

Apple Metal

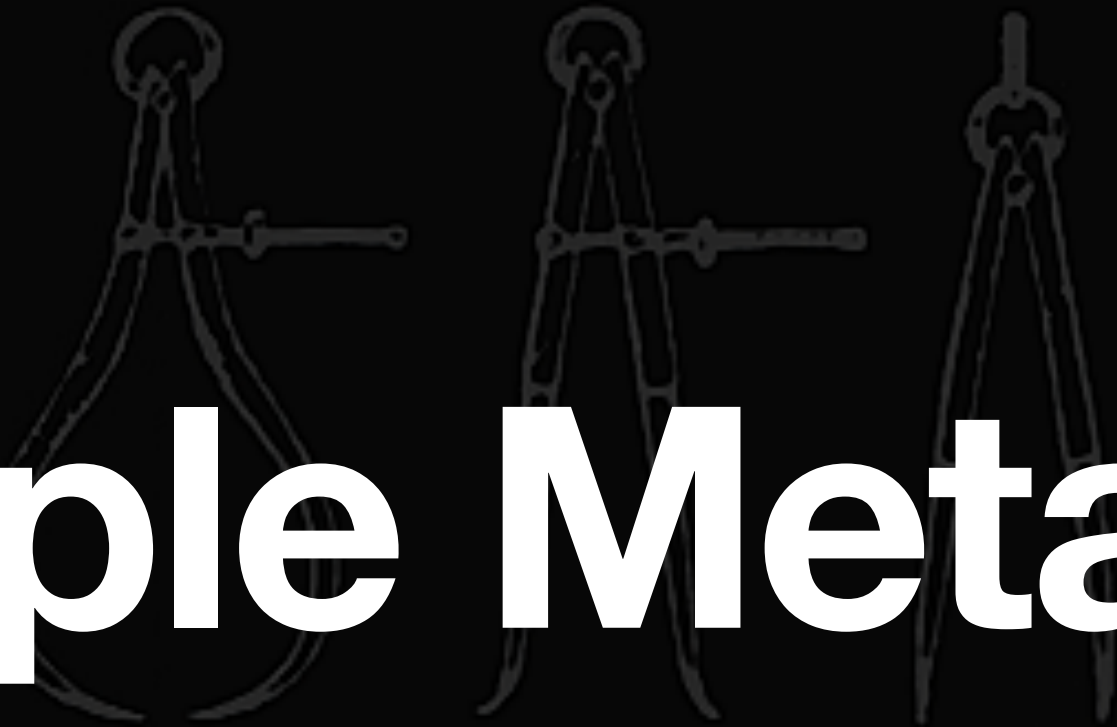
Tools and Debugging Techniques



straightedge rule



snips



spring caliper

vernier caliper



hermaphrodite caliper



twist drill



reamer



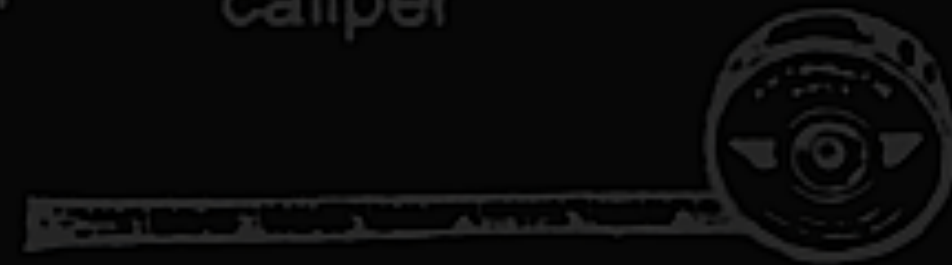
try square



long nose pliers



diagonal cutting pliers



tape rule



chisel



scraper



vernier caliper



micrometer



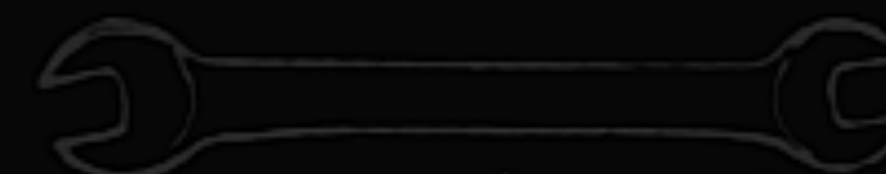
hacksaw



hand drill



scriber



open-end wrench

George Ostrobrod

Savage Software
(Creator of Procreate)

Apple Metal Tools

- Metal System Trace (Instruments)
- GPU Frame Capture (XCode)
- GPU Frame Capture (in code)

Apple Metal Tools

- Metal System Trace (Instruments)
- GPU Frame Capture (XCode)
- GPU Frame Capture (in code)





(丿 𐄂益𐄂) 丿

THANK YOU

Questions?

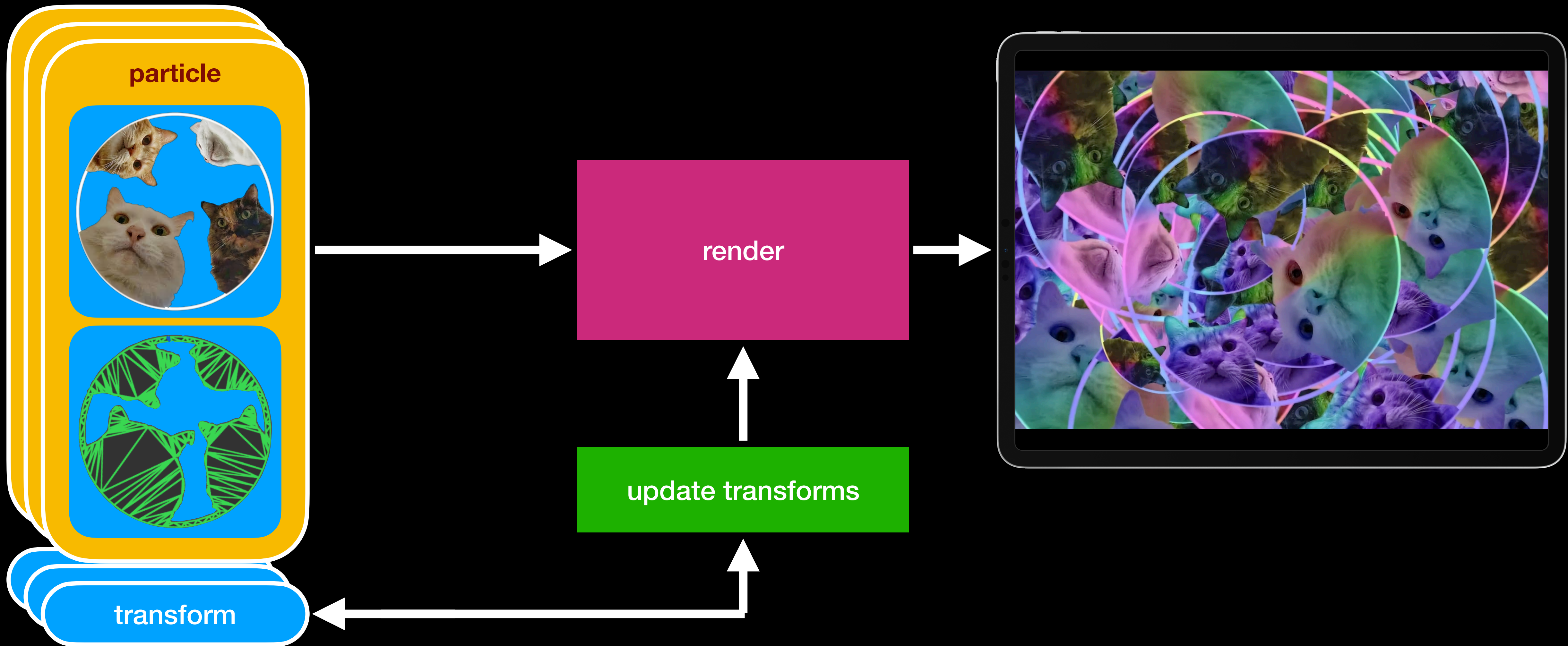
Presentation Plan

- Problem (and experimental task)
- Apple's tools
 - Documentation
 - Metal System Trace (Instruments)
 - GPU Frame Capture (XCode + Metal API)
- Manual solving problems

EXPERIMENTAL TASK

Cats (compute + render)

EXPERIMENTAL TASK



PERFORMANCE

Profiling

CORRECTNESS

Debugging

INSTRUMENTS

Apple Documentation

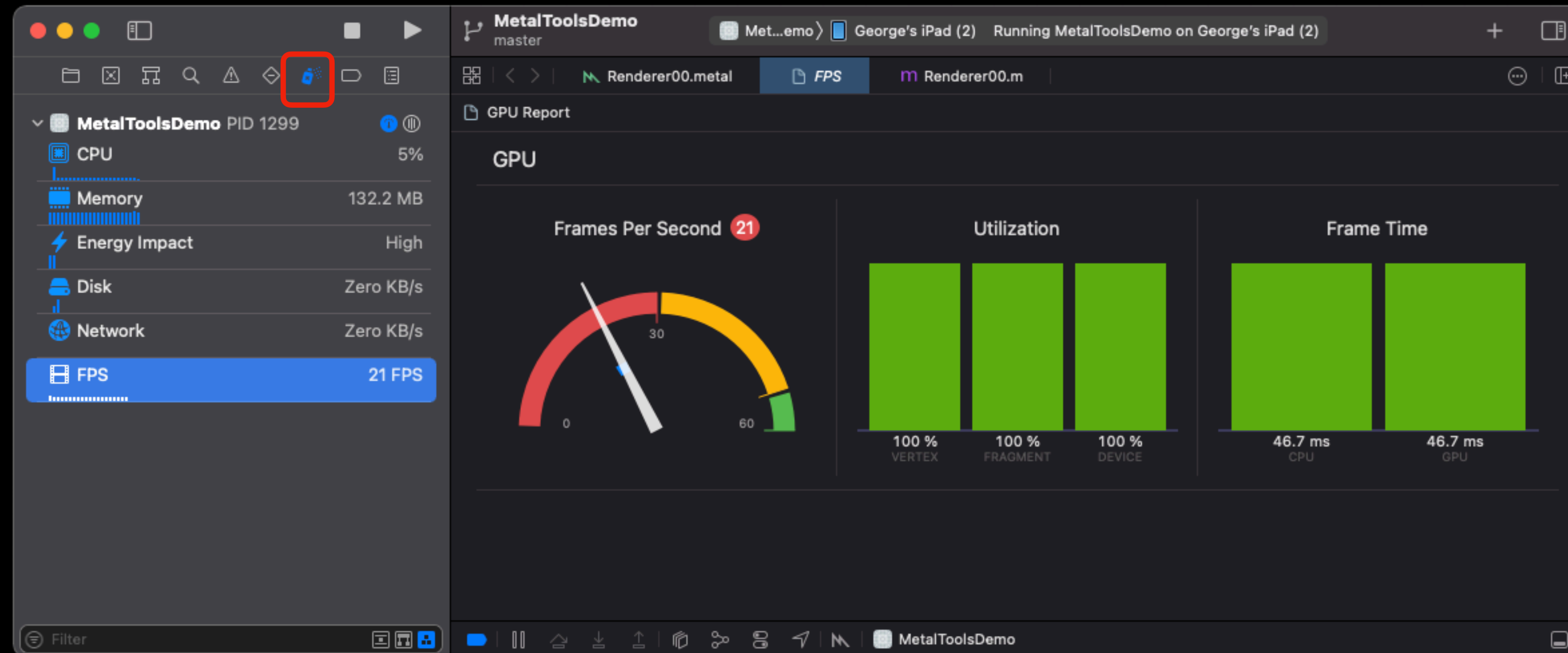
Documentation

- [Metal Feature Set Tables](#)
- [Metal Shading Language Specification](#)
- [Metal Performance Shaders](#)
- [Metal API Documentation](#)

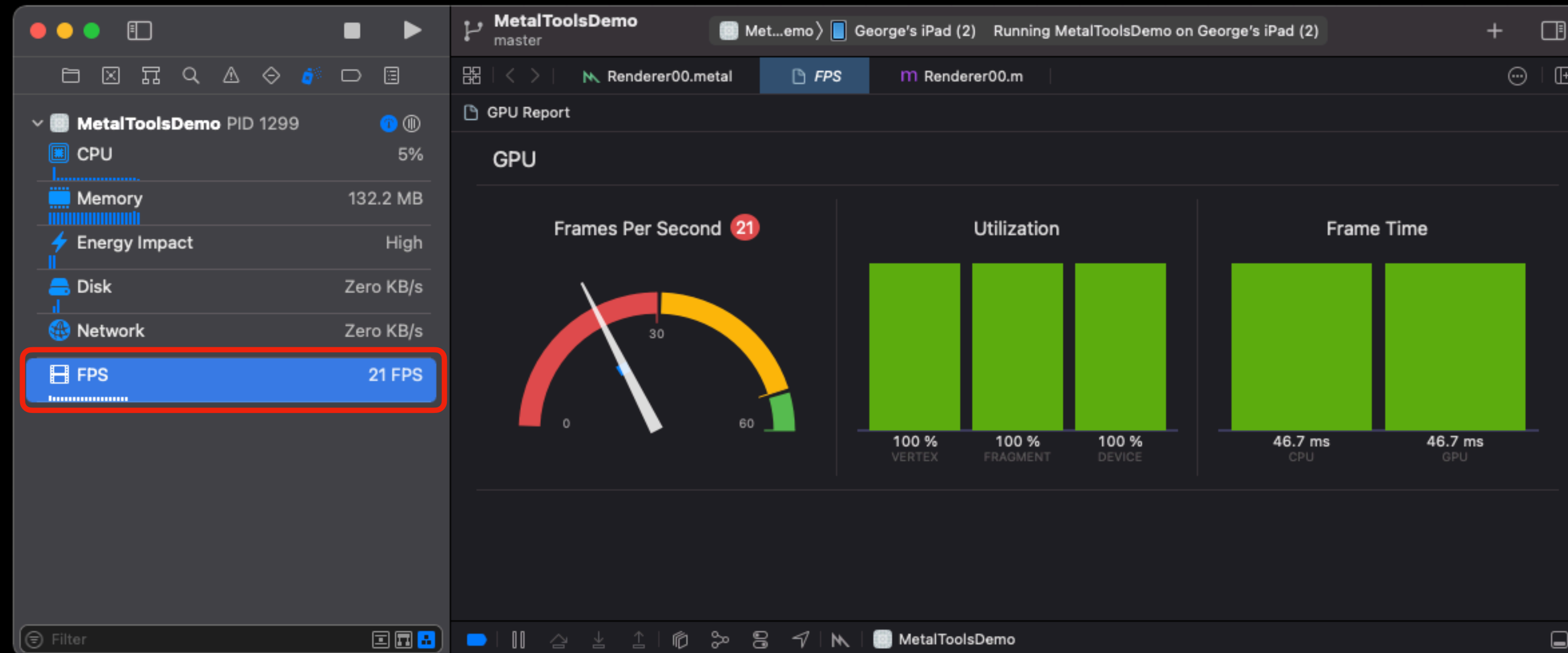
XCODE

Debug Navigator: GPU

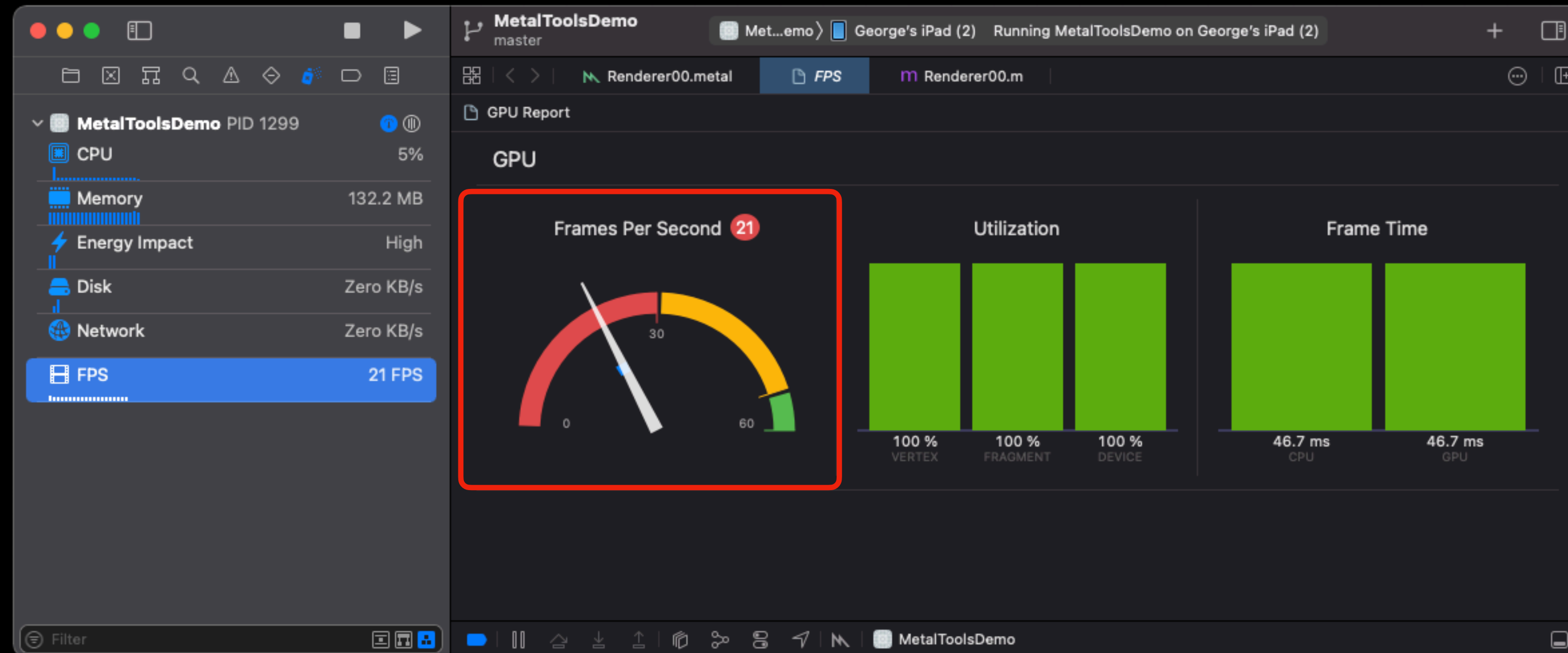
Debug Navigator: GPU



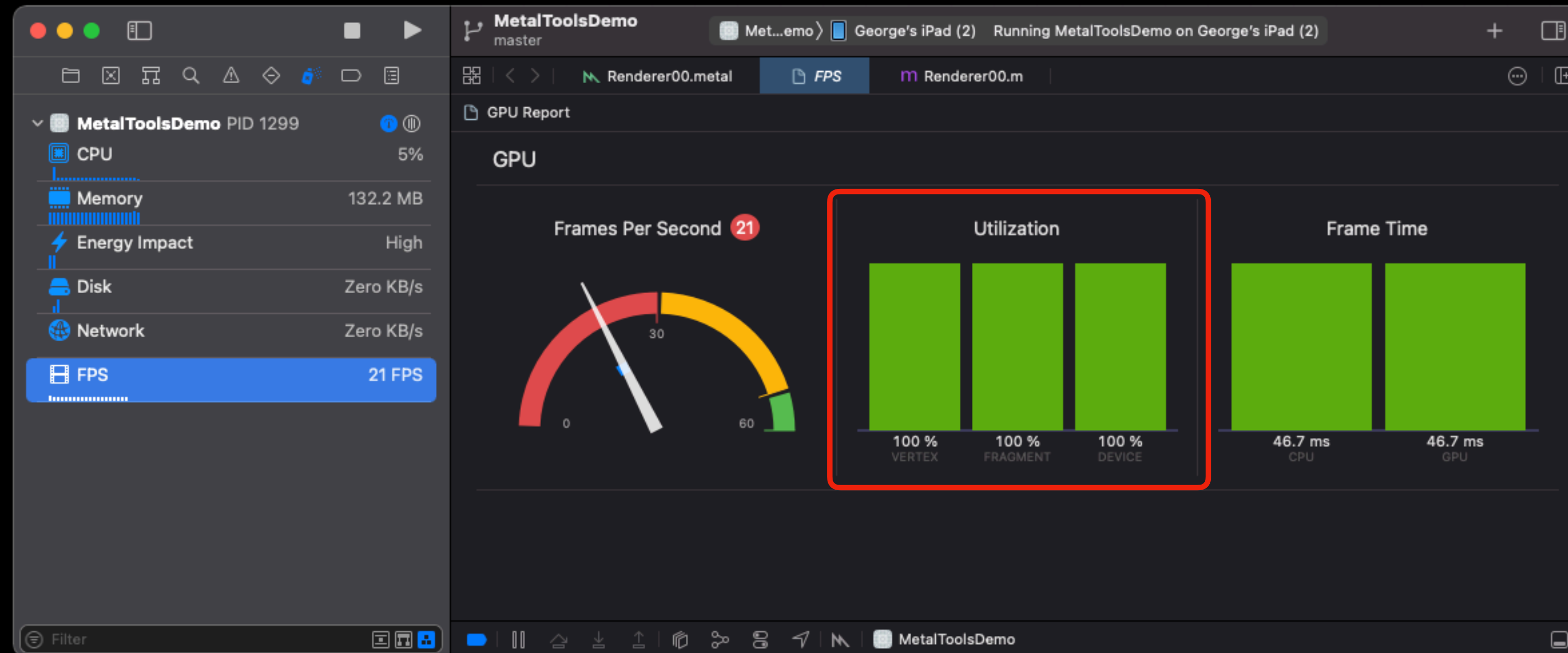
Debug Navigator: GPU



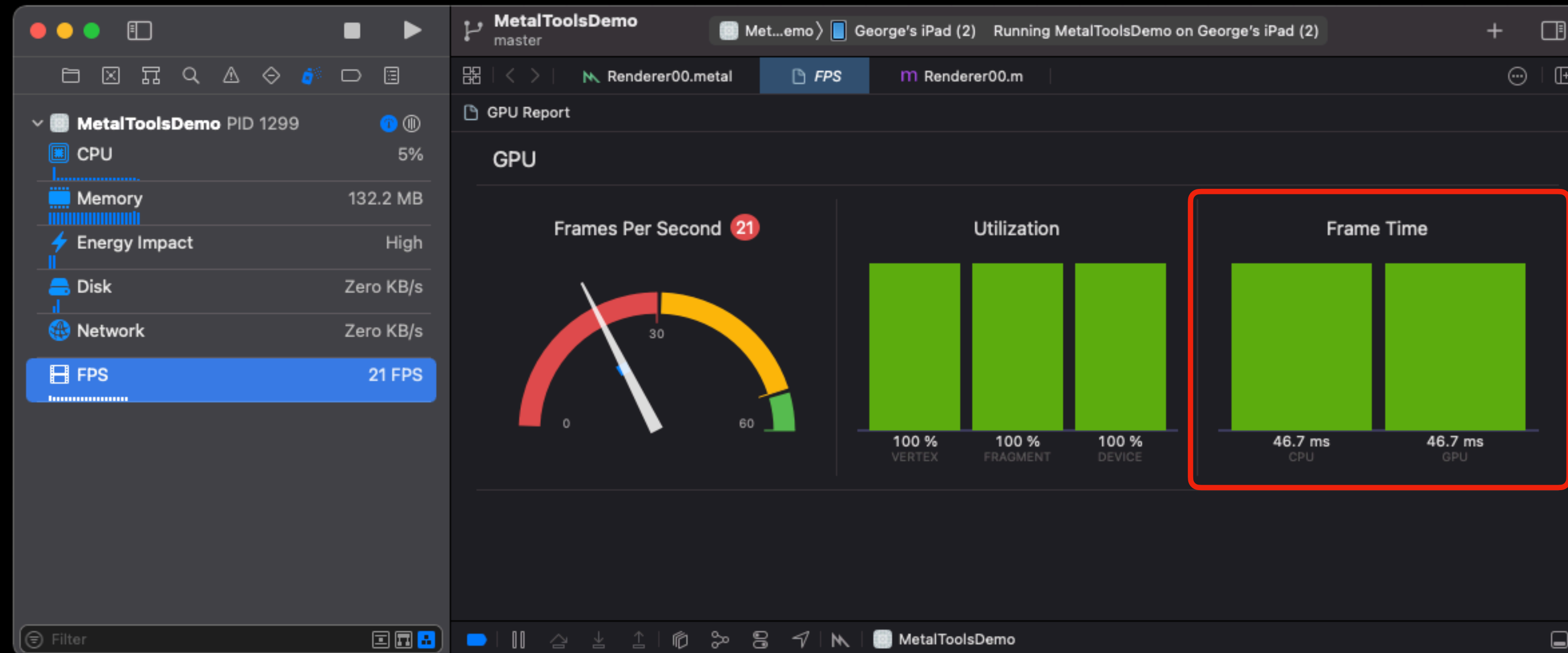
Debug Navigator: GPU



Debug Navigator: GPU



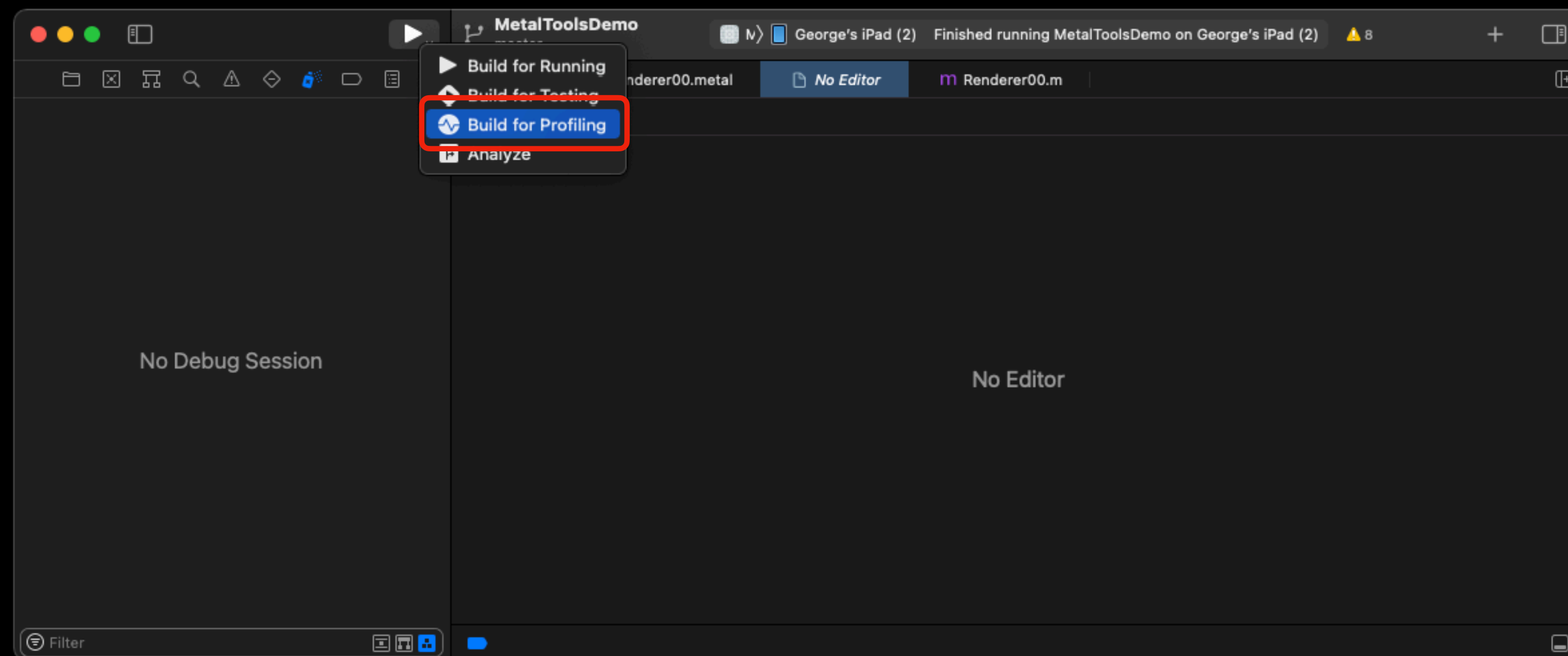
Debug Navigator: GPU



INSTRUMENTS

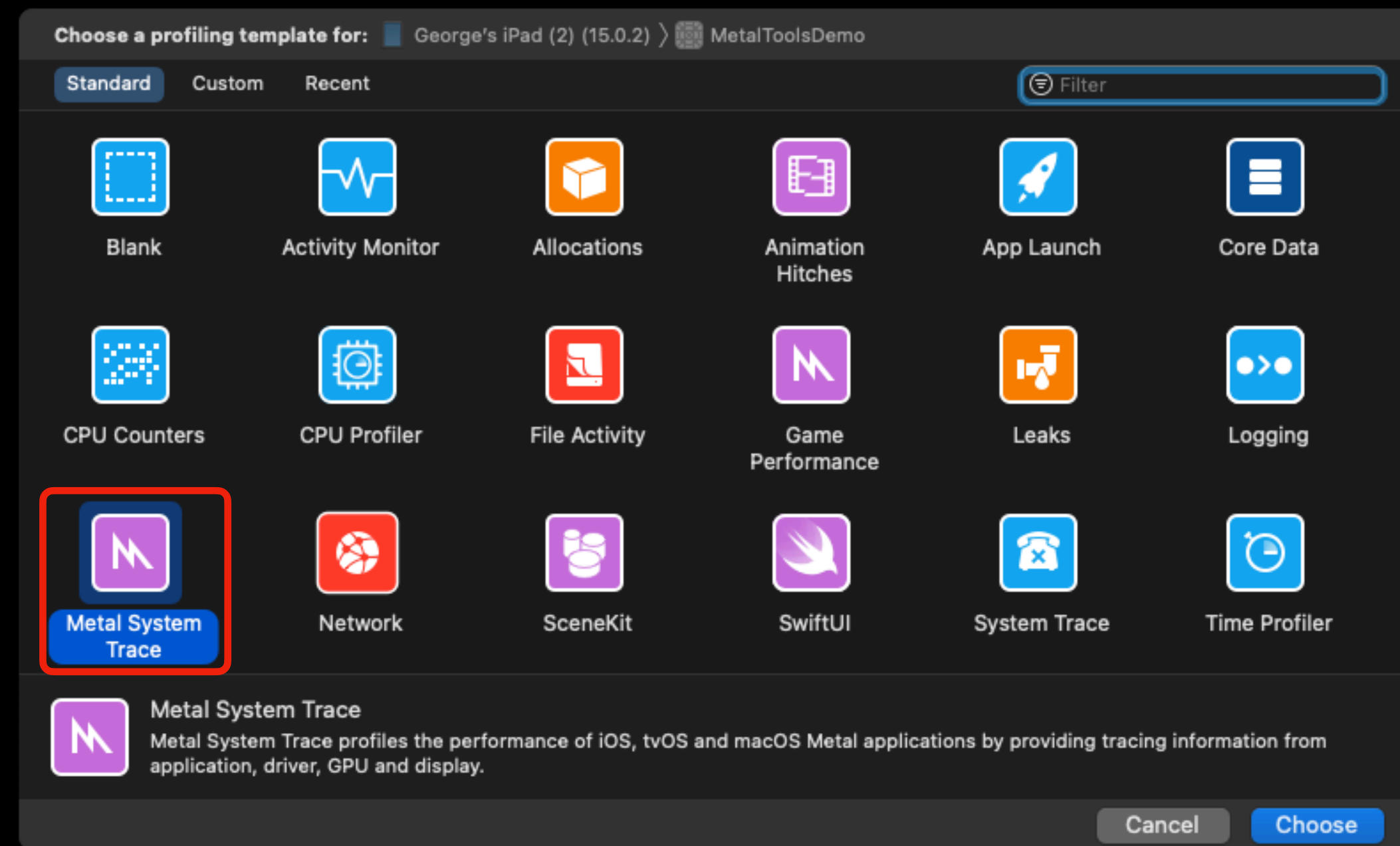
Metal System Trace

Profiling: Metal System Trace

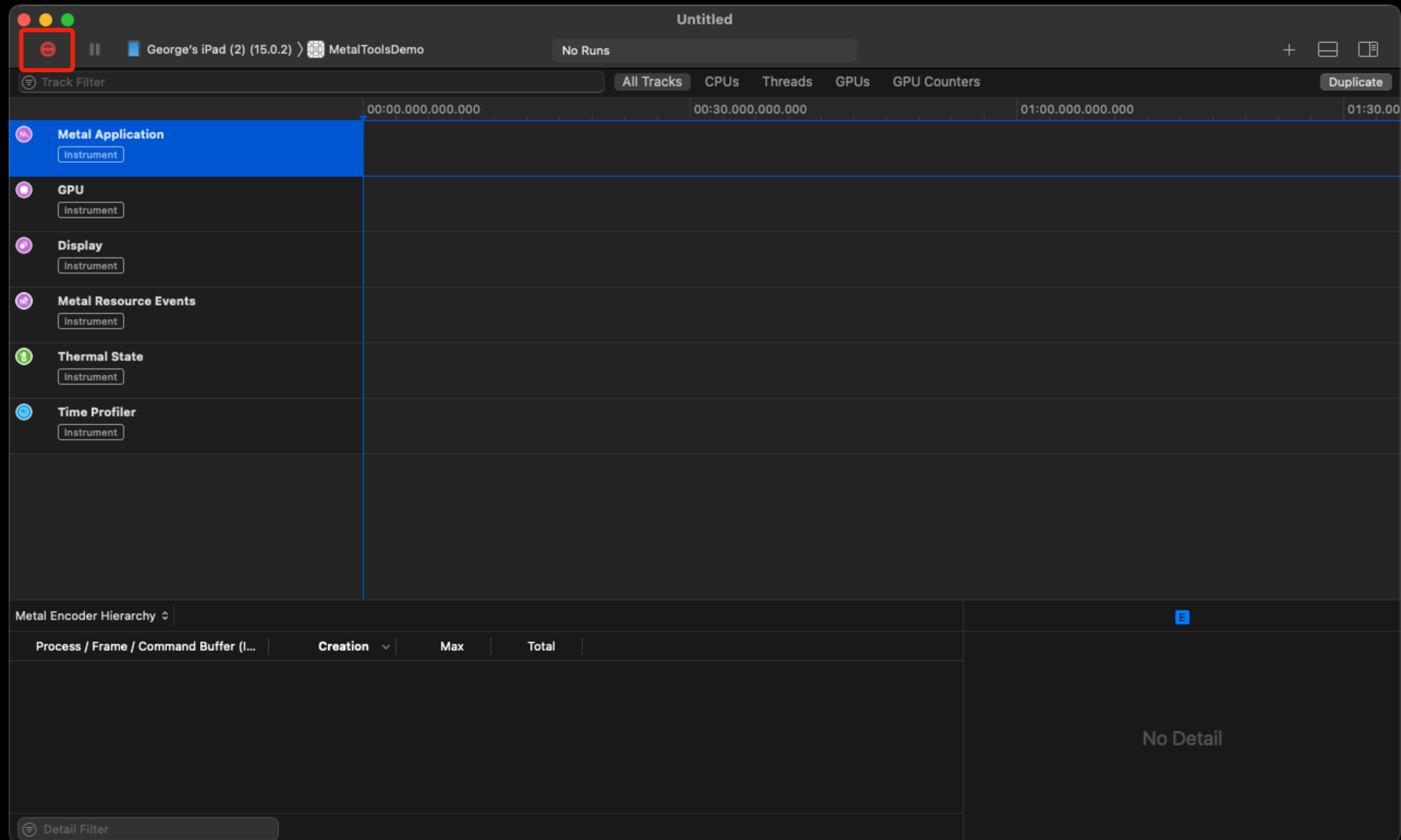


Product → Profile (⌘ + I)

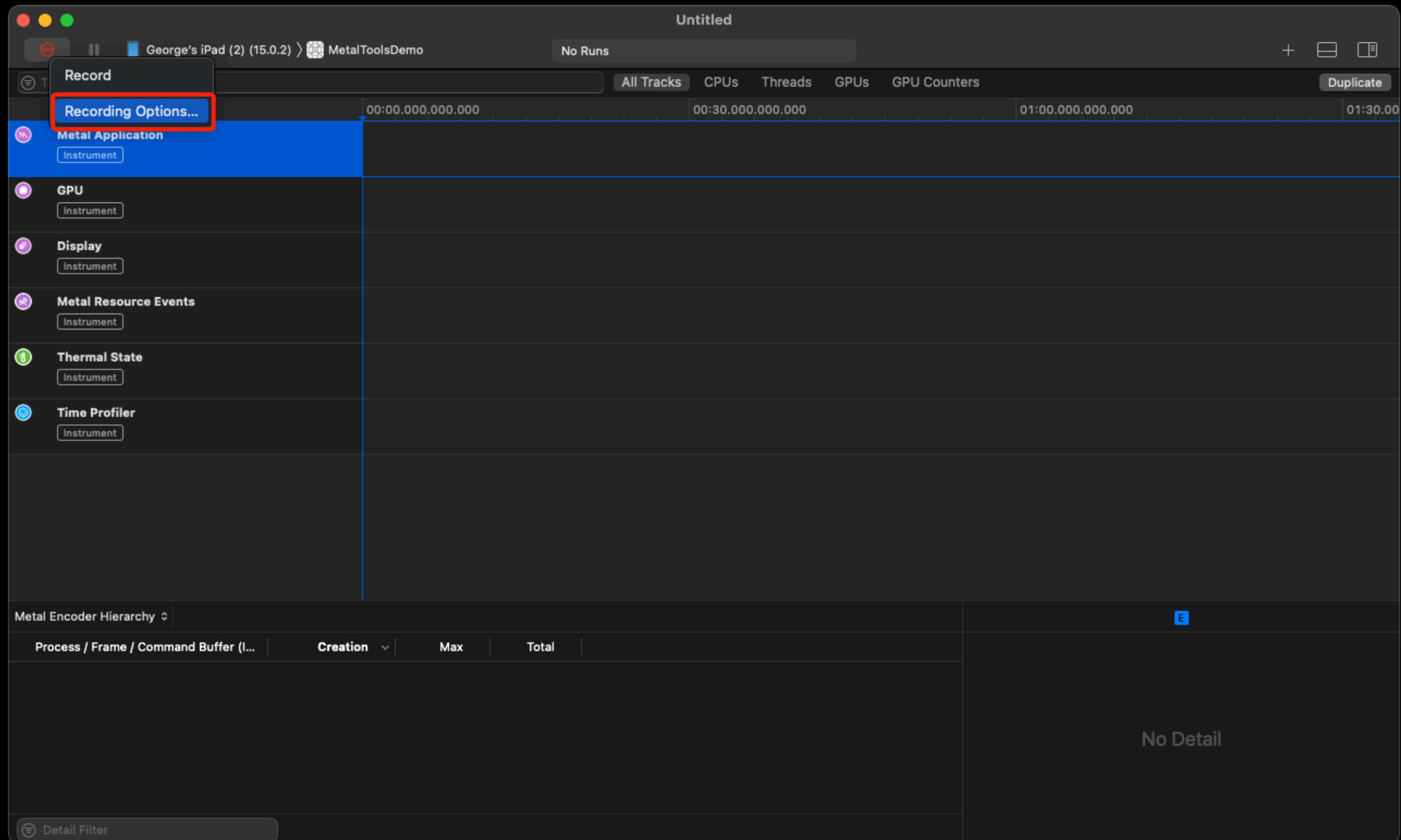
Profiling: Metal System Trace



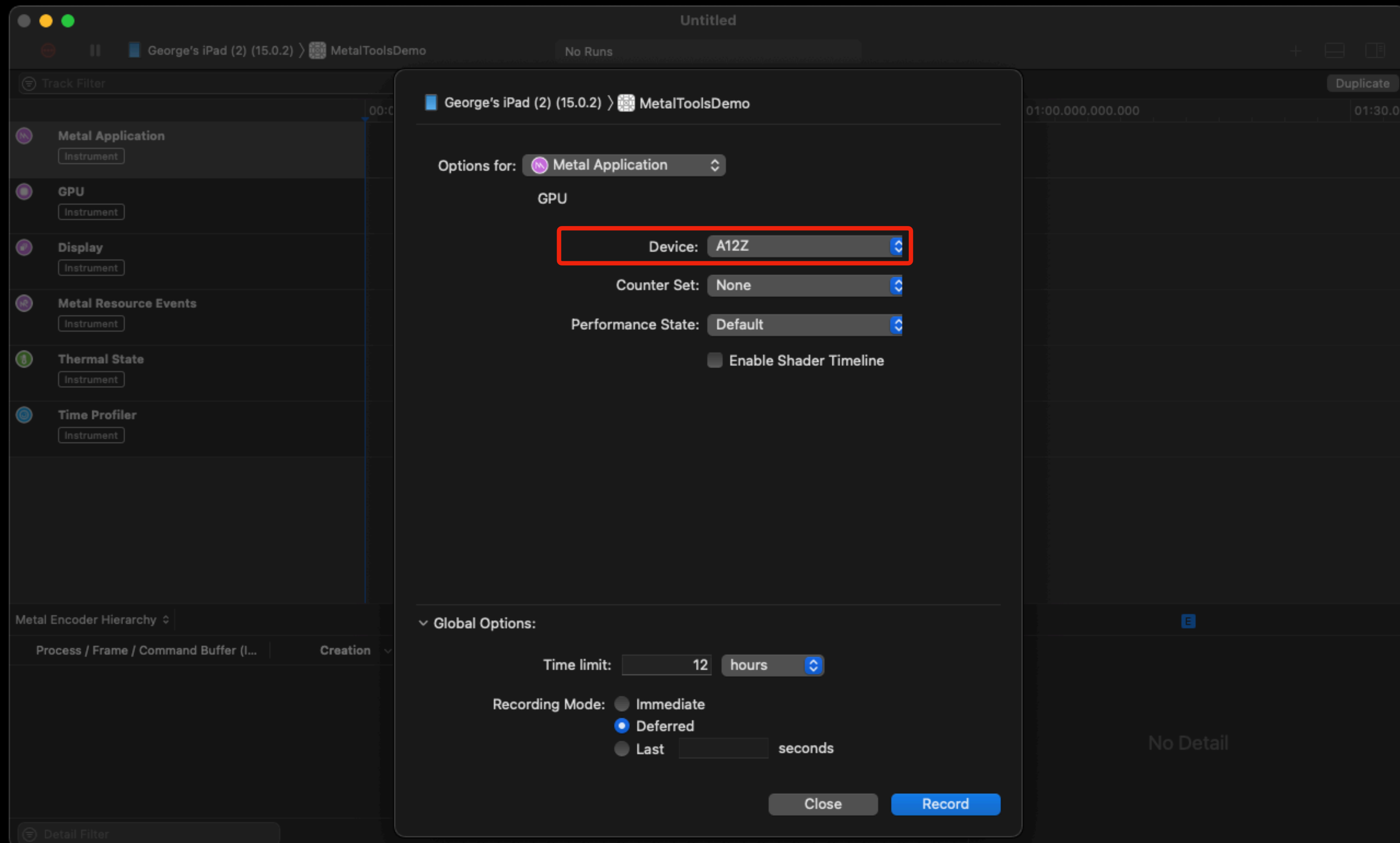
Recording Options



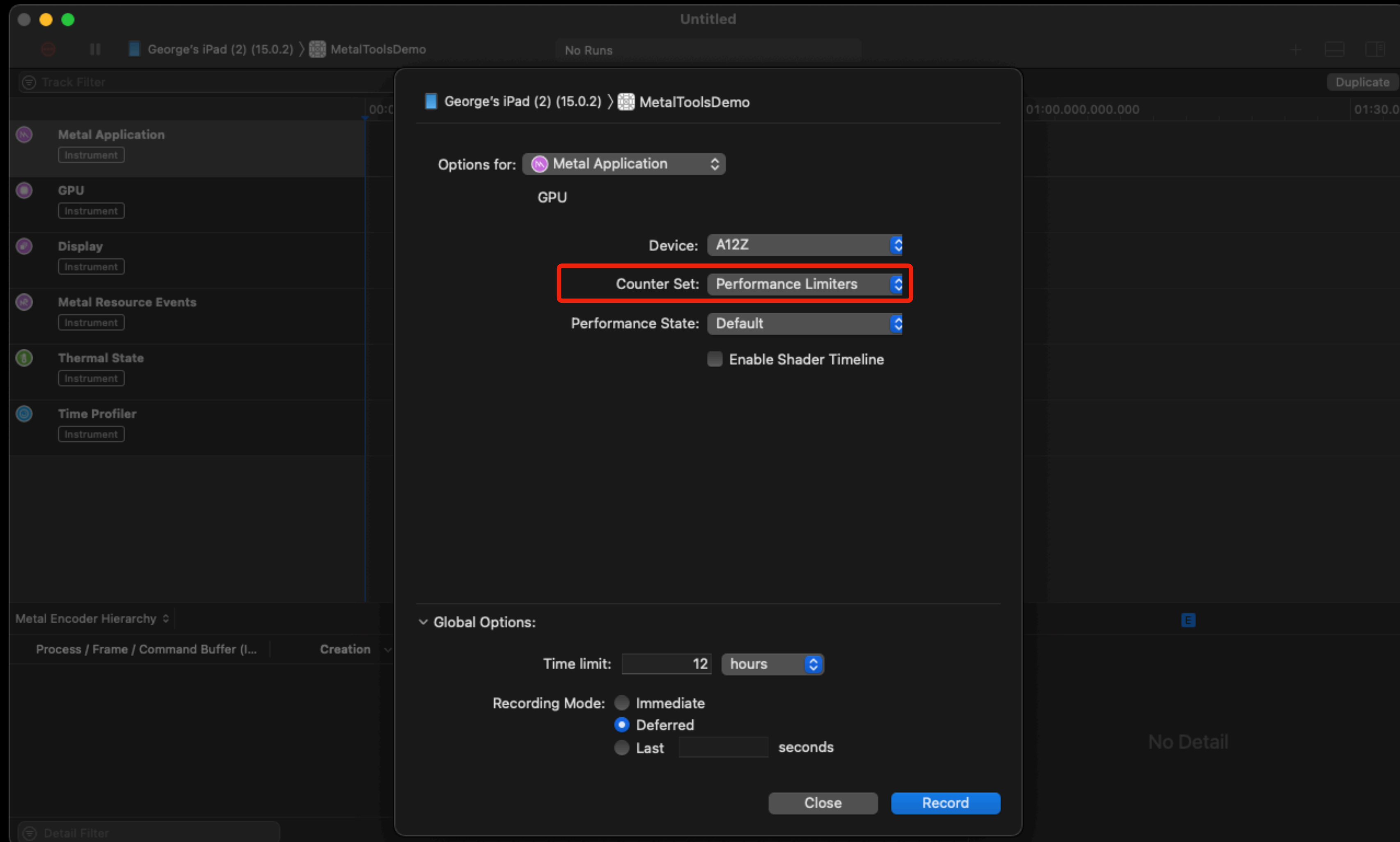
Recording Options



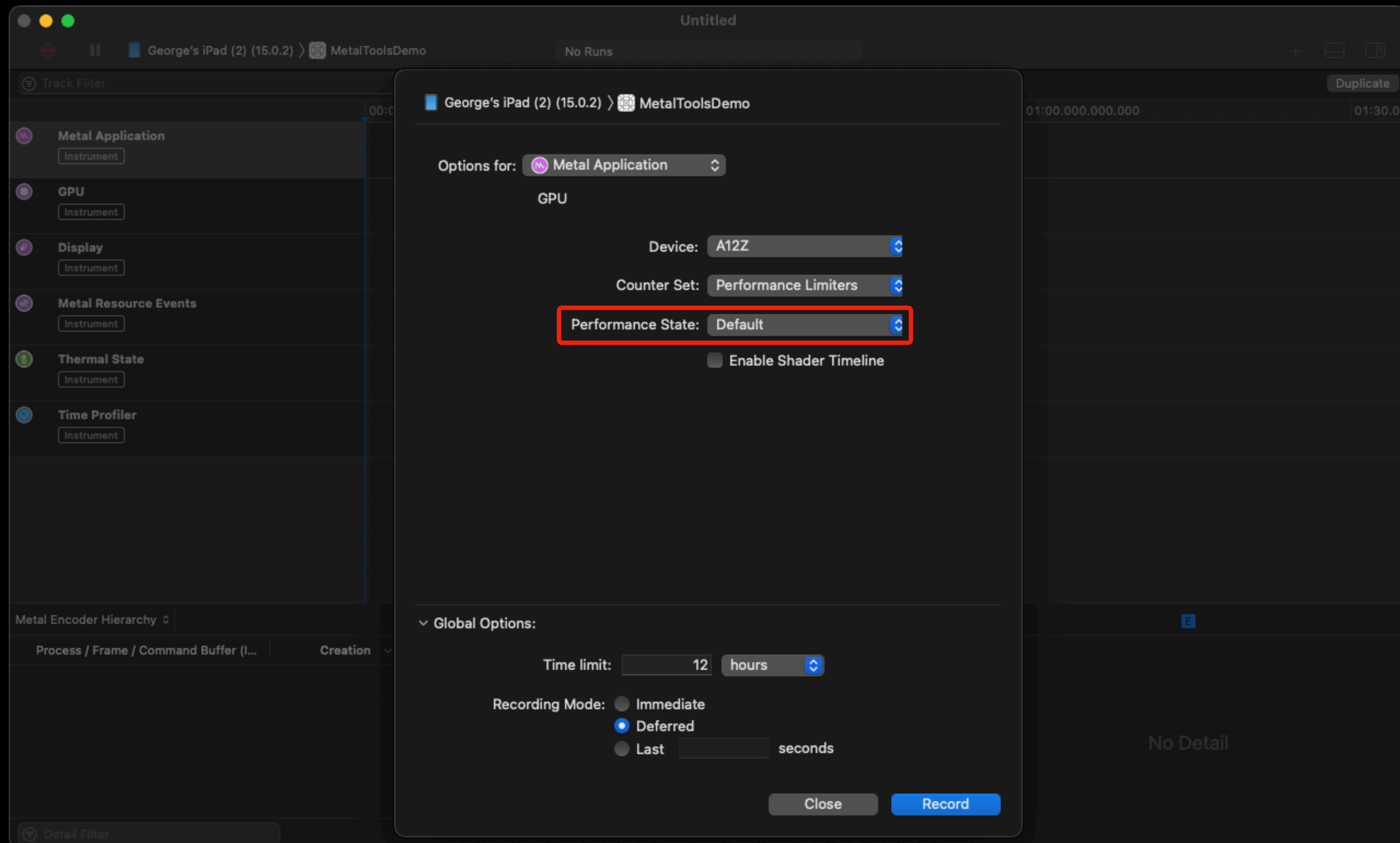
Recording Options



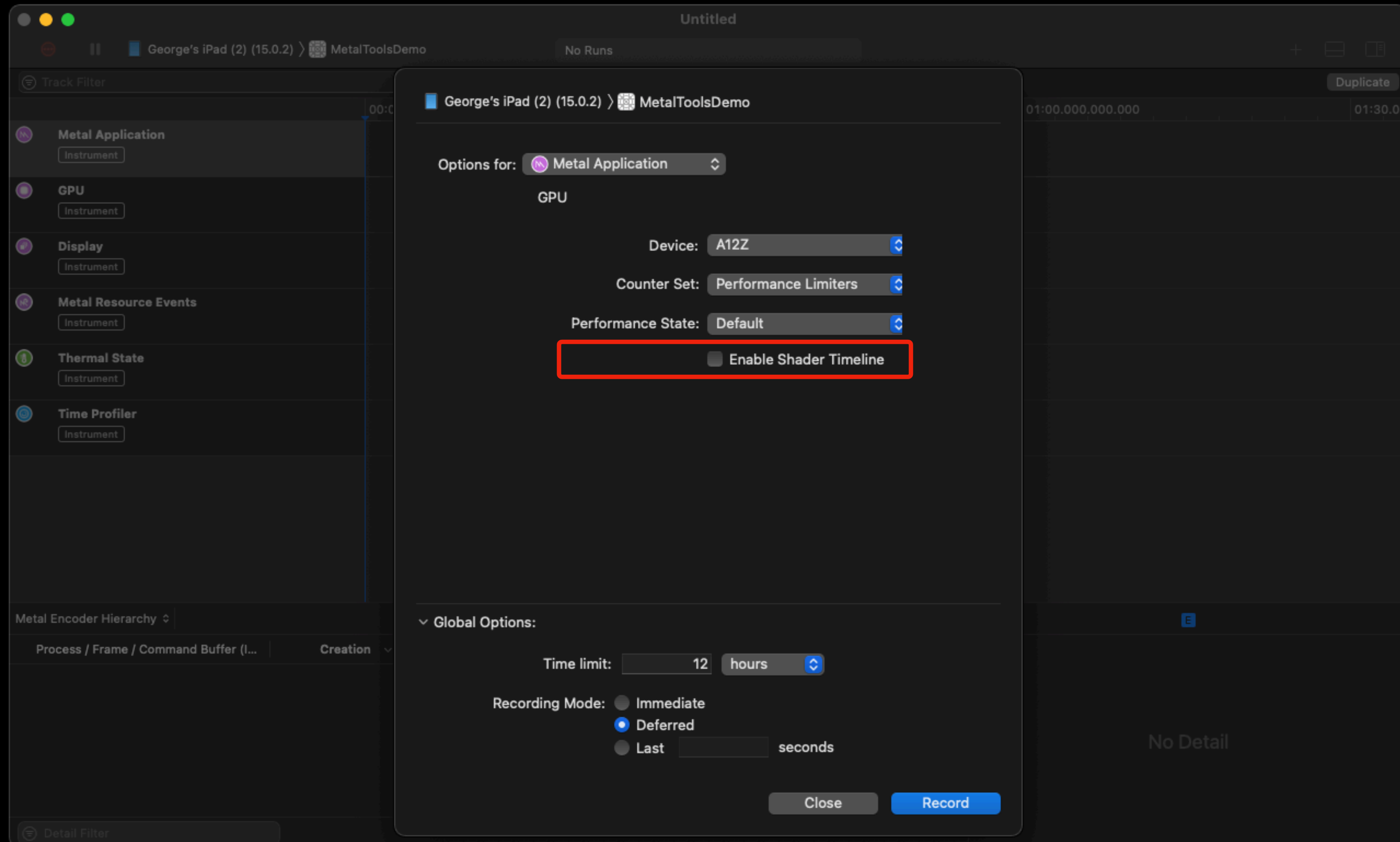
Recording Options



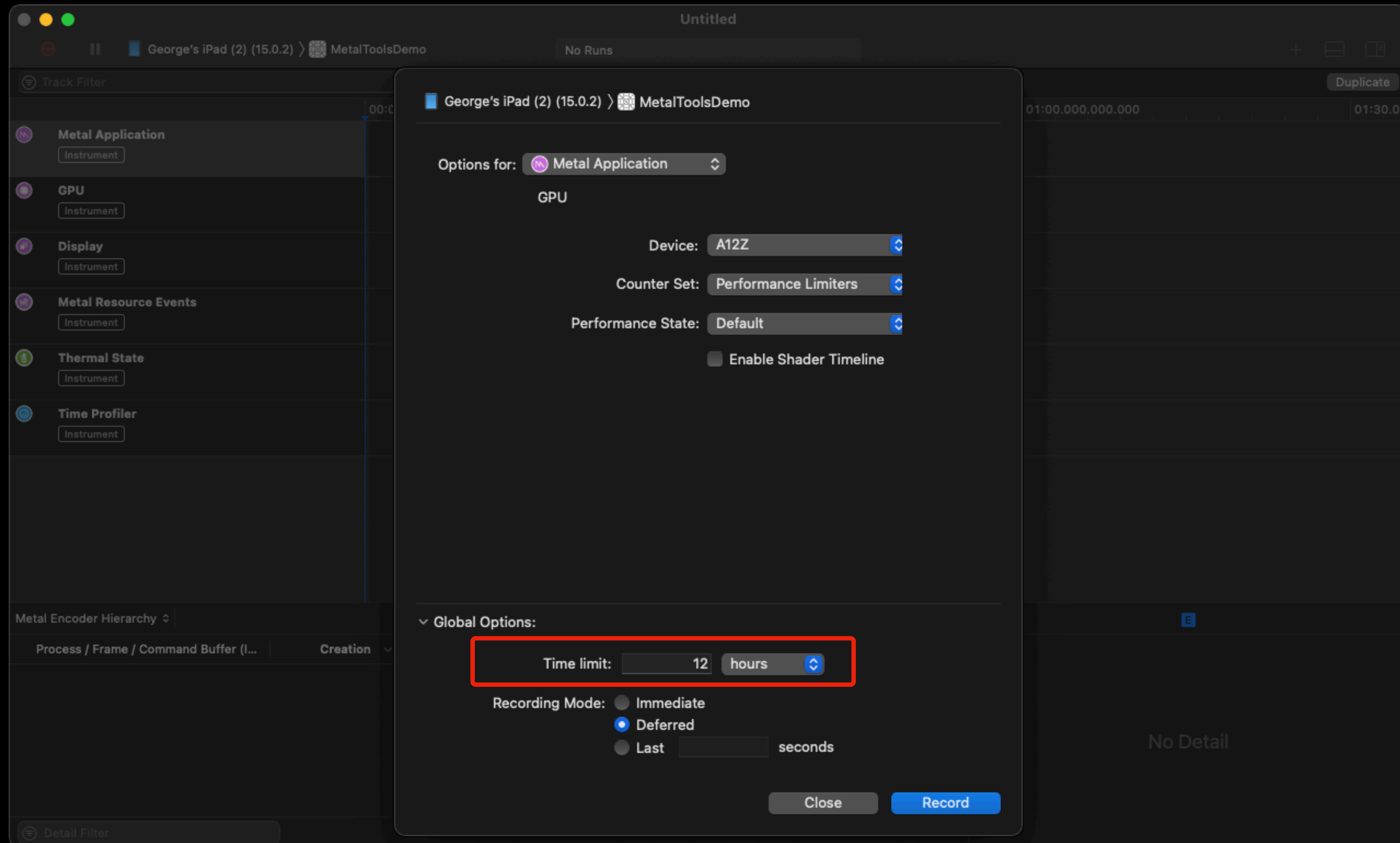
Recording Options



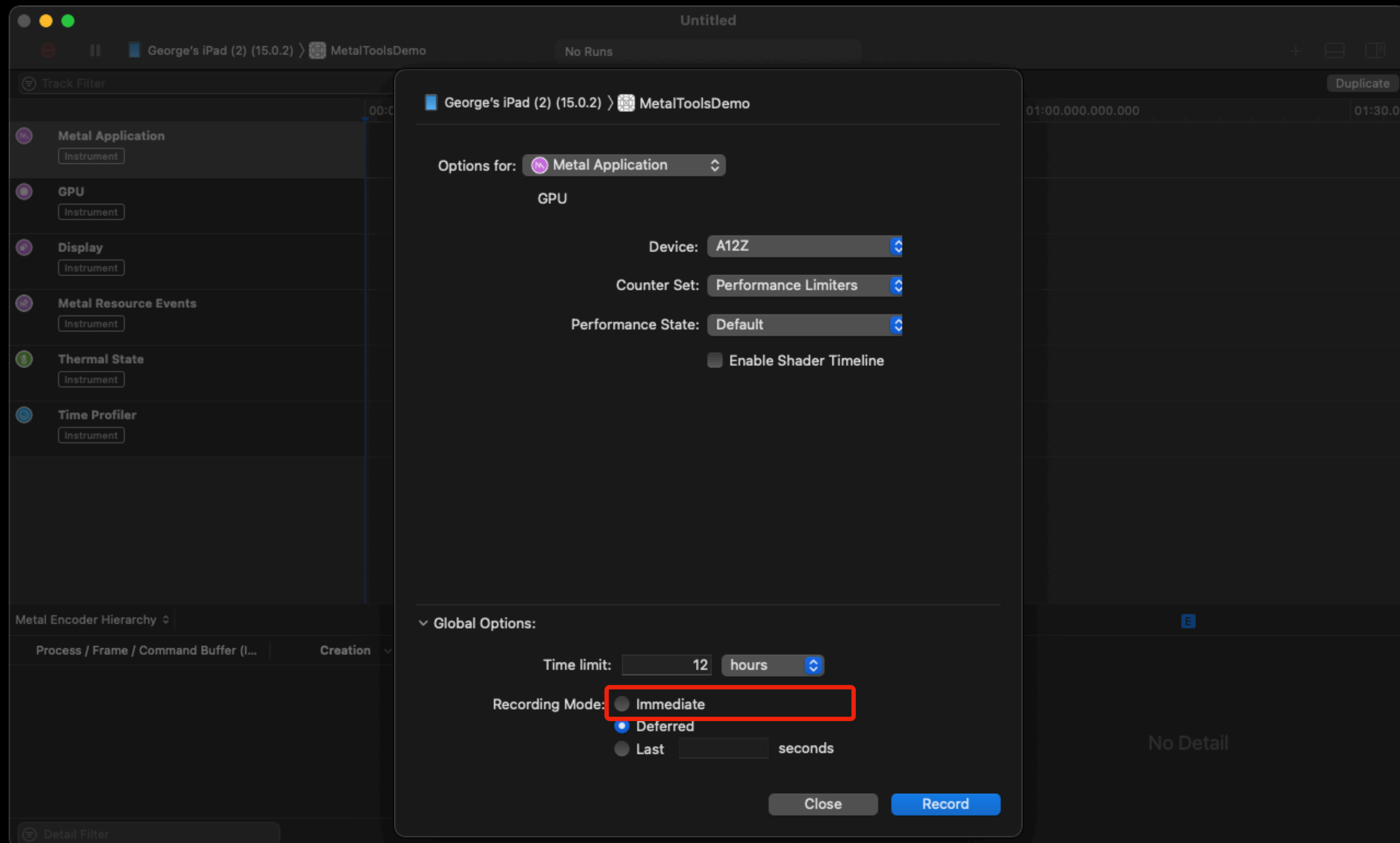
Recording Options



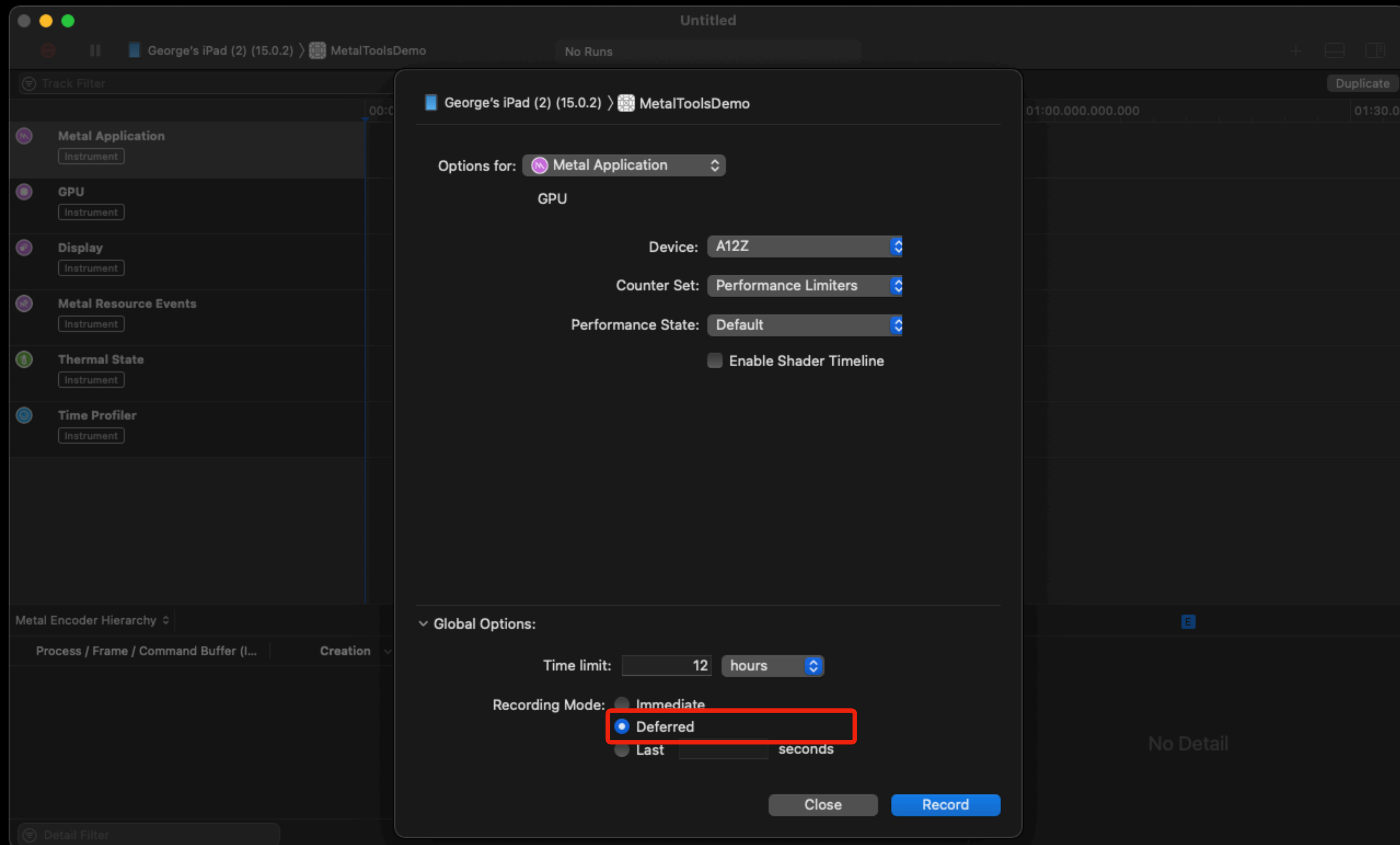
Recording Options



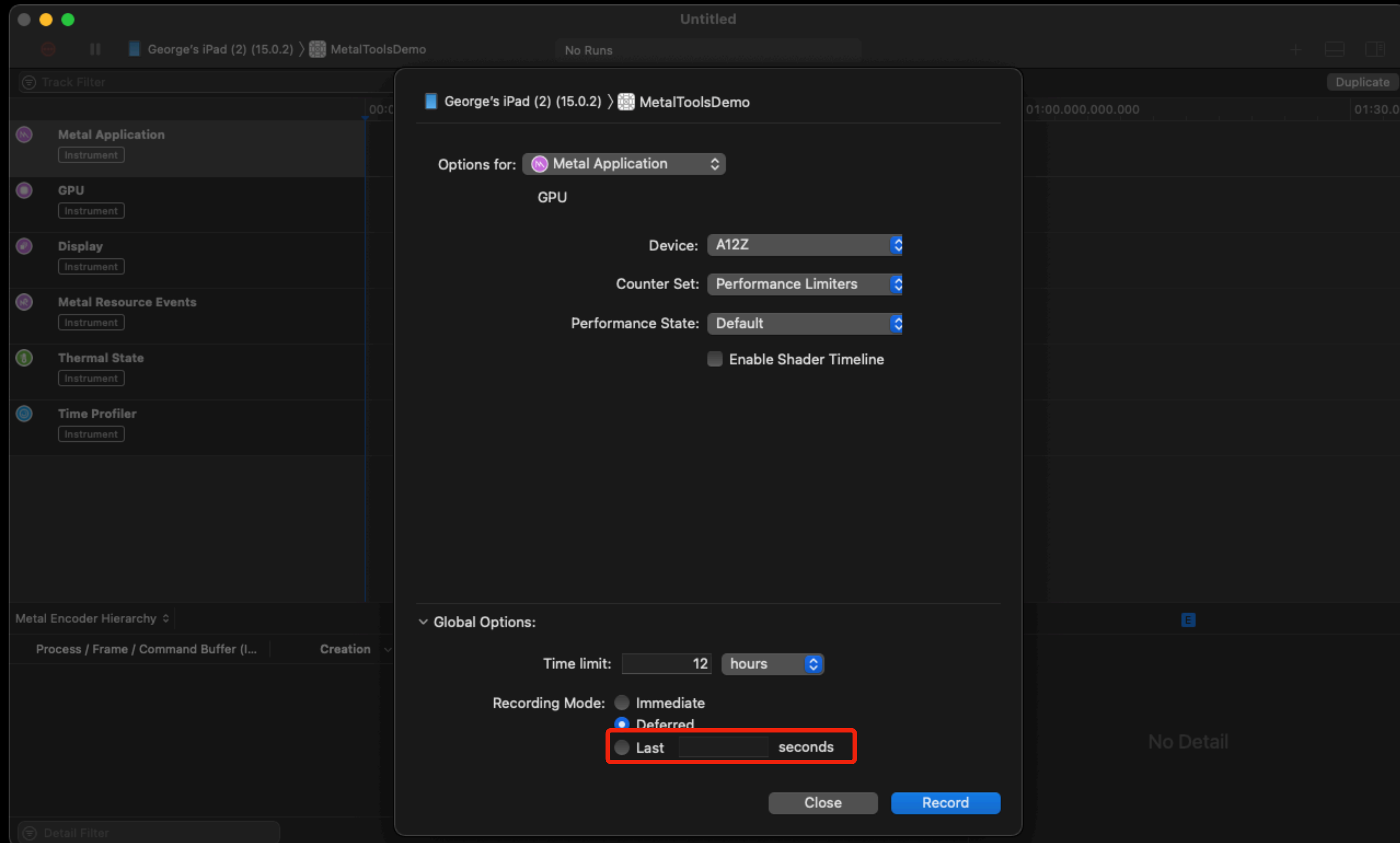
Recording Options



Recording Options



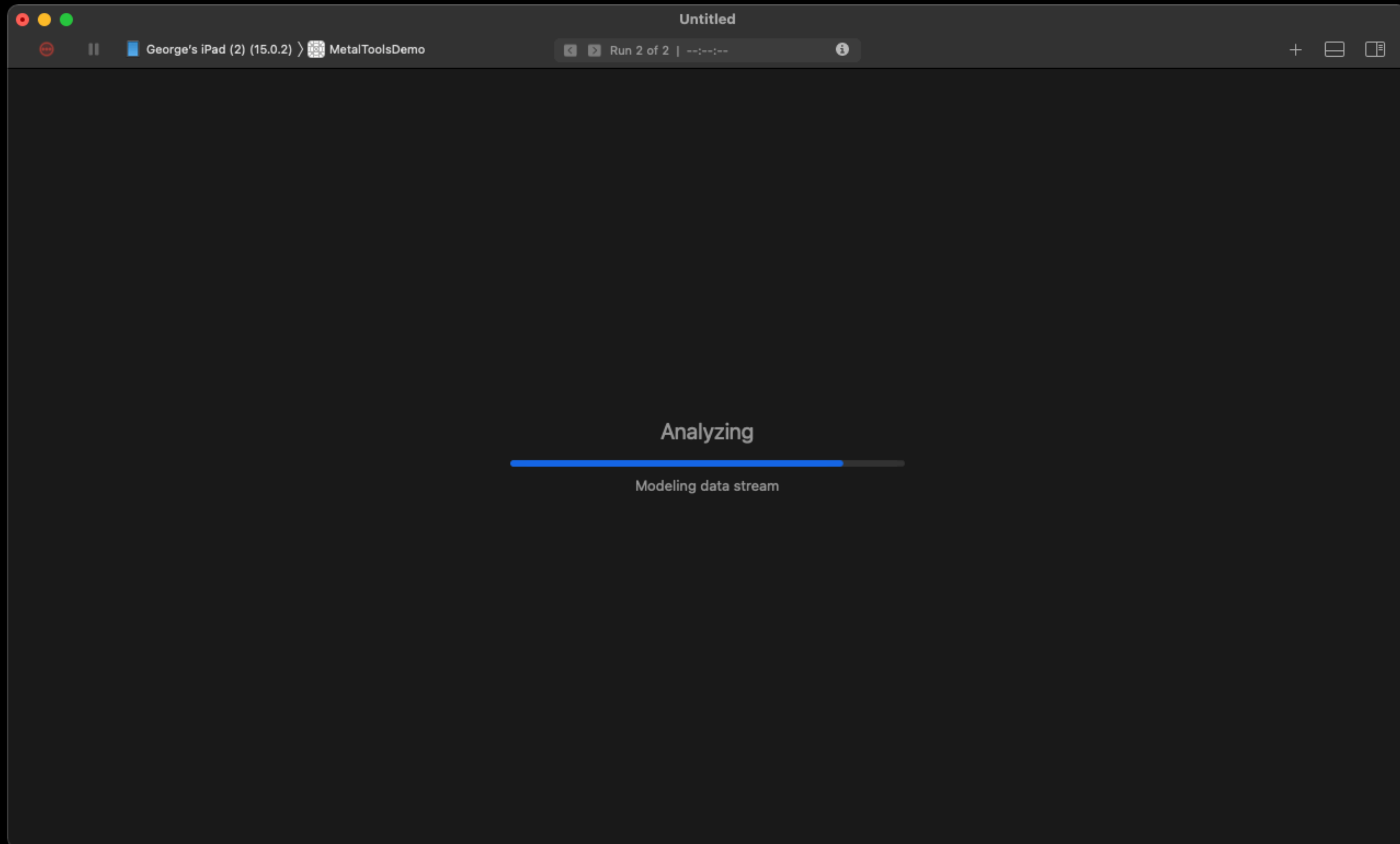
Recording Options



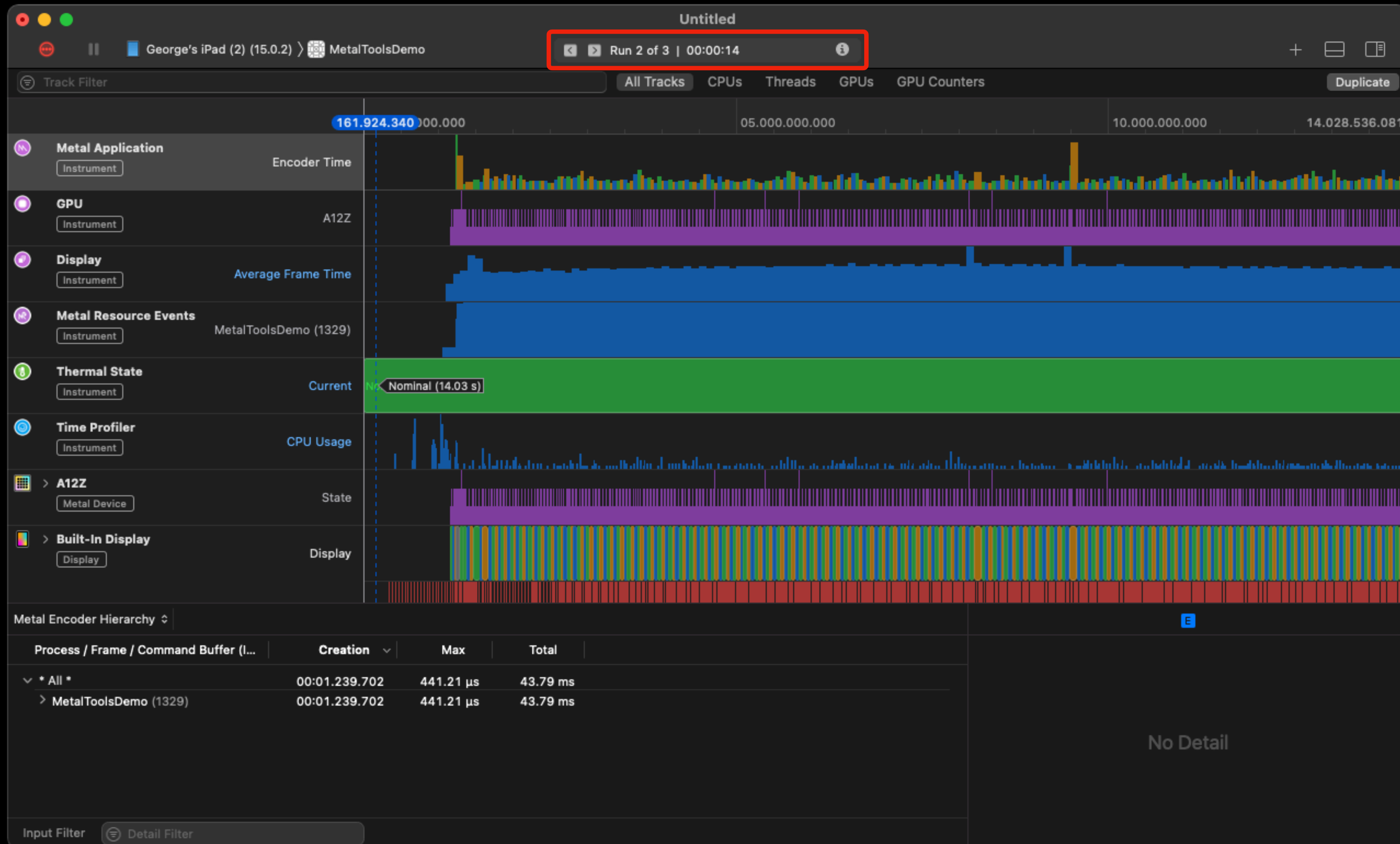
Recording



Recording



Records



Records

The screenshot displays the Xcode Instruments interface for a recording on George's iPad (2). The main window shows a timeline with several instrument tracks: Metal Application (Encoder Time), GPU (A12Z), Display (Average Frame Time), Metal Resource Events (MetalToolsDemo (1329)), Thermal State (Current: Nominal), Time Profiler (CPU Usage), A12Z (State), and Built-In Display (Display). A 'Recording Information' dialog is open, providing details about the recording session.

Recording Information

- Target Name: George's iPad (2)
- Target Model: iPad Pro (12.9-inch) (4th generation)
- Target IOS: 15.0.2 (19A404)
- Host Name: George's MacBook Pro (1466)
- Host Model: MacBook Pro
- Host macOS: 11.6 (20G165)
- Template: Metal System Trace
- Start: 25 Oct 2021 at 10:01:45 am
- End: 25 Oct 2021 at 10:01:59 am
- Duration: 14 seconds
- End Reason: User pressed Stop
- Instruments Version: 13.0 (13A233)

Recording Settings

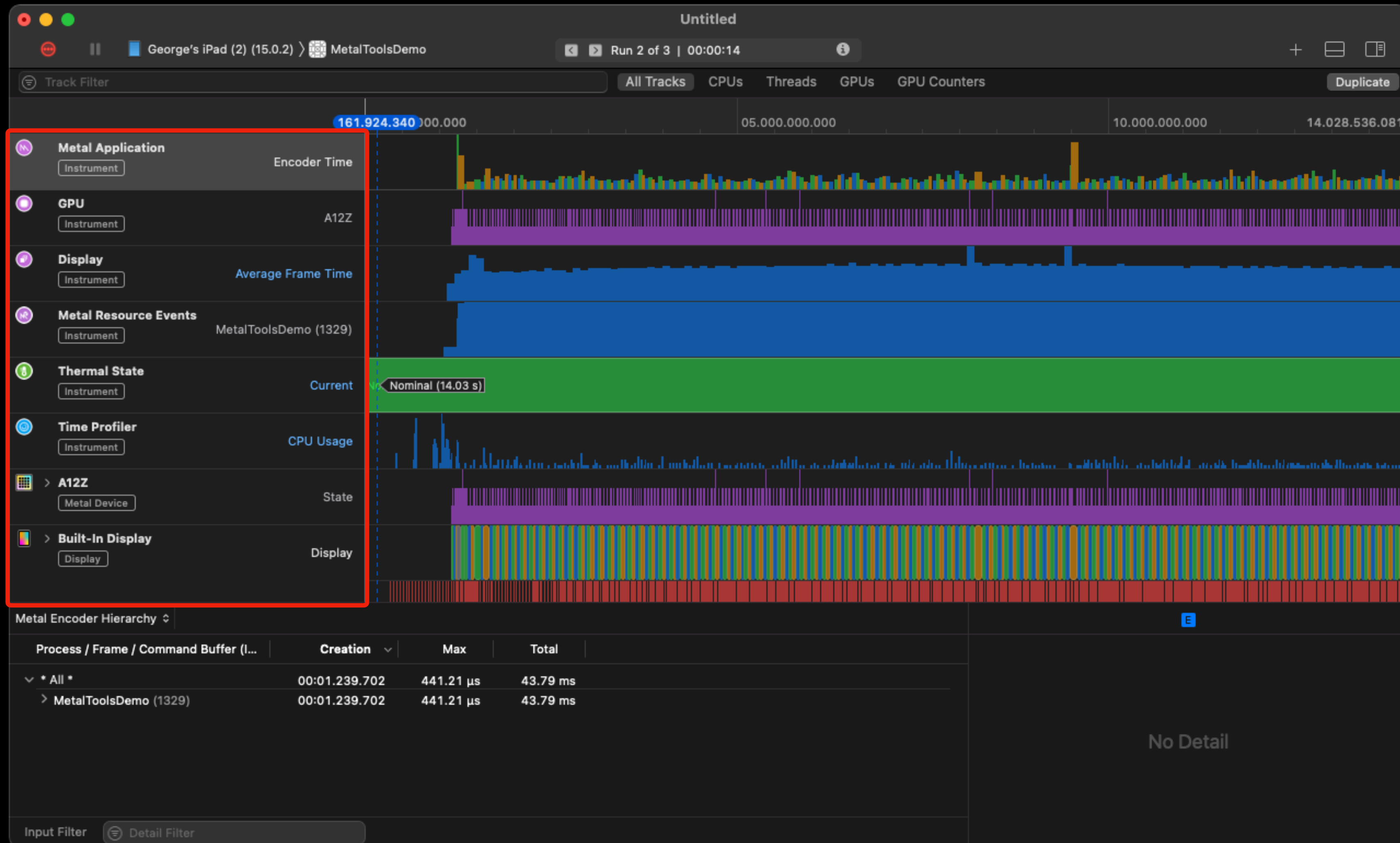
- Target: MetalToolsDemo (1329)
- Recording Mode: Deferred
- Time Limit: 12 hours
- GPU Counter Set: Performance Limiters
- GPU Shader Timeline: Enabled
- GPU Induced GPU Performance State: Default

Metal Encoder Hierarchy

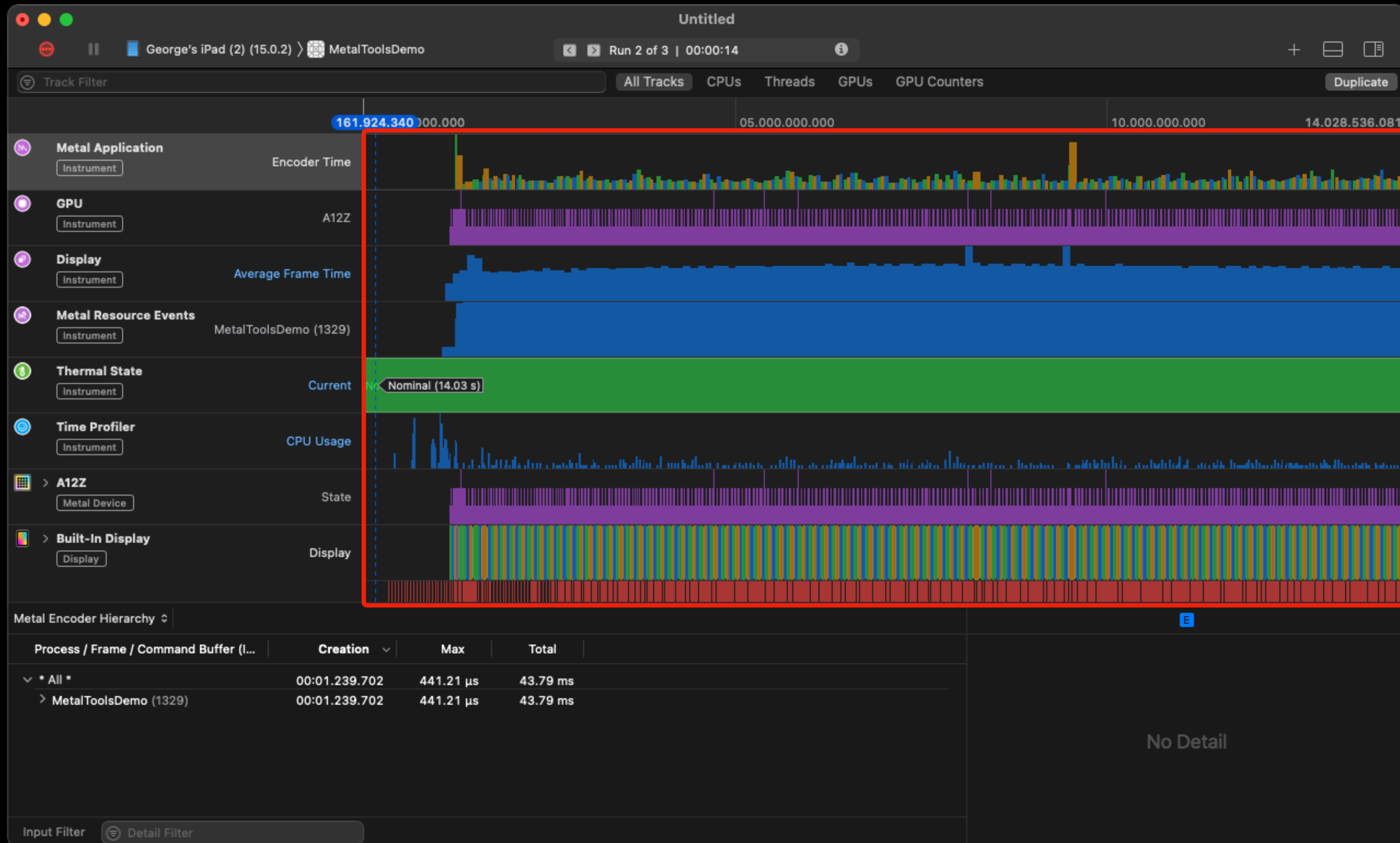
Process / Frame / Command Buffer (l...	Creation	Max	Total
* All *	00:01.239.702	441.21 μ s	43.79 ms
> MetalToolsDemo (1329)	00:01.239.702	441.21 μ s	43.79 ms

Input Filter: Detail Filter

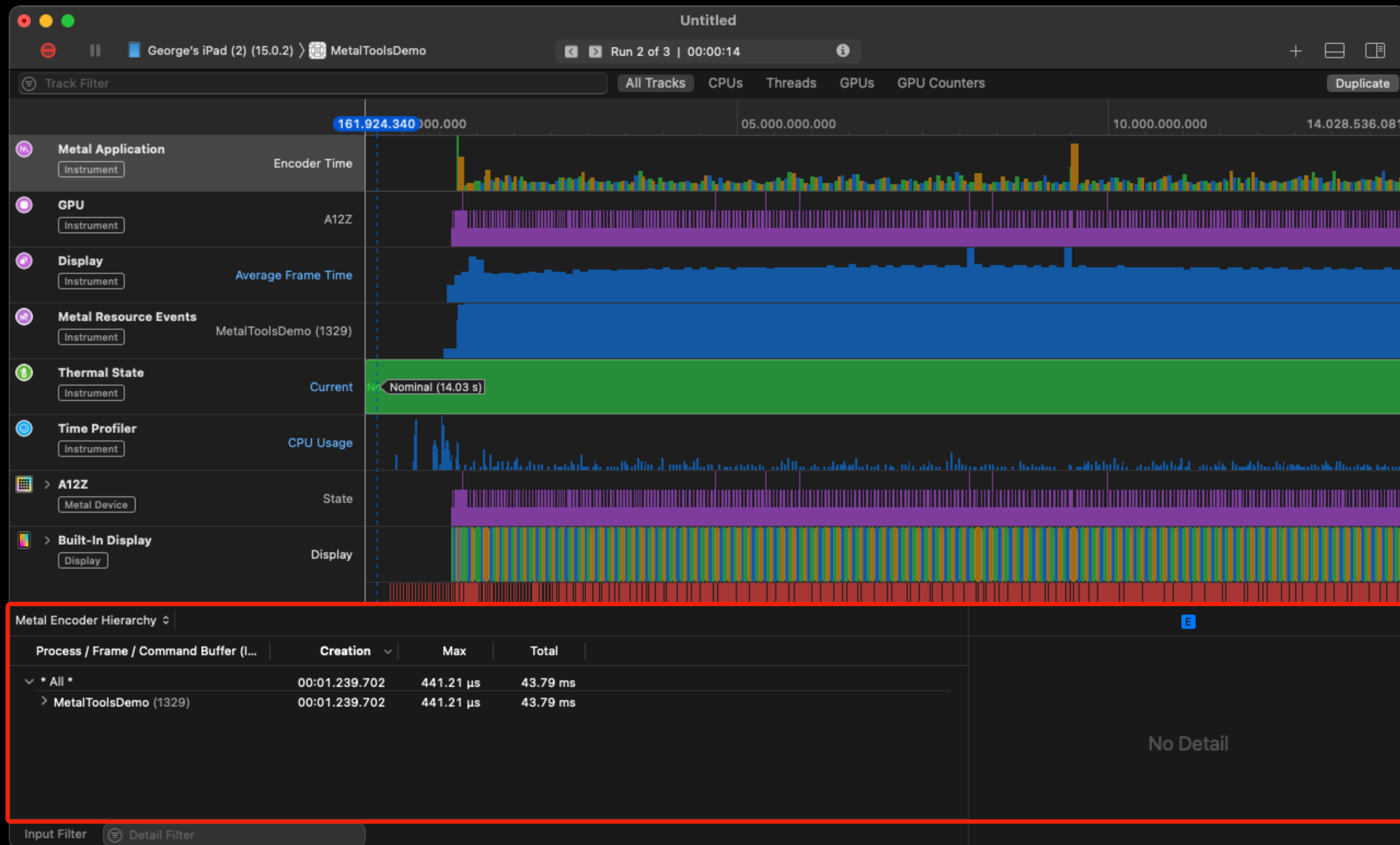
Tracks



Tracks



Details



Tracks: Filters

Untitled

George's iPad (2) (15.0.2) MetalToolsDemo Run 2 of 3 | 00:00:14

METAL DEVICE All Tracks CPUs Threads GPUs GPU Counters Duplicate

00.000.000.000 05.000.000.000 10.000.000.000 14.028.536.081

> A12Z
Metal Device State

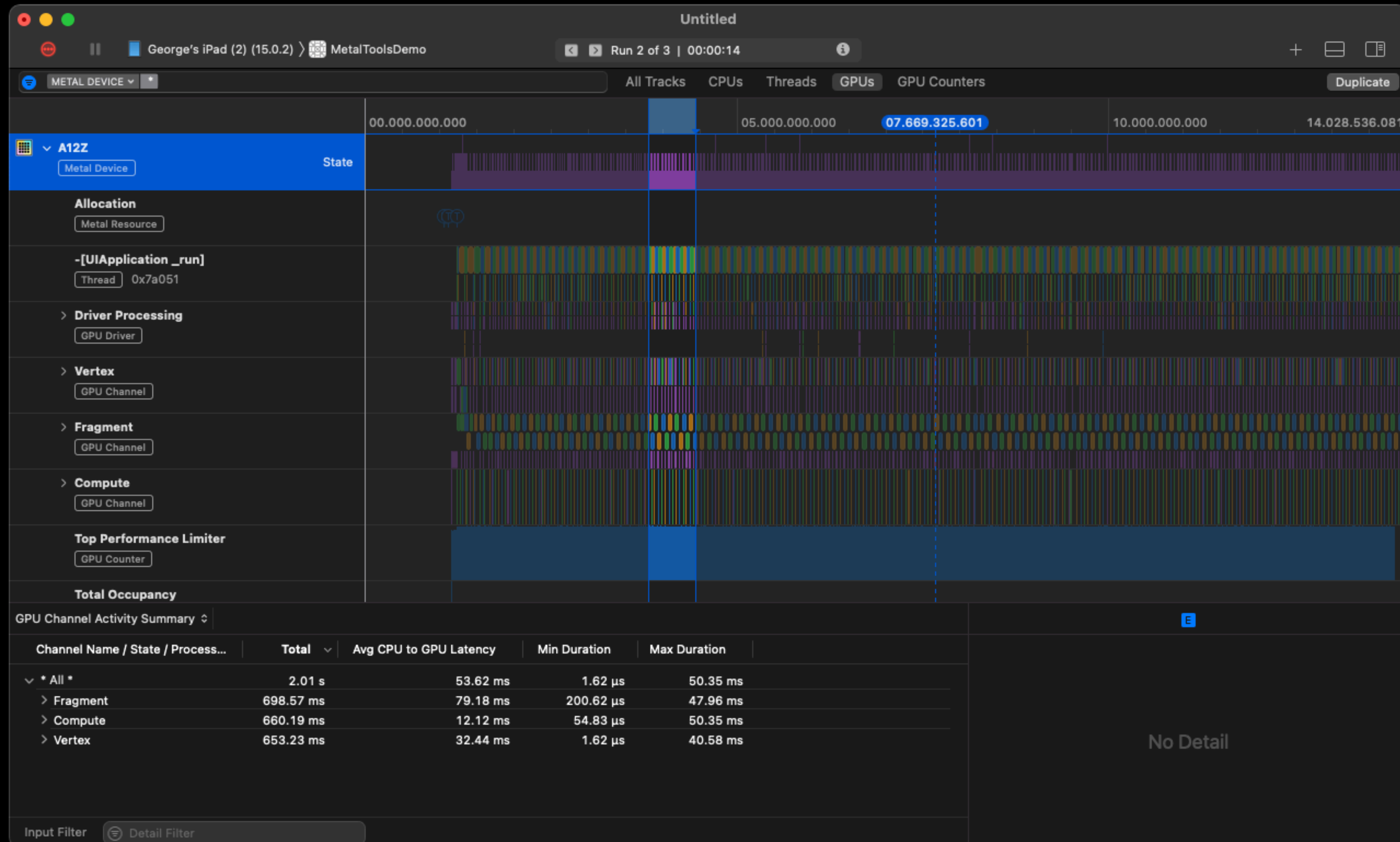
Metal Encoder Hierarchy

Process / Frame / Command Buffer (l...	Creation	Max	Total
* All *	00:01.239.702	441.21 μs	43.79 ms
> MetalToolsDemo (1329)	00:01.239.702	441.21 μs	43.79 ms

No Detail

Input Filter Detail Filter

Tracks: Zoom

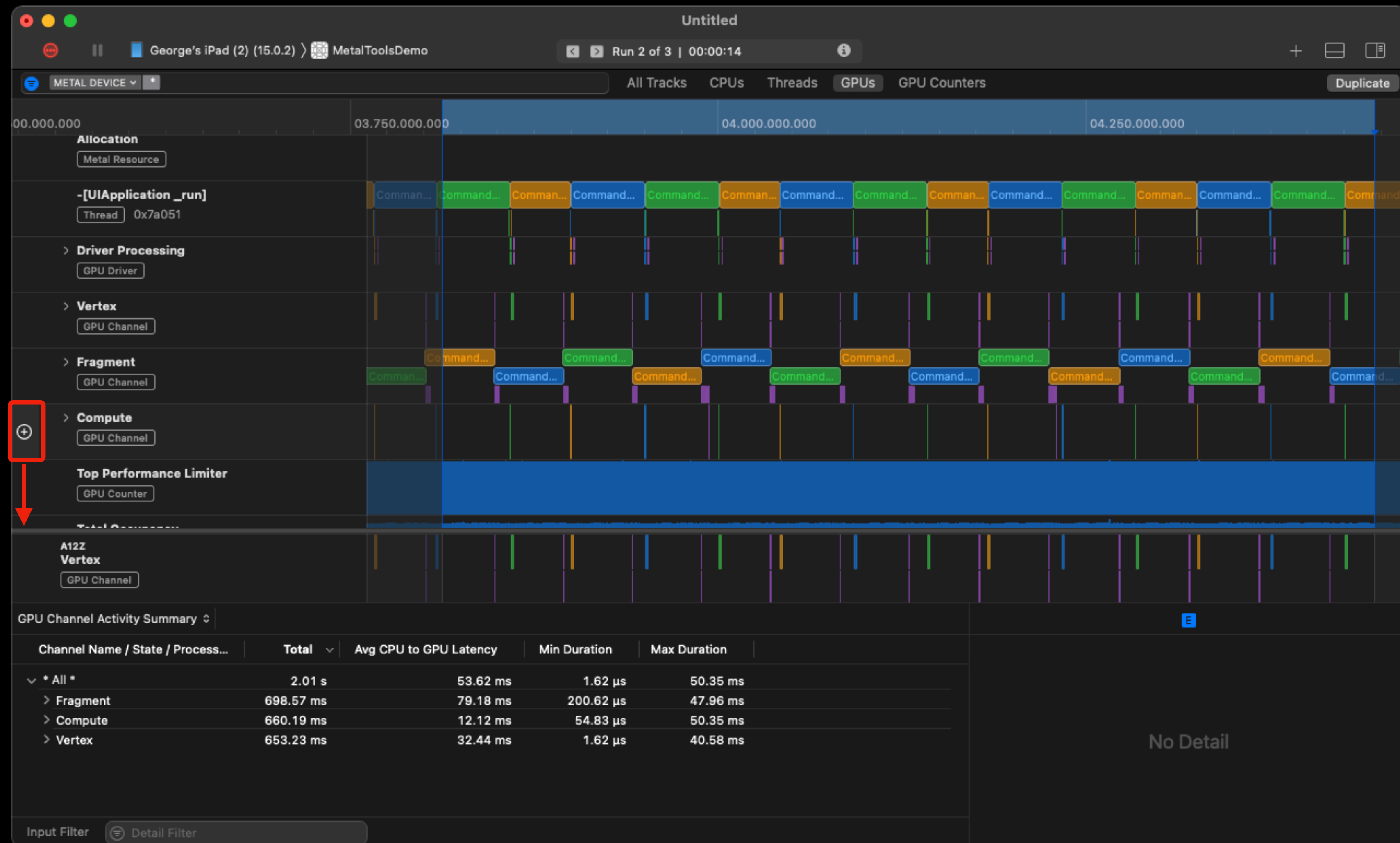


Tracks: Zoom

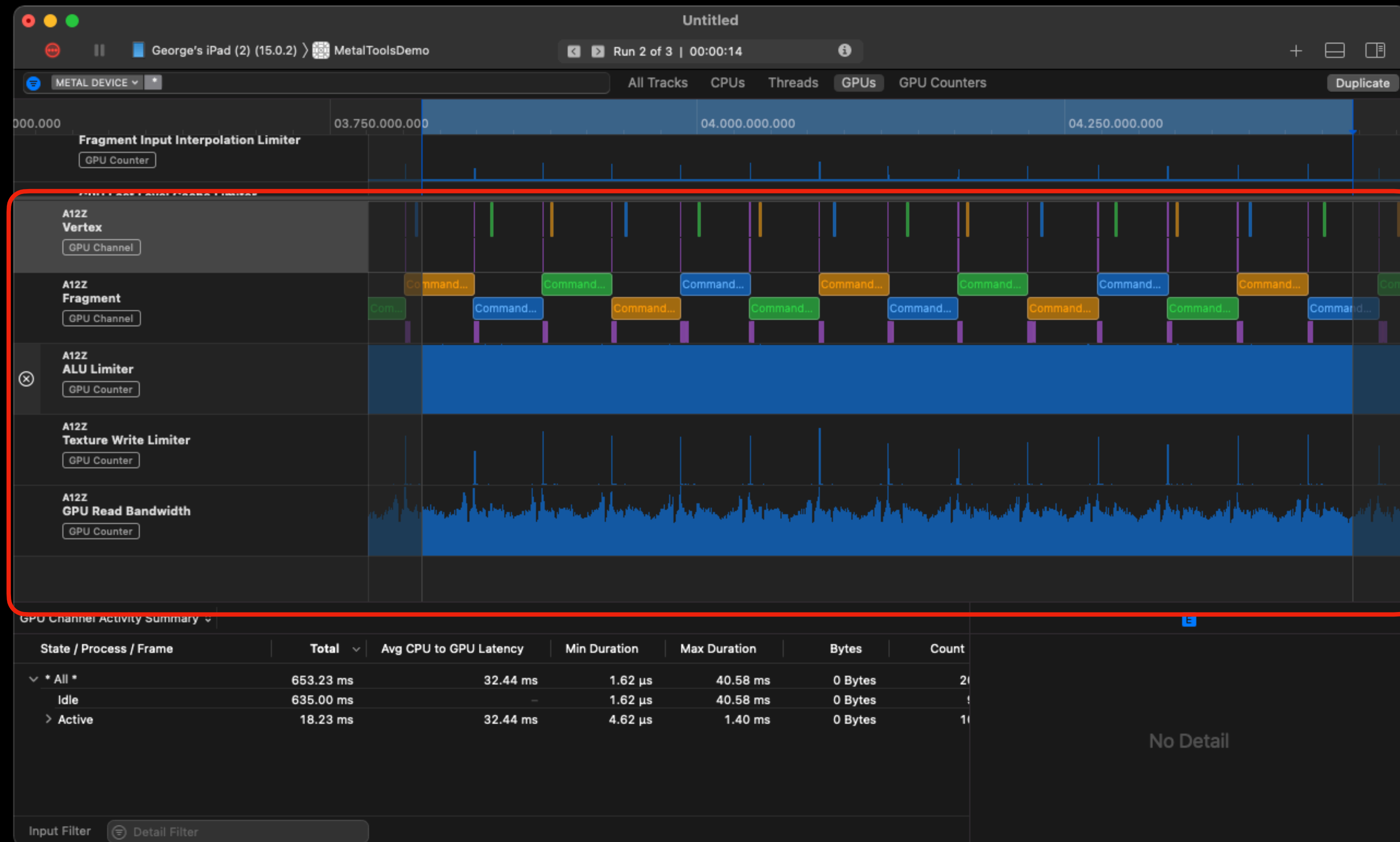
View → Zoom → Zoom to Inspection Range (^ + ⌘ + I)

Channel Name / State / Process...	Total	Avg CPU to GPU Latency	Min Duration	Max Duration
* All *	2.01 s	53.62 ms	1.62 μs	50.35 ms
> Fragment	698.57 ms	79.18 ms	200.62 μs	47.96 ms
> Compute	660.19 ms	12.12 ms	54.83 μs	50.35 ms
> Vertex	653.23 ms	32.44 ms	1.62 μs	40.58 ms

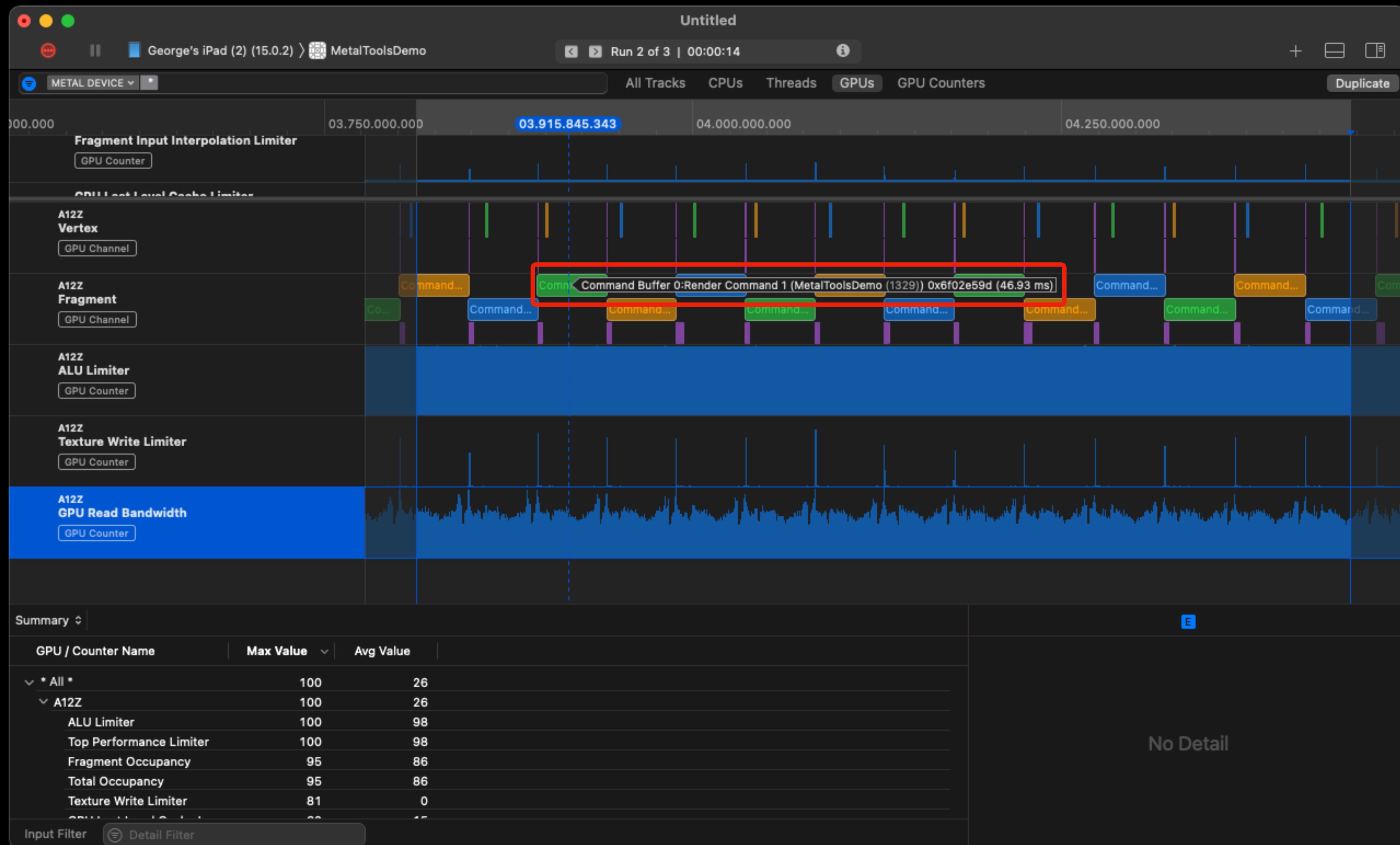
Tracks: Selecting



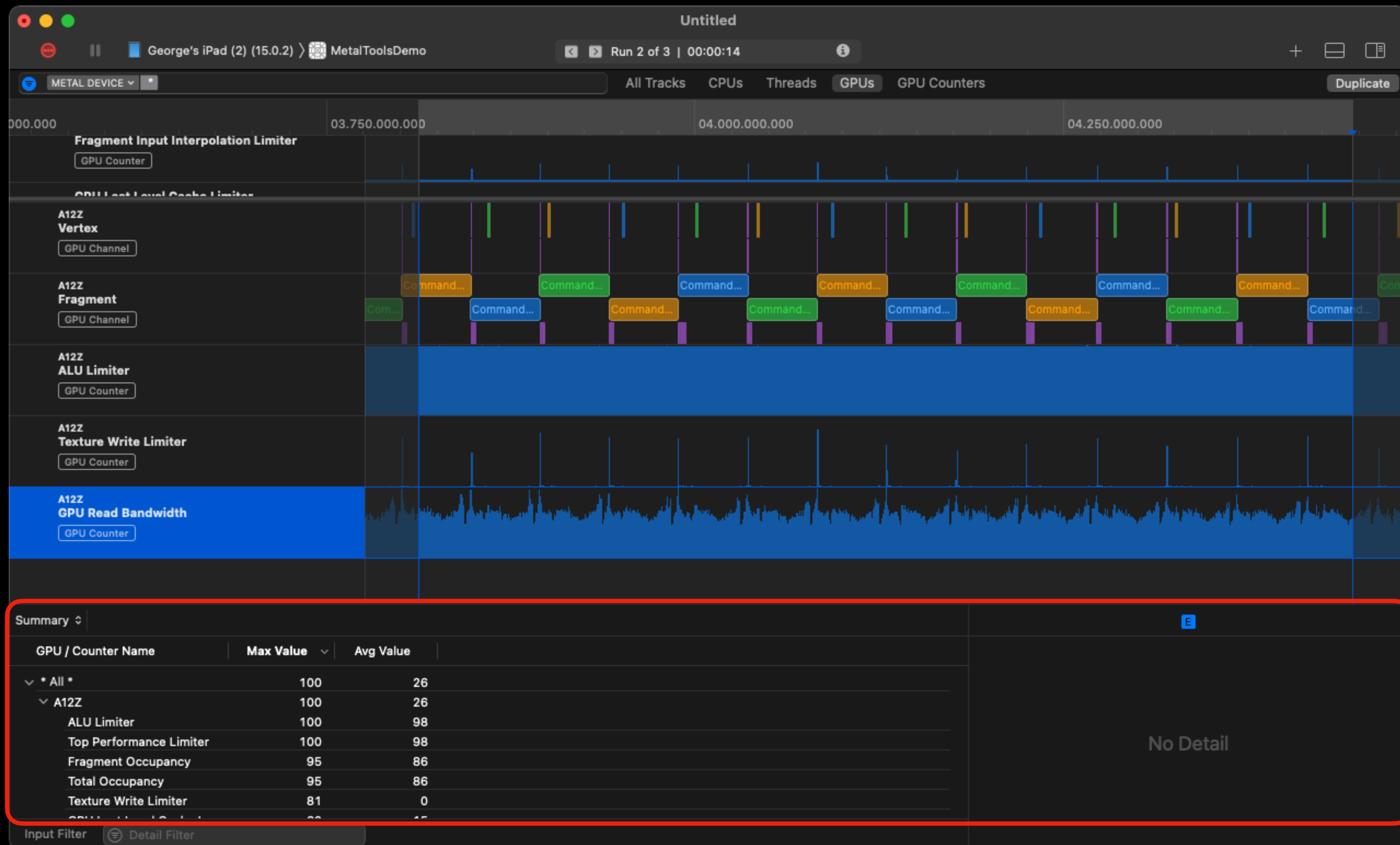
Tracks: Selecting



Tracks: Event Preview



Tracks: Event Details



PREPARATION

Devices and Simulators

Device Setup

Window → Devices and Simulators

Device Setup

The screenshot shows the Xcode interface for setting up a virtual device. The left sidebar has 'Devices' selected, showing 'George's iPhone' under 'Connected'. The main panel displays the device's details: 'George's iPhone', iOS 15.0.2 (19A404), Model: iPhone XS, and Capacity: 51.3 GB (2.52 GB available). There are checkboxes for 'Show as run destination' (checked) and 'Connect via network' (unchecked). Buttons for 'Take Screenshot', 'View Device Logs', and 'Open Console' are visible. Below this are sections for 'PAIRED WATCHES' (empty table), 'INSTALLED APPS' (table with MetalToolsDemo), and 'DEVICE CONDITIONS' (GPU Performance State and Profile dropdowns). A 'Start' button is at the bottom.

George's iPhone

iOS 15.0.2 (19A404)
Model: iPhone XS
Capacity: 51.3 GB (2.52 GB available)

Show as run destination
 Connect via network

Take Screenshot
View Device Logs
Open Console

PAIRED WATCHES

Name	Model	watchOS	Identifier
------	-------	---------	------------

INSTALLED APPS

Name	Version	Identifier
MetalToolsDemo	1	wdf.sandbox.MetalToolsDemo

DEVICE CONDITIONS

Condition: GPU Performance State
Profile: Minimum

Start

Device Setup

✓ None

GPU Performance State

Network Link

Thermal State

Device Setup

✓ None

GPU Performance State

Network Link

Thermal State

Device Setup



08:27



Device Setup

GPU Performance State Condition Active

"Minimum" is active on this iPhone.
Stop running this condition?

Cancel

Stop

Simulators

- Nothing interesting in Instruments
- No conditions
- No record options
- Nothing interesting in GPU Capture

Simulators

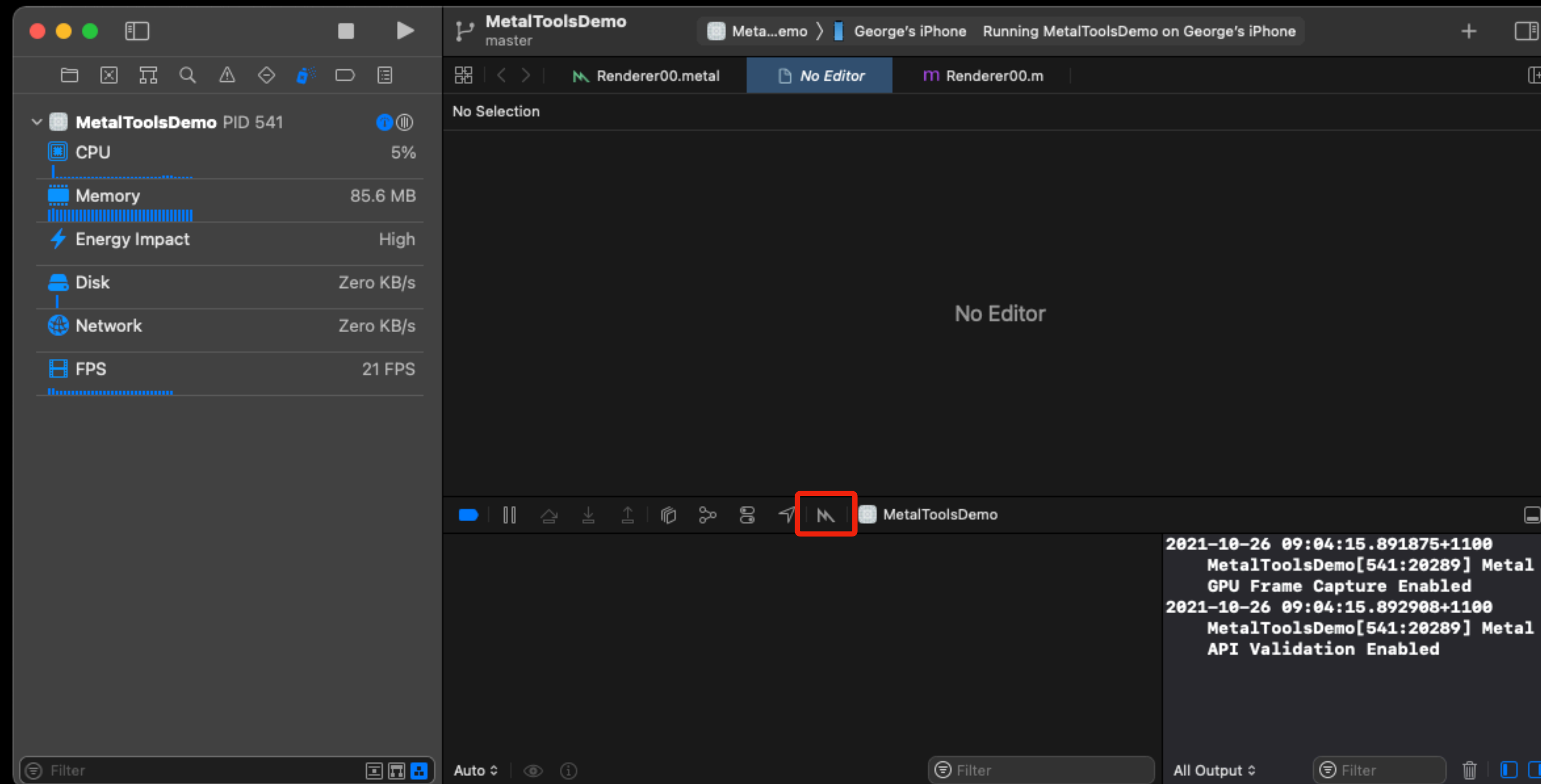
- Nothing interesting in Instruments
- No conditions
- No record options
- Nothing interesting in GPU Capture



XCODE

GPU Frame Capture

GPU Frame Capture



Debug → Capture GPU Workload

GPU Frame Capture

- ✓ Frame
- Metal Layers
 - 0x28125f750 [0,0 0x0]
- Command Queues
 - Command Queue <0x280969980>
- Devices
 - Apple A12 GPU

The screenshot shows the Xcode interface for MetalToolsDemo. On the left, the Activity Monitor panel displays system metrics for the process: CPU (4%), Memory (85.8 MB), Energy Impact (High), Disk (Zero KB/s), Network (Zero KB/s), and FPS (20 FPS). A modal dialog is open in the center, allowing configuration of the GPU frame capture. The 'Scope' is set to 'Frame' and the 'Count' is set to 5. A 'Capture' button is visible at the bottom of the dialog. The bottom right corner of the interface shows a console window with the following log output:

```
2021-10-26 09:04:15.891875+1100
MetalToolsDemo[541:20289] Metal
GPU Frame Capture Enabled
2021-10-26 09:04:15.892908+1100
MetalToolsDemo[541:20289] Metal
API Validation Enabled
```


GPU Frame Capture

- ✓ Frame
- Metal Layers
 - 0x28125f750 [0,0 0x0]
- Command Queues
 - Command Queue <0x280969980>
- Devices
 - Apple A12 GPU

The screenshot shows the Xcode MetalToolsDemo interface. On the left, the Activity Monitor sidebar displays system metrics for the MetalToolsDemo process (PID 541): CPU (4%), Memory (85.8 MB), Energy Impact (High), Disk (Zero KB/s), Network (Zero KB/s), and FPS (20 FPS). The main editor shows the MetalToolsDemo master project with a 'Renderer00.metal' file open. A 'Capture' dialog box is displayed, allowing configuration of the GPU frame capture. The 'Scope' is set to '0x28125f750 [0,0 0x0]' and 'Frames' is set to 5. A 'Capture' button is visible. The bottom right corner shows a log window with the following output:

```
2021-10-26 09:04:15.891875+1100
MetalToolsDemo[541:20289] Metal
GPU Frame Capture Enabled
2021-10-26 09:04:15.892908+1100
MetalToolsDemo[541:20289] Metal
API Validation Enabled
```

GPU Frame Capture

✓ Frame

Metal Layers

- 0x28125f750 [0,0 0x0]

Command Queues

- Command Queue <0x280969980>

Devices

- Apple A12 GPU

The screenshot shows the Xcode MetalToolsDemo interface. On the left, the Activity Monitor shows the process 'MetalToolsDemo' with a CPU usage of 4%, Memory of 85.8 MB, and Energy Impact of High. The FPS is 19. In the center, a dialog box is open for configuring the capture scope. The 'Scope' is set to 'Command Queue <0x280969980>' and 'Command Buffers' is set to 1. The dialog text states: 'Capture will start after command buffer creation on "Command Queue <0x280969980>" and will finish after a command buffer commit on "Command Queue <0x280969980>".' A 'Capture' button is visible. On the right, the console output shows the following logs:

```
2021-10-26 09:04:15.891875+1100
MetalToolsDemo[541:20289] Metal
GPU Frame Capture Enabled
2021-10-26 09:04:15.892908+1100
MetalToolsDemo[541:20289] Metal
API Validation Enabled
```

GPU Frame Capture

✓ Frame

Metal Layers
0x28125f750 [0,0 0x0]

Command Queues
Command Queue <0x280969980>

Devices
Apple A12 GPU

The screenshot shows the Xcode MetalToolsDemo interface. On the left, a sidebar displays system metrics for the MetalToolsDemo process (PID 541): CPU (4%), Memory (85.8 MB), Energy Impact (High), Disk (Zero KB/s), Network (Zero KB/s), and FPS (19 FPS). The main editor area shows a code editor with a 'No Selection' message and a 'Capture' dialog box. The dialog box is titled 'Scope' and has a dropdown menu set to 'Apple A12 GPU'. Below the dropdown, there is a 'Command Buffers' field with a value of '1'. A text box explains: 'Capture will start after command buffer creation on "Apple A12 GPU" and will finish after a command buffer commit.' A blue 'Capture' button is at the bottom of the dialog. At the bottom right, a console window displays the following log output:

```
2021-10-26 09:04:15.891875+1100  
MetalToolsDemo[541:20289] Metal  
GPU Frame Capture Enabled  
2021-10-26 09:04:15.892908+1100  
MetalToolsDemo[541:20289] Metal  
API Validation Enabled
```

Debug Navigator

The screenshot displays the Debug Navigator interface for a MetalToolsDemo application. The interface is divided into several sections:

- Left Panel (Summary):** A list of captured GPU workload items, grouped by API call. The items include:
 - 0 0x283234700 = [MTLLayer nextDra...
 - 1 CAMetalLayer Display Drawable = [M...
 - Command Buffer 0 0x148e055e0
 - 20 0x28321c780 = [MTLLayer nextDra...
 - 21 CAMetalLayer Display Drawable = [...]
 - Command Buffer 1 0x148e06220
 - 40 0x283228180 = [MTLLayer nextDra...
 - 41 CAMetalLayer Display Drawable = [...]
 - Command Buffer 2 0x148c04530
 - 60 0x283228b00 = [MTLLayer nextDr...
 - 61 CAMetalLayer Display Drawable = [...]
 - Command Buffer 3 0x148d063b0
 - 80 0x28322cb80 = [MTLLayer nextDra...
 - 81 CAMetalLayer Display Drawable = [...]
 - Command Buffer 4 0x148c04280
- Summary View:** A central view showing a rendered image of a cat's face with colorful, overlapping circles. Below the image, the resolution is 1125x2436 and the pixel format is BGRA8Unorm.
- Overview:** A table showing the number of various GPU calls:

Category	Count
Command Buffers	5
Render Encoders	5
Blit Encoders	0
Compute Encoders	5
Draw Calls	5
Dispatch Calls	5
- Performance:** A table showing GPU time and vertices:

Metric	Value
GPU Time	215.33 ms
Vertices	2,376,960
- Memory:** A table showing memory usage:

Category	Value
Textures	48.1 MB
Buffers	58 KB
Other	Zero KB
- Related Articles:** A list of links to related articles:
 - Monitoring Basic Memory Statistics
 - Exporting Memory Viewer Information
 - Analyzing Resources
 - Investigating Resource Issues
 - Reducing Your Memory Footprint

Summary: Export

The screenshot shows the MetalToolsDemo application interface. The left sidebar contains a tree view with 'Summary' selected. The main content area displays the 'Summary' page, which includes a central image of a cat, an 'Overview' table, a 'Performance' table, a 'Memory' table, and an 'Insights' section. An orange box highlights the 'Summary' header and the 'Export' button in the top right corner of the main content area.

Summary Export

Overview Show Dependencies

Command Buffers	5
Render Encoders	5
Blit Encoders	0
Compute Encoders	5
Draw Calls	5
Dispatch Calls	5

Performance Show Performance

GPU Time	215.33 ms
Vertices	2,376,960

Memory Show Memory

Textures	48.1 MB
Buffers	58 KB
Other	Zero KB

Insights

Memory No Insights | Bandwidth No Insights | Performance No Insights | API No Insights

Savings Insight

No Memory Insights

Related Articles

- Monitoring Basic Memory Statistics
- Exporting Memory Viewer Information
- Analyzing Resources
- Investigating Resource Issues
- Reducing Your Memory Footprint

Summary: Insights

The screenshot displays the MetalToolsDemo application interface. The main window shows a 'Summary' page for a captured GPU workload. The left sidebar contains a tree view of the workload, grouped by API call, listing five command buffers. The main content area features a central image of a cat's face rendered with colorful, overlapping circles. Below the image, the resolution is 1125x2436 and the pixel format is BGRA8Unorm. To the right, there are three summary panels: Overview, Performance, and Memory. The Overview panel shows counts for Command Buffers (5), Render Encoders (5), Blit Encoders (0), Compute Encoders (5), Draw Calls (5), and Dispatch Calls (5). The Performance panel shows GPU Time (215.33 ms) and Vertices (2,376,960). The Memory panel shows Textures (48.1 MB), Buffers (58 KB), and Other (Zero KB). At the bottom, an 'Insights' section is highlighted with a red border, showing tabs for Memory, Bandwidth, Performance, and API, all with 'No Insights'. A 'Savings' dropdown is set to 'Insight', and the main area displays 'No Memory Insights'. A 'Related Articles' section on the right lists four links: 'Monitoring Basic Memory Statistics', 'Exporting Memory Viewer Information', 'Analyzing Resources', 'Investigating Resource Issues', and 'Reducing Your Memory Footprint'.

MetalToolsDemo master

MetalToolsDemo > George's iPhone MetalToolsDemo - Debugging GPU Workload 2

Renderer00.metal Summary Renderer00.m

Summary

Summary

Export

Overview [Show Dependencies](#)

Command Buffers	5
Render Encoders	5
Blit Encoders	0
Compute Encoders	5
Draw Calls	5
Dispatch Calls	5

Performance [Show Performance](#)

GPU Time	215.33 ms
Vertices	2,376,960

Memory [Show Memory](#)

Textures	48.1 MB
Buffers	58 KB
Other	Zero KB

Resolution 1125x2436
Pixel Format BGRA8Unorm

Insights

[Memory](#) No Insights [Bandwidth](#) No Insights [Performance](#) No Insights [API](#) No Insights

Savings Insight

No Memory Insights

Related Articles

- [Monitoring Basic Memory Statistics](#)
- [Exporting Memory Viewer Information](#)
- [Analyzing Resources](#)
- [Investigating Resource Issues](#)
- [Reducing Your Memory Footprint](#)

Insights

Memory	Bandwidth	Performance	API
No Insights	12.7 GB	No Insights	No Insights

Savings	Insight
<ul style="list-style-type: none"> Render cats (12.7 GB) Coalesce Encoders x 1256 <ul style="list-style-type: none"> Render cats (5.1 MB) Coalesce Encoders Render cats (10.1 MB) Coalesce Encoders Render cats (10.1 MB) Coalesce Encoders Render cats (10.1 MB) Coalesce Encoders Render cats (10.1 MB) Coalesce Encoders Render cats (10.1 MB) Coalesce Encoders Render cats (10.1 MB) Coalesce Encoders Render cats (10.1 MB) Coalesce Encoders Render cats (10.1 MB) Coalesce Encoders 	

R Render cats

Draw Calls 1


Encoder 'Render cats' is the root of 1257 encoders that can be coalesced

Your application created separate command encoders which can be combined into a single encoder. By combining these encoders you may reduce your application's load/store bandwidth usage.

Insights

Memory	Bandwidth	Performance	API
No Insights	268.4 MB	1 Insight	No Insights

Savings	Insight
Texture 0x100e34fc0 (268.4 MB) Lossless Compression	



Texture 0x100e34fc0

Type	2D
Pixel Format	BGRA8Unorm
Mipmap Level	0
Width	8192
Height	8192

Resource "Texture 0x100e34fc0" can't be lossless compressed.

'Texture:0x100e34fc0' opted out of lossless compression because its usage flags include 'ShaderWrite' which prevents lossless compression

Insights

Memory	Bandwidth	Performance	API
No Insights	No Insights	1 Insight	No Insights

Count	Insight
fshShader00 (vshShader -> fshShader00) (1 Insight)	High float to half ratio

S fshShader00

Shader "vshShader" in pipeline "vshShader -> fshShader00" uses a relatively high number of 32-bit floating point operations. (69.78 of Instructions, 68.96 of Shader Cost, 246.67 ms Shader Time)

• For optimal performance, use 16-bit half data types when possible. Avoid 32-bit floating point inputs (textures / buffers). Avoid implicit conversions.

[Show Resource](#)

Related Articles

- [Optimizing Performance with the Shader Profiler](#)
- [Optimizing Performance with Pipeline Statistics](#)

Summary: Overview

The screenshot shows the MetalToolsDemo Summary page. The left sidebar lists 'MetalToolsDemo Captured GPU Worklo...' with a 'Summary' tab selected. The main content area displays a 'Summary' view for 'Renderer00.metal'. A central image shows a 3D scene with a cat and foliage, overlaid with a complex network of colored lines representing GPU resources. Below the image, the resolution is 1125x2436 and the pixel format is BGRA8Unorm. On the right, a red-bordered box highlights the 'Overview', 'Performance', and 'Memory' sections.

Overview [Show Dependencies](#)

Command Buffers	5
Render Encoders	5
Blit Encoders	0
Compute Encoders	5
Draw Calls	5
Dispatch Calls	5

Performance [Show Performance](#)

GPU Time	215.33 ms
Vertices	2,376,960

Memory [Show Memory](#)

Textures	48.1 MB
Buffers	58 KB
Other	Zero KB

Insights

[Memory](#) No Insights | [Bandwidth](#) No Insights | [Performance](#) No Insights | [API](#) No Insights

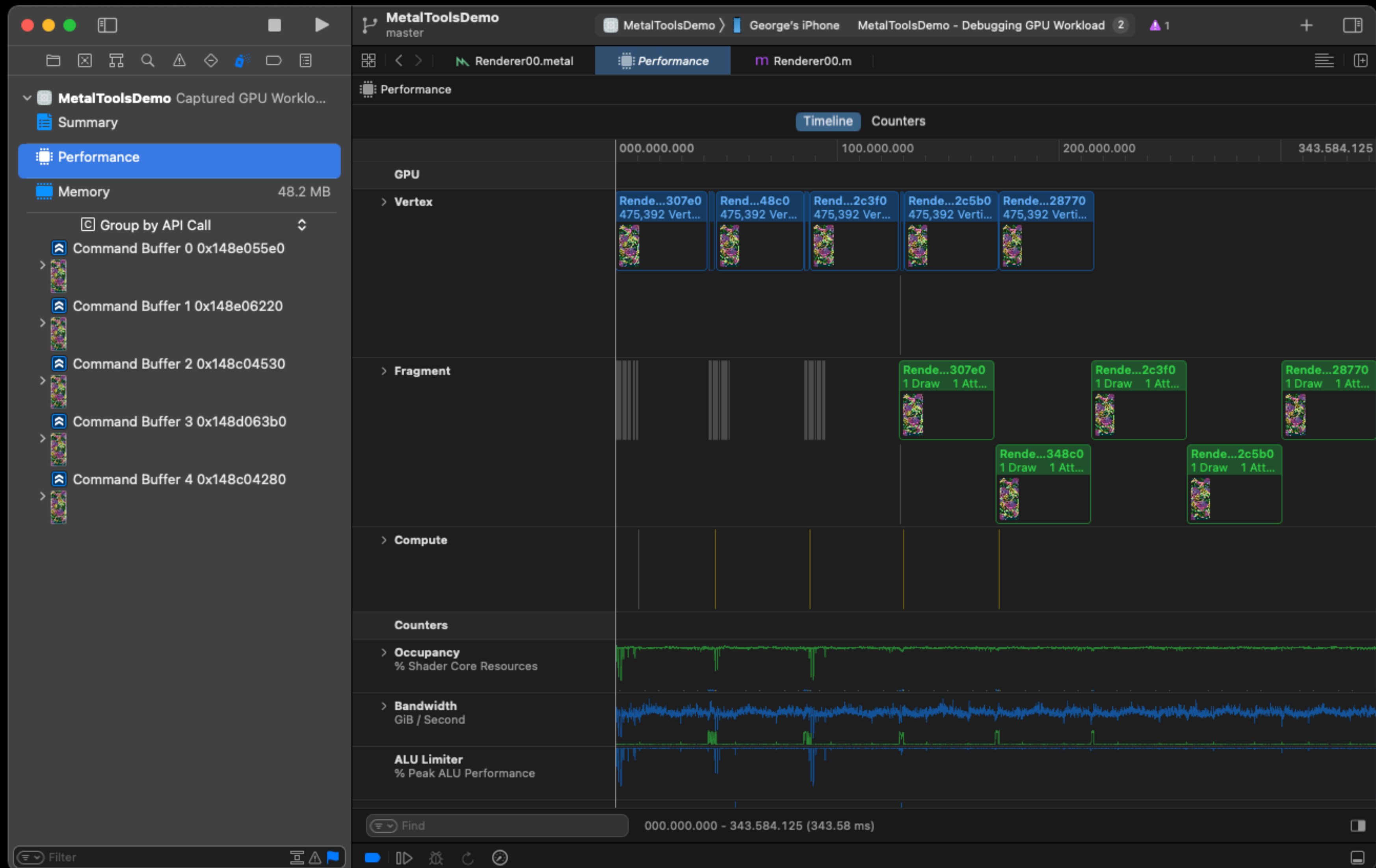
[Savings](#) | [Insight](#)

No Memory Insights

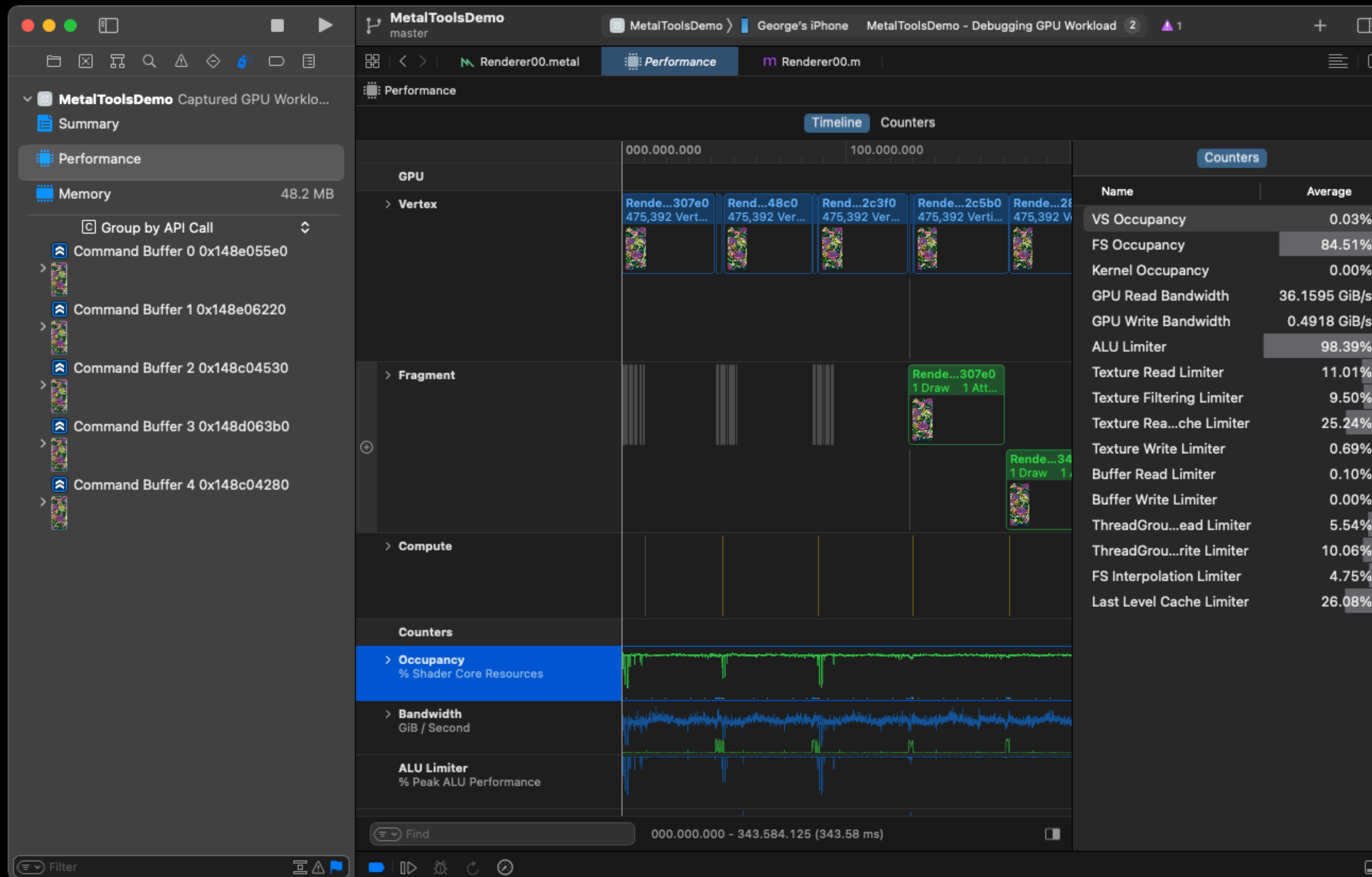
Related Articles

- [Monitoring Basic Memory Statistics](#)
- [Exporting Memory Viewer Information](#)
- [Analyzing Resources](#)
- [Investigating Resource Issues](#)
- [Reducing Your Memory Footprint](#)

Performance: Timeline



Performance: Details



Performance: Counters

The screenshot displays the MetalToolsDemo application interface, specifically the Performance Counters window. The window is titled "MetalToolsDemo" and shows the "Performance" tab selected. The "Counters" sub-tab is active, displaying a table of performance metrics for various GPU commands. The table has columns for "Thumbnails", "Name", "Shader Profiler Time", "ALU Limiter", and "ALU Utilization". The data is grouped by API call, showing multiple instances of RenderEncoder and ComputeEncoder commands. The RenderEncoder commands show high ALU utilization (around 75-76%) and significant shader profiler times (around 43 ms). The ComputeEncoder commands show much lower ALU utilization (around 1-3%) and shorter shader profiler times (around 3-7 microseconds).

Thumbnails	Name	Shader Profiler Time	ALU Limiter	ALU Utilization
	RenderEncoder 0x2837348c0	43.11 ms	95.97%	75.65%
	ComputeEncoder 0x28372c4d0	7.17 μs	3.06%	2.61%
	ComputeEncoder 0x28372c620	6.10 μs	1.49%	0.82%
	ComputeEncoder 0x283730930	3.04 μs	3.10%	2.58%
	ComputeEncoder 0x2837308c0	5.82 μs	2.90%	2.53%
	RenderEncoder 0x28372c5b0	43.00 ms	95.97%	75.65%
	RenderEncoder 0x283728770	42.95 ms	95.95%	75.63%
	ComputeEncoder 0x283728540	7.14 μs	1.90%	1.43%
	RenderEncoder 0x28372c3f0	43.05 ms	95.96%	75.63%
	RenderEncoder 0x2837307e0	43.20 ms	95.91%	75.61%

Performance: Counters

The screenshot displays the MetalToolsDemo Performance Counters interface. The left sidebar shows the 'Performance' section with a 'Memory' usage of 48.2 MB and a 'Group by API Call' view listing five command buffers. The main panel shows the 'Counters' view with a table of GPU commands. The table columns are: Thumbnails, Name, Encoder, Pipeline, Shader Profiler Time, and Primitives. The data rows show various dispatch and drawIndexedPrimitive commands with their respective execution times and primitive counts.

Thumbnails	Name	Encoder	Pipeline	Shader Profiler Time	Primitives
	7 [dispatchThreadg..	ComputeEncoder 0x28...	MTLComputePipelineSt..	3.04 μ s	
	16 [drawIndexedPri..	RenderEncoder 0x283...	vshRenderer00 -> fshR..	43.19 ms	1
	27 [dispatchThread..	ComputeEncoder 0x28...	MTLComputePipelineSt..	0 ns	
	36 [drawIndexedPri..	RenderEncoder 0x283...	vshRenderer00 -> fshR..	43.11 ms	1
	47 [dispatchThread..	ComputeEncoder 0x28...	MTLComputePipelineSt..	0 ns	
	56 [drawIndexedPri..	RenderEncoder 0x283...	vshRenderer00 -> fshR..	43.05 ms	1
	67 [dispatchThread..	ComputeEncoder 0x28...	MTLComputePipelineSt..	0 ns	
	76 [drawIndexedPri..	RenderEncoder 0x283...	vshRenderer00 -> fshR..	43.00 ms	1
	87 [dispatchThread..	ComputeEncoder 0x28...	MTLComputePipelineSt..	0 ns	
	96 [drawIndexedPri..	RenderEncoder 0x283...	vshRenderer00 -> fshR..	42.94 ms	1

Performance: Edit Counters

The screenshot shows the Xcode Performance tool interface. The 'Performance' tab is active, displaying a list of performance counters. A red box highlights the 'Edit Counters...' button in the top right corner. The dialog box is open, showing a list of performance counters with checkboxes for each. The 'Primitives' counter is selected. The dialog also includes a 'Reset' button, a 'Find' search bar, and a 'Close' button.

Show	Item	Time
<input checked="" type="checkbox"/>	GPU Time GPU time in nanoseconds	3.04 μ s
<input checked="" type="checkbox"/>	Primitives Number of primitives submitted to the render pipeline	43.19 ms
<input checked="" type="checkbox"/>	Primitives Culled (Zero-Area) Percentage of primitives culled due to having zero area coverage, relative to the total number of primitives s...	0 ns
<input checked="" type="checkbox"/>	Primitives Culled (Back-Face) Percentage of primitives culled due to backface relative to the total number of primitives submitted	43.11 ms
<input checked="" type="checkbox"/>	Primitives Culled (Guard-Band) Percentage of primitives culled due to being outside the guard band, relative to the total number of primitiv...	0 ns
<input checked="" type="checkbox"/>	Primitives Culled (Off-Screen) Percentage of primitives culled due to being off-screen, relative to the total number of primitives submitted	43.05 ms
<input checked="" type="checkbox"/>	Primitives Clipped Percentage of primitives discarded by the clipper, relative to the total number of primitives submitted	0 ns
<input checked="" type="checkbox"/>	Primitives Rendered Percentage of primitives rendered, relative to the total number of primitives submitted	43.00 ms
<input checked="" type="checkbox"/>	Pre Cull Primitive Processing Measures the time during which Primitive processing block processes primitives prior to culling as a percen...	0 ns
<input checked="" type="checkbox"/>	Post Clip Cull Primitive Processing Measures the time during which Primitive processing block processes primitives post clipping and culling as...	42.94 ms

Memory

MetalToolsDemo master
MetalToolsDemo > George's iPhone MetalToolsDemo - Debugging GPU Workload 2

Render00.metal | **Memory** | Render00.m

Memory 48.2 MB

Group by API Call

- Command Buffer 0 0x148e055e0
- Command Buffer 1 0x148e06220
- Compute 0 0x2837308c0 5.82 μs
 - No Previews
 - 27 [dispatchThreadgroups:{1... < 1μs
 - Render 1 0x2837348c0 43.11 ms
 - 29 0x2837348c0 = [renderComm...
 - 36 [drawIndexedPrimitive... 43.11 ms
 - 38 [presentDrawable:0x28321c780]
 - Command Buffer 2 0x148c04530
 - Command Buffer 3 0x148d063b0
 - Command Buffer 4 0x148c04280

Memory

- Non-Volatile 48.2 MB
- Volatile / Empty Zero KB
- Textures 48.1 MB
- Buffers 58 KB
- Pipeline States 2 KB
- Private 16 KB
- Shared 48.2 MB
- Used 48.2 MB
- Unused Zero KB

Label	Type	Allocated Size	Storage Mo...	Purgeable Sta...	CPU Access	Time Since Last Bound
Buffer:0x148807080	Buffer	32 KB	Shared	NonVolatile	None	Now
Buffer:0x148809e00	Buffer	4 KB	Shared	NonVolatile	None	Now
Buffer:0x14880c780	Buffer	16 KB	Private	NonVolatile	None	Now
Buffer:0x14880c8d0	Buffer	6 KB	Shared	NonVolatile	None	Now
CAMetalLayer Display Drawa...	Texture 2D	11.0 MB	Shared	NonVolatile	None	Now
CAMetalLayer Display Drawa...	Texture 2D	11.0 MB	Shared	NonVolatile	None	Now
CAMetalLayer Display Drawa...	Texture 2D	11.0 MB	Shared	NonVolatile	None	Now
Compute Pipeline State:0x1...	Compute Pip...	558 bytes	--	--	--	Now
Texture:0x147d11020	Texture 2D	15.0 MB	Shared	NonVolatile	Write	Now
vshRender00 -> fshRende...	Render Pipel...	1 KB	--	--	--	Now

Memory: Preview

The screenshot displays the MetalToolsDemo application's memory management interface. The left sidebar shows a tree view of GPU workload components, with 'Memory' selected, showing a total usage of 48.2 MB. The main panel shows a breakdown of memory usage:

- Non-Volatile: 48.2 MB
- Volatile / Empty: Zero KB
- Textures: 48.1 MB
- Buffers: 58 KB
- Pipeline States: 2 KB
- Private: 16 KB
- Shared: 48.2 MB
- Used: 48.2 MB
- Unused: Zero KB

A tooltip for a texture (0x147d11020) is shown, displaying a preview of a cat's face and indicating a size of 15.0 MB. Below the memory summary is a table of allocated memory resources:

Label	Type	Allocated Size	Storage Mo...	Purgeable Sta...	CPU Access	Time Since Last Bound
Buffer:0x148807080	Buffer	32 KB	Shared	NonVolatile	None	Now
Buffer:0x148809e00	Buffer	4 KB	Shared	NonVolatile	None	Now
Buffer:0x14880c780	Buffer	16 KB	Private	NonVolatile	None	Now
Buffer:0x14880c8d0	Buffer	6 KB	Shared	NonVolatile	None	Now
CAMetalLayer Display Drawa...	Texture 2D	11.0 MB	Shared	NonVolatile	None	Now
CAMetalLayer Display Drawa...	Texture 2D	11.0 MB	Shared	NonVolatile	None	Now
CAMetalLayer Display Drawa...	Texture 2D	11.0 MB	Shared	NonVolatile	None	Now
Compute Pipeline State:0x1...	Compute Pip...	558 bytes	--	--	--	Now
Texture:0x147d11020	Texture 2D	15.0 MB	Shared	NonVolatile	Write	Now
vshRenderer00 -> fshRende...	Render Pipel...	1 KB	--	--	--	Now

Memory: Filter

The screenshot displays the MetalToolsDemo application interface, specifically the Memory filter view. The left sidebar shows a tree view of the GPU workload, with the 'Memory' filter selected, showing a total of 48.2 MB. The main area displays a summary of memory usage and a table of allocated buffers.

Memory Summary:

- Non-Volatile: 42 KB
- Volatile / Empty: Zero KB
- Textures: Zero KB
- Buffers: 42 KB
- Pipeline States: Zero KB
- Private: Zero KB
- Shared: 42 KB
- Used: 42 KB
- Unused: Zero KB

Buffer Table:

Label	Type	Allocated Size	Storage Mo...	Purgeable Sta...	CPU Access	Time Since Last Bound
Buffer:0x148807080	Buffer	32 KB	Shared	NonVolatile	None	Now
Buffer:0x148809e00	Buffer	4 KB	Shared	NonVolatile	None	Now
Buffer:0x14880c8d0	Buffer	6 KB	Shared	NonVolatile	None	Now

The interface also includes a filter bar at the bottom with the following settings: Filter, ALL, BUFFERS, STORAGE MODE, Shared.

Calls: Show Only Markers And Commands

The screenshot displays the MetalToolsDemo application interface. The main window shows a 'Summary' view for a captured GPU workload. The left sidebar contains a tree view with 'Summary' selected, and a 'Group by API Call' section listing five command buffers. The main content area features a central image of a cat with overlaid colored circles, representing the rendered scene. To the right of the image is an 'Overview' table with the following data:

Overview	
Command Buffers	5
Render Encoders	5
Blit Encoders	0
Compute Encoders	5
Draw Calls	5
Dispatch Calls	5

Below the overview is a 'Performance' section with the following data:

Performance	
GPU Time	215.33 ms
Vertices	2,376,960

At the bottom of the main content area is a 'Memory' section with the following data:

Memory	
Textures	48.1 MB
Buffers	58 KB
Other	Zero KB

The bottom of the interface shows an 'Insights' section with tabs for 'Memory', 'Bandwidth', 'Performance', and 'API'. The 'Memory' tab is selected, and it displays 'No Memory Insights'. A 'Related Articles' section is visible on the right side of the bottom panel, listing several articles related to memory monitoring and optimization.

Calls: Group By API Call

The screenshot displays the MetalToolsDemo interface for debugging GPU workloads. The left sidebar shows a tree view of captured GPU workloads, with the 'Group by API Call' option highlighted in red. The main area shows a summary of the workload, including a rendered image of a cat, performance metrics, and memory usage.

MetalToolsDemo master
MetalToolsDemo - Debugging GPU Workload 2

Summary

Dispatch Calls 5

Performance [Show Performance](#)

GPU Time	173.47 ms
Vertices	2,376,960

Memory [Show Memory](#)

Textures	48.1 MB
Buffers	58 KB
Other	Zero KB

Resolution 1125x2436
Pixel Format BGRA8Unorm

Insights

Memory No Insights	Bandwidth No Insights	Performance No Insights	API No Insights
-----------------------	--------------------------	----------------------------	--------------------

[Savings](#) [Insight](#)

No Memory Insights

Related Articles

- [Monitoring Basic Memory Statistics](#)
- [Exporting Memory Viewer Information](#)
- [Analyzing Resources](#)
- [Investigating Resource Issues](#)
- [Reducing Your Memory Footprint](#)

Command Buffer 0 0x1070054b0

- Compute 0 0x2820a4460 6.02 μs
- No Previews
- 7 [dispatchThreadgroups:{...} 6.02 μs
- Render 1 0x2820a0930 38.04 ms
- 9 0x2820a0930 = [renderComma...
- 16 [drawIndexedPrimitive... 38.04 ms
- 18 [presentDrawable:0x2825bd280]

Command Buffer 1 0x104f05350

- Compute 0 0x2820a0230 10.10 μs
- No Previews
- 27 [dispatchThreadgroups:{1... < 1μs
- Render 1 0x2820bc540 33.81 ms
- 29 0x2820bc540 = [renderComm...
- 36 [drawIndexedPrimitive... 33.80 ms
- 38 [presentDrawable:0x2825b8300]

Command Buffer 2 0x1070065c0

Calls: Group By Pipeline State

The screenshot displays the MetalToolsDemo application interface. The left sidebar shows a tree view of the workload hierarchy. The 'Group by Pipeline State' option is highlighted with a red box. The main view shows a summary of the workload, including a render pipeline and a compute pipeline. The render pipeline is expanded to show its components and draws. The compute pipeline is also expanded to show its components. The right sidebar shows performance and memory statistics.

MetalToolsDemo

Summary

Performance

Memory 48.2 MB

Group by Pipeline State

- Render Pipeline 0x10301e800 147.35 ms
 - vshRenderer00 2.24 ms
 - fshRenderer00 145.11 ms
 - Draws
 - 16 [drawIndexedPrimitive... 33.52 ms
 - 36 [drawIndexedPrimitive... 28.55 ms
 - 56 [drawIndexedPrimitive... 28.46 ms
 - 76 [drawIndexedPrimitive... 28.40 ms
 - 96 [drawIndexedPrimitive... 28.43 ms
- Compute Pipeline 0x102e11540 6.84 μs
 - krnRenderer00 6.84 μs
 - DispatchThreadgroups
 - 7 [dispatchThreadgroups:{... 6.84 μs
 - 27 [dispatchThreadgroups:{1... < 1μs
 - 47 [dispatchThreadgroups:{1... < 1μs
 - 67 [dispatchThreadgroups:{1... < 1μs
 - 87 [dispatchThreadgroups:{1... < 1μs

Dispatch Calls 5

Performance [Show Performance](#)

GPU Time 147.39 ms

Vertices 2,376,960

Memory [Show Memory](#)

Textures 48.1 MB

Buffers 58 KB

Other Zero KB

Resolution 1125x2436

Pixel Format BGRA8Unorm

Insights

Memory No Insights

Bandwidth No Insights

Performance No Insights

API No Insights

Savings Insight

No Memory Insights

Related Articles

- [Monitoring Basic Memory Statistics](#)
- [Exporting Memory Viewer Information](#)
- [Analyzing Resources](#)
- [Investigating Resource Issues](#)
- [Reducing Your Memory Footprint](#)

Labels: Pipelines

```
MTLComputePipelineDescriptor *descriptor = [MTLComputePipelineDescriptor new];  
descriptor.label = @"label";
```

```
MTLRenderPipelineDescriptor *descriptor = [MTLComputePipelineDescriptor new];  
descriptor.label = @"label";
```

Calls: Labels

The screenshot displays the MetalToolsDemo Profiling GPU trace interface. The window title is "MetalToolsDemo - Profiling GPU trace" with a notification badge for 2 items. The interface is divided into several sections:

- Summary:** Shows a central image of a cat with overlaid colorful lines representing GPU calls. Below the image, it displays "Resolution 1125x2436" and "Pixel Format BGRA8Unorm".
- Dispatch Calls:** Shows 5 dispatch calls.
- Performance:** Displays "Profiling GPU Trace..." with a loading spinner.
- Memory:** Shows a total of 48.2 MB. A "Show Memory" button is present. The breakdown is:

Textures	48.1 MB
Buffers	58 KB
Other	Zero KB
- Insights:** Shows "No Memory Insights" under the "Savings" tab. Other tabs include "Memory", "Bandwidth", "Performance", and "API", all showing "No Insights".
- Related Articles:** Lists several links:
 - Monitoring Basic Memory Statistics
 - Exporting Memory Viewer Information
 - Analyzing Resources
 - Investigating Resource Issues
 - Reducing Your Memory Footprint
- Left Sidebar:** Shows a tree view of the captured GPU workload:
 - MetalToolsDemo Captured GPU Worklo...
 - Summary
 - Performance
 - Memory 48.2 MB
 - Group by Pipeline State
 - Pipeline: Transform cats
 - krnRenderer00
 - DispatchThreadgroups
 - 9 [dispatchThreadgroups:{16, 1, 1}...
 - 32 [dispatchThreadgroups:{16, 1, ...
 - 55 [dispatchThreadgroups:{16, 1, ...
 - 78 [dispatchThreadgroups:{16, 1, 1...]
 - 101 [dispatchThreadgroups:{16, 1, ...
 - Pipeline: Render cats
 - vshRenderer00
 - fshRenderer00
 - Draws
 - 19 [drawIndexedPrimitives:Triangl...
 - 42 [drawIndexedPrimitives:Triangl...
 - 65 [drawIndexedPrimitives:Triangl...
 - 88 [drawIndexedPrimitives:Triangl...
 - 111 [drawIndexedPrimitives:Triangl...

Labels: Command Buffer / Encoder / ...

```
// ...  
commandBuffer.label = @"label";
```

```
// ...  
encoder.label = @"label";
```

Calls: Labels

The screenshot displays the MetalToolsDemo interface for debugging GPU workloads. The main window shows a summary of a captured GPU workload, including performance and memory statistics. The interface is divided into several sections:

- Summary:** Displays a central image of a cat with overlaid colorful lines representing GPU calls. Below the image, it shows the resolution (1125x2436) and pixel format (BGRA8Unorm).
- Dispatch Calls:** Shows 5 dispatch calls.
- Performance:** Displays GPU Time (145.33 ms) and Vertices (2,376,960).
- Memory:** Displays Textures (48.1 MB), Buffers (58 KB), and Other (Zero KB).
- Insights:** Shows no insights for Memory, Bandwidth, Performance, or API. A dropdown menu is set to 'Savings' and 'Insight'.
- Related Articles:** Lists several articles related to memory statistics and resource issues.

The left sidebar shows a tree view of the workload, grouped by API call. The tree includes:

- Main Frame
- Transform (4.27 μs)
- Render (31.92 ms)
- 21 [presentDrawable:0x280a08f80]
- Main Frame
- Transform (4.30 μs)
- Render (28.41 ms)
- 44 [presentDrawable:0x280a09780]
- Main Frame
- Main Frame
- Main Frame

The bottom of the interface features a filter bar and a status bar with 'Auto' and 'Filter' options.

Debug Groups

```
[commandBuffer pushDebugGroup:@"Group #1"];  
  [commandBuffer pushDebugGroup:@"Group #2"];  
    [self updateTransformsIn:commandBuffer];  
  [commandBuffer popDebugGroup];  
  [commandBuffer pushDebugGroup:@"Group #3"];  
    [self renderIn:commandBuffer];  
  [commandBuffer popDebugGroup];  
[commandBuffer popDebugGroup];
```

Calls: Labels

The screenshot displays the MetalToolsDemo interface for debugging GPU workloads. The left sidebar shows a tree view of the captured GPU workload, with a red box highlighting a specific call path: Group #1 (32.76 ms) containing Group #2 (5.81 μs) and its sub-call Transform (5.81 μs). Below this, Group #3 (32.75 ms) contains a Render call (32.75 ms). The main window shows a summary of the workload, including a rendered image of a cat with overlaid colored lines representing GPU calls. The image has a resolution of 1125x2436 and a pixel format of BGRA8Unorm. The right sidebar provides performance and memory statistics:

Dispatch Calls	
Dispatch Calls	5

Performance	
GPU Time	146.35 ms
Vertices	2,376,960

Memory	
Textures	48.1 MB
Buffers	58 KB
Other	Zero KB

The Insights section at the bottom shows no insights for Memory, Bandwidth, Performance, or API. A 'No Memory Insights' message is displayed with a small thumbnail of the rendered image. The bottom right corner features a 'Related Articles' section with links to various debugging guides.

Command Buffer: Dependencies

The screenshot displays the MetalToolsDemo application interface. On the left, a sidebar shows a tree view of the captured GPU workload. The 'Main Frame' is expanded to show 'Group #2' selected, which contains a 'Transform' operation. The main area shows a dependency graph for the selected operation, with a red box highlighting the graph. The graph consists of several interconnected nodes and lines, representing the execution flow and dependencies of the GPU commands. The text 'No Assistant Results' is visible on the right side of the main area.

MetalToolsDemo Captured GPU Workload

- Summary
- Performance
- Memory: 48.2 MB
- Group by API Call
 - Main Frame
 - Group #1: 32.76 ms
 - Group #2: 5.81 μs**
 - Transform: 5.81 μs
 - No Previews
 - Group #3: 32.75 ms
 - Render: 32.75 ms
 - 27 [presentDrawable:0x282a29080]
 - Main Frame
 - Main Frame
 - Main Frame
 - Main Frame
 - Main Frame

MetalToolsDemo.gputrace

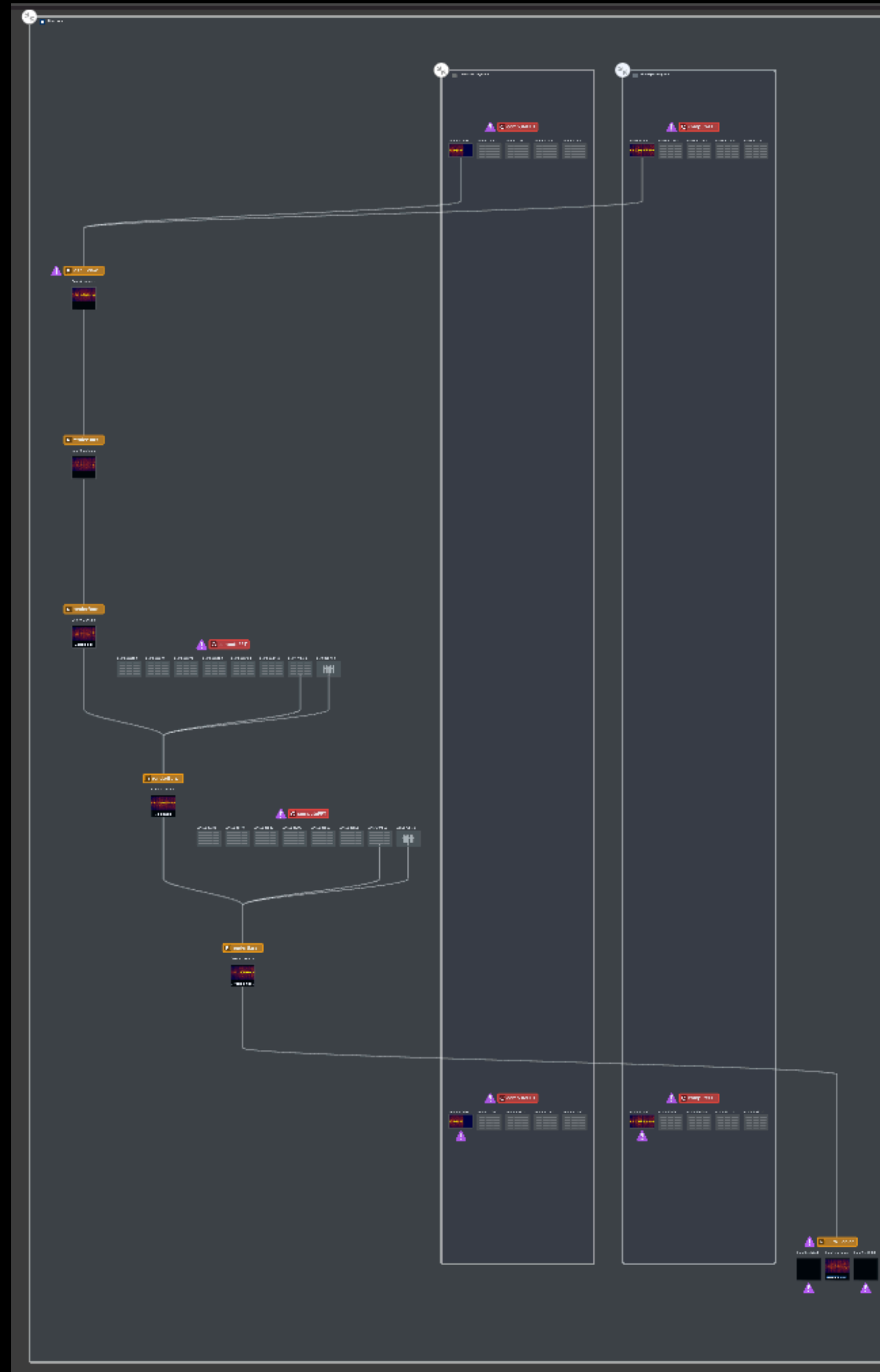
- Main Frame
 - Group #1
 - Group #2**

Find

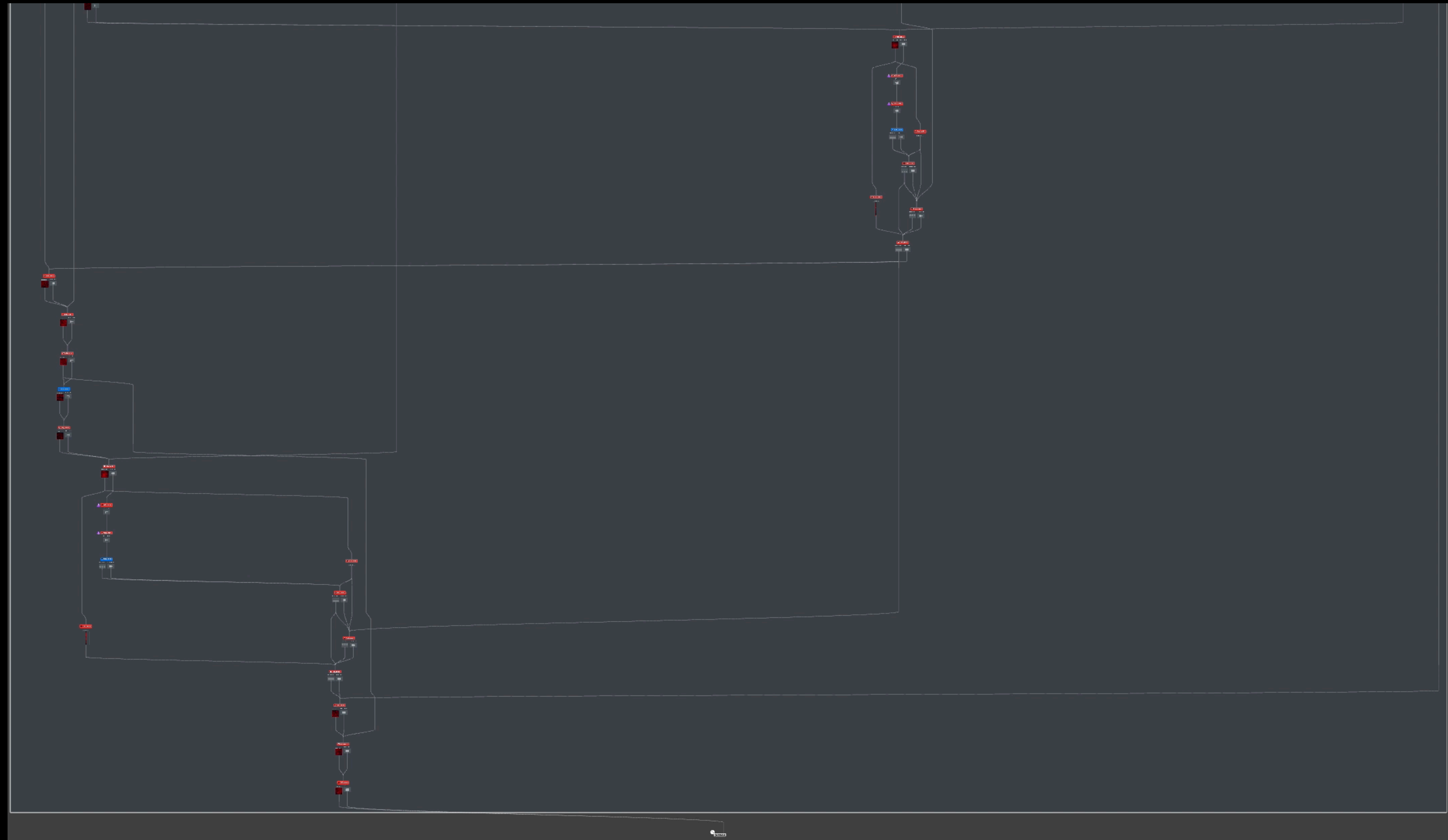
No Assistant Results

33%

Command Buffer: Dependencies



Command Buffer: Dependencies



Encoder → Call

The screenshot displays the Xcode GPU Debugger interface for a MetalToolsDemo application. The left sidebar shows a tree view of captured GPU workload, with call 52 [drawIndexedPrimitives:Triangle indexCount:1857...] highlighted in a red box. The main pane shows the details for this call, including Pipeline States (Vertex and Fragment), Attachments, and a detailed resource usage table.

Label	Type	Size	Details	Parameter Name	Resource Usage
Pipeline States					
Pipeline: Render cats					
Vertex					
Buffer 0x10310eb50	Index	4 KB	Offset: 0x0		
Buffer 0x10310e740	Buffer 0	18 KB	Offset: 0x0	vertexBuffer.0	Read
Buffer 0x10310eca0	Buffer 1	16 KB	Offset: 0x0	transforms	Read
Vertex Bytes	Buffer 2 (Bytes)	4 bytes		ratio	Read
Geometry	Post Vertex Tran...				
Vertex Attributes	Vertex Attributes				
vshRenderer00	Vertex Function		Library 0x282452e80 [Pr...		
Fragment					
Texture 0x10310e380	Texture 0	1936 x 1936	BGRA8Unorm	image	Read
Fragment Bytes	Buffer 0 (Bytes)	4 bytes		time_	Read
fshRenderer00	Fragment Function		Library 0x282452e80 [Pr...		
Attachments					
CAMetalLayer Display Dra...	Color 0	1125 x 2436	BGRA8Unorm		Write

Below the table, a detailed resource usage list is shown:

- "Pipeline: Render cats" (0x104035a00) vshRenderer00 - fshRenderer00
- RenderPipeline Performance 140.60 ms (100.0%)
- Vertex Buffer 0 (MTLBuffer) 0x10310e740
- Vertex Buffer 1 (MTLBuffer) 0x10310eca0
- Vertex Buffer 2 (MTLBuffer) Bytes
- Vertex Buffer 3 (MTLBuffer) Bytes
- Vertex Buffer 4 (MTLBuffer) Bytes
- Vertex Buffer 5 (MTLBuffer) Bytes
- Vertex Buffer 6 (MTLBuffer) Bytes

Encoder → Call

The screenshot displays the MetalToolsDemo debugger interface, showing a captured GPU workload. The left sidebar contains a tree view of the workload, with the 'Render' group selected. The main panel shows a table of resource usage for the selected pipeline state.

Label	Type	Size	Details	Parameter Name	Resource Usage
Pipeline States					
Pipeline: Render cats					
Vertex					
Buffer 0x10310eb50	Index	4 KB	Offset: 0x0		
Buffer 0x10310e740	Buffer 0	18 KB	Offset: 0x0	vertexBuffer.0	Read
Buffer 0x10310eca0	Buffer 1	16 KB	Offset: 0x0	transforms	Read
Vertex Bytes	Buffer 2 (Bytes)	4 bytes		ratio	Read
Geometry	Post Vertex Tran...				
Vertex Attributes	Vertex Attributes				
	Vertex Function		Library 0x282452e80 [Pr...		
Texture 0		1936 × 1936	BGRA8Unorm	image	Read
Buffer 0 (Bytes)		4 bytes		time_	Read
Fragment Function			Library 0x282452e80 [Pr...		
Attachments					
CAMetalLayer Display Dra...	Color 0	1125 × 2436	BGRA8Unorm		Write

The bottom panel shows the callstack for the selected pipeline state, listing various buffers and their sizes.

- "Pipeline: Render cats" (0x104035a00) vshRenderer00 - fshRenderer00
- RenderPipeline Performance 140.60 ms (100.0%)
- Vertex Buffer 0 (MTLBuffer) 0x10310e740
- Vertex Buffer 1 (MTLBuffer) 0x10310eca0
- Vertex Buffer 2 (MTLBuffer) Bytes
- Vertex Buffer 3 (MTLBuffer) Bytes
- Vertex Buffer 4 (MTLBuffer) Bytes
- Vertex Buffer 5 (MTLBuffer) Bytes
- Vertex Buffer 6 (MTLBuffer) Bytes

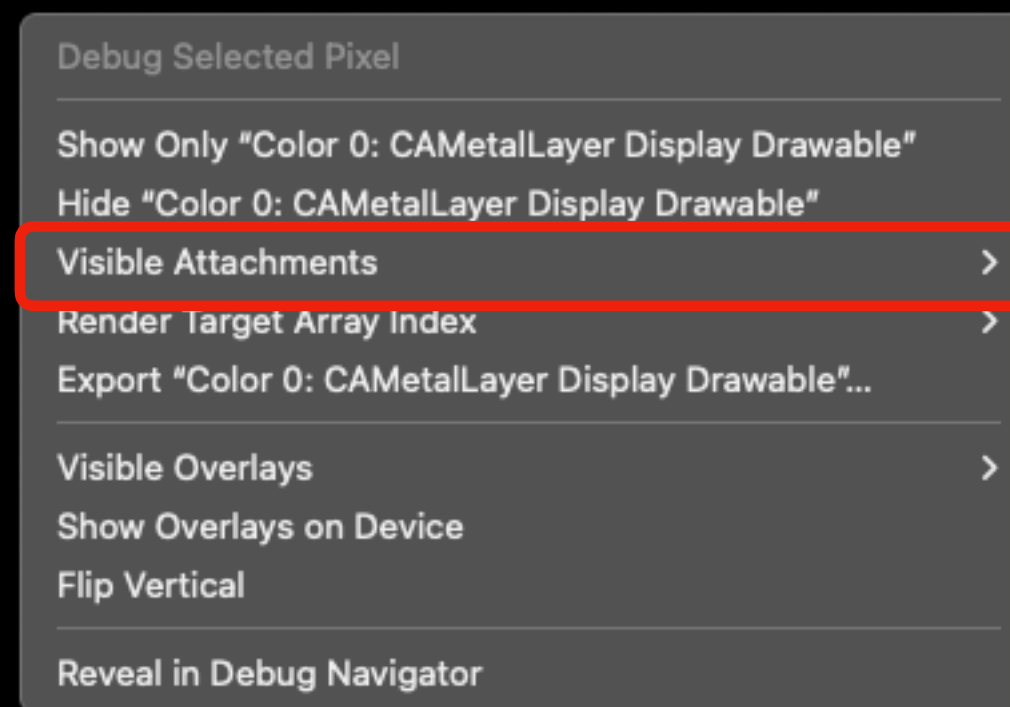
Attachments

The screenshot displays the MetalToolsDemo application interface for debugging GPU workloads. The left sidebar shows a tree view of the captured GPU workload, with the 'Attachments' section selected for a specific render command (ID 52). The main area shows a 3D visualization of the rendered scene, which is currently obscured by a large, semi-transparent green rectangular overlay. Below the visualization, a detailed list of attachments is visible, including pipeline performance, vertex buffers, and geometry data.

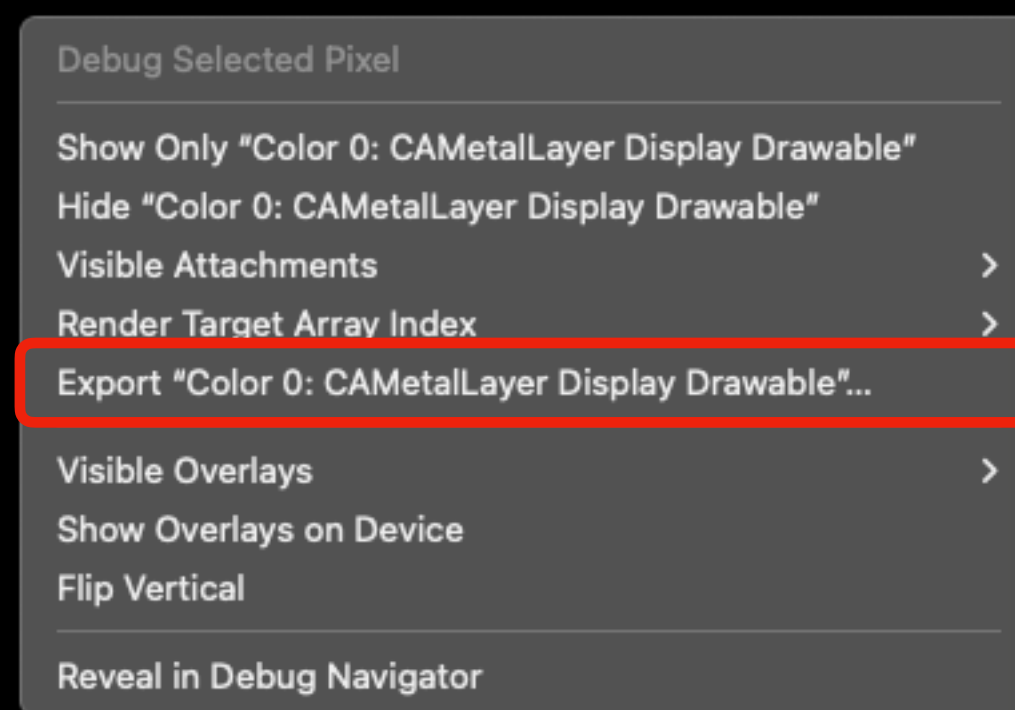
Attachments List:

- Color 0: CAMetalLayer Display Drawable
- "Pipeline: Render cats" (0x104035a00) vshRenderer00 - fshRenderer00
- RenderPipeline Performance 140.60 ms (100.0%)
- Vertex Buffer 0 (MTLBuffer) 0x10310e740
- Vertex Buffer 1 (MTLBuffer) 0x10310eca0
- Vertex Buffer 2 (MTLBuffer) Bytes
- Vertex Buffer 3 (MTLBuffer) Bytes
- Vertex Buffer 4 (MTLBuffer) Bytes
- Vertex Buffer 5 (MTLBuffer) Bytes
- Vertex Buffer 6 (MTLBuffer) Bytes

Attachments



Attachments



Attachments

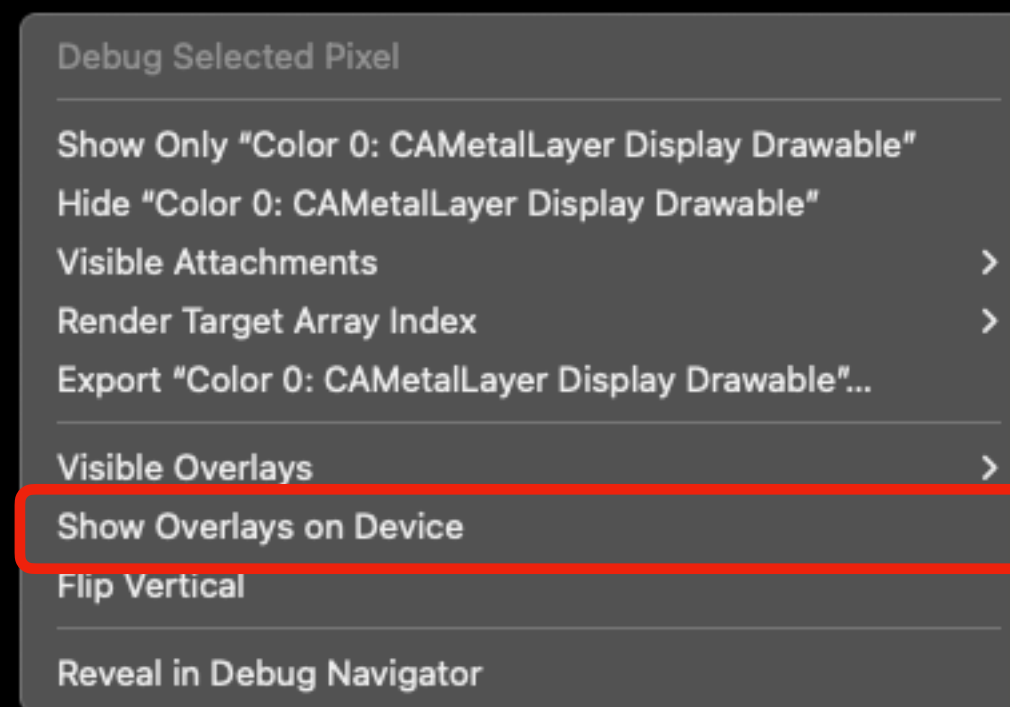
Debug Selected Pixel

- Show Only "Color 0: CAMetalLayer Display Drawable"
- Hide "Color 0: CAMetalLayer Display Drawable"
- Visible Attachments >
- Render Target Array Index >
- Export "Color 0: CAMetalLayer Display Drawable"...
- Visible Overlays >**
- Show Overlays on Device
- Flip Vertical

Reveal in Debug Navigator

Outline
✓ Wireframe

Attachments



Attachment: Information

The screenshot displays the MetalToolsDemo application interface for debugging GPU workloads. The left sidebar shows a tree view of the captured GPU workload, including a 'Render' group with a '44 Render = [renderComman...]' and a '52 [drawIndexedPrim...]' entry. The main window shows a detailed view of the '52 [drawIndexedPrim...]' entry, which is a 'Color 0: CAMetalLayer Display Drawable'. A tooltip provides details for this attachment: Type 2D, Pixel Format BGRA8Unorm, Mipmap Level 0, Width 1125, and Height 2436. The bottom panel shows a callstack for the pipeline 'Render cats' and lists various buffers, including Vertex Buffer 0 through 6.

MetalToolsDemo master
MetalToolsDemo - Debugging GPU Workload 2

Renderer00.metal | Renderer00.m | MetalToolsDemo.gputrace | MetalUtils.m | MetalUtils.h

52 [drawIndexedPrimitives:Triangle indexCount:1857 indexType:...dexBuffer:0x10310eb50 indexBufferOffset:0 instanceCount:256] Attachments

Summary
Performance
Memory 48.2 MB

Group by API Call

- Main Frame
- Main Frame
- Group #1 27.54 ms
 - Group #2 4.58 μs
 - Group #3 27.53 ms
 - Render 27.53 ms
 - 44 Render = [renderComman...
 - 52 [drawIndexedPrim... 27.53 ms
 - Attachments
 - Geometry
 - Bound Resources
 - All Resources
 - Pipeline Statistics
 - Performance
 - Callstack

56 [presentDrawable:0x281f24100]

- Main Frame
- Main Frame
- Main Frame

Type 2D
Pixel Format BGRA8Unorm
Mipmap Level 0
Width 1125
Height 2436

Color 0: CAMetalLayer Display Drawable

"Pipeline: Render cats" (0x104035a00) vshRenderer00 - fshRenderer00

- RenderPipeline Performance 140.60 ms (100.0%)
- Vertex Buffer 0 (MTLBuffer) 0x10310e740
- Vertex Buffer 1 (MTLBuffer) 0x10310eca0
- Vertex Buffer 2 (MTLBuffer) Bytes
- Vertex Buffer 3 (MTLBuffer) Bytes
- Vertex Buffer 4 (MTLBuffer) Bytes
- Vertex Buffer 5 (MTLBuffer) Bytes
- Vertex Buffer 6 (MTLBuffer) Bytes

Attachment: Settings

The screenshot displays the MetalToolsDemo application interface, which is used for debugging GPU workloads. The interface is divided into several sections:

- Left Panel (Summary/Performance/Memory):** Shows a tree view of the captured GPU workload. The 'Render' group is expanded, showing a list of render commands. The selected command is '52 [drawIndexedPrim...]' with a duration of 27.53 ms.
- Bottom Left Panel (Attachments):** Lists various attachments for the selected command, including 'Attachments', 'Geometry', 'Bound Resources', 'All Resources', 'Pipeline Statistics', 'Performance', and 'Callstack'. The 'Attachments' attachment is currently selected.
- Center Panel (Attachment View):** Displays the 'Attachments' view for the selected command. It shows a 3D scene rendered with a green color. The color is labeled 'Color 0: CAMetalLayer Display Drawable'. Below the scene, there is a 'Pipeline: Render cats' section with a performance breakdown: 'RenderPipeline Performance 140.60 ms (100.0%)', followed by six 'Vertex Buffer' entries (0-6) with their respective MTLBuffer addresses and sizes.
- Right Panel (Settings):** Shows the settings for the selected attachment. It includes a 'Color' section with a 'RGB' color picker (set to 0) and an 'Alpha' section (set to 1). The 'Combine RGB' checkbox is checked. Below these are color selection options for Red, Green, Blue, and Alpha, each with a corresponding color swatch and a 'Pre-Multiplied Alpha' checkbox.

Attachment: Inspect Pixel

The screenshot displays the MetalToolsDemo application interface. On the left, a sidebar shows a tree view of the captured GPU workload, including a 'Render' group with a selected item '52 [drawIndexedPrim...]'. The main area shows a 3D scene with a green pixel color overlay. A circular magnifying glass highlights a specific pixel, with a tooltip displaying its color values: R: 0.117647, G: 0.407843, B: 0.352941, and A: 1.000000. The bottom right corner features a 'Debug' button and a red square icon.

Color 0: CAMetalLayer Display Drawable

R:	0.117647
G:	0.407843
B:	0.352941
A:	1.000000

Debug

Attachment: Debug

The screenshot displays the MetalToolsDemo application in a debugger, showing a captured GPU workload. The interface is divided into several sections:

- Left Panel (Summary/Performance):** Shows the workload summary, performance metrics (48.2 MB memory), and a tree view of API calls. The selected call is `52 [drawIndexedPrim...` (27.53 ms).
- Right Panel (Attachments):** Shows a rendered scene with a color debug overlay. A circular selection highlights a specific pixel, with its color values displayed: R: 0.117647, G: 0.407843, B: 0.352941, A: 1.000000. The overlay is labeled `Color 0: CAMetalLayer Display Drawable`.
- Bottom Panel (Callstack):** Shows the callstack for the selected call, including the pipeline `"Pipeline: Render cats" (0x104035a00)` and various vertex buffers.

The `Debug` button in the bottom right corner is highlighted with a red box, indicating the active debug mode.

Geometry

The screenshot displays the MetalToolsDemo GPU debugger interface. The main window shows a 3D view of a green cat geometry with a white bounding box and a small blue square. The left sidebar shows a tree view of the GPU workload, with the 'Geometry' node selected. The bottom panel shows a table of vertex data for Instance 0.

Index	in - ushort Vertex	in - float4 position		in - float2 texture		out - float4 position		out - float texCoord					
0	367	0.000	-0.243	0.000	1.000	0.500	0.621	0.565	-0.105	0.000	1.000	0.500	0.000
1	399	-0.152	0.005	0.000	1.000	0.424	0.497	0.348	-0.147	0.000	1.000	0.424	0.000
2	397	-0.067	-0.042	0.000	1.000	0.466	0.521	0.396	-0.118	0.000	1.000	0.466	0.000
3	399	-0.152	0.005	0.000	1.000	0.424	0.497	0.348	-0.147	0.000	1.000	0.424	0.000

Below the table, the debugger shows the pipeline state for 'Pipeline: Render cats' (0x104035a00) and lists the vertex buffers used: Vertex Buffer 0 (MTLBuffer 0x10310e740), Vertex Buffer 1 (MTLBuffer 0x10310eca0), and Vertex Buffers 2-6 (MTLBuffer Bytes).

Geometry

- Debug Selected Vertex
- Visualize Vertex Attribute >
- Change Cull Mode >

- Show Render Attachment
- Hide Wireframe
- Hide Vertices

- Zoom to Selected Primitive
- Reset View

- Export Geometry...

Geometry

The screenshot displays the MetalToolsDemo GPU debugger interface. The left sidebar shows a tree view of the captured GPU workload, with the 'Geometry' node selected under the 'Render' group. The main window shows a 3D scene of overlapping, colorful spheres (red, orange, yellow, green) with a white bounding box around a central cluster. Below the scene is a table of vertex data for Instance 0.

Index	in - ushort Vertex	in - float4 position			in - float2 texture		out - float4 position			out - float texCoord			
0	367	0.000	-0.243	0.000	1.000	0.500	0.621	0.565	-0.105	0.000	1.000	0.500	0.000
1	399	-0.152	0.005	0.000	1.000	0.424	0.497	0.348	-0.147	0.000	1.000	0.424	0.000
2	397	-0.067	-0.042	0.000	1.000	0.466	0.521	0.396	-0.118	0.000	1.000	0.466	0.000
3	399	-0.152	0.005	0.000	1.000	0.424	0.497	0.348	-0.147	0.000	1.000	0.424	0.000

The bottom panel shows the pipeline state for 'Pipeline: Render cats' (0x104035a00) and lists the vertex buffers used in the pipeline:

- RenderPipeline Performance 140.60 ms (100.0%)
- Vertex Buffer 0 (MTLBuffer) 0x10310e740
- Vertex Buffer 1 (MTLBuffer) 0x10310eca0
- Vertex Buffer 2 (MTLBuffer) Bytes
- Vertex Buffer 3 (MTLBuffer) Bytes
- Vertex Buffer 4 (MTLBuffer) Bytes
- Vertex Buffer 5 (MTLBuffer) Bytes
- Vertex Buffer 6 (MTLBuffer) Bytes

Geometry

- Debug Selected Vertex
- Visualize Vertex Attribute >
- Change Cull Mode >
- Show Render Attachment
- Hide Wireframe
- Hide Vertices
- Zoom to Selected Primitive
- Reset View
- Export Geometry...

Geometry

- Debug Selected Vertex
- Visualize Vertex Attribute >
- Change Cull Mode >
- Show Render Attachment
- Hide Wireframe
- Hide Vertices
- Zoom to Selected Primitive
- Reset View
- Export Geometry...

Geometry

- Debug Selected Vertex
- Visualize Vertex Attribute >
- Change Cull Mode >
- Show Render Attachment
- Hide Wireframe
- Hide Vertices
- Zoom to Selected Primitive
- Reset View
- Export Geometry...

Bound Resources: Pipeline

The screenshot displays the MetalToolsDemo application interface. The left sidebar shows a tree view of the GPU workload, with the 'Bound Resources' view selected. The main panel shows a table of resources for the pipeline 'Pipeline: Render cats'.

Label	Type	Size	Details	Parameter Name	Resource Usage
Pipeline States					
R Pipeline: Render cats					
Vertex					
B Buffer 0x10310eb50	Index	4 KB	Offset: 0x0		
B Buffer 0x10310e740	Buffer 0	18 KB	Offset: 0x0	vertexBuffer.0	Read
B Buffer 0x10310eca0	Buffer 1	16 KB	Offset: 0x0	transforms	Read
B Vertex Bytes	Buffer 2 (Bytes)	4 bytes		ratio	Read
Geometry	Post Vertex Tran...				
B Vertex Attributes	Vertex Attributes				
S vshRenderer00	Vertex Function		Library 0x282452e80 [Pr...		
Fragment					
Texture 0x10310e380	Texture 0	1936 x 1936	BGRA8Unorm	image	Read
B Fragment Bytes	Buffer 0 (Bytes)	4 bytes		time_	Read
S fshRenderer00	Fragment Function		Library 0x282452e80 [Pr...		
Attachments					
CAMetalLayer Display Dra...	Color 0	1125 x 2436	BGRA8Unorm		Write

Below the table, a list of resources is shown for the pipeline: "Pipeline: Render cats" (0x104035a00) vshRenderer00 - fshRenderer00. The list includes: RenderPipeline Performance (140.60 ms (100.0%)), Vertex Buffer 0 (MTLBuffer) 0x10310e740, Vertex Buffer 1 (MTLBuffer) 0x10310eca0, Vertex Buffer 2 (MTLBuffer) Bytes, Vertex Buffer 3 (MTLBuffer) Bytes, Vertex Buffer 4 (MTLBuffer) Bytes, Vertex Buffer 5 (MTLBuffer) Bytes, and Vertex Buffer 6 (MTLBuffer) Bytes.

Bound Resources: Pipeline

The screenshot displays the MetalToolsDemo application interface for debugging GPU workloads. The left sidebar shows a tree view of the captured GPU workload, with the 'Bound Resources' view selected for a specific render command. The top breadcrumb navigation indicates the current view: 'MetalToolsDemo.gputrace > Main Frame > Group #1 > Group #3 > Render > 52 [drawIndexe...anceCount:256] > Bound Resources > Pipeline: Render cats'. The central pane shows the details of the 'Pipeline: Render cats', including functions, buffers, attachments, and rasterization/visibility state. The bottom pane shows a timeline of the pipeline execution, including performance metrics and resource usage.

Left Sidebar: Captured GPU Workload

- MetalToolsDemo Captured GPU Worklo...
- Summary
- Performance
- Memory 48.2 MB
- Group by API Call
- Main Frame
- Main Frame
- Group #1 27.54 ms
 - Group #2 4.58 μs
 - Group #3 27.53 ms
 - Render 27.53 ms
 - 44 Render = [renderComman...
 - 52 [drawIndexedPrim... 27.53 ms
 - Attachments
 - Geometry
 - Bound Resources**
 - All Resources
 - Pipeline Statistics
 - Performance
 - Callstack

- Main Frame
- Main Frame

Top Breadcrumb: Pipeline: Render cats

Central Pane: Pipeline: Render cats

Label	Details
Functions	
> Vertex Function	vshRenderer00
> Fragment Function	fshRenderer00
Buffers	
> Vertex Descriptor	
> Vertex Buffers	31 Items
> Fragment Buffers	31 Items
Attachments	
> Color Attachments 0	
Depth Attachment Pixel Forma	Invalid
Stencil Attachment Pixel Form	Invalid
Rasterization and Visibility State	
Alpha To Coverage Enabled	false
Alpha To One Enabled	false
Rasterization Enabled	true
Input Primitive Topology	Unspecified

Bottom Timeline: Pipeline: Render cats

- "Pipeline: Render cats" (0x104035a00) vshRenderer00 - fshRenderer00
- RenderPipeline Performance 140.60 ms (100.0%)
- Vertex Buffer 0 (MTLBuffer) 0x10310e740
- Vertex Buffer 1 (MTLBuffer) 0x10310eca0
- Vertex Buffer 2 (MTLBuffer) Bytes
- Vertex Buffer 3 (MTLBuffer) Bytes
- Vertex Buffer 4 (MTLBuffer) Bytes
- Vertex Buffer 5 (MTLBuffer) Bytes
- Vertex Buffer 6 (MTLBuffer) Bytes

Bound Resources: Buffer

The screenshot displays the MetalToolsDemo interface, showing a captured GPU workload. The left sidebar contains a tree view of the workload, with 'Bound Resources' selected. The main panel shows a table of bound resources for the pipeline 'Render cats'.

Label	Type	Size	Details	Parameter Name	Resource Usage
Pipeline States					
Pipeline: Render cats					
Vertex					
Buffer 0x10310eb50	Index	4 KB	Offset: 0x0		
Buffer 0x10310e740	Buffer 0	18 KB	Offset: 0x0	vertexBuffer.0	Read
Buffer 0x10310eca0	Buffer 1	16 KB	Offset: 0x0	transforms	Read
Vertex Bytes	Buffer 2 (Bytes)	4 bytes		ratio	Read
Geometry	Post Vertex Tran...				
Vertex Attributes	Vertex Attributes				
vshRenderer00	Vertex Function		Library 0x282452e80 [Pr...		
Fragment					
Texture 0x10310e380	Texture 0	1936 x 1936	BGRA8Unorm	image	Read
Fragment Bytes	Buffer 0 (Bytes)	4 bytes		time_	Read
fshRenderer00	Fragment Function		Library 0x282452e80 [Pr...		
Attachments					
CAMetalLayer Display Dra...	Color 0	1125 x 2436	BGRA8Unorm		Write

The bottom panel shows the pipeline state: "Pipeline: Render cats" (0x104035a00) vshRenderer00 - fshRenderer00. It lists various buffers, including Vertex Buffer 0 through 6, and their respective sizes and types.

Bound Resources: Buffer

The screenshot displays the MetalToolsDemo application interface for debugging GPU workloads. The left sidebar shows a tree view of the captured GPU workload, with the 'Bound Resources' section selected. The main area shows a table of bound resources, specifically 'Vertex Buffer 1' at offset 0x10310eca0. The table lists 7 rows of data, each representing a vertex buffer with a specific offset and a 4x4 matrix of float values. The bottom panel shows the pipeline details for 'Pipeline: Render cats' and lists various resources used, including Vertex Buffers 0 through 6.

Row	Offset	float4x4 transforms
0	0x0	0.124 0.801 0.000 0.000 -0.801 0.124 0.000 0.000 0.000 0.000 1.000 0.000 0.371 -0.197 0.000 1.000
1	0x40	0.170 0.934 0.000 0.000 -0.934 0.170 0.000 0.000 0.000 0.000 1.000 0.000 0.029 -0.128 0.000 1.000
2	0x80	-0.078 0.974 0.000 0.000 -0.974 -0.078 0.000 0.000 0.000 0.000 1.000 0.000 -1.308 1.349 0.000 1.000
3	0xC0	-0.283 -0.186 0.000 0.000 0.186 -0.283 0.000 0.000 0.000 0.000 1.000 0.000 -1.296 0.786 0.000 1.000
4	0x100	-0.013 1.000 0.000 0.000 -1.000 -0.013 0.000 0.000 0.000 0.000 1.000 0.000 0.413 0.350 0.000 1.000
5	0x140	-0.219 0.368 0.000 0.000 -0.368 -0.219 0.000 0.000 0.000 0.000 1.000 0.000 0.664 0.764 0.000 1.000
6	0x180	0.484 0.659 0.000 0.000 -0.659 0.484 0.000 0.000 0.000 0.000 1.000 0.000 0.221 0.148 0.000 1.000

Bound Resources: Vertex Attributes

The screenshot displays the MetalToolsDemo debugger interface. The left sidebar shows a tree view of the GPU workload, with 'Bound Resources' selected under the 'Render' group. The main panel shows a table of resources for the selected pipeline state.

Label	Type	Size	Details	Parameter Name	Resource Usage
Pipeline States					
Pipeline: Render cats					
Vertex					
Buffer 0x10310eb50	Index	4 KB	Offset: 0x0		
Buffer 0x10310e740	Buffer 0	18 KB	Offset: 0x0	vertexBuffer.0	Read
Buffer 0x10310eca0	Buffer 1	16 KB	Offset: 0x0	transforms	Read
Vertex Bytes	Buffer 2 (Bytes)	4 bytes		ratio	Read
Geometry	Post Vertex Tran...				
Vertex Attributes	Vertex Attributes				
vshRenderer00	Vertex Function		Library 0x282452e80 [Pr...		
Fragment					
Texture 0x10310e380	Texture 0	1936 x 1936	BGRA8Unorm	image	Read
Fragment Bytes	Buffer 0 (Bytes)	4 bytes		time_	Read
fshRenderer00	Fragment Function		Library 0x282452e80 [Pr...		
Attachments					
CAMetalLayer Display Dra...	Color 0	1125 x 2436	BGRA8Unorm		Write

The 'Vertex Attributes' row is highlighted with a red border. Below the table, a callstack view shows the pipeline state: 'Pipeline: Render cats' (0x104035a00) using 'vshRenderer00' and 'fshRenderer00'. A performance summary shows 'RenderPipeline Performance' at 140.60 ms (100.0%). A list of vertex buffers is also visible: 'Vertex Buffer 0' (MTLBuffer) 0x10310e740, 'Vertex Buffer 1' (MTLBuffer) 0x10310eca0, and 'Vertex Buffer 2' (MTLBuffer) Bytes.

Bound Resources: Vertex Attributes

The screenshot displays the MetalToolsDemo GPU workload debugger. The left sidebar shows a tree view of the captured GPU workload, with the 'Bound Resources' section selected. The main window shows a table of vertex attributes for Instance 0, with columns for Index, in-ushort Vertex, in-float4 position, in-float2 texture, out-float4 position, and out-float4 texCoord. Below the table, the 'Vertex Attributes' section shows the pipeline 'Render cats' and lists the vertex buffers used, including Vertex Buffer 0 (MTLBuffer 0x10310e740) and Vertex Buffer 1 (MTLBuffer 0x10310eca0).

Index	in - ushort Vertex	in - float4 position		in - float2 texture		out - float4 position		out - float4 texCoord					
0	367	0.000	-0.243	0.000	1.000	0.500	0.621	0.565	-0.105	0.000	1.000	0.500	0.000
1	399	-0.152	0.005	0.000	1.000	0.424	0.497	0.348	-0.147	0.000	1.000	0.424	0.000
2	397	-0.067	-0.042	0.000	1.000	0.466	0.521	0.396	-0.118	0.000	1.000	0.466	0.000
3	399	-0.152	0.005	0.000	1.000	0.424	0.497	0.348	-0.147	0.000	1.000	0.424	0.000
4	367	0.000	-0.243	0.000	1.000	0.500	0.621	0.565	-0.105	0.000	1.000	0.500	0.000
5	400	-0.184	0.013	0.000	1.000	0.408	0.493	0.337	-0.158	0.000	1.000	0.408	0.000
6	0	-0.084	0.928	0.000	1.000	0.458	0.036	-0.383	-0.069	0.000	1.000	0.458	0.000
7	1	-0.099	0.923	0.000	1.000	0.450	0.039	-0.380	-0.075	0.000	1.000	0.450	0.000
8	2	-0.175	0.913	0.000	1.000	0.413	0.043	-0.382	-0.103	0.000	1.000	0.413	0.000
9	1	-0.099	0.923	0.000	1.000	0.450	0.039	-0.380	-0.075	0.000	1.000	0.450	0.000
10	0	-0.084	0.928	0.000	1.000	0.458	0.036	-0.383	-0.069	0.000	1.000	0.458	0.000
11	586	-0.072	0.884	0.000	1.000	0.464	0.058	-0.346	-0.067	0.000	1.000	0.464	0.000
12	2	-0.175	0.913	0.000	1.000	0.413	0.043	-0.382	-0.103	0.000	1.000	0.413	0.000
13	1	-0.099	0.923	0.000	1.000	0.450	0.039	-0.380	-0.075	0.000	1.000	0.450	0.000
14	585	-0.163	0.872	0.000	1.000	0.418	0.064	-0.348	-0.102	0.000	1.000	0.418	0.000
15	3	-0.250	0.896	0.000	1.000	0.375	0.052	-0.378	-0.132	0.000	1.000	0.375	0.000
16	2	-0.175	0.913	0.000	1.000	0.413	0.043	-0.382	-0.103	0.000	1.000	0.413	0.000
17	584	-0.233	0.854	0.000	1.000	0.383	0.073	-0.342	-0.129	0.000	1.000	0.383	0.000
18	535	-0.160	0.522	0.000	1.000	0.420	0.239	-0.067	-0.121	0.000	1.000	0.420	0.000
19	533	-0.167	0.506	0.000	1.000	0.416	0.247	-0.055	-0.124	0.000	1.000	0.416	0.000
20	531	-0.171	0.499	0.000	1.000	0.414	0.251	-0.050	-0.126	0.000	1.000	0.414	0.000
21	533	-0.167	0.506	0.000	1.000	0.416	0.247	-0.055	-0.124	0.000	1.000	0.416	0.000
22	535	-0.160	0.522	0.000	1.000	0.420	0.239	-0.067	-0.121	0.000	1.000	0.420	0.000
23	534	-0.159	0.519	0.000	1.000	0.420	0.241	-0.064	-0.120	0.000	1.000	0.420	0.000
24	4	-0.304	0.879	0.000	1.000	0.348	0.060	-0.371	-0.153	0.000	1.000	0.348	0.000
25	3	-0.250	0.896	0.000	1.000	0.375	0.052	-0.378	-0.132	0.000	1.000	0.375	0.000

Vertex Attributes: 1
Offset:

"Pipeline: Render cats" (0x104035a00) vshRenderer00 - fshRenderer00

- RenderPipeline Performance 140.60 ms (100.0%)
- Vertex Buffer 0 (MTLBuffer) 0x10310e740
- Vertex Buffer 1 (MTLBuffer) 0x10310eca0
- Vertex Buffer 2 (MTLBuffer) Bytes
- Vertex Buffer 3 (MTLBuffer) Bytes
- Vertex Buffer 4 (MTLBuffer) Bytes
- Vertex Buffer 5 (MTLBuffer) Bytes
- Vertex Buffer 6 (MTLBuffer) Bytes

Bound Resources: Texture

The screenshot displays the MetalToolsDemo interface for debugging GPU workloads. The left sidebar shows a tree view of the captured GPU workload, with the 'Bound Resources' view selected. The main panel shows a table of resources used by the pipeline, with a red box highlighting the texture resource.

Label	Type	Size	Details	Parameter Name	Resource Usage
Pipeline States					
Pipeline: Render cats					
Vertex					
Buffer 0x10310eb50	Index	4 KB	Offset: 0x0		
Buffer 0x10310e740	Buffer 0	18 KB	Offset: 0x0	vertexBuffer.0	Read
Buffer 0x10310eca0	Buffer 1	16 KB	Offset: 0x0	transforms	Read
Vertex Bytes	Buffer 2 (Bytes)	4 bytes		ratio	Read
Geometry	Post Vertex Tran...				
Vertex Attributes	Vertex Attributes				
vshRenderer00	Vertex Function		Library 0x282452e80 [Pr...		
Fragment					
Texture 0x10310e380	Texture 0	1936 x 1936	BGRA8Unorm	image	Read
Fragment Bytes	Buffer 0 (Bytes)	4 bytes		time_	Read
fshRenderer00	Fragment Function		Library 0x282452e80 [Pr...		
Attachments					
CAMetalLayer Display Dra...	Color 0	1125 x 2436	BGRA8Unorm		Write

The bottom panel shows a callstack for the pipeline: "Pipeline: Render cats" (0x104035a00) vshRenderer00 - fshRenderer00. The callstack includes the following items:

- RenderPipeline Performance 140.60 ms (100.0%)
- Vertex Buffer 0 (MTLBuffer) 0x10310e740
- Vertex Buffer 1 (MTLBuffer) 0x10310eca0
- Vertex Buffer 2 (MTLBuffer) Bytes
- Vertex Buffer 3 (MTLBuffer) Bytes
- Vertex Buffer 4 (MTLBuffer) Bytes
- Vertex Buffer 5 (MTLBuffer) Bytes
- Vertex Buffer 6 (MTLBuffer) Bytes

Bound Resources: Texture

The screenshot displays the MetalToolsDemo application interface. The left sidebar shows a tree view of the captured GPU workload, with the 'Bound Resources' section selected. The main window shows a circular texture resource containing four cat images. The bottom panel displays the pipeline details for the 'Render cats' pipeline, listing various buffers and their addresses.

Bound Resources

- Texture 0x10310e380

Pipeline: Render cats (0x104035a00) vshRenderer00 - fshRenderer00

- RenderPipeline Performance 140.60 ms (100.0%)
- Vertex Buffer 0 (MTLBuffer) 0x10310e740
- Vertex Buffer 1 (MTLBuffer) 0x10310eca0
- Vertex Buffer 2 (MTLBuffer) Bytes
- Vertex Buffer 3 (MTLBuffer) Bytes
- Vertex Buffer 4 (MTLBuffer) Bytes
- Vertex Buffer 5 (MTLBuffer) Bytes
- Vertex Buffer 6 (MTLBuffer) Bytes

Bound Resources: Shader

The screenshot displays the MetalToolsDemo debugger interface. The left sidebar shows a tree view of the GPU workload, with the 'Bound Resources' view selected. The main panel shows a table of bound resources for the 'Pipeline: Render cats'.

Label	Type	Size	Details	Parameter Name	Resource Usage
Pipeline States					
Pipeline: Render cats					
Vertex					
Buffer 0x10310eb50	Index	4 KB	Offset: 0x0		
Buffer 0x10310e740	Buffer 0	18 KB	Offset: 0x0	vertexBuffer.0	Read
Buffer 0x10310eca0	Buffer 1	16 KB	Offset: 0x0	transforms	Read
Vertex Bytes	Buffer 2 (Bytes)	4 bytes		ratio	Read
Geometry	Post Vertex Tran...				
Vertex Attributes	Vertex Attributes				
vshRenderer00	Vertex Function		Library 0x282452e80 [Pr...		
Fragment					
Texture 0x10310e380	Texture 0	1936 x 1936	BGRA8Unorm	image	Read
Fragment Bytes	Buffer 0 (Bytes)	4 bytes		time	Read
fshRenderer00	Fragment Function		Library 0x282452e80 [Pr...		
Attachments					
CAMetalLayer Display Dra...	Color 0	1125 x 2436	BGRA8Unorm		Write

The 'fshRenderer00' entry in the Fragment section is highlighted with a red box. Below the table, a callstack is visible, showing the pipeline state and various buffers.

Bound Resources: Shader

The screenshot displays the MetalToolsDemo application's GPU workload debugging interface. The left sidebar shows a tree view of the captured GPU workload, with the 'Bound Resources' view selected for the 'Render' stage. The main window shows the shader code for 'fshRenderer00' and its performance breakdown.

Shader Code:

```
87 fragment half4 fshRenderer00(ColorInOut in [[stage_in]],
88     constant float &time_ [[buffer(0)]],
89     texture2d<half> image [[ texture(0) ]])
90 {
91     constexpr sampler imageSampler(address::clamp_to_zero, filter::nearest);
92
93     float2 shift = in.texCoord - 0.5;
94     float angle = atan2(shift.y, shift.x);
95
96     half time = half(time_);
97     half4 src = image.sample(imageSampler, in.texCoord);
98     half3 hsv = rgb2hsv(src.rgb);
99     half3 col = time_color(half2(in.texCoord), time);
100    hsv.x = pos_fract(fma(half(angle), 0.5h / M_PI_H, 0.5h) - fma(time, 0.25h,
101        -rgb2hsv(col).x));
102
103    half k = smoothstep(0.0h, 1.0h, length(half2(shift)));
104    hsv.y = fma(hsv.y, 1.0h - k, k);
105    src.rgb = hsv2rgb(max(hsv, 0.0h));
106
107    return src;
108 }
109
110
```

Performance Breakdown:

Line	Code Snippet	Percentage
87	fragment half4 fshRenderer00...	138.87 ms
91	constexpr sampler imageSampler...	1.22%
93-94	float2 shift = in.texCoord - 0.5; float angle = atan2...	6.84%
97	half4 src = image.sample...	28.78%
98	half3 hsv = rgb2hsv...	23.14%
99	half3 col = time_color...	16.86%
100	hsv.x = pos_fract...	6.68%
103	half k = smoothstep...	3.00%
104	hsv.y = fma...	0.67%
105	src.rgb = hsv2rgb...	8.35%
107	return src;	4.47%

Resource List:

- "Pipeline: Render cats" (0x104035a00) vshRenderer00 - fshRenderer00
- RenderPipeline Performance 140.60 ms (100.0%)
- Vertex Buffer 0 (MTLBuffer) 0x10310e740
- Vertex Buffer 1 (MTLBuffer) 0x10310eca0
- Vertex Buffer 2 (MTLBuffer) Bytes
- Vertex Buffer 3 (MTLBuffer) Bytes
- Vertex Buffer 4 (MTLBuffer) Bytes
- Vertex Buffer 5 (MTLBuffer) Bytes
- Vertex Buffer 6 (MTLBuffer) Bytes

Shader: Profiling

MetalToolsDemo master

Renderer00.metal | Renderer00.m | MetalToolsDemo.gputrace

Deployment Target is lower than device OS version. Shader performance data may not be accurate.

```
87 fragment half4 fshRenderer00(ColorInOut in [[st
88         constant float &ti
89         texture2d<half> im
90 {
91     constexpr sampler imageSampler(address::cla
92
93     float2 shift = in.texCoord - 0.5;
94     float angle = atan2(shift.y, shift.x);
95
96     half time = half(time_);
97     half4 src = image.sample(imageSampler, in.t
98     half3 hsv = rgb2hsv(src.rgb);
99     half3 col = time_color(half2(in.texCoord), time);
100    hsv.x = pos_fract(fma(half(angle), 0.5h / M_PI_H, 0.5h) - fma(time, 0.25h,
101                    -rgb2hsv(col).x));
102
103    half k = smoothstep(0.0h, 1.0h, length(half2(shift)));
104    hsv.y = fma(hsv.y, 1.0h - k, k);
105    src.rgb = hsv2rgb(max(hsv, 0.0h));
106
107    return src;
108 }
109
110
```

Summary

Performance

Memory 48.2 MB

Group by API Call

- Main Frame
- Main Frame
- Group #1 27.54 ms
 - Group #2 4.58 μs
 - Group #3 27.53 ms
 - Render 27.53 ms
 - 44 Render = [renderComman...
 - 52 [drawIndexedPrim... 27.53 ms
 - Attachments
 - Geometry
 - Bound Resources
 - All Resources
 - Pipeline Statistics
 - Performance
 - Callstack
- 56 [presentDrawable:0x281f24100]
- Main Frame
- Main Frame
- Main Frame

Callstack

- "Pipeline: Render cats" (0x104035a00) vshRenderer00 - fshRenderer00
- RenderPipeline Performance 140.60 ms (100.0%)
- Vertex Buffer 0 (MTLBuffer) 0x10310e740
- Vertex Buffer 1 (MTLBuffer) 0x10310eca0
- Vertex Buffer 2 (MTLBuffer) Bytes
- Vertex Buffer 3 (MTLBuffer) Bytes
- Vertex Buffer 4 (MTLBuffer) Bytes
- Vertex Buffer 5 (MTLBuffer) Bytes
- Vertex Buffer 6 (MTLBuffer) Bytes

Shader Profile

Category	Sub-category	Percentage
ALU		51.46%
Memory	Load	0.52%
	Store	0.61%
	Sample	28.11%
	Control Flow	0.17%
Synchronization	Wait Memory	19.13%
		19.13%
Complex	Float	16.02%
	Half	24.54%
	Complex	10.35%
	Input Interpolat...	0.56%

Line: 87 Col: 1

Shader: Profiling

The screenshot displays the Xcode IDE with the MetalToolsDemo project open. The interface is divided into three main sections:

- Left Panel (GPU Workload Hierarchy):** Shows a tree view of the captured GPU workload. The 'Render' call is selected, showing its sub-calls: '44 Render = [renderComman...', '52 [drawIndexedPrim...', and '56 [presentDrawable:0x281f24100]'. The '52 [drawIndexedPrim...' call is further expanded to show 'Attachments', 'Geometry', 'Bound Resources', 'All Resources', 'Pipeline Statistics', 'Performance', and 'Callstack'.
- Center Panel (Code Editor):** Displays the source code for the 'fshRenderer00' fragment shader. The code is as follows:

```
87 fragment half4 fshRenderer00(ColorInOut in [[stage_in]],
88     constant float &time_ [[buffer(0)]],
89     texture2d<half> image [[ texture(0) ]])
90 {
91     constexpr sampler imageSampler(address::clamp_to_zero, filter::nearest);
92
93     float2 shift = in.texCoord - 0.5;
94     float angle = atan2(shift.y, shift.x);
95
96     half time = half(time_);
97     half4 src = image.sample(imageSampler, in.texCoord);
98     half3 hsv = rgb2hsv(src.rgb);
99     half3 col = time_color(half2(in.texCoord), time);
100    hsv.x = pos_fract(fma(half(angle), 0.5h / M_PI_H, 0.5h) - fma(time, 0.25h,
101        -rgb2hsv(col).x));
102
103    half k = smoothstep(0.0h, 1.0h, length(half2(shift)));
104    hsv.y = fma(hsv.y, 1.0h - k, k);
105    src.rgb =
106
107    return si
108 }
109
110
```
- Right Panel (Performance Metrics):** Shows the execution time and percentage of total GPU time for each line of code. The data is as follows:

Line	Time	Percentage
87	138.87 ms	
91		1.22%
94		6.84%
97		28.78%
98		23.14%
99		16.86%
100		6.68%
103		3.00%
104		0.67%
105		8.35%
106		4.47%

A context menu is open over the code editor, showing the following options:

- Profile with Induced GPU Performance State
- Minimum
- Medium
- Maximum
- Off (checked)

The bottom of the interface shows the 'Pipeline: Render cats' (0x104035a00) vshRenderer00 - fshRenderer00. The pipeline performance is 140.60 ms (100.0%). The pipeline consists of several vertex buffers (0-6) and a fragment shader.

Shader: Debug

MetalToolsDemo master
MetalToolsDemo - Debugging GPU Workload 2

Renderer00.metal | Renderer00.m | MetalToolsDemo.gputrace | MetalUtils.m | MetalUtils.h

MetalToolsDemo.gputrace > Main Frame > Group #1 > Group #3 > Render > 52 [dra...unt:256] > Bound Resources > Fragment Function — fshRenderer00

Deployment Target is lower than device OS version. Shader performance data may not be available.

```
87 fragment half4 fs
88
89
90 {
91     constexpr sam
92
93     float2 shift
94     float angle =
95
96     half time = h
97     half4 src = i
98     half3 hsv = r
99     half3 col = t
100    hsv.x = pos_f
        -rgb2hsv(
101
102    half k = smoo
103    hsv.y = fma(h
104    src.rgb = hsv
105
106    return src;
107 }
108
109
110
```

Vertex | Fragment

Color 0: CAMetalLayer Display Drawable

	138.87 ms
	1.22%
	6.84%
	28.78%
	23.14%
	16.86%
	6.68%
	3.00%
	0.67%
	8.35%
	4.47%

5h,

Line: 87 Col: 1

56 [presentDrawable:0x281f24100]

- RenderPipeline Performance 140.60 ms (100.0%)
- Vertex Buffer 0 (MTLBuffer) 0x10310e740
- Vertex Buffer 1 (MTLBuffer) 0x10310eca0
- Vertex Buffer 2 (MTLBuffer) Bytes
- Vertex Buffer 3 (MTLBuffer) Bytes
- Vertex Buffer 4 (MTLBuffer) Bytes
- Vertex Buffer 5 (MTLBuffer) Bytes
- Vertex Buffer 6 (MTLBuffer) Bytes

Shader: Debug

The screenshot displays the MetalToolsDemo Shader Debugger interface. The main window shows the MetalToolsDemo.gputrace file with the following code:

```
87 fragment half4 fshRenderer00(ColorInOut in [[stage_in]],
88     constant float &time_ [[buffer(0)]],
89     texture2d<half> image [[ texture(0) ]])
90 {
91     constexpr sampler imageSampler(address::clamp_to_zero, filter::nearest);
92
93     float2 shift = in.texCoord - 0.5;
94     float angle = atan2(shift.y, shift.x);
95
96     half time = half(time_);
97     half4 src = image.sample(imageSampler, in.texCoord);
98     half3 hsv = rgb2hsv(src.rgb);
99     half3 col = time_color(half2(in.texCoord), time);
100    hsv.x = pos_fract(fma(half(angle), 0.5h / M_PI_H, 0.5h) - fma(time, 0.25h,
101        -rgb2hsv(col).x));
102
103    half k = smoothstep(0.0h, 1.0h, length(half2(shift)));
104    hsv.y = fma(hsv.y, 1.0h - k, k);
105    src.rgb = hsv2rgb(max(hsv, 0.0h));
106
107    return src;
108 }
109
110
```

The left sidebar shows the Shader Debugger with a small texture preview and the following variable list:

- constant float &time_ [[buffer(0)]]
- texture2d<half> image [[texture(0)]]
- constexpr sampler imageSampler(ad...
- float2 shift = in.texCoord - 0.5;
- float angle = atan2(shift.y, shift.x);
- half time = half(time_);
- half4 src = image.sample(imageSam...
- half3 hsv = rgb2hsv(src.rgb);
- half3 col = time_color(half2(in.texCo...
- hsv.x = pos_fract(fma(half(angle), 0....
- hsv.x = pos_fract(fma(half(angle), 0....
- half k = smoothstep(0.0h, 1.0h, lengt...
- hsv.y = fma(hsv.y, 1.0h - k, k);
- src.rgb = hsv2rgb(max(hsv, 0.0h));
- return src;

The right sidebar shows the variable values:

- in = { [583.5, 60.5, 0.0, 1] }
- time_ ≈ 3.8
- image = Texture 0x1031C
- imageSampler = Sampler
- shift = [≈0.282, ≈0.217]
- angle ≈ 0.655
- time = 3.801
- src = [0.341, ≈0.286, ≈0]
- hsv = [≈0.084, ≈0.322, C]
- col = [≈0.24, ≈0.852, ≈0]
- hsv = [≈0.059, ≈0.322, C]
- k ≈ 0.289
- hsv = [≈0.059, 0.518, 0.]
- src = [0.341, ≈0.226, ≈0]
- ret = [0.341, ≈0.226, ≈0]

The bottom status bar shows: Line: 87 Col: 1

At the bottom, the input structure is defined: > in = (ColorInOut) { position = [583.5, 60.5, 0.0, 1.0], texCoord = [≈0.782, ≈0.717] }

Shader: Debug

The screenshot displays the MetalToolsDemo Shader Debugger interface. The main window shows the MetalToolsDemo.gputrace file with the following code:

```
87 fragment half4 fshRenderer00(ColorInOut in [[stage_in]],
88     constant float &time_ [[buffer(0)]],
89     texture2d<half> image [[ texture(0) ]])
90 {
91     constexpr sampler imageSampler(address::clamp_to_zero, filter::nearest);
92
93     float2 shift = in.texCoord - 0.5;
94     float angle = atan2(shift.y, shift.x);
95
96     half time = half(time_);
97     half4 src = image.sample(imageSampler, in.texCoord);
98     half3 hsv = rgb2hsv(src.rgb);
99     half3 col = time_color(half2(in.texCoord), time);
100    hsv.x = pos_fract(fma(half(angle), 0.5h / M_PI_H, 0.5h) - fma(time, 0.25h,
101        -rgb2hsv(col).x));
102
103    half k = smoothstep(0.0h, 1.0h, length(half2(shift)));
104    hsv.y = fma(hsv.y, 1.0h - k, k);
105    src.rgb = hsv2rgb(max(hsv, 0.0h));
106
107    return src;
108 }
109
110
```

The Shader Debugger panel on the left shows the following variable values:

- X: 583.5
- Y: 60.5
- Z: 0

The variable list includes:

- constant float &time_ [[buffer(0)]]
- texture2d<half> image [[texture(0)]]
- constexpr sampler imageSampler(ad...
- float2 shift = in.texCoord - 0.5;
- float angle = atan2(shift.y, shift.x);
- half time = half(time_);
- half4 src = image.sample(imageSam...
- half3 hsv = rgb2hsv(src.rgb);
- half3 col = time_color(half2(in.texCo...
- hsv.x = pos_fract(fma(half(angle), 0....
- hsv.x = pos_fract(fma(half(angle), 0....
- half k = smoothstep(0.0h, 1.0h, lengt...
- hsv.y = fma(hsv.y, 1.0h - k, k);
- src.rgb = hsv2rgb(max(hsv, 0.0h));
- return src;

The right panel shows the variable values:

- in = { [583.5, 60.5, 0.0, 1] }
- time_ ≈ 3.8
- image = Texture 0x1031C
- imageSampler = Sampler
- shift = [≈0.282, ≈0.217]
- angle ≈ 0.655
- time = 3.801
- src = [0.341, ≈0.286, ≈0]
- hsv = [≈0.084, ≈0.322, C]
- col = [≈0.24, ≈0.852, ≈0]
- hsv = [≈0.059, ≈0.322, C]
- k ≈ 0.289
- hsv = [≈0.059, 0.518, 0.]
- src = [0.341, ≈0.226, ≈0]
- ret = [0.341, ≈0.226, ≈0]

The bottom status bar shows: Line: 87 Col: 1

Shader: Debug

The screenshot displays the MetalToolsDemo Shader Debugger interface. On the left, a sidebar shows the project structure and a list of variables for the selected fragment function. The main area shows the Metal shader code with line numbers 87 to 110. On the right, a variable inspector shows the current values of variables in the scope.

Shader Code:

```
87 fragment half4 fshRenderer00(ColorInOut in [[stage_in]],
88     constant float &time_ [[buffer(0)]],
89     texture2d<half> image [[ texture(0) ]])
90 {
91     constexpr sampler imageSampler(address::clamp_to_zero, filter::nearest);
92
93     float2 shift = in.texCoord - 0.5;
94     float angle = atan2(shift.y, shift.x);
95
96     half time = half(time_);
97     half4 src = image.sample(imageSampler, in.texCoord);
98     half3 hsv = rgb2hsv(src.rgb);
99     half3 col = time_color(half2(in.texCoord), time);
100    hsv.x = pos_fract(fma(half(angle), 0.5h / M_PI_H, 0.5h) - fma(time, 0.25h,
101        -rgb2hsv(col).x));
102
103    half k = smoothstep(0.0h, 1.0h, length(half2(shift)));
104    hsv.y = fma(hsv.y, 1.0h - k, k);
105    src.rgb = hsv2rgb(max(hsv, 0.0h));
106
107    return src;
108 }
109
110
```

Variable Inspector (Right Panel):

- in = { [583.5, 60.5, 0.0, 1] }
- time_ ≈ 3.8
- image = Texture 0x1031C
- imageSampler = Sampler
- shift = [≈0.282, ≈0.217]
- angle ≈ 0.655
- time = 3.801
- src = [0.341, ≈0.286, ≈0]
- hsv = [≈0.084, ≈0.322, C]
- col = [≈0.24, ≈0.852, ≈0]
- hsv = [≈0.059, ≈0.322, C]
- k ≈ 0.289
- hsv = [≈0.059, 0.518, 0.]
- src = [0.341, ≈0.226, ≈0]
- ret = [0.341, ≈0.226, ≈0]

Variable List (Left Panel):

- constant float &time_ [[buffer(0)]]
- texture2d<half> image [[texture(0)]]
- constexpr sampler imageSampler(ad...
- float2 shift = in.texCoord - 0.5;
- float angle = atan2(shift.y, shift.x);
- half time = half(time_);
- half4 src = image.sample(imageSam...
- half3 hsv = rgb2hsv(src.rgb);
- half3 col = time_color(half2(in.texCo...
- hsv.x = pos_fract(fma(half(angle), 0....
- hsv.x = pos_fract(fma(half(angle), 0....
- half k = smoothstep(0.0h, 1.0h, lengt...
- hsv.y = fma(hsv.y, 1.0h - k, k);
- src.rgb = hsv2rgb(max(hsv, 0.0h));
- return src;

Current Scope (Bottom):

```
> in = (ColorInOut) { position = [ 583.5, 60.5, 0.0, 1.0 ], texCoord = [ ≈0.782, ≈0.717 ] }
```

Shader: Debug

The screenshot displays the Xcode Shader Debugger for a Metal shader. The interface is divided into several sections:

- Left Sidebar:** Contains project information for "MetalToolsDemo" (Captured GPU Workload) and a list of shader instructions for the "fragment half4 fshRenderer00" function. The instruction "half k = smoothstep(0.0h, 1.0h, length...)" is currently selected.
- Main Editor:** Shows the Metal shader code. A red box highlights the line: `half4 src = image.sample(imageSampler, in.texCoord);`. Below this line, a visualization shows the texture being sampled (a grayscale image) and a white mask. The values for the texture are shown as `src = (half4) [0.3410644531, 0.2863769531, 0.2313232422, 1.0]`.
- Right Panel:** Displays the current state of variables. For the highlighted line, it shows `time = 3.801` and `src = [0.341, ≈0.286, ≈0.231, 1.0]`.
- Bottom Console:** Shows the current state of variables for the entire shader function, including `in`, `time_`, `image`, `imageSampler`, `shift`, `angle`, `time`, `src`, and `hsv`.

Shader: Debug

The screenshot displays the Xcode Shader Debugger for a Metal shader. The interface is divided into several sections:

- Left Panel:** Contains a 'Shader Debugger' section with a small texture preview and coordinates (X: 583.5, Y: 60.5, Z: 0). Below it is a list of shader instructions for the fragment shader 'fshRenderer00'. The instruction 'half k = smoothstep(0.0h, 1.0h, length(half2(shift)));' is highlighted in green.
- Center Panel:** Shows the source code of the shader. Line 102 is highlighted in green: `half k = smoothstep(0.0h, 1.0h, length(half2(shift)));`. A debugger overlay is visible over the code, showing a variable 'src' of type '(half4) [0.3410644531, 0.2863769531, 0.2313232422, 1.0]'. Below this, two square visualizations are shown: 'Values' (a grayscale texture) and 'Mask' (a white square). A legend below indicates 'Min Value [0.0, 0.0, 0.0, 0.0]' and 'Max Value [1.0, 1.0, 1.0, 1.0]'.
- Right Panel:** Displays the current values of variables: `time = 3.801`, `src = [0.341, ≈0.286, ≈0`, `hsv = [≈0.084, ≈0.322, (`, `col = [≈0.24, ≈0.852, ≈0`, `hsv = [≈0.059, ≈0.322, (`, and `k ≈ 0.289`.
- Bottom Panel (Red Box):** A console window showing the state of variables at the current execution point: `> in = (ColorInOut) { position = [583.5, 60.5, 0.0, 1.0], texCoord = [≈0.782, ≈0.717] }`, `> time_ = (float&) 3.7999970913`, `> image = (texture2d<half, metal::access::sample, vold>) n/a`, `> imageSampler = (sampler) n/a`, `> shift = (float2) [0.2819499969, 0.2166078687]`, `> angle = (float) 0.6550787091`, `> time = (half) 3.80078125`, `> src = (half4) [0.3410644531, 0.2863769531, 0.2313232422, 1.0]`, and `> hsv = (half3) [0.05859375, 0.3217773438, 0.3410644531]`.

Shader: Debug

The screenshot displays the MetalToolsDemo Shader Debugger interface. The main window shows the shader code for the `Fragment Function — fshRenderer00`. The code includes the following lines:

```
96 half time = half(time_);  
97 half4 src = image.sample(imageSampler, in.texCoord);  
  
half3 hsv = rgb2hsv(src.rgb);  
half3 col = time_color(half2(in.texCoord), time);  
hsv.x = pos_fract(fma(half(angle), 0.5h / M_PI_H, 0.5h) - fma(time, 0.25h,  
-rgb2hsv(col).x));  
half k = smoothstep(0.0h, 1.0h, length(half2(shift)));  
k = (half) 0.2893066406  
src.rgb = hsv2rgb(max(hsv, 0.0h));  
return src;
```

The debugger shows the current state of variables and textures:

- `time = 3.801`
- `src = [0.341, ≈0.286, ≈0]`
- `hsv = [≈0.084, ≈0.322, (]`
- `col = [≈0.24, ≈0.852, ≈0]`
- `hsv = [≈0.059, ≈0.322, (]`
- `k ≈ 0.289`

The `src` variable is expanded to show its values and mask:

```
src = (half4) [ 0.3410644531, 0.2863769531, 0.2313232422, 1.0 ]  
Values: [Image showing a noisy texture sample]  
Mask: [Image showing a white square mask]
```

The `Values` image shows a noisy texture sample, and the `Mask` image shows a white square. The `Min Value` is `[0.0, 0.0, 0.0, 0.0]` and the `Max Value` is `[1.0, 1.0, 1.0, 1.0]`.

The `half k = smoothstep(0.0h, 1.0h, length(half2(shift)));` line is highlighted in green, and the debugger shows the result: `k = (half) 0.2893066406`.

The `in` variable is expanded to show its position and texCoord:

```
> in = (ColorInOut) { position = [ 583.5, 60.5, 0.0, 1.0 ], texCoord = [ ≈0.782, ≈0.717 ] }  
> time_ = (float&) 3.7999970913  
> image = (texture2d<half, metal::access::sample, vold>) n/a  
> imageSampler = (sampler) n/a  
> shift = (float2) [ 0.2819499969, 0.2166078687 ]  
> angle = (float) 0.6550787091  
> time = (half) 3.80078125  
> src = (half4) [ 0.3410644531, 0.2863769531, 0.2313232422, 1.0 ]  
> hsv = (half3) [ 0.05859375, 0.3217773438, 0.3410644531 ]
```

The `Shader Debugger` panel on the left shows a histogram of the `src` variable's values. The histogram shows a distribution of values across the range `0 to 1`. The `Alpha` value is `0.756`. The `Combine RGB` checkbox is checked. The `Pre-Multiplied Alpha` checkbox is also checked.

The `Shader Debugger` panel also shows the `fragmen` variable and its components:

- `const`
- `textur`
- `const`
- `float2`
- `float a`
- `half ti`
- `half4 s`
- `half3 l`
- `half3 c`
- `hsv.x :`
- `hsv.x :`
- `half k`
- `hsv.y :`
- `src.rgb = hsv2rgb(max(hsv, 0.0h));`
- `return src;`

The `Shader Debugger` panel also shows the `fragmen` variable and its components:

- `const`
- `textur`
- `const`
- `float2`
- `float a`
- `half ti`
- `half4 s`
- `half3 l`
- `half3 c`
- `hsv.x :`
- `hsv.x :`
- `half k`
- `hsv.y :`
- `src.rgb = hsv2rgb(max(hsv, 0.0h));`
- `return src;`

Shader: Edit+Replay

The screenshot displays the Xcode Shader Debugger interface for a Metal shader. The main window is titled "MetalToolsDemo" and shows the "MetalToolsDemo.gputrace" file. The code editor displays the following shader code:

```
91 constexpr sampler imageSampler(address::clamp_to_zero, filter::nearest);
92
93 float2 shift = in.texCoord - 0.5;
94 float angle = atan2(shift.y, shift.x);
95
96 half time = half(time_);
97 half4 src = image.sample(imageSampler, in.texCoord);
98 half3 hsv = rgb2hsv(src.rgb);
99 half3 col = time_color(half2(in.texCoord), time);
100 hsv.x = pos_fract(fma(half(angle), 0.5h / M_PI_H, 0.5h) - fma(time, 0.25h,
    -rgb2hsv(col).x));
101
102 half k = smoothstep(0.0h, 1.0h, length(half2(shift)));
103 hsv.y = fma(hsv.y, 1.0h - k, k);
104 src.rgb = hsv2rgb(max(hsv, 0.0h));
105
106 return src.gggg;
107 }
108
109
110
111 kernel void krnRenderer00(device PositionData *positions [[buffer(0)]],
112                          device float4x4 *transforms [[buffer(1)]],
113                          uint gid [[thread_position_in_grid]])
114 {
115     PositionData pos = positions[gid];
116 }
```

The variable inspector on the right shows the following values:

- imageSampler = Sampler
- shift = [≈0.261, ≈0.21]
- angle ≈ 0.677
- time = 3.801
- src = [0.353, 0.298, ≈0.1]
- hsv = [0.098, 0.378, 0.3]
- col = [≈0.321, ≈0.852, ≈0.1]
- hsv = [≈0.047, 0.378, 0.3]
- k = 0.261
- hsv = [≈0.047, ≈0.541, 0.3]
- src = [0.353, ≈0.216, ≈0.1]
- ret = [0.353, ≈0.216, ≈0.1]

The console at the bottom shows the following output:

```
> in = (ColorInOut) { position = [ 570.5, 80.5, 0.0, 1.0 ], texCoord = [ ≈0.761, ≈0.71 ] }
> time_ = (float&) 3.7999970913
> image = (texture2d<half, metal::access::sample, vold>) n/a
> imageSampler = (sampler) n/a
> shift = (float2) [ 0.2607759237, 0.2095175982 ]
> angle = (float) 0.6768345833
> time = (half) 3.80078125
> src = (half4) [ 0.3530273438, 0.2163085938, 0.1622314453, 1.0 ]
> hsv = (half3) [ 0.0473632812, 0.5405273438, 0.3530273438 ]
```

The Shader Debugger interface includes a "Shader Debugger" section on the left with a "Summary" tab and a "Performance" tab. The "Memory" section shows 48.2 MB used. The "Shader Debugger" section shows the current shader function "fragment half4 fshRenderer00(ColorInOut)" and a list of variables and their values. The "Console" section at the bottom shows the output of the shader execution.

Shader: Edit+Replay

The screenshot displays the Xcode Shader Debugger interface for a Metal shader. The main window shows the source code for `Renderer00.metal` with the following content:

```
91 constexpr sampler imageSampler(address::clamp_to_zero, filter::nearest);
92
93 float2 shift = in.texCoord - 0.5;
94 float angle = atan2(shift.y, shift.x);
95
96 half time = half(time_);
97 half4 src = image.sample(imageSampler, in.texCoord);
98 half3 hsv = rgb2hsv(src.rgb);
99 half3 col = time_color(half2(in.texCoord), time);
100 hsv.x = pos_fract(fma(half(angle), 0.5h / M_PI_H, 0.5h) - fma(time, 0.25h,
    -rgb2hsv(col).x));
101
102 half k = smoothstep(0.0h, 1.0h, length(half2(shift)));
103 hsv.y = fma(hsv.y, 1.0h - k, k);
104 src.rgb = hsv2rgb(max(hsv, 0.0h));
105
106 return src.gggg;
107 }
108
109
110
111 kernel void krnRenderer00(device PositionData *positions [[buffer(0)]],
112                          device float4x4 *transforms [[buffer(1)]],
113                          uint gid [[thread_position_in_grid]])
114 {
115     PositionData pos = positions[gid];
116 }
```

The right-hand pane shows the state of variables at the current execution point (Line 106, Column 17):

- `imageSampler = Sampler`
- `shift = [≈0.261, ≈0.21]`
- `angle ≈ 0.677`
- `time = 3.801`
- `src = [0.353, 0.298, ≈0.1]`
- `hsv = [0.098, 0.378, 0.3]`
- `col = [≈0.321, ≈0.852, ≈0.1]`
- `hsv = [≈0.047, 0.378, 0.3]`
- `k = 0.261`
- `hsv = [≈0.047, ≈0.541, 0.3]`
- `src = [0.353, ≈0.216, ≈0.1]`
- `ret = [0.353, ≈0.216, ≈0.1]`

The bottom console shows the current execution context:

```
> in = (ColorInOut) { position = [ 570.5, 80.5, 0.0, 1.0 ], texCoord = [ ≈0.761, ≈0.71 ] }
> time_ = (float&) 3.7999970913
> image = (texture2d<half, metal::access::sample, vold>) n/a
> imageSampler = (sampler) n/a
> shift = (float2) [ 0.2607759237, 0.2095175982 ]
> angle = (float) 0.6768345833
> time = (half) 3.80078125
> src = (half4) [ 0.3530273438, 0.2163085938, 0.1622314453, 1.0 ]
> hsv = (half3) [ 0.0473632812, 0.5405273438, 0.3530273438 ]
```

The interface includes a left sidebar with a Shader Debugger panel showing a list of variables and their values, and a bottom console for execution logs. A red box highlights the 'Replay' button in the bottom toolbar.

All Resources

The screenshot displays the Xcode interface for MetalToolsDemo, showing a captured GPU workload. The left sidebar contains a tree view of the workload, with 'All Resources' selected. The main area shows a table of resources categorized into Textures, Buffers, Libraries, and Functions. The bottom panel shows a pipeline view for 'Render cats'.

Label	Type	Size	Details
Textures			
CAMetalLayer Display Dra...	Texture	1125 x 2436	BGRA8Unorm
CAMetalLayer Display Dra...	Texture	1125 x 2436	BGRA8Unorm
CAMetalLayer Display Dra...	Texture	1125 x 2436	BGRA8Unorm
Texture 0x10310e380	Texture	1936 x 1936	BGRA8Unorm
Buffers			
Buffer 0x10310e740	Buffer	18 KB	
Buffer 0x10310eb50	Buffer	4 KB	
Buffer 0x10310eca0	Buffer	16 KB	
Buffer 0x10310edf0	Buffer	6 KB	
Libraries			
Library 0x282452e80			
Library 0x28245ebc0			
Functions			
fshRenderer00	Fragment Function		Library 0x282452e80 [Precompiled]
krnRenderer00	Kernel Function		Library 0x28245ebc0 [Precompiled]
vshRenderer00	Vertex Function		Library 0x282452e80 [Precompiled]

Pipeline: Render cats (0x104035a00) vshRenderer00 - fshRenderer00

- RenderPipeline Performance 140.22 ms (100.0%) -0.37 ms
- Vertex Buffer 0 (MTLBuffer) 0x10310e740
- Vertex Buffer 1 (MTLBuffer) 0x10310eca0
- Vertex Buffer 2 (MTLBuffer) Bytes
- Vertex Buffer 3 (MTLBuffer) Bytes
- Vertex Buffer 4 (MTLBuffer) Bytes
- Vertex Buffer 5 (MTLBuffer) Bytes
- Vertex Buffer 6 (MTLBuffer) Bytes

Pipeline Statistics

The screenshot displays the MetalToolsDemo application's Pipeline Statistics window. The interface is divided into several sections:

- Left Panel:** A tree view showing the captured GPU workload. The 'Render' group is expanded, showing 44 render commands and 52 drawIndexedPrimitives calls. The 'Pipeline Statistics' option is highlighted.
- Top Panel:** Shows the current pipeline being analyzed: '52 [drawIndexedPrimitives:Triangle indexCount:1857 indexTyp...xBuffer:0x10310eb50 indexBufferOffset:0 instanceCount:256]'. The breadcrumb path is 'Pipeline Statistics'.
- Remarks Section (highlighted with a red box):**
 - High Float-to-Half Ratio:** A warning icon indicates a performance issue. The text states: 'A relatively high amount of floating-point operations use 32-bit float data types. Instead, use 16-bit half data types to increase performance. Ratio is 85.0% - Renderer00.metal:vshRenderer00 (Pipeline: Render cats)'.
- Vertex Shader Performance:**
 - Renderer00.metal: vshRenderer00 (Pipeline: Render cats):** Total Function Execution Time: 1.89 ms.
 - ALU Time, Memory Time, Control Flow Time, Synchronization Time, and Wait Time are shown as horizontal bars.
 - Total Number of Instructions:** 37.
 - ALU, Memory, Control Flow, Synchronization, and Maximum Theoretical Occupancy are also shown as horizontal bars.
- Fragment Shader Performance:**
 - Renderer00.metal: fshRenderer00 (Pipeline: Render cats):** Total Function Execution Time: 138.34 ms.
 - ALU Time and Memory Time are shown as horizontal bars.
- Bottom Panel:** A callstack view showing the pipeline execution flow: 'Pipeline: Render cats' (0x104035a00) vshRenderer00 - fshRenderer00. It lists performance metrics for the pipeline and its buffers: Vertex Buffer 0-6 (MTLBuffer) Bytes.

Pipeline Statistics

The screenshot displays the MetalToolsDemo interface. The left sidebar shows a tree view of the GPU workload, with 'Pipeline Statistics' selected. The main panel shows the 'Fragment Shader' 'fshRenderer00' with a total execution time of 138.34 ms. Below this, a table of 'Draw Calls' is highlighted with a red box, showing a total of 140.23 ms and a differential of -373.03 μs. The table includes columns for 'Total' and 'Differential' time, and a bar chart showing the distribution of draw call times.

Draw Calls	Total	Differential
23 [drawIndexedPrimitives:Triangle indexC...	30.77 ms	-161.63 μs
52 [drawIndexedPrimitives:Triangle indexC...	27.44 ms	-92.1 μs
81 [drawIndexedPrimitives:Triangle indexC...	27.37 ms	-40.61 μs
110 [drawIndexedPrimitives:Triangle index...	27.36 ms	+15.74 μs
139 [drawIndexedPrimitives:Triangle index...	27.3 ms	-94.45 μs
Total	140.23 ms	-373.03 μs

Below the table, the callstack shows the pipeline 'Render cats' with a performance of 140.22 ms (100.0%) and a differential of -0.37 ms. The callstack includes several vertex buffers (0-6) and their associated MTLBuffer objects.

Performance

The screenshot displays the MetalToolsDemo application interface for analyzing GPU performance. The left sidebar shows a tree view of the captured GPU workload, with the 'Performance' section selected. The main panel shows a summary of the performance analysis for a specific GPU command (52 [drawIndexedPrimitives:Triangle]).

Potential Hotspots/Bottlenecks
No potential bottlenecks detected

Counters

	Draw	Encoder
GPU (Command 52 - Render - Main Frame)		
GPU Time	27.44 ms	27.44 ms

Performance Limiters

ALU Limiter	N/A	75.63%
ALU Utilization	N/A	57.77%
F32 Utilization	N/A	26.59%
F16 Utilization	N/A	22.97%
Integer and Complex Utilization	N/A	38.73%
Integer and Conditional Utilization	N/A	11.78%
Texture Read Limiter	N/A	93.38%
Texture Read Utilization	N/A	7.92%
Texture Write Limiter	N/A	0.56%
Texture Write Utilization	N/A	0.56%
Buffer Read Limiter	N/A	0.13%
Buffer Read Utilization	N/A	0.13%
Buffer Write Limiter	N/A	0.00%
Buffer Write Utilization	N/A	0.00%
Main Memory Access Limiter	N/A	62.98%
ThreadGroup/ImageBlock Read Limiter	N/A	4.63%
ThreadGroup/ImageBlock Read Utilization	N/A	3.96%
ThreadGroup/ImageBlock Write Limiter	N/A	8.42%

Callstack

- 56 [presentDrawable:0x281f24100]
- Main Frame
- Main Frame
- Main Frame

Performance

The screenshot displays the MetalToolsDemo application's performance analysis interface. The left sidebar shows a tree view of the captured GPU workload, with the 'Performance' section selected. The main panel shows a detailed performance report for a specific GPU command (52).

Potential Hotspots/Bottlenecks
No potential bottlenecks detected

Counters	Draw	Encoder
GPU (Command 52 - Render - Main Frame)		
GPU Time	27.44 ms	27.44 ms
Performance Limiters		
ALU Limiter	N/A	75.63%
ALU Utilization	N/A	57.77%
F32 Utilization	N/A	26.59%
F16 Utilization	N/A	22.97%
Integer and Complex Utilization	N/A	38.73%
Integer and Conditional Utilization	N/A	11.78%
Texture Read Limiter	N/A	93.38%
Texture Read Utilization	N/A	7.92%
Texture Write Limiter	N/A	0.56%
Texture Write Utilization	N/A	0.56%
Buffer Read Limiter	N/A	0.13%
Buffer Read Utilization	N/A	0.13%
Buffer Write Limiter	N/A	0.00%
Buffer Write Utilization	N/A	0.00%
Main Memory Access Limiter	N/A	62.98%
ThreadGroup/ImageBlock Read Limiter	N/A	4.63%
ThreadGroup/ImageBlock Read Utilization	N/A	3.96%
ThreadGroup/ImageBlock Write Limiter	N/A	8.42%

Callstack

- 56 [presentDrawable:0x281f24100]
- Main Frame
- Main Frame
- Main Frame

Performance Report Details:

- "Pipeline: Render cats" (0x104035a00) vshRenderer00 - fshRenderer00
- RenderPipeline Performance 140.22 ms (100.0%) -0.37 ms
- Vertex Buffer 0 (MTLBuffer) 0x10310e740
- Vertex Buffer 1 (MTLBuffer) 0x10310eca0
- Vertex Buffer 2 (MTLBuffer) Bytes
- Vertex Buffer 3 (MTLBuffer) Bytes
- Vertex Buffer 4 (MTLBuffer) Bytes
- Vertex Buffer 5 (MTLBuffer) Bytes
- Vertex Buffer 6 (MTLBuffer) Bytes

Performance

The screenshot displays the MetalToolsDemo application interface. The left sidebar shows a tree view of the captured GPU workload, with the 'Performance' section selected. The main panel shows the following information:

- Potential Hotspots/Bottlenecks:** No potential bottlenecks detected.
- Counters:** A table showing performance metrics for 'GPU (Command 52 - Render - Main Frame)'. The table has columns for 'Draw' and 'Encoder'.
- Performance Limiters:** A list of various GPU limiters and their utilization percentages.
- Callstack:** A list of pipeline and buffer operations.

	Draw	Encoder
GPU (Command 52 - Render - Main Frame)		
GPU Time	27.44 ms	27.44 ms
Performance Limiters		
ALU Limiter	N/A	75.63%
ALU Utilization	N/A	57.77%
F32 Utilization	N/A	26.59%
F16 Utilization	N/A	22.97%
Integer and Complex Utilization	N/A	38.73%
Integer and Conditional Utilization	N/A	11.78%
Texture Read Limiter	N/A	93.38%
Texture Read Utilization	N/A	7.92%
Texture Write Limiter	N/A	0.56%
Texture Write Utilization	N/A	0.56%
Buffer Read Limiter	N/A	0.13%
Buffer Read Utilization	N/A	0.13%
Buffer Write Limiter	N/A	0.00%
Buffer Write Utilization	N/A	0.00%
Main Memory Access Limiter	N/A	62.98%
ThreadGroup/ImageBlock Read Limiter	N/A	4.63%
ThreadGroup/ImageBlock Read Utilization	N/A	3.96%
ThreadGroup/ImageBlock Write Limiter	N/A	8.42%

Performance

The screenshot displays the MetalToolsDemo application's performance analysis interface. The left sidebar shows a tree view of the captured GPU workload, with the 'Performance' section selected. The main panel shows the following information:

- Potential Hotspots/Bottlenecks:** No potential bottlenecks detected.
- Counters:** A table comparing 'Draw' and 'Encoder' performance.
- GPU (Command 52 - Render - Main Frame):** GPU Time is 27.44 ms for both Draw and Encoder.
- Performance Limiters:** A list of various GPU limiters and their utilization percentages.

Counter	Draw	Encoder
GPU Time	27.44 ms	27.44 ms
ALU Limiter	N/A	75.63%
ALU Utilization	N/A	57.77%
F32 Utilization	N/A	26.59%
F16 Utilization	N/A	22.97%
Integer and Complex Utilization	N/A	38.73%
Integer and Conditional Utilization	N/A	11.78%
Texture Read Limiter	N/A	93.38%
Texture Read Utilization	N/A	7.92%
Texture Write Limiter	N/A	0.56%
Texture Write Utilization	N/A	0.56%
Buffer Read Limiter	N/A	0.13%
Buffer Read Utilization	N/A	0.13%
Buffer Write Limiter	N/A	0.00%
Buffer Write Utilization	N/A	0.00%
Main Memory Access Limiter	N/A	62.98%
ThreadGroup/ImageBlock Read Limiter	N/A	4.63%
ThreadGroup/ImageBlock Read Utilization	N/A	3.96%
ThreadGroup/ImageBlock Write Limiter	N/A	8.42%

The bottom panel shows a detailed view of the pipeline performance, including the pipeline name, total time, and a list of buffers and their utilization.

Filter

Callstack

The screenshot displays the Xcode IDE interface for a project named "MetalToolsDemo". The interface is divided into several panels:

- Left Panel (Callstack):** Shows a callstack for the "Render" phase, which took 27.44 ms. The stack includes:
 - m 0 -[Renderer00 renderIn:]
 - m 1 -[Renderer00 drawInM...]
 - 2 -[MTKView draw]
 - 3 -[MTKViewDisplayLinkTar...]
 - 4 CA::Display::DisplayLink::...
 - 5 display_timer_callback(__...)
 - 6 __CFMachPortPerform
 - 7 __CFRUNLOOP_IS_CALLI...
 - 8 __CFRunLoopDoSource1
 - 9 __CFRunLoopRun
 - 10 CFRunLoopRunSpecific
 - 11 GSEventRunModal
 - 12 -[UIApplication _run]
 - 13 UIApplicationMain
 - m 14 main
 - 15 start
- Top Panel (Code Editor):** Displays the source code for "Renderer00.m". The current function is "-renderIn:". The code includes Metal shader calls:

```
212 [encoder setVertexBytes:&ratio length:sizeof(float) atIndex:2];
213 [encoder setFragmentBytes:&_time length:sizeof(float) atIndex:0];
214 [encoder setFragmentTexture:_texture atIndex:0];
215 [encoder drawIndexedPrimitives:MTLPrimitiveTypeTriangle
216         indexCount:_mesh.indices_count
217         indexType:MTLIndexTypeUInt16
218         indexBuffer:_bufferIndices
219         indexBufferOffset:0
220         instanceCount:_instances];
221
222 [encoder endEncoding];
223 }
224 }
```
- Bottom Panel (GPU Workload):** Shows a list of GPU workload items:
 - "Pipeline: Render cats" (0x104035a00) vshRenderer00 - fshRenderer00
 - RenderPipeline Performance 140.22 ms (100.0%) -0.37 ms
 - Vertex Buffer 0 (MTLBuffer) 0x10310e740
 - Vertex Buffer 1 (MTLBuffer) 0x10310eca0
 - Vertex Buffer 2 (MTLBuffer) Bytes
 - Vertex Buffer 3 (MTLBuffer) Bytes
 - Vertex Buffer 4 (MTLBuffer) Bytes
 - Vertex Buffer 5 (MTLBuffer) Bytes
 - Vertex Buffer 6 (MTLBuffer) Bytes

METAL API

GPU Frame Capture

Simple Capture

```
+ (void)startCaptureIn:(id)target {
    MTLCaptureManager *captureManager = MTLCaptureManager.sharedCaptureManager;
    MTLCaptureDescriptor *captureDescriptor = [MTLCaptureDescriptor new];
    captureDescriptor.captureObject = target;
    [captureManager startCaptureWithDescriptor:captureDescriptor error:nil];
}

+ (void)endCapture {
    MTLCaptureManager *captureManager = MTLCaptureManager.sharedCaptureManager;
    [captureManager stopCapture];
}
```

Scope Capture

```
id<MtlCaptureScope> customScope =  
    [[MtlCaptureManager sharedCaptureManager] newCaptureScopeWithDevice:_device];  
customScope.label = @"Custom scope";  
  
// ...  
  
[Renderer00 startCaptureIn:customScope];  
  
// ...  
  
[customScope beginScope];  
// captured command buffers  
[customScope endScope];  
  
// ...  
  
[Renderer00 endCapture];
```

TOOLS DON'T WORK

Manual Solutions

PERFORMANCE

FPS (in code)

FPS (in code)

- Check frame time every frame
- Store time history
- Compute average time with history

FPS: Preparing

```
// Previous timestamp.  
CTimeInterval _prevTime;  
// A callback for updating some FPS representation.  
FPSUpdater _fpsUpdater;  
// A history of time per frames.  
double _prevFPS[MaxFPSHistory];  
// A size of the history.  
NSUInteger _prevFPSNum;
```

FPS: Computing

```
- (void)updateFPS {
    static dispatch_once_t once;

    dispatch_once(&once, ^{
        _prevTime = CACurrentMediaTime() - 1.0 / 60.0;
    });

    CTimeInterval curTime = CACurrentMediaTime();
    double fps = 1.0 / (curTime - _prevTime);

    double fpsSum = fps;
    int fpsCount = 1;
    for (NSUInteger i = 0; i < _prevFPSNum; ++i) {
        fpsSum += _prevFPS[i];
        ++fpsCount;
    }
    for (int i = (int)_prevFPSNum - 1; i > 0; --i) {
        _prevFPS[i] = _prevFPS[i - 1];
    }
    _prevFPS[0] = fps;
    _prevFPSNum = MIN(_prevFPSNum + 1, MaxFPSHistory);
    _fpsUpdater(fpsSum / fpsCount);
    _prevTime = curTime;
}
```

PERFORMANCE

Command Buffer Profiling

```
- (void) foo {
    // ...
    CFAbsoluteTime startTime = CACurrentMediaTime();
    NSUInteger iterations = 100;
    for (NSUInteger i = 0; i < iterations - 1; ++i) {
        id <MTLCommandBuffer> commandBuffer = [_commandQueue commandBuffer];
        [self renderIn:commandBuffer];
        [commandBuffer commit];
    }
    id <MTLCommandBuffer> commandBuffer = [_commandQueue commandBuffer];
    [self renderIn:commandBuffer];
    [commandBuffer commit];
    [commandBuffer waitUntilCompleted];

    CFAbsoluteTime curTime = CACurrentMediaTime();
    double fps = (double)iterations / (curTime - startTime);
    // ...

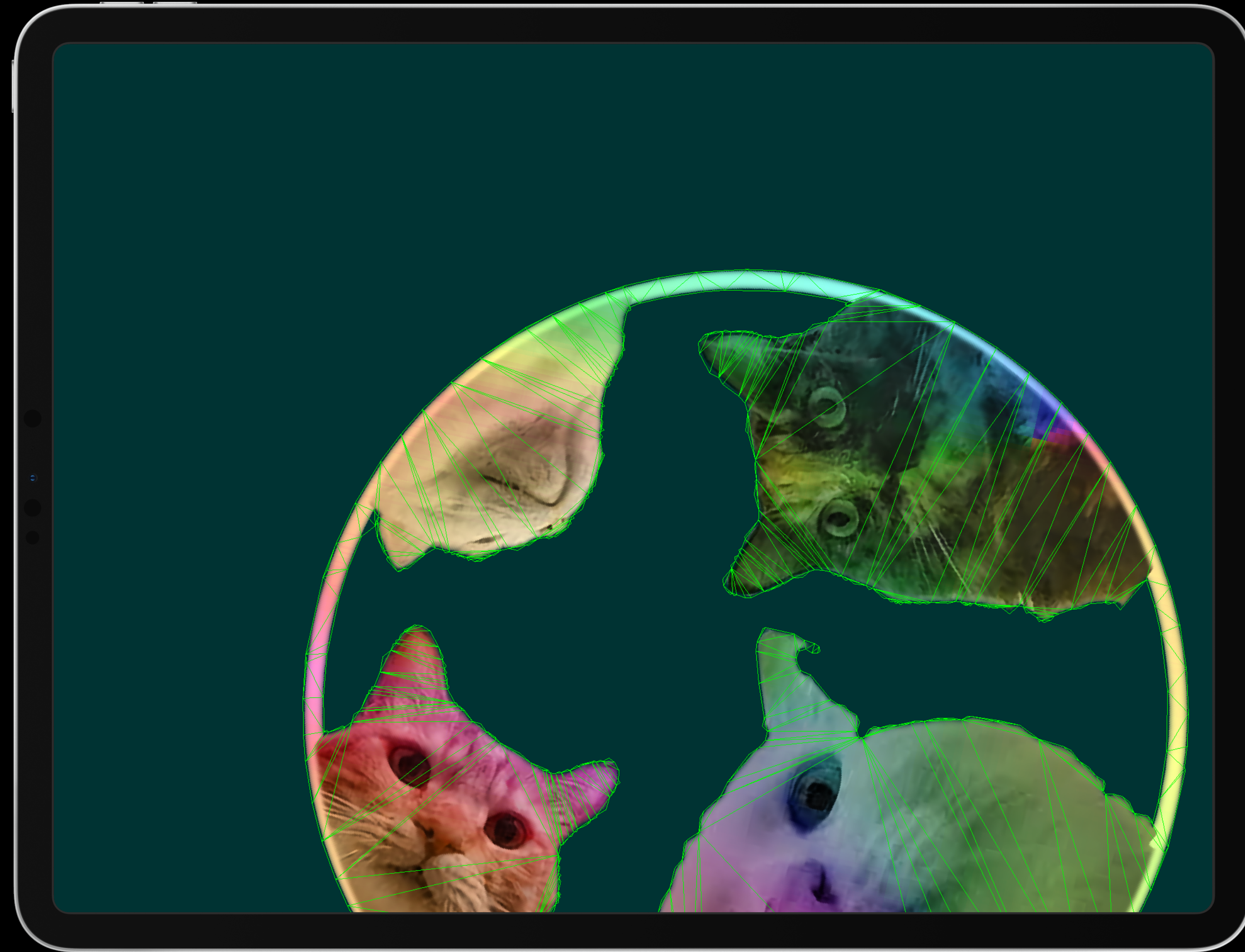
    (void) fps;
}
```

DEBUG SHADERS

Strokes & Fills

GEOMETRY

Wireframe



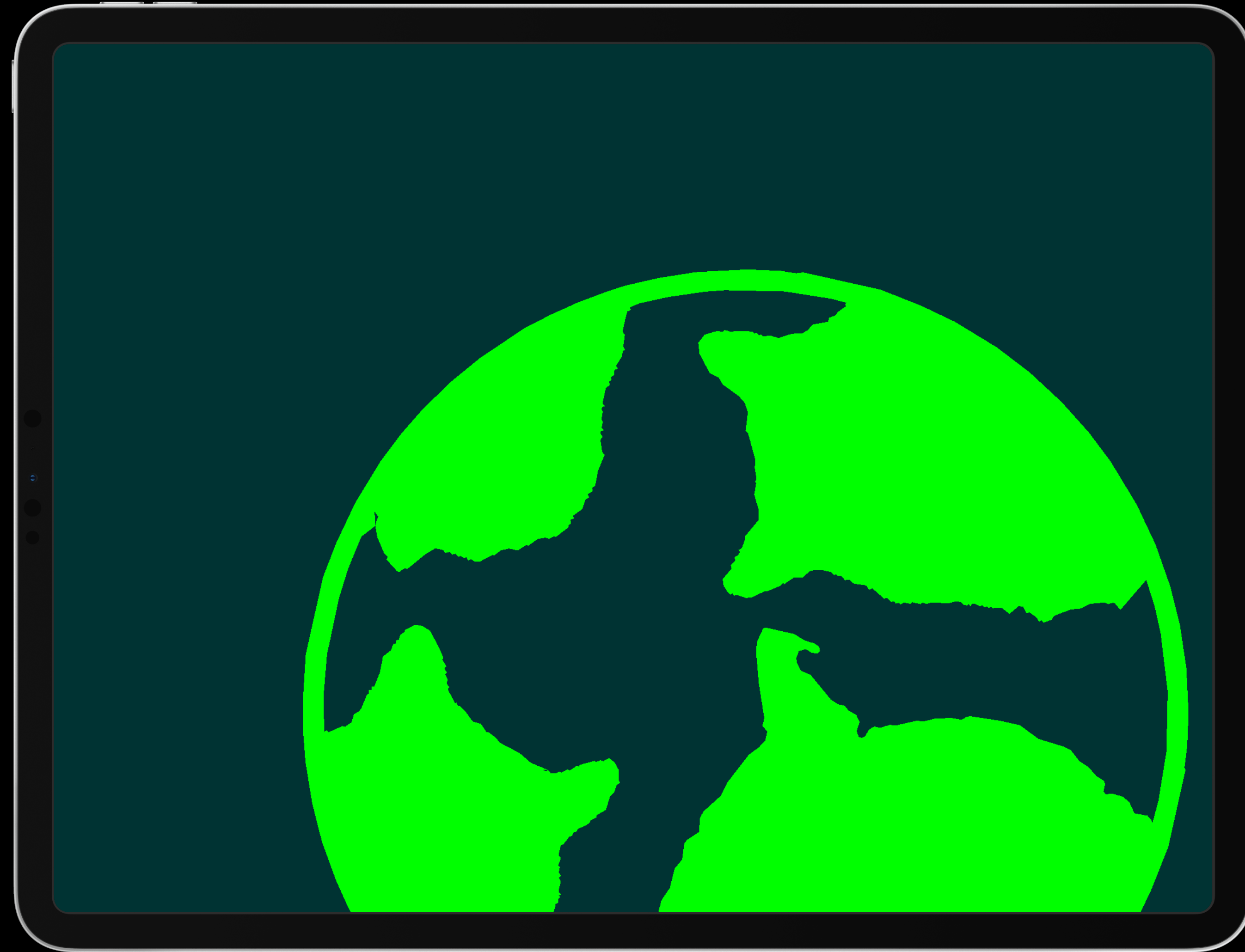
GEOMETRY

Wireframe

```
[encoder setTriangleFillMode:MTLTriangleFillModeLines];
```

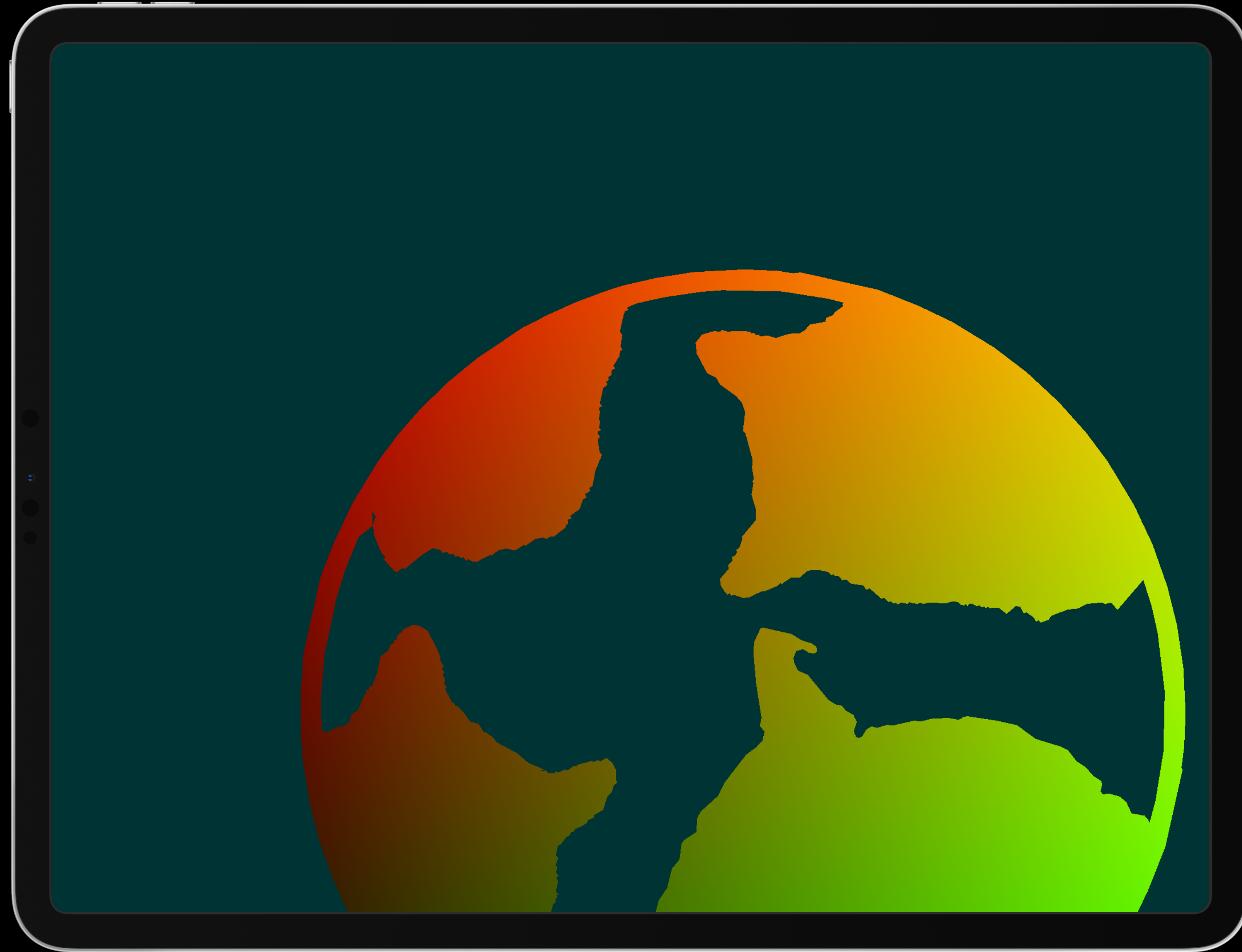

PROCESSED AREA

Color Fill



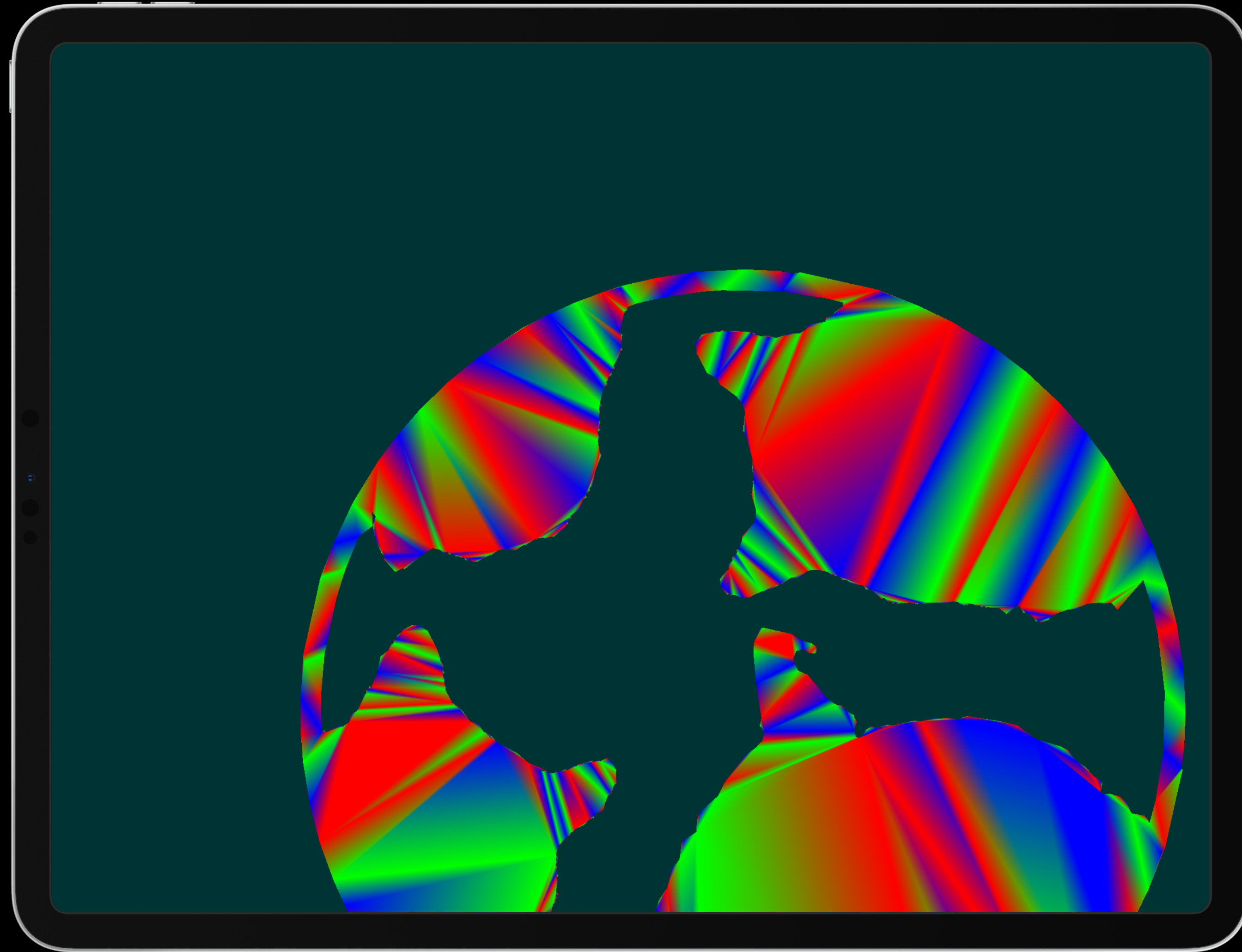
COORDINATES

RG-Fill



SPECIAL CASES

Special Fills



OFF-SCREEN DEBUGGING

Preview MTLTexture in XCode



> **com.apple.uikit.eventfetch-thread (6)**
> **Thread 7** Queue: com.a...s (concurrent)

_time = (float) 0.016666675
> **_texture = (CaptureMTLTexture *) 0x2830a0340**

Running MetalToolsDemo on George's iPhone

metalUtils.m | h Renderer00.h | h MetalUtils.h

n > **M** -drawInMTKView:

0 matches + Aa Contains < > Done

```
erWithDescriptor:renderPassDescriptor];

.pipelineRender];
vertices offset:0 atIndex:0];
transforms offset:0 atIndex:1];
length:sizeof(float) atIndex:2];
length:sizeof(float) atIndex:0];
texture atIndex:0];
TLPrimitiveTypeTriangle
mesh.indices_count
TLIndexTypeUInt16
bufferIndices

instances];
```

Thread 1: breakpoint 1.1 (1)

MTLCaptureManager.sharedCaptureManager:

o > **Thread 1** > 0 -[Renderer00 renderIn:] 22 characters

2021-10-28 08:59:29.509055+1100
MetalToolsDemo[4363:940692] Metal GPU Frame
Capture Enabled

2021-10-28 08:59:29.510329+1100
MetalToolsDemo[4363:940692] Metal API Validation
Enabled
(lldb)

All Output > Filter

OFF-SCREEN DEBUGGING

Download Buffer/Texture

```
+ (vImage_Buffer)bufferOf:(id<MTLTexture>)texture {
    vImage_Buffer buffer = (vImage_Buffer){
        .data = calloc(texture.width * texture.height, 4),
        .width = texture.width,
        .height = texture.height,
        .rowBytes = texture.width * 4
    };

    [texture getBytes:buffer.data
        bytesPerRow:buffer.rowBytes
        fromRegion:MTLRegionMake2D(0, 0, texture.width, texture.height) mipmapLevel:0];

    return buffer;
}
```



buffer

Open With Preview

ToolsDemo on George's iPhone

tils.m | MetalUtils.h

Texture.height)

Thread 1: breakpoint 1.1 (1)

er00 bufferOf:] Line: 232 Col: 26

8 09:12:52.822537+1100
ToolsDemo[4417:946123] Metal GPU Frame
re Enabled
8 09:12:52.823249+1100
ToolsDemo[4417:946123] Metal API Validation
ed

Filter

THANK YOU

Questions?



Contacts



george.ostrobrod@gmail.com

jefferson@savage.si



[@GOstrobrod](https://twitter.com/GOstrobrod)

<https://savage.si/>



<https://wdf-dev.gitlab.io>

[@jeffersonstjohn](https://www.linkedin.com/company/jeffersonstjohn)



[@georgeostrobrod](https://www.linkedin.com/company/georgeostrobrod)