GPU TECHNOLOGY CONFERENCE

DATA VISUALIZATION OF THE GRAPHICS PIPELINE:

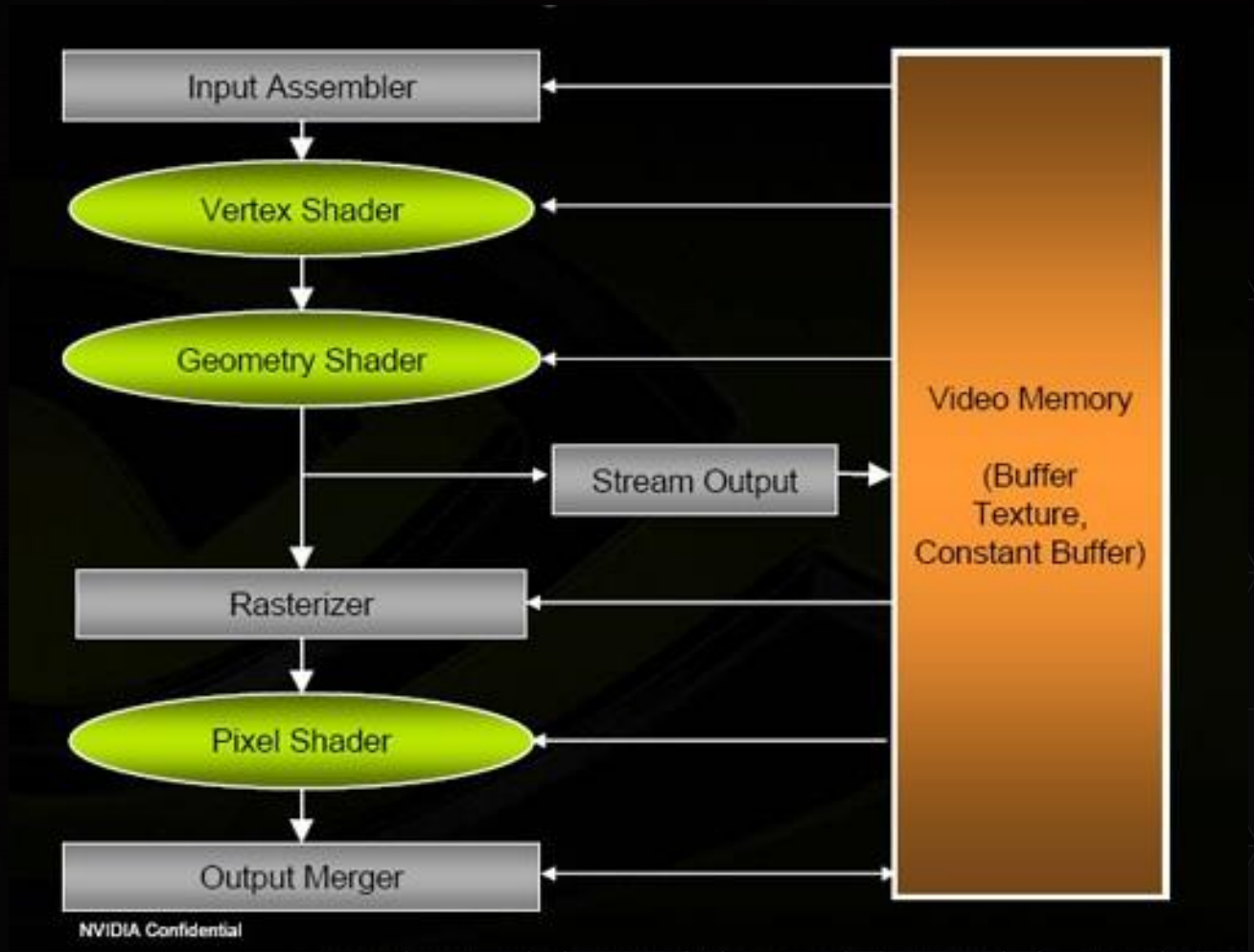
TRACKING STATE WITH THE STATEVIEWER

RAMA HOETZLEIN, DEVELOPER TECHNOLOGY, NVIDIA

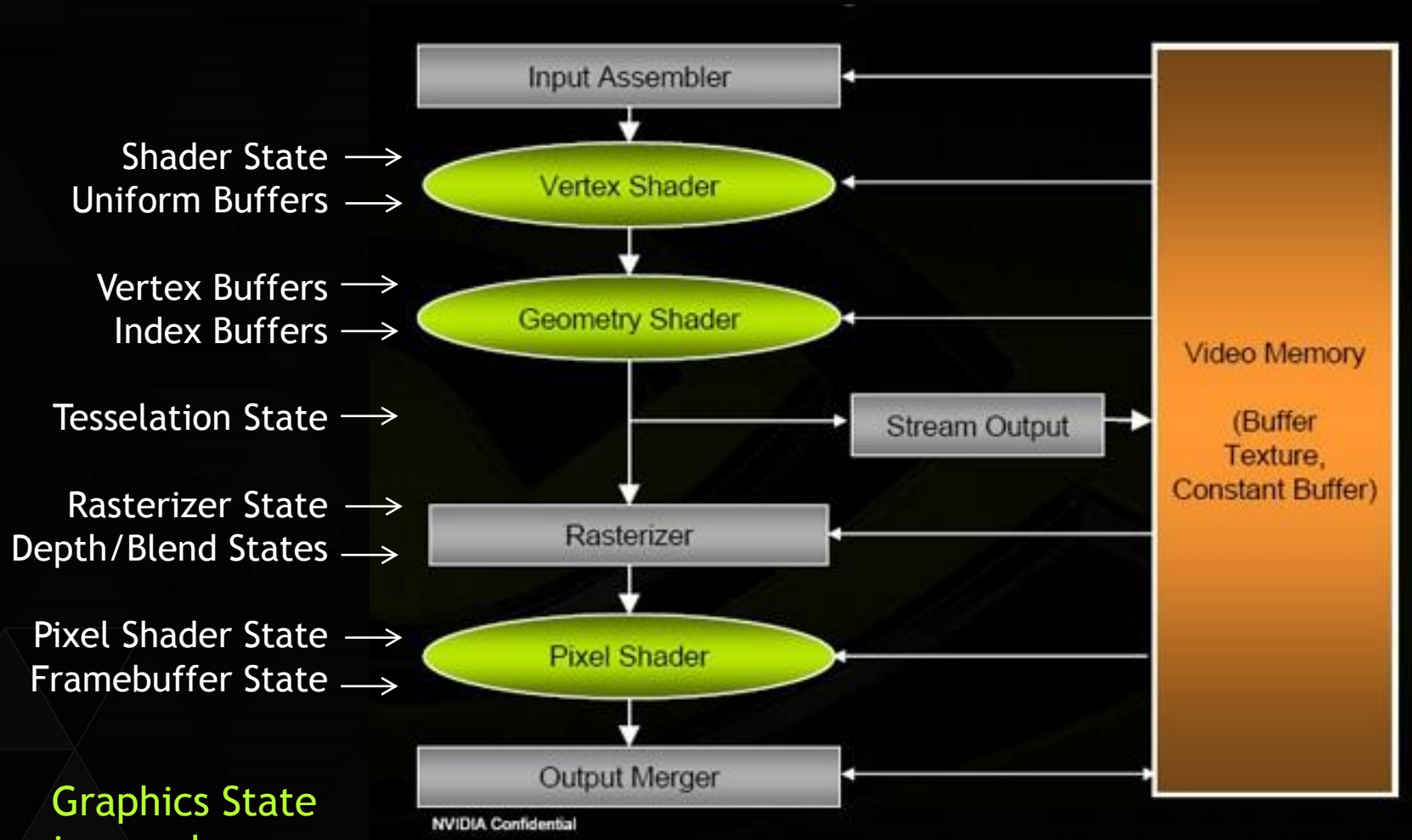
"Data Visualizations assist humans with data analysis by representing information visually.. These mechanisms rely on human perception to help understand data."

Human Factors in Visualization Research, Melanie Tory & Torsten Moller
IEEE Transactions on Visualization and Computer Graphics, Vol 10, No 1, Jan 2004.

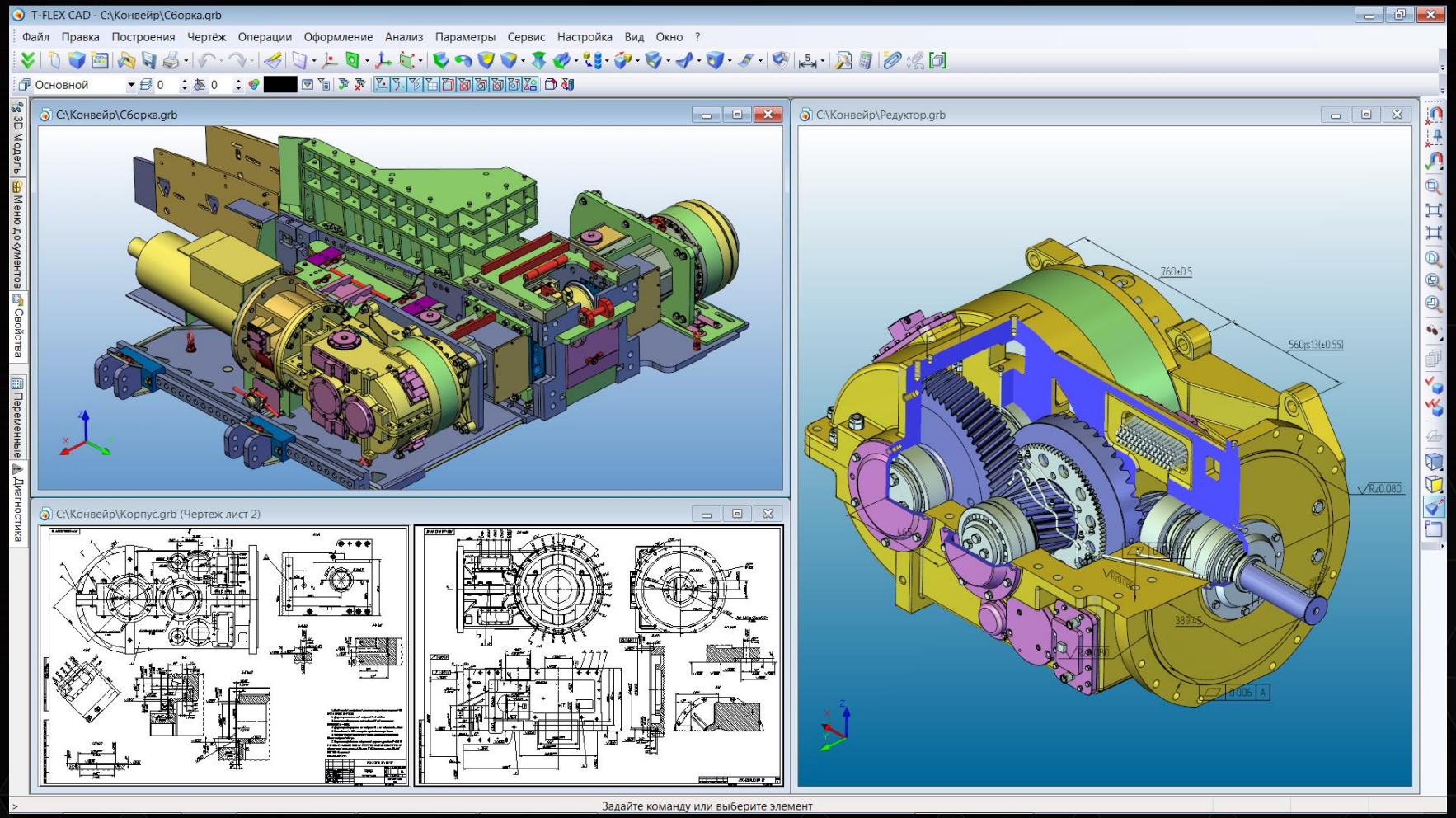
GRAPHICS PIPELINE



GRAPHICS PIPELINE



Graphics State is complex.



T-FLEX CAD, 2012. Image from wikimedia commons.

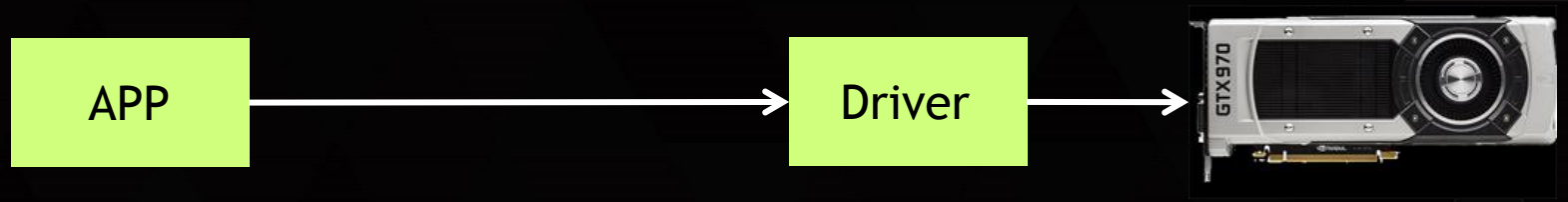
CAD/Workstation Applications solve *complex*, real world problems

CPU Bound: Traversal of CPU scene graph, or drawing setup, outweighs GPU rendering.

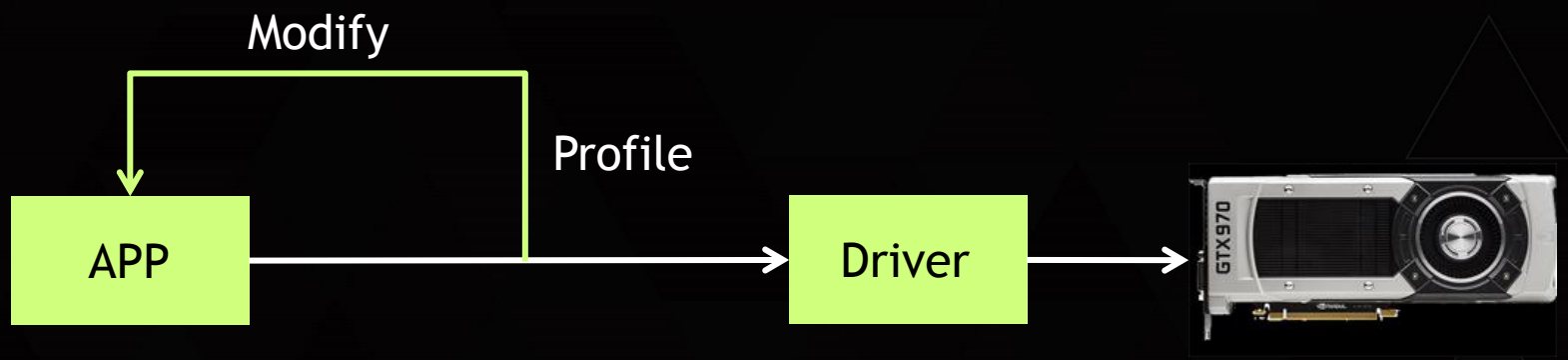
Many CAD/Professional Workstation applications are CPU Bound.

These are ideal candidates for next-gen APIs.

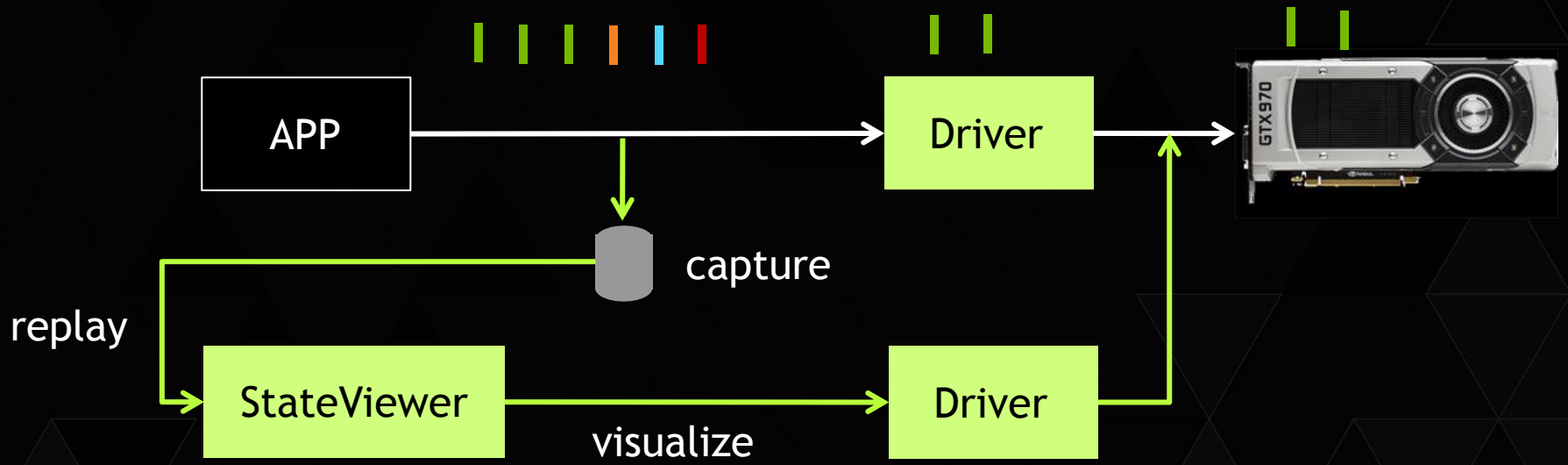
PROFILING PRACTICE



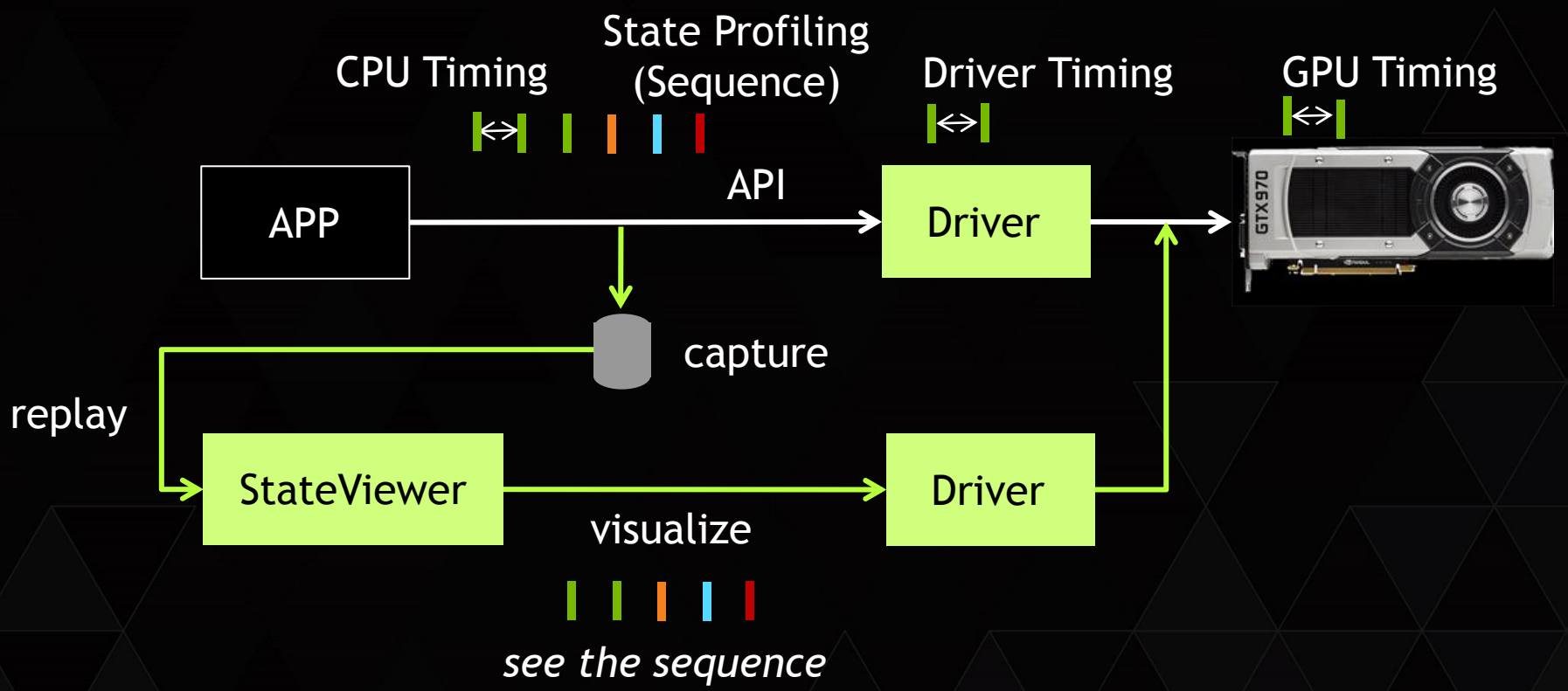
PROFILING PRACTICE



PROFILING PRACTICE



PROFILING PRACTICE



GOALS

API Tracing

Identify named buffers at the time of API calls.

Value Tracing

Identify which state arbitrary buffers belong to.
Identify *values* inside named buffers.
Identify *values* transferred by memcpy/map

Value-Delta Tracing

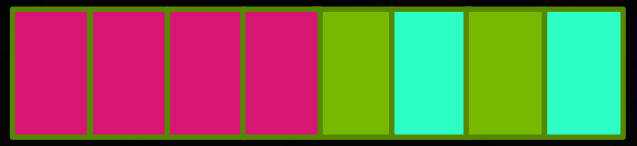
Identifies changes in values *in the same buffer.*
Identifies when switching buffers with *same value*.

We want tools that identify all of the above.

EXAMPLE

VISUALIZATION DESIGN

Colored rectangles map state *values*.



Value of this buffer stays the same for the first 4 draws. Then, value flip-flops between 2 values. Colors are random.. It's about seeing *patterns*.

Colored *flags* map state value *changes*.

Create/write – app is allocating a new buffer, or rewriting it.

Switch – app is switching to another buffer.

Reuse – app is reusing buffer from last draw, no switch.

ALGORITHM

PASS #1

Replay all API calls to determine state bins.
Example:

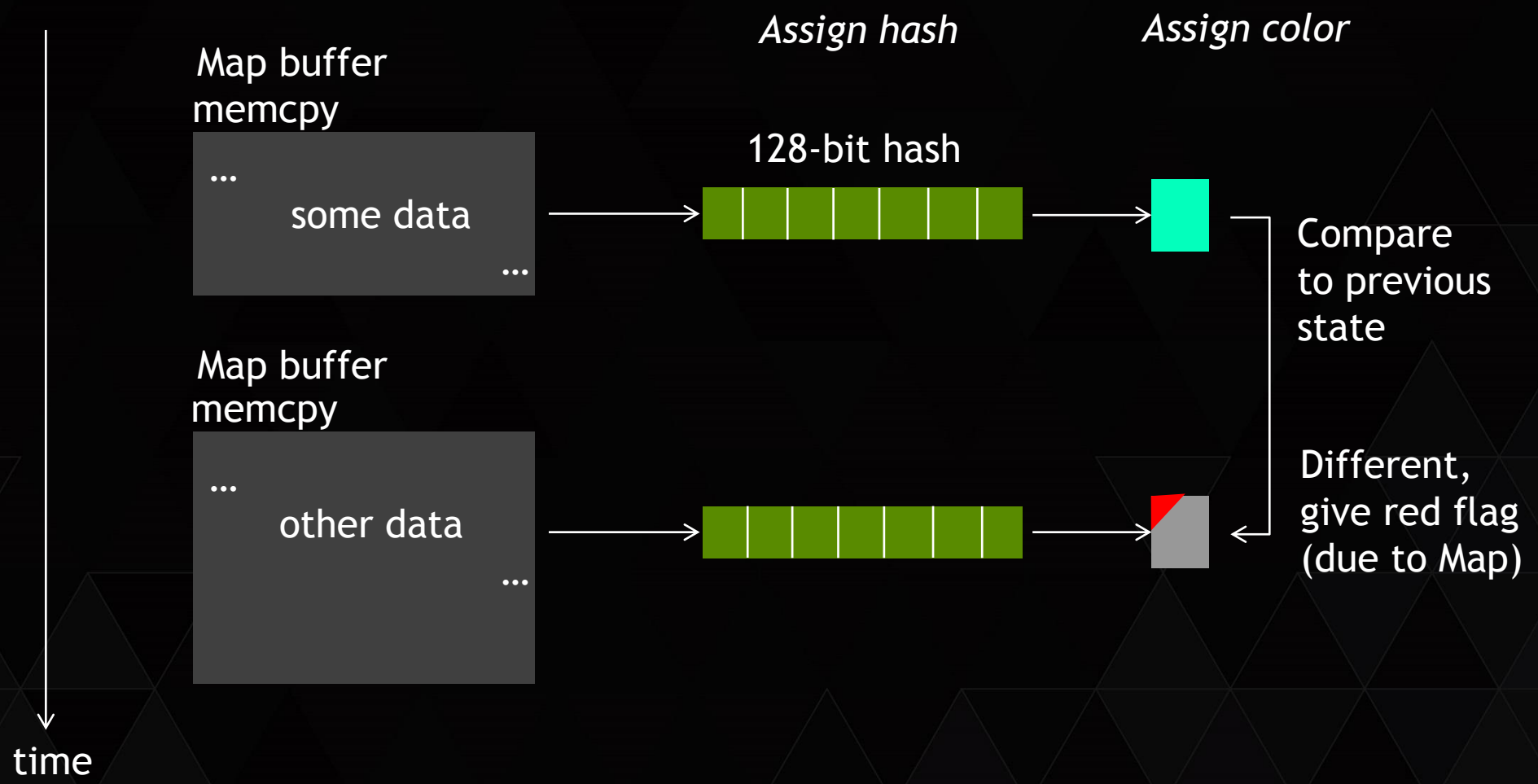
DXCreateBuffer	How will it be used? Unknown until later.
IASetVertexBuffer	*Now*, we know it is a VBO.

PASS #2

Replay all API calls *again*, and record both input and output values.
Compress all values using a 128-bit **hash.**
Assign colors and track deltas based on the **hash**.

Every API call specifies a unique state bin, named object, and value.

ALGORITHM



WHAT STATES TO TRACK?

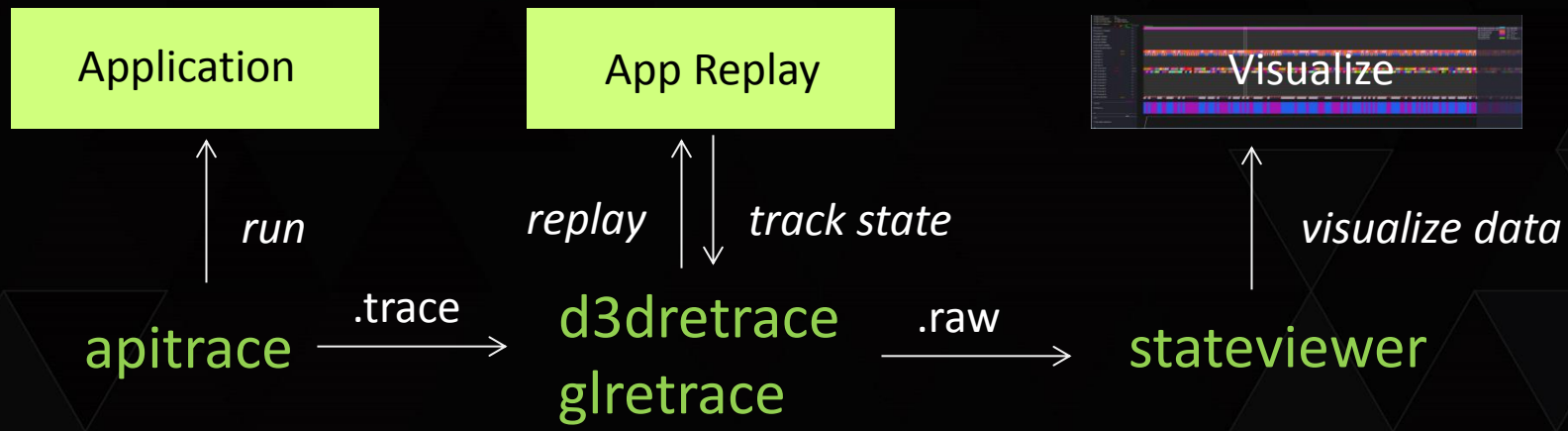
0 Shader
1 Render Target
2 Viewport
3 Rasterizer State
4 Depth State
5 Blend State
6 Sampler State
7 Input
8 Texture
9 Vertex Buffer (IA Slot 0)
10 Vertex Buffer (IA Slot 1)
11 Vertex Buffer (IA Slot 2)
12 Vertex Buffer (IA Slot 3)
13 Vertex Buffer (IA Slot 4)
14 VS Const Buffer 0
15 VS Const Buffer 1
16 VS Const Buffer 2
19 PS Const Buffer 0
20 PS Const Buffer 1
21 PS Const Buffer 2
24 Index Buffer

STATEVIEWER

Contributed to *apitrace*, open source.

A free tool for deep state tracking /w value deltas.

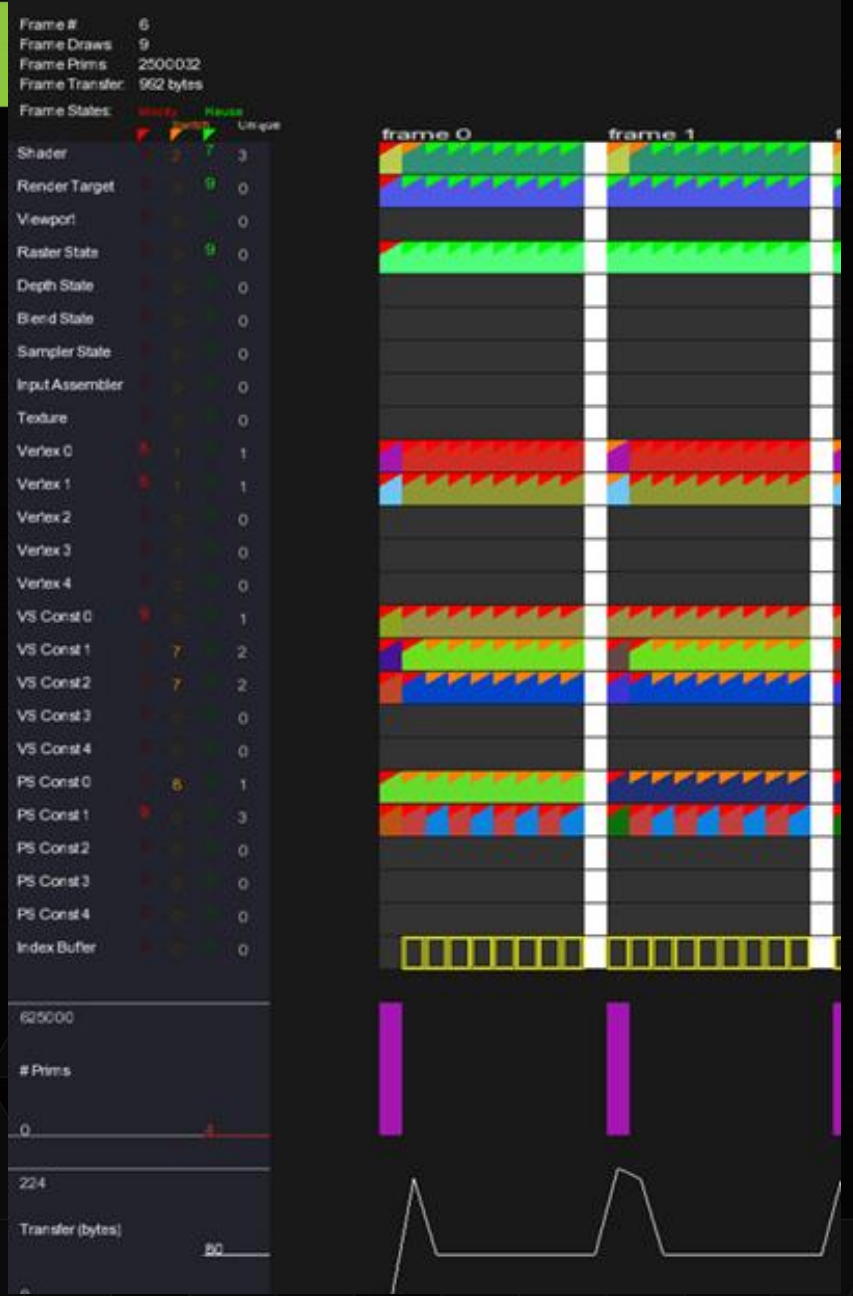
Simple trace and view workflow.



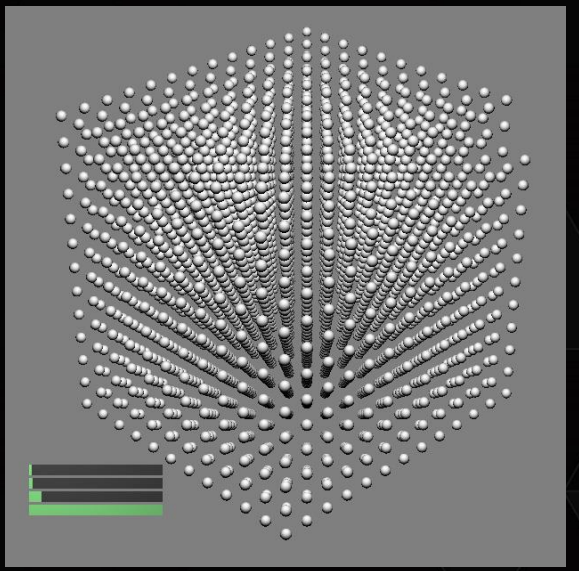
** Now availabe on github! **

GPU

STATEVIEWER: SIMPLE EXAMPLE



StateViewer output

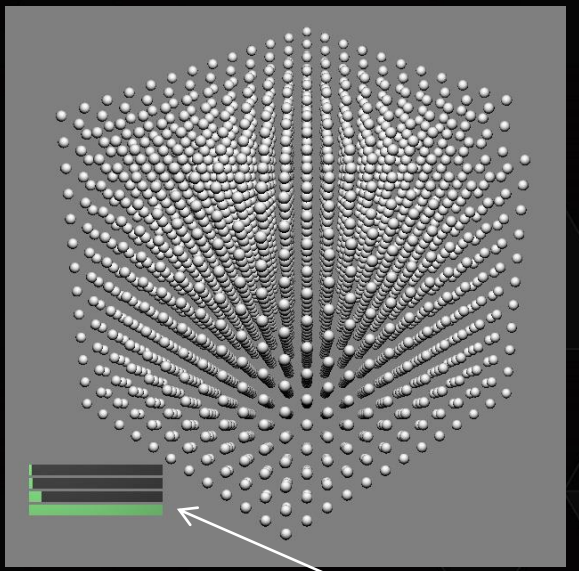


Example:

Draw instanced spheres with some GUI controls.

GPU

STATEVIEWER: SIMPLE EXAMPLE



Observe:

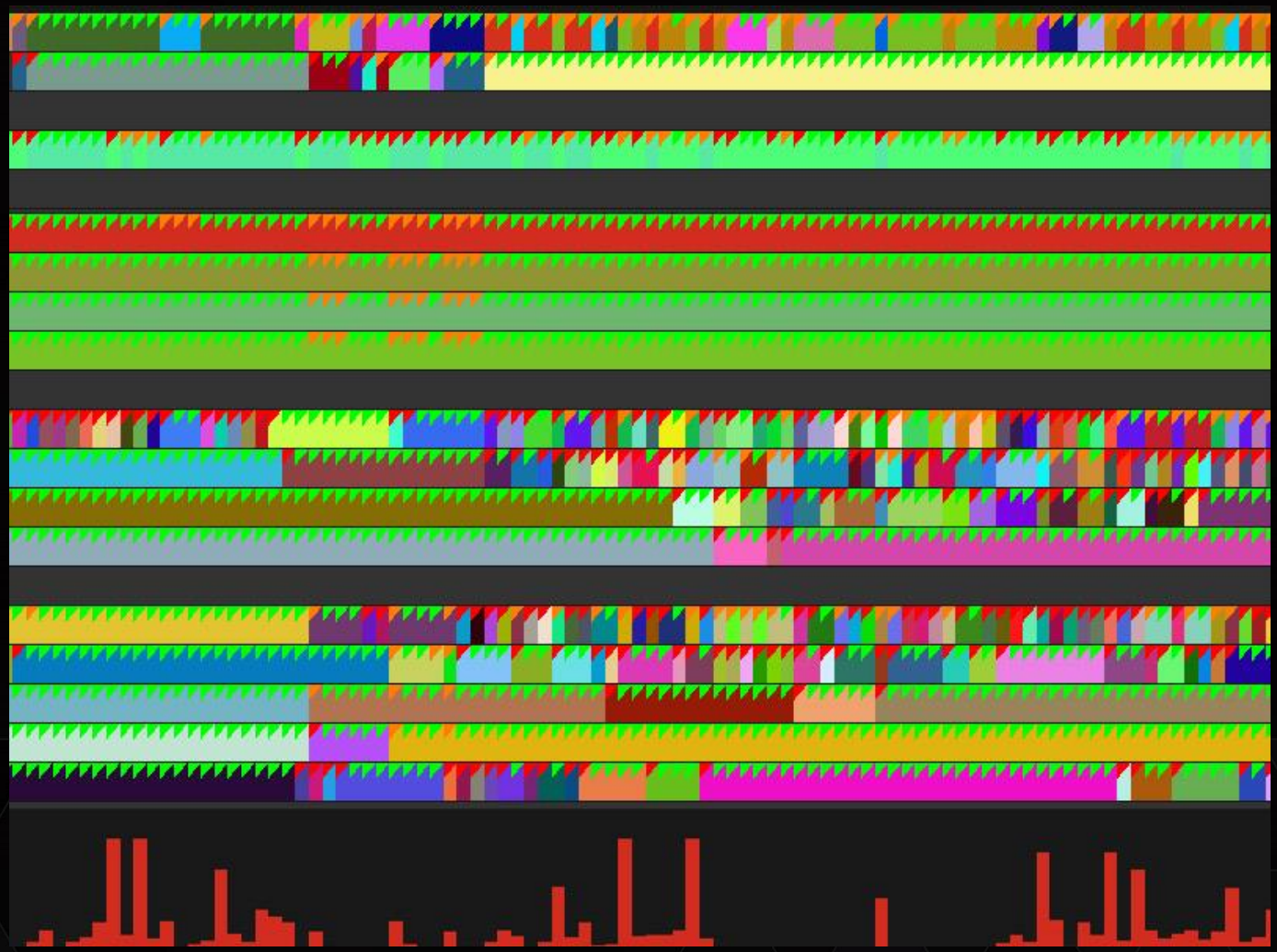
Frames separate by white bars.

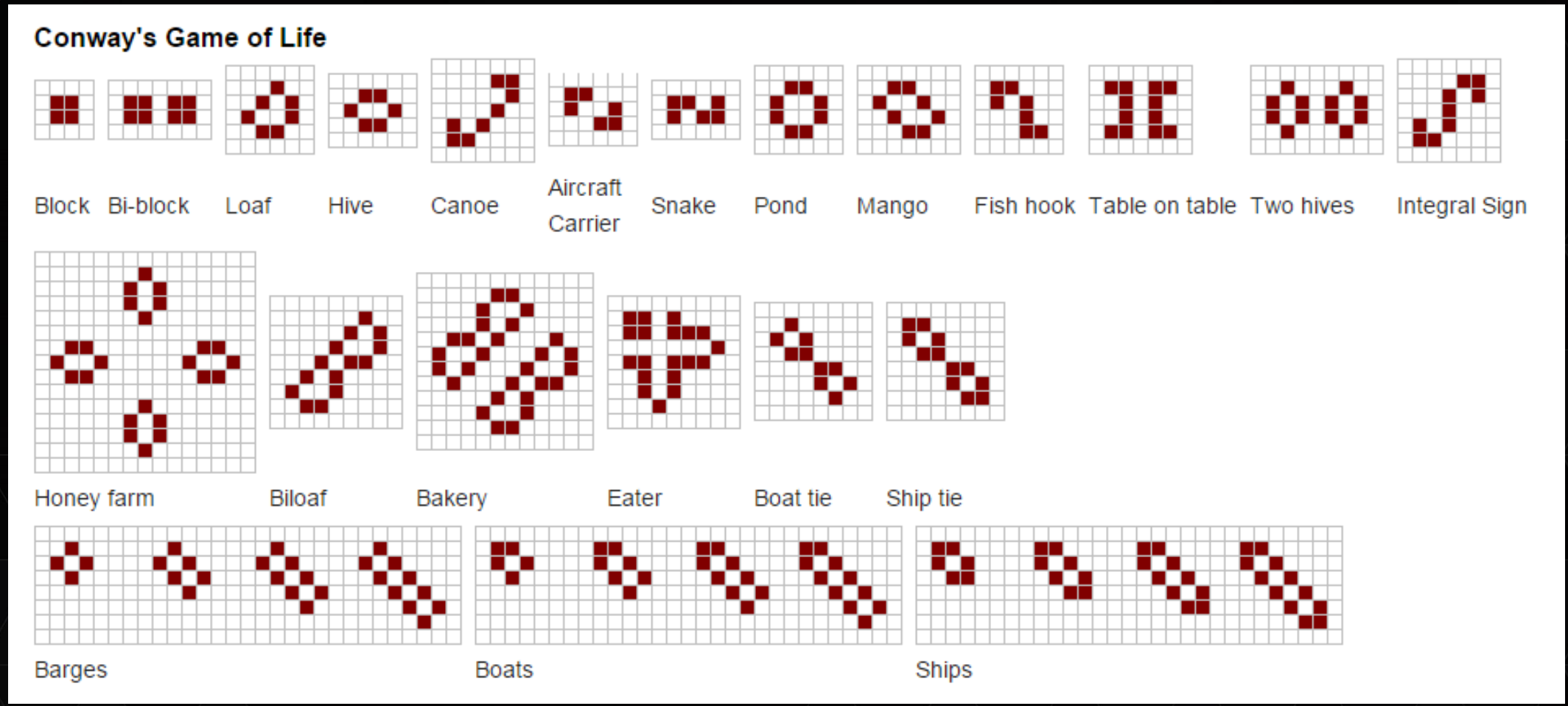
Each column is one draw call.

First draw uses different shader, VBO, and VS constant. *This draws instanced spheres.*

Eight other calls use same shader, and VBO. *These draw the GUI bars.*

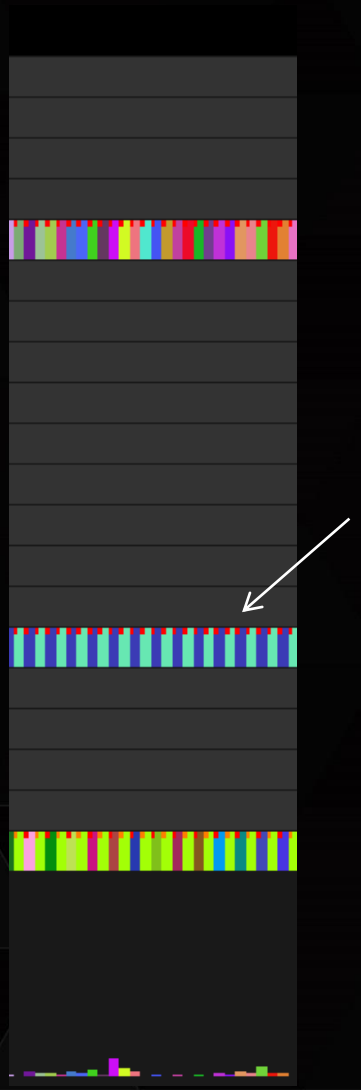
PS Const1 flip-flops between 2 states. **This is the grey and green bars in the GUI of the app.**





"Mathematical Games – The fantastic combinations of John Conway's new solitaire game 'life'". John Horton Conway, 1970.
Image from wikimedia commons.

COMMON PATTERNS

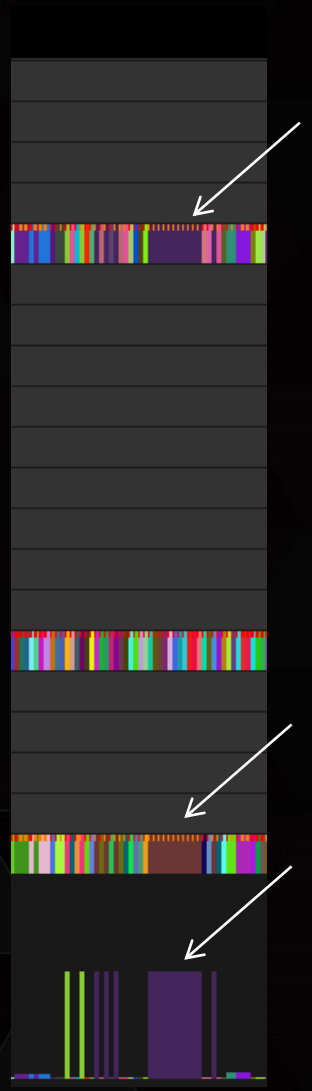


The Flip-Flop

Bars oscillate between values. Indicates potentially unnecessary switch between two states.

Example:
Draw faces, then edges, then faces, then edges.

COMMON PATTERNS



Flatliner

Set of draw calls which use the
same shader, VBO and number of primitives.
Draw may be unnecessarily repeated.

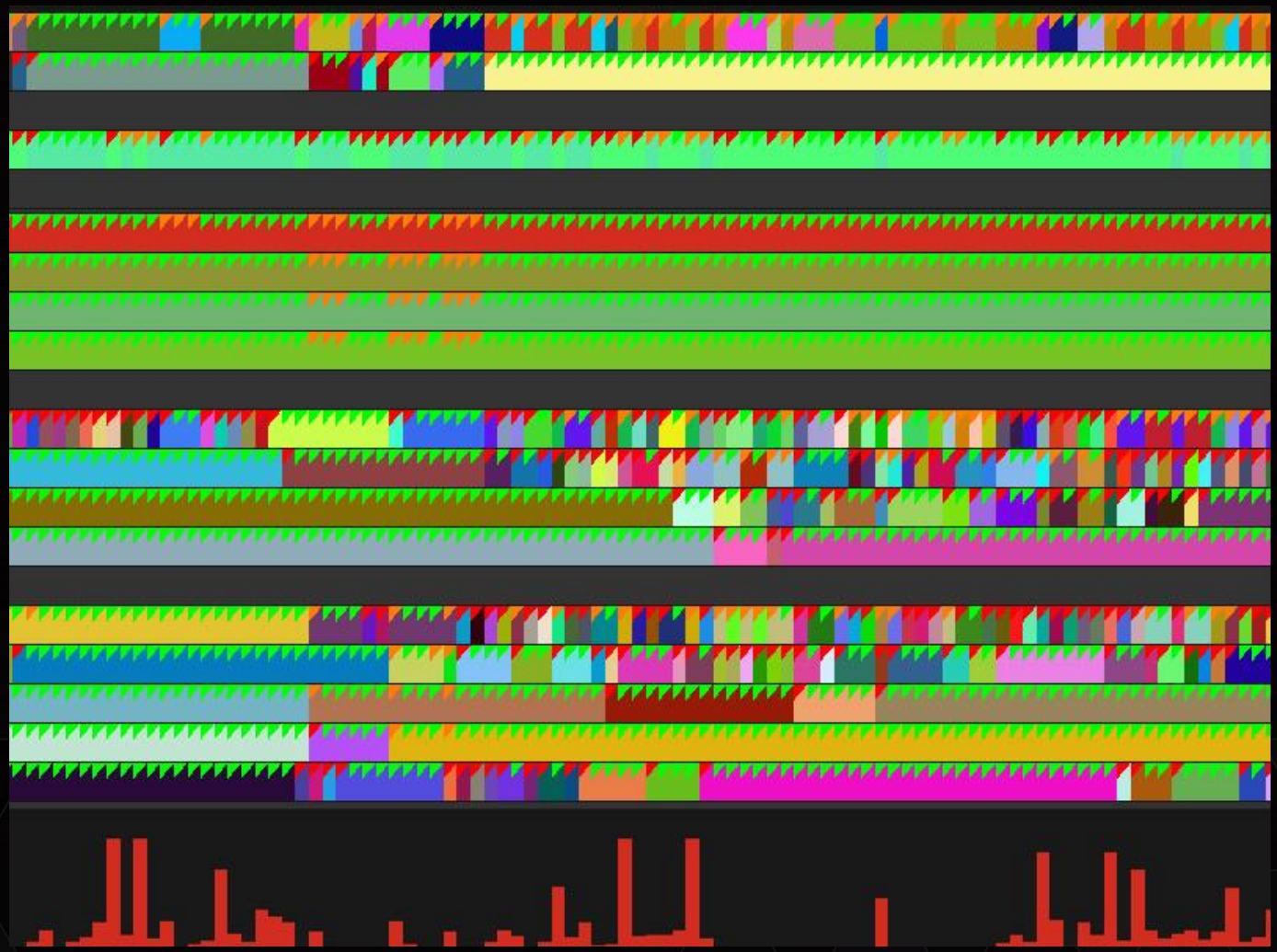
Example:
Drawing multiple copies of an object
in the different locations.

COMMON PATTERNS

The Repeater

A set of states that is similar
to an earlier group.
Strongly suggests candidates for grouping.

Example:
Draw legs, arms, back and seat of a chair.
Then draw whole chair again!



ALL ABOUT THE (DATA) PATTERNS

Worst Case

1. Recreating buffers
2. Using single buffer /w different data

StateViewer Output

Create
Map / Unmap
IASetVB / Draw
Create
Map / Unmap
IASetVB / Draw
Create
Map / Unmap
IASetVB / Draw
Create
Map / Unmap
IASetVB / Draw
Create
Map / Unmap
IASetVB / Draw
Map / Unmap
IASetVB / Draw
Map / Unmap
IASetVB / Draw
Map / Unmap
IASetVB / Draw
Map / Unmap
IASetVB / Draw
Present

REPEAT !

all red flags

Bad Case

1. Good - Write once
2. Lots of switching

StateViewer Output

Create
Create
Create
Create
Map / Unmap
Map / Unmap
Map / Unmap
Map / Unmap
Map / Unmap
IASetVB
Draw
IASetVB
Draw
IASetVB
Draw
IASetVB
Draw
IASetVB
Draw
Present
IASetVB
Draw
Present
IASetVB
Draw
Present

mostly orange

Better Case

1. Good - Write once
2. State sorting !
3. Draw without switch.

StateViewer Output

Create
Create
Create
Create
Map / Unmap
Map / Unmap
Map / Unmap
Map / Unmap
Map / Unmap
State Sort !
IASetVB
Draw
Draw
Draw
IASetVB
Draw
Draw
Present
IASetVB
Draw
Draw
Draw
IASetVB
Draw
Draw
Present

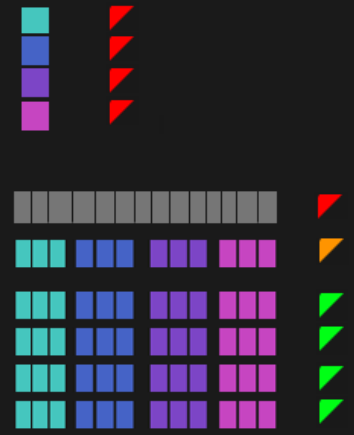
mostly green

Great Case

1. Good - Write once
2. State sort!
3. Make global buffers for other state
4. Draw ONCE with MultiDraw

StateViewer Output

Create
Create
Create
Create
Map / Unmap
Map / Unmap
Map / Unmap
Map / Unmap
Map / Unmap
State Sort !
Create Const Buffer
Create Transform Buffer
Map / Unmap
MultiDraw
Present
MultiDraw
Present
MultiDraw
Present

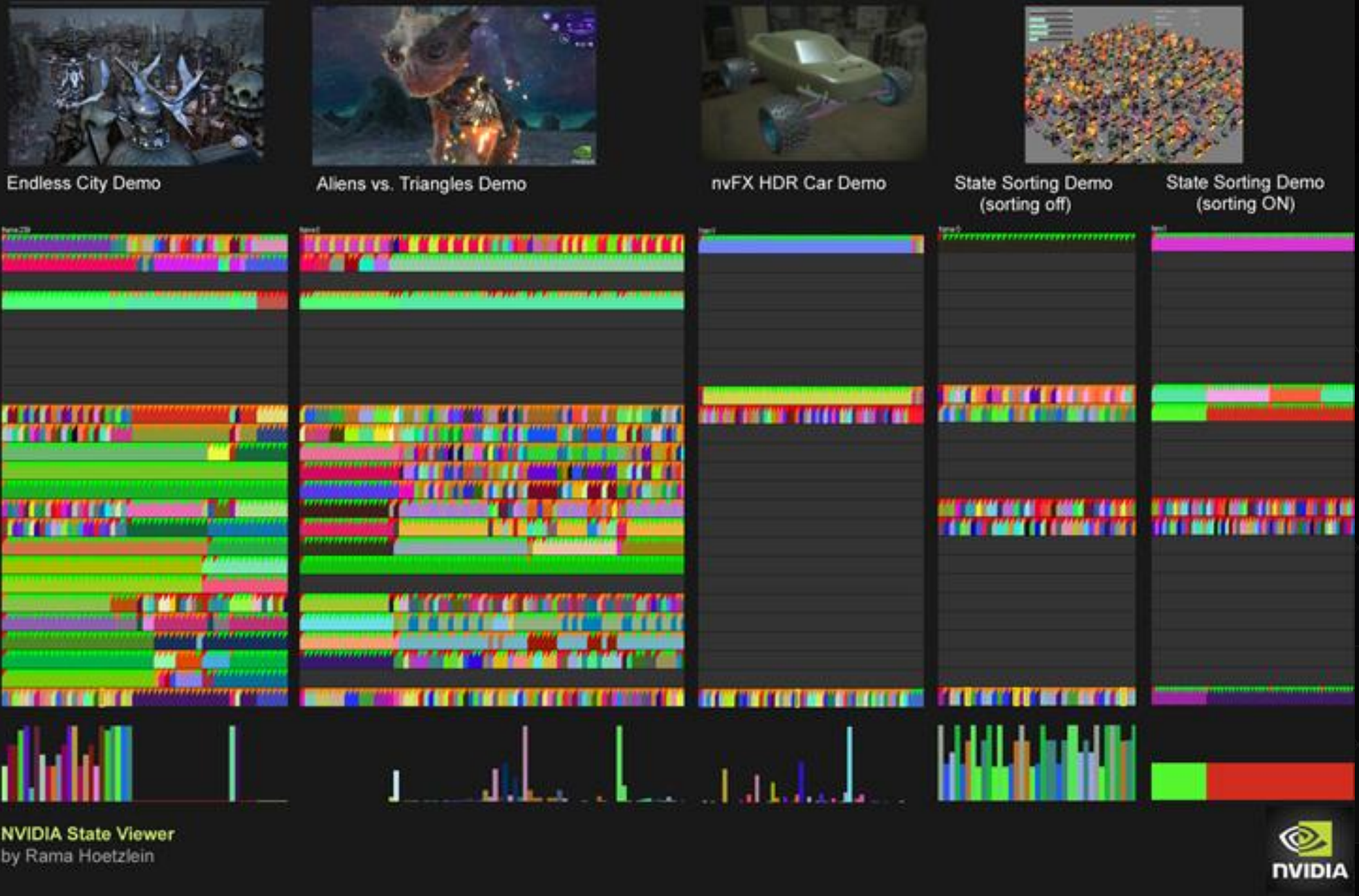


all green
(except first frames)

Too many red flags.

Too many orange flags.

Green flags!


Endless City Demo
Aliens vs. Triangles Demo
nvFX HDR Car Demo
State Sorting Demo (sorting off)
State Sorting Demo (sorting ON)
NVIDIA State Viewer
by Rama Hoetzlein
NVIDIA

Endless City Demo

Frame #	2
Frame Draws	29
Frame Prims	1308
Frame Transfer	5632 bytes

Frame States:

	Modify	Switch	Reuse	Unique
Shader		23		90
Render Target		23		12
Viewport				0
Raster State		22		3
Depth State				0
Blend State				0
Sampler State				0
Input Assembler				0
Texture				0
Vertex 0		29		104
Vertex 1		28		103
Vertex 2			27	1
Vertex 3			28	0
Vertex 4			28	0
VS Const 0			26	16
VS Const 1			26	100
VS Const 2			27	6
VS Const 3			27	8
VS Const 4			29	0
PS Const 0	82	5		83
PS Const 1		13	13	14
PS Const 2		11	17	6
PS Const 3		6	16	13
PS Const 4		9	16	16
Index Bufer		29		104

Aliens vs. Triangles Demo

Frame #	0
Frame Draws	267
Frame Prims	714832
Frame Transfer	11200 bytes

Frame States:

	Modify	Switch	Reuse	Unique
Shader		141	125	82
Render Target		46	193	36
Viewport				0
Raster State		66	125	2
Depth State				0
Blend State				0
Sampler State				0
Input Assembler				0
Texture				0
Vertex 0		266		74
Vertex 1		260		75
Vertex 2		91	175	44
Vertex 3		71	195	43
Vertex 4		71	195	43
VS Const 0		37	189	41
VS Const 1		25	216	39
VS Const 2			248	38
VS Const 3			265	0
VS Const 4				0
PS Const 0		36	189	96
PS Const 1		36	217	34
PS Const 2		68	197	7
PS Const 3			222	117
PS Const 4				0
Index Bufer		266		139

nvFX HDR Car Demo

Frame #	4
Frame Draws	56
Frame Prims	1259690
Frame Transfer	1008 bytes

Frame States:

	Modify	Switch	Reuse	Unique
Shader			52	9
Render Target				0
Viewport				0
Raster State				0
Depth State				0
Blend State				0
Sampler State				0
Input Assembler				0
Texture			51	6
Vertex 0	50			101
Vertex 1				0
Vertex 2				0
Vertex 3				0
Vertex 4				0
VS Const 0				0
VS Const 1				0
VS Const 2				0
VS Const 3				0
VS Const 4				0
PS Const 0				0
PS Const 1				0
PS Const 2				0
PS Const 3				0
PS Const 4				0
Index Bufer		52		51

State Sorting Demo (sorting off)

Frame #	0
Frame Draws	400
Frame Prims	25078284
Frame Transfer	31920 bytes

Frame States:

	Modify	Switch	Reuse	Unique
Shader			399	6
Render Target				0
Viewport				0
Raster State				0
Depth State				0
Blend State				0
Sampler State				0
Input Assembler				0
Texture		399		8
Vertex 0		399		9
Vertex 1				0
Vertex 2				0
Vertex 3				0
Vertex 4				0
VS Const 0	400			399
VS Const 1	400			802
VS Const 2				0
VS Const 3				0
VS Const 4				0
PS Const 0				0
PS Const 1				0
PS Const 2				0
PS Const 3				0
PS Const 4				0
Index Bufer		399		7

State Sorting Demo (sorting ON)

Frame #	0
Frame Draws	400
Frame Prims	25078284
Frame Transfer	31920 bytes

Frame States:

	Modify	Switch	Reuse	Unique
Shader			399	5
Render Target				0
Viewport				0
Raster State				0
Depth State				0
Blend State				0
Sampler State				0
Input Assembler				0
Texture			396	4
Vertex 0			393	7
Vertex 1				0
Vertex 2				0
Vertex 3				0
Vertex 4				0
VS Const 0	400			400
VS Const 1	400			800
VS Const 2				0
VS Const 3				0
VS Const 4				0
PS Const 0				0
PS Const 1				0
PS Const 2				0
PS Const 3				0
PS Const 4				0
Index Bufer			393	7

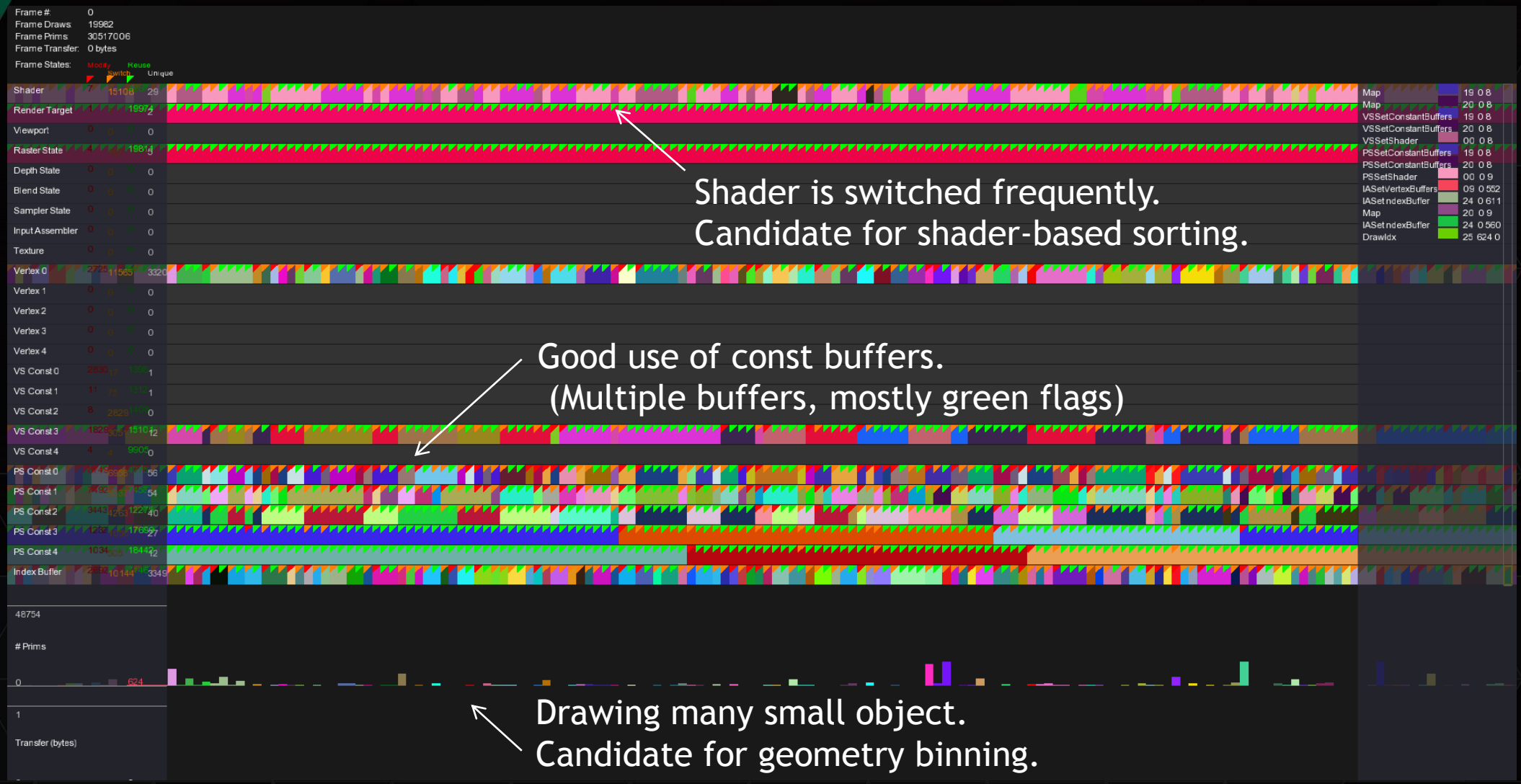
NVIDIA State Viewer
by Rama Hoetzlein

NVIDIA

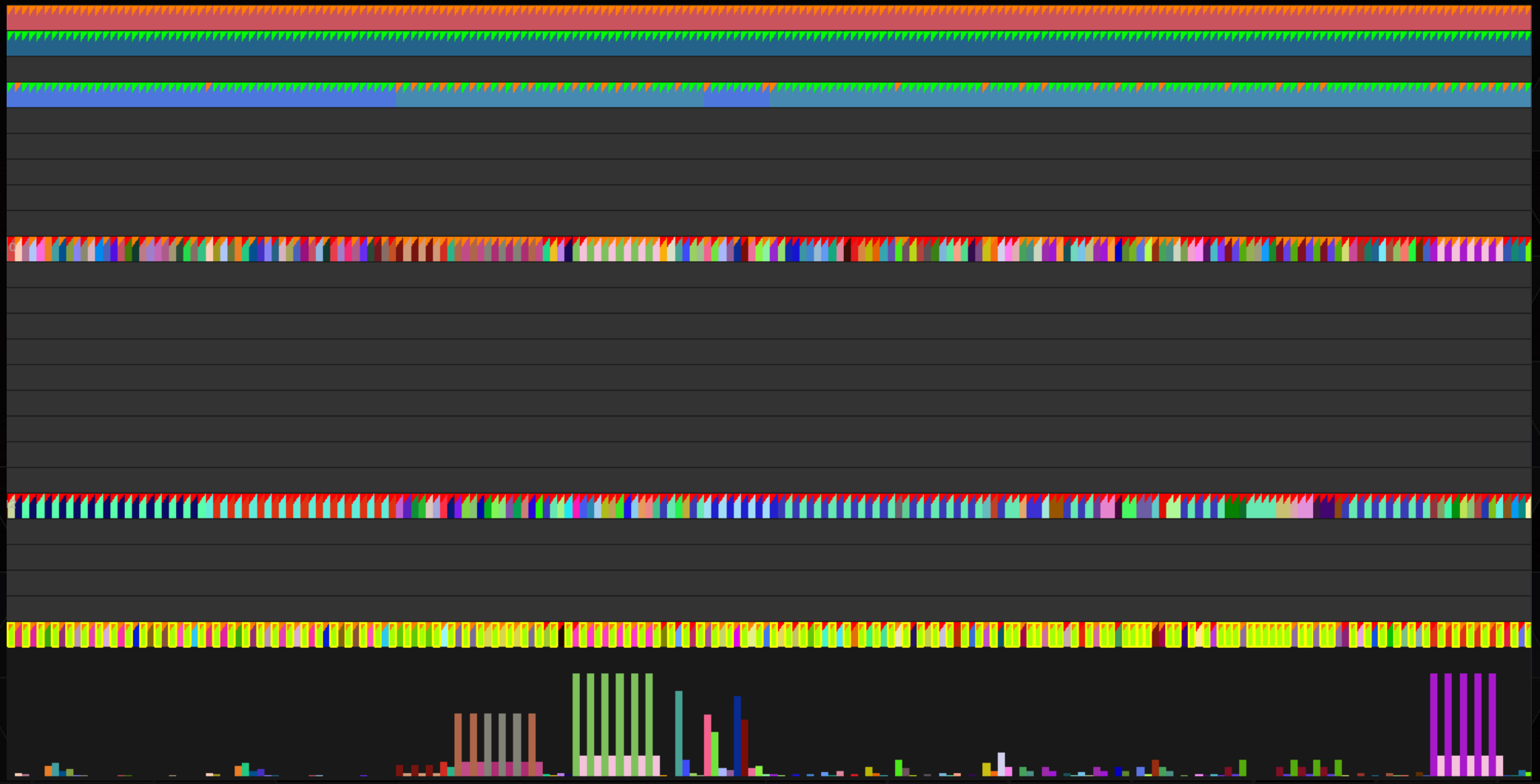
GPU TECHNOLOGY CONFERENCE

REAL-WORLD APPLICATIONS

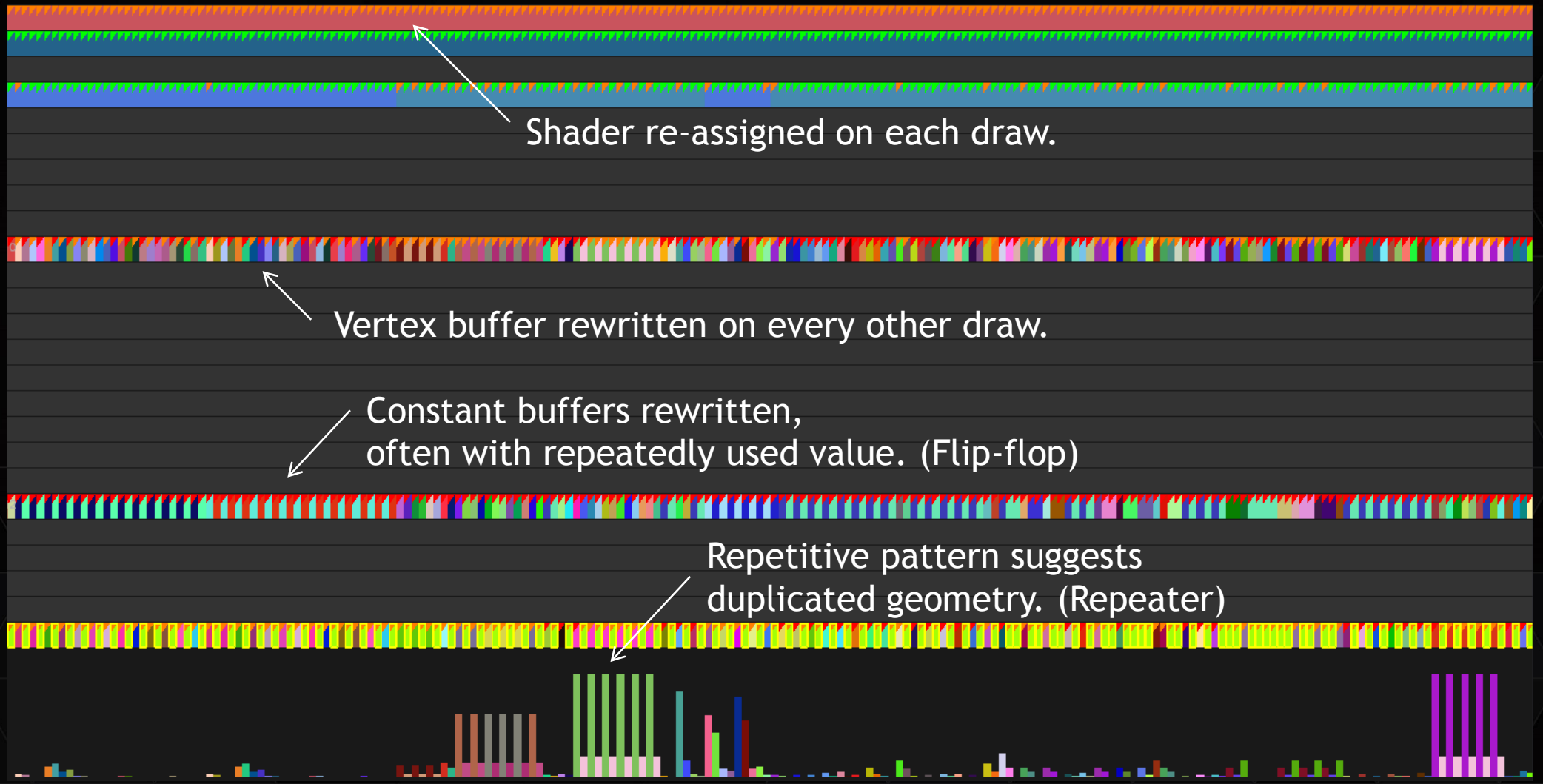
REAL-WORLD APPLICATIONS



REAL-WORLD APPLICATIONS



REAL-WORLD APPLICATIONS



STATEVIEWER: TOOL COMPARISON

GPU Timing:
Gives valuable information about what the graphics API and GPU are doing.
Good for GPU-bound apps. Use NSight.

e.g. Does GPU spend more time in vertex or pixel shader?

CPU Function Profiling:
Gives valuable information about which are the slowest functions.
Good for Algorithm-bound apps.

e.g. Which specific part of a CPU algorithm is slowest?

StateViewer:
Gives systematic information about design patterns in the application.
Good for Data-bound apps.
*Tells us **why** the app is slow, without access to code!*

e.g. How could the data be better organized for submission to graphics pipeline?

SUMMARY

StateViewer has identified unknown issues
in several large CAD/Workstation applications.

Provides an overall picture of the
application's ***systematic*** behavior.

Gives feedback with direct indicators
on areas of improvement.

Thank You!

DATA VISUALIZATION OF THE GRAPHICS PIPELINE

JOIN THE CONVERSATION

#GTC15