▷ 0.013    0.006  RenderD3D9::D3D9RenderingContext::BeginScene
              0.002    0.002  ScaleformSystem::RunUpdate
            ◢ 16.624   0.001  ActiveState::Draw
               ◢ 16.624   0.013  GameSession::Draw
                    0.004    0.004  RenderD3D9::D3D9RenderingContext::PushRende
                  ▷ 0.138    0.002  Scene::SkeletonScene::FlushGutter
                  ◢ 16.468   0.006  Scene::Scene3D::Draw
                       0.028    0.028  DebugUI::Context::Update
                     ▷ 0.019    0.001  OnDraw - call subscribers
                     ▷ 0.172    0.031  Context3D::Update
                       0.012    0.012  Scene::GlobalLight::Update
                     ▷ 0.02     0.002  Space::Scanner::MoveTo
                     ◢ 16.167   0.038  Scene::Scene3D::Draw3D
                          0.018    0.018  Scene::SceneShaderParams::SetPassParam
                          0.001    0.001  LogListener3D::Update
                        ◢ 9.398    0.05   GBufferPass
```

S	Function	Self%	Self(ms)	Total(ms)	Max(ms)	Count	Path
□	WaitForSingleObjectEx	69.559	71.062	71.062	10.014	42	Hook.cpp:
□	Scene::BatchCollector::FlushPass	4.962	5.07	6.187	2.804	19	BatchColle
□	Scaleform::Client::MovieController::FrameBegin	2.32	2.37	2.374	2.374	1	MovieCon
□	Scaleform::Client::MovieController::Draw	1.712	1.749	1.833	1.833	1	MovieCon
□	MorphemeAnimation::SkeletalAnimInstance::NetworkStartUpdate	1.339	1.368	1.368	0.196	16	Morphem
□	Scene::BatchCollector::SortRequests	1.093	1.117	1.117	0.42	12	BatchColle
□	Terrain::FoliageTemplate::DrawInstances<struct FoliageInstancedPart>	1.024	1.046	1.905	0.266	26	FoliageTer
□	Scene::VisTestCallback::OnReportObjects	0.963	0.984	0.984	0.235	12	CommonS
□	ServerConnection::Tick	0.952	0.972	1.079	1.079	1	System.cp
□	Scene::UmbraCam::EndVisibilityTest	0.939	0.959	10.602	9.224	3	UmbraInte
□	Terrain::FoliageGeometry::SubmitRenderRequests	0.84	0.859	0.859	0.044	472	FoliageGe
□	Visual/FlowScripts/Parts/Other/ForEachComponent-FlowScript.xdb	0.552	0.564	2.62	1.386	12	
□	Scene::GeometryMaterialBase::Create	0.422	0.431	0.454	0.034	58	GeometryM
□	FlowScript::Scheduler::Execute	0.411	0.42	0.595	0.168	211	Scheduler.
□	Scene::VisTestCallbackCascadedShadows::SubmitCascadeBatches	0.396	0.405	0.405	0.277	3	GlobalLigh
□	VisualObject::VisObject::ForcedSetTransform	0.384	0.392	0.509	0.104	7	VisualObje
□	DeferredLightingPass	0.381	0.389	3.9	3.9	1	Scene3D.c
□	Scene::ParticleEmitter::SampleParticles	0.365	0.373	0.373	0.078	67	ParticleSys
□	MorphemeAnimation::SkeletalAnimInstance::NetworkTaskUpdate	0.294	0.3	0.59	0.107	16	Morphem
□	pCustomSoundSystem->GetEventSystem()->update	0.285	0.292	0.292	0.292	1	SoundSyst
□	MorphemeAnimation::SkeletalAnimSource::computeAtTime	0.256	0.262	0.29	0.017	42	Morphem
□	Scene::GeometryBase::RecreateShaderInstanceParameters	0.248	0.254	0.299	0.043	14	GeometryB
□	VisualObject::JointConstraintComponent::UpdateTransform	0.233	0.238	0.266	0.013	258	JointConst
□	GameBase::EventProcessorBase<class GameEvents::Event>::Cleanup	0.232	0.237	0.237	0.237	1	EventProc

Brofiler

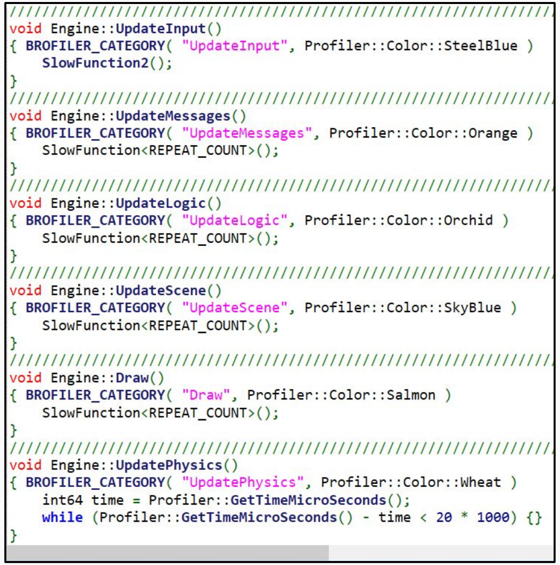
- We will integrate Brofiler
- It requires [code instrumentation]
- This means that we need to use its library to inject C++ code in our game



```
//////////////////////////////////////////////////////////////////
void Engine::UpdateInput()
{ BROFILER_CATEGORY( "UpdateInput", Profiler::Color::SteelBlue )
    SlowFunction2();
}
//////////////////////////////////////////////////////////////////
void Engine::UpdateMessages()
{ BROFILER_CATEGORY( "UpdateMessages", Profiler::Color::Orange )
    SlowFunction<REPEAT_COUNT>();
}
//////////////////////////////////////////////////////////////////
void Engine::UpdateLogic()
{ BROFILER_CATEGORY( "UpdateLogic", Profiler::Color::Orchid )
    SlowFunction<REPEAT_COUNT>();
}
//////////////////////////////////////////////////////////////////
void Engine::UpdateScene()
{ BROFILER_CATEGORY( "UpdateScene", Profiler::Color::SkyBlue )
    SlowFunction<REPEAT_COUNT>();
}
//////////////////////////////////////////////////////////////////
void Engine::Draw()
{ BROFILER_CATEGORY( "Draw", Profiler::Color::Salmon )
    SlowFunction<REPEAT_COUNT>();
}
//////////////////////////////////////////////////////////////////
void Engine::UpdatePhysics()
{ BROFILER_CATEGORY( "UpdatePhysics", Profiler::Color::Wheat )
    int64 time = Profiler::GetTimeMicroSeconds();
    while (Profiler::GetTimeMicroSeconds() - time < 20 * 1000) {}
}
```

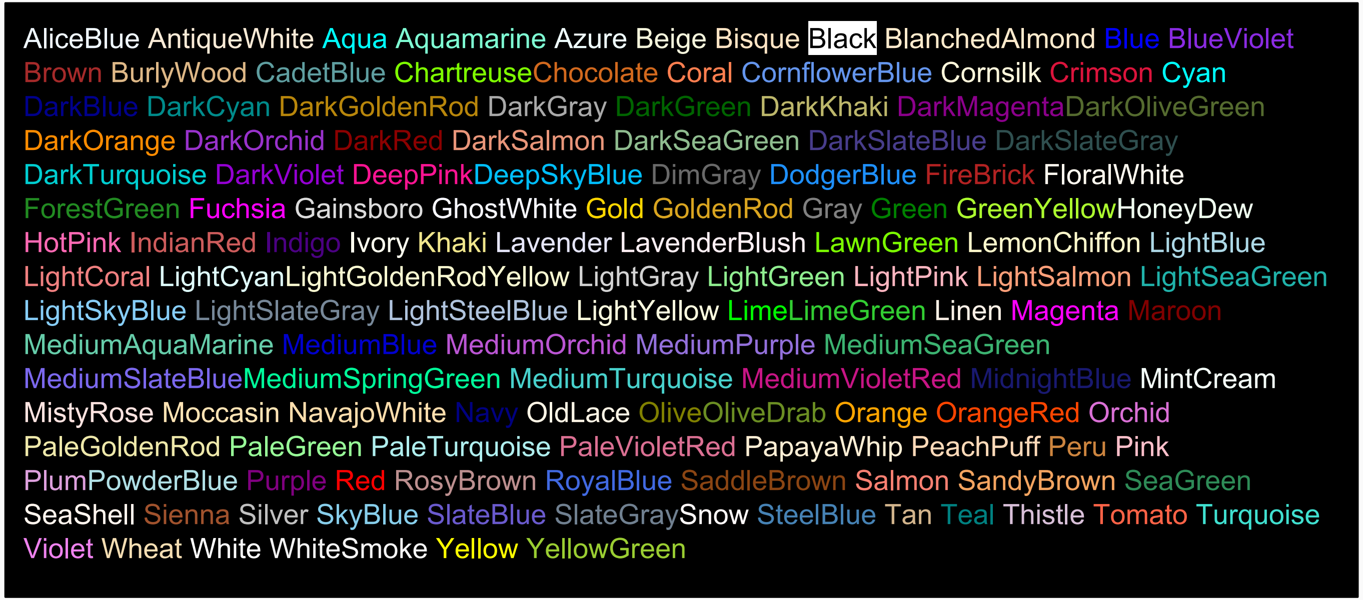
Brofiler: integration steps

- Include Brofiler header and lib
- Mark the beginning of the main loop using:
  - `BROFILER_FRAME("YourThreadName")`
- Mark any function that you want measured:
  - `BROFILER_CATEGORY( "UpdateLogic", Profiler::Color::Orchid )`



```
//////////////////////////////////////////////////////////////////
void Engine::UpdateInput()
{ BROFILER_CATEGORY( "UpdateInput", Profiler::Color::SteelBlue )
    SlowFunction2();
}
//////////////////////////////////////////////////////////////////
void Engine::UpdateMessages()
{ BROFILER_CATEGORY( "UpdateMessages", Profiler::Color::Orange )
    SlowFunction<REPEAT_COUNT>();
}
//////////////////////////////////////////////////////////////////
void Engine::UpdateLogic()
{ BROFILER_CATEGORY( "UpdateLogic", Profiler::Color::Orchid )
    SlowFunction<REPEAT_COUNT>();
}
//////////////////////////////////////////////////////////////////
void Engine::UpdateScene()
{ BROFILER_CATEGORY( "UpdateScene", Profiler::Color::SkyBlue )
    SlowFunction<REPEAT_COUNT>();
}
//////////////////////////////////////////////////////////////////
void Engine::Draw()
{ BROFILER_CATEGORY( "Draw", Profiler::Color::Salmon )
    SlowFunction<REPEAT_COUNT>();
}
//////////////////////////////////////////////////////////////////
void Engine::UpdatePhysics()
{ BROFILER_CATEGORY( "UpdatePhysics", Profiler::Color::Wheat )
    int64 time = Profiler::GetTimeMicroSeconds();
    while (Profiler::GetTimeMicroSeconds() - time < 20 * 1000) {}
}
```

Brofiler Colors Reference



AliceBlue AntiqueWhite Aqua Aquamarine Azure Beige Bisque Black BlanchedAlmond Blue BlueViolet
Brown BurlyWood CadetBlue ChartreuseChocolate Coral CornflowerBlue Cornsilk Crimson Cyan
DarkBlue DarkCyan DarkGoldenRod DarkGray DarkGreen DarkKhaki DarkMagentaDarkOliveGreen
DarkOrange DarkOrchid DarkRed DarkSalmon DarkSeaGreen DarkSlateBlue DarkSlateGray
DarkTurquoise DarkViolet DeepPinkDeepSkyBlue DimGray DodgerBlue FireBrick FloralWhite
ForestGreen Fuchsia Gainsboro GhostWhite Gold GoldenRod Gray Green GreenYellowHoneyDew
HotPink IndianRed Indigo Ivory Khaki Lavender LavenderBlush LawnGreen LemonChiffon LightBlue
LightCoral LightCyanLightGoldenRodYellow LightGray LightGreen LightPink LightSalmon LightSeaGreen
LightSkyBlue LightSlateGray LightSteelBlue LightYellow LimeLimeGreen Linen Magenta Maroon
MediumAquaMarine MediumBlue MediumOrchid MediumPurple MediumSeaGreen
MediumSlateBlueMediumSpringGreen MediumTurquoise MediumVioletRed MidnightBlue MintCream
MistyRose Moccasin NavajoWhite Navy OldLace OliveOliveDrab Orange OrangeRed Orchid
PaleGoldenRod PaleGreen PaleTurquoise PaleVioletRed PapayaWhip PeachPuff Peru Pink
PlumPowderBlue Purple Red RosyBrown RoyalBlue SaddleBrown Salmon SandyBrown SeaGreen
SeaShell Sienna Silver SkyBlue SlateBlue SlateGraySnow SteelBlue Tan Teal Thistle Tomato Turquoise
Violet Wheat White WhiteSmoke Yellow YellowGreen

TODO 1 / 2 / 3

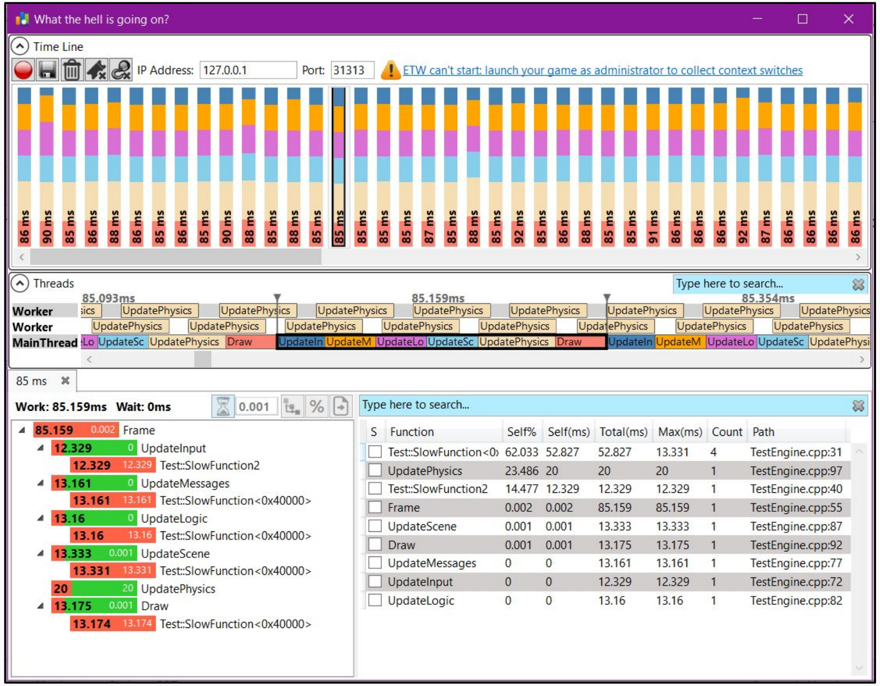
- Add brofiler in main.cpp in
  - Add the header
  - Add the library
  - Add the macro to mark the beginning of the main loop
- Now trace App methods:
  - Add the Brofile header
  - Add Brofiler macros to trace all *Update methods from App

Brofiler GUI

You can see the distribution of use of millisecond on the top panel.

Middle area is for threads

Bottom has the hierarchy and the details of performance usage



IP Address: 127.0.0.1 Port: 31313 ETW can't start: launch your game as administrator to collect context switches

Work: 85.159ms Wait: 0ms 0.001

```
85.159  0.002  Frame
  12.329  0  UpdateInput
    12.329  12.329  Test::SlowFunction2
  13.161  0  UpdateMessages
    13.161  13.161  Test::SlowFunction<0x40000>
  13.16  0  UpdateLogic
    13.16  13.16  Test::SlowFunction<0x40000>
  13.333  0.001  UpdateScene
    13.331  13.331  Test::SlowFunction<0x40000>
  20  20  UpdatePhysics
  13.175  0.001  Draw
    13.174  13.174  Test::SlowFunction<0x40000>
```

Type here to search...

S	Function	Self%	Self(ms)	Total(ms)	Max(ms)	Count	Path
□	Test::SlowFunction<0›	62.033	52.827	52.827	13.331	4	TestEngine.cpp:31
□	UpdatePhysics	23.486	20	20	20	1	TestEngine.cpp:97
□	Test::SlowFunction2	14.477	12.329	12.329	12.329	1	TestEngine.cpp:40
□	Frame	0.002	0.002	85.159	85.159	1	TestEngine.cpp:55
□	UpdateScene	0.001	0.001	13.333	13.333	1	TestEngine.cpp:87
□	Draw	0.001	0.001	13.175	13.175	1	TestEngine.cpp:92
□	UpdateMessages	0	0	13.161	13.161	1	TestEngine.cpp:77
□	UpdateInput	0	0	12.329	12.329	1	TestEngine.cpp:72
□	UpdateLogic	0	0	13.16	13.16	1	TestEngine.cpp:82

References

- Brofiler is in active development, check it’s latest code here
- Research from students here and here
- Will need to follow the development of PIX

Homework

Add code instrumentalization to measure in ms:

- Cost of your A* / Dijkstra / BFS
- Cost of each module predupdate / update / postupdate
- Check cost of Render::PostUpdate with and without vsync
- Check j1Map::Draw