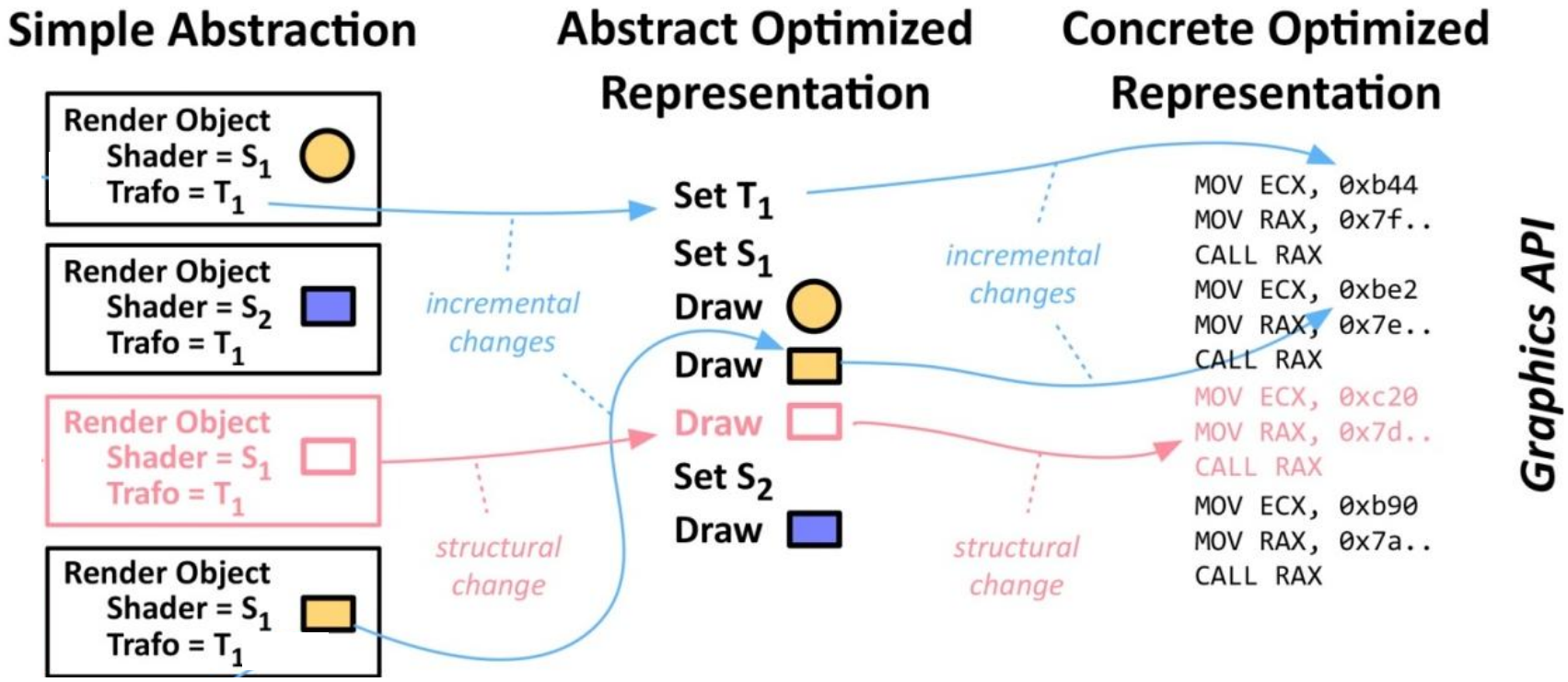


An Incremental Rendering VM

Georg Haaser, Harald Steinlechner
Stefan Maierhofer, and Robert F. Tobler

VRVis Research Center
Vienna, Austria

Motivation

Rendering huge scenes with dynamism

- Editor applications / strategy games
- Everything is changeable (add / remove / modify)
- Changes are relatively “rare“

Problems

- Many driver calls
- Driver calls impose cost (e.g. redundancy, validation)
- Hardware not utilized

Typical approaches

Static

- Re-organize the scene (e.g. grouping)
- Merge geometries
- Render caches (e.g. Durbin et. al. 1995)
- **Difficult to update**

Dynamic

- Filter redundant API calls
- State sorting (e.g. sort by shaders)
- **Runtime overhead**

Ultimate Goal

Dynamic flexibility

without compromising performance

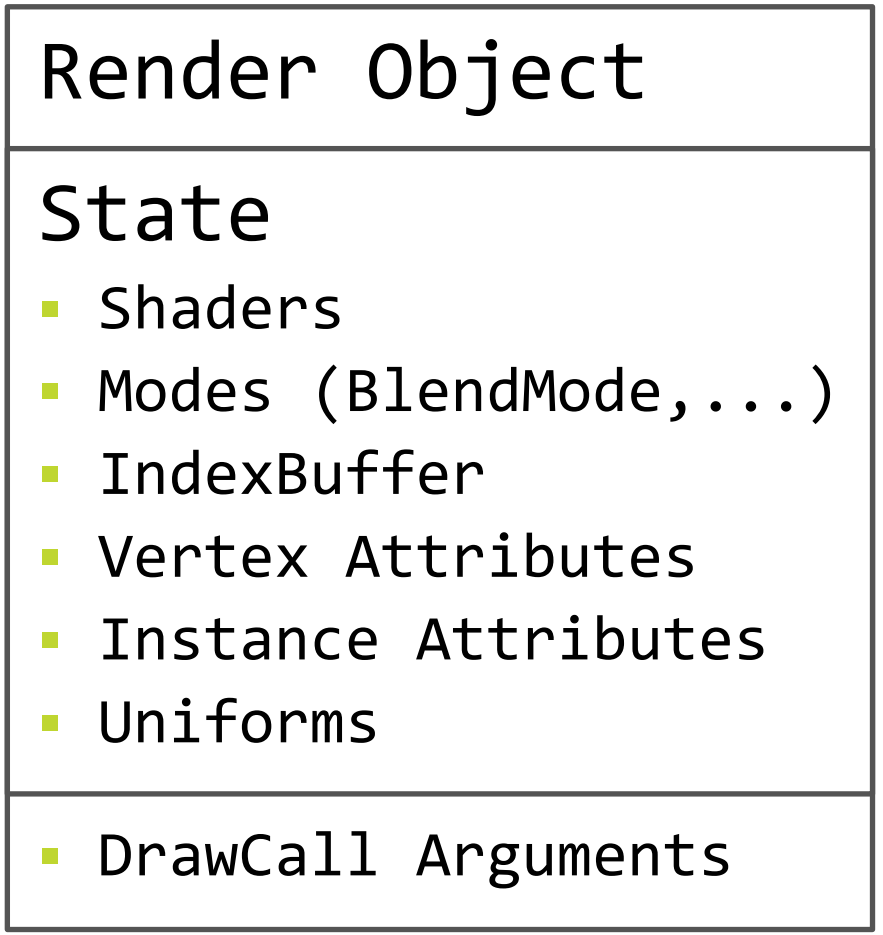
Minimal frontend

Expected input

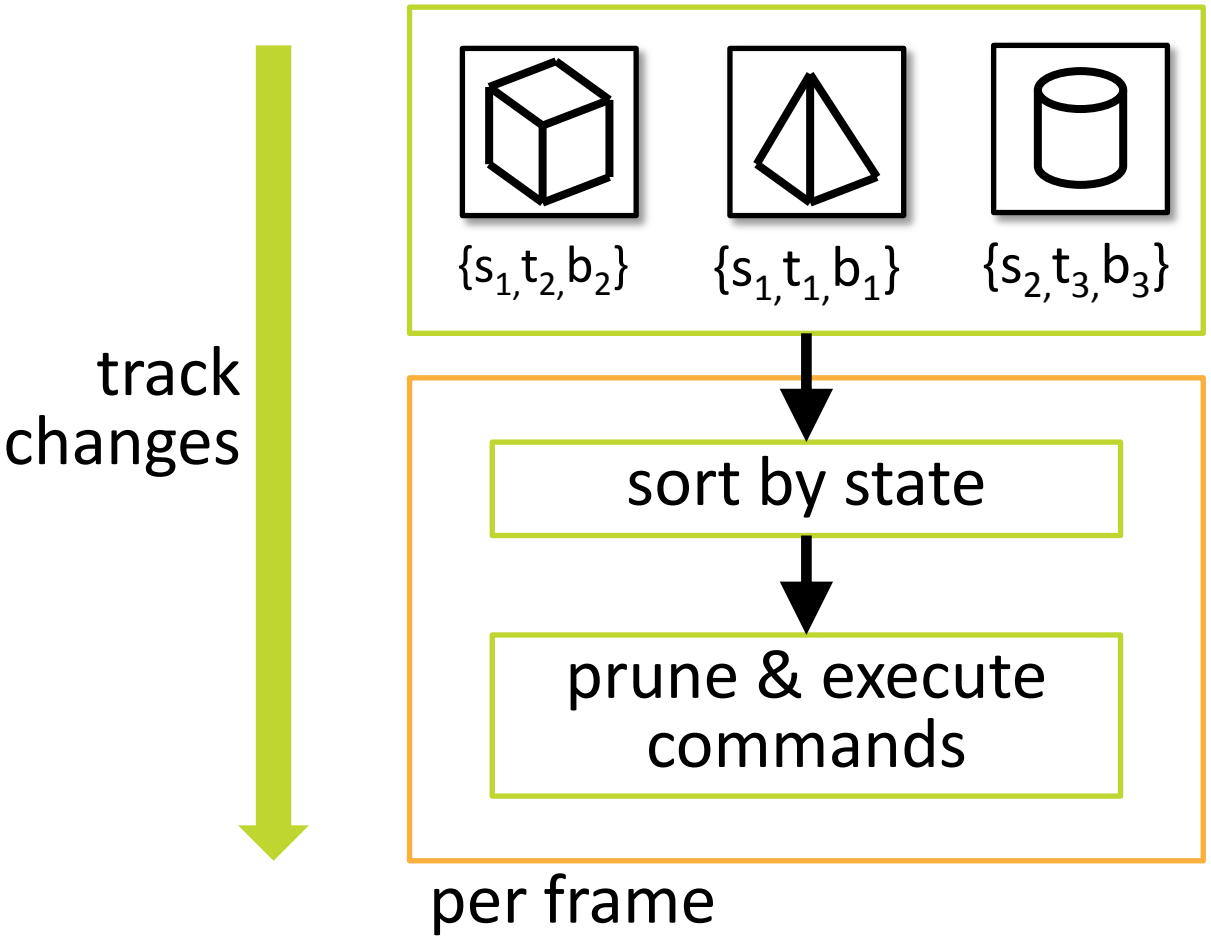
- Set of Render Objects
- Order independent

Render Objects

- Draw call arguments
- Stateless



How to render those objects?



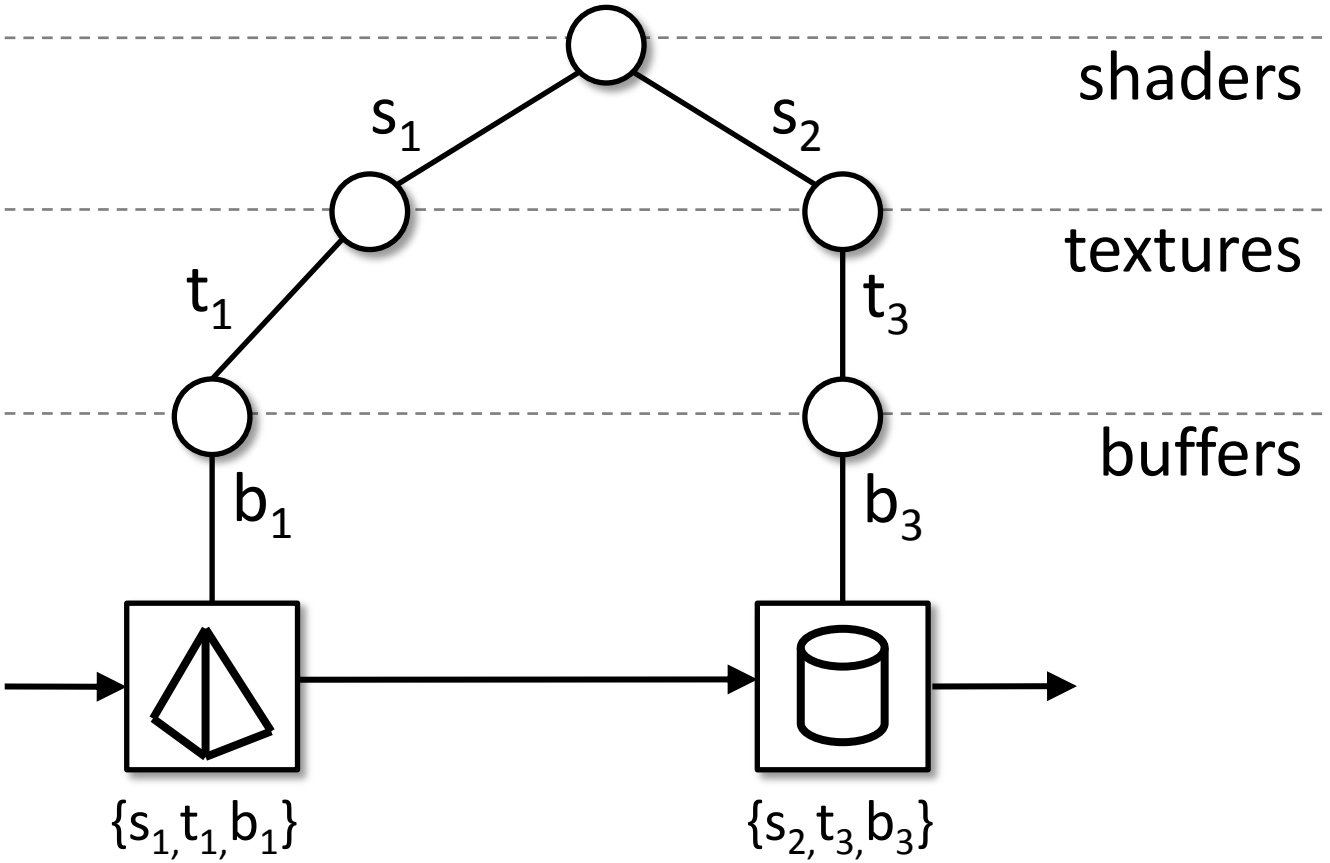
State Sorting

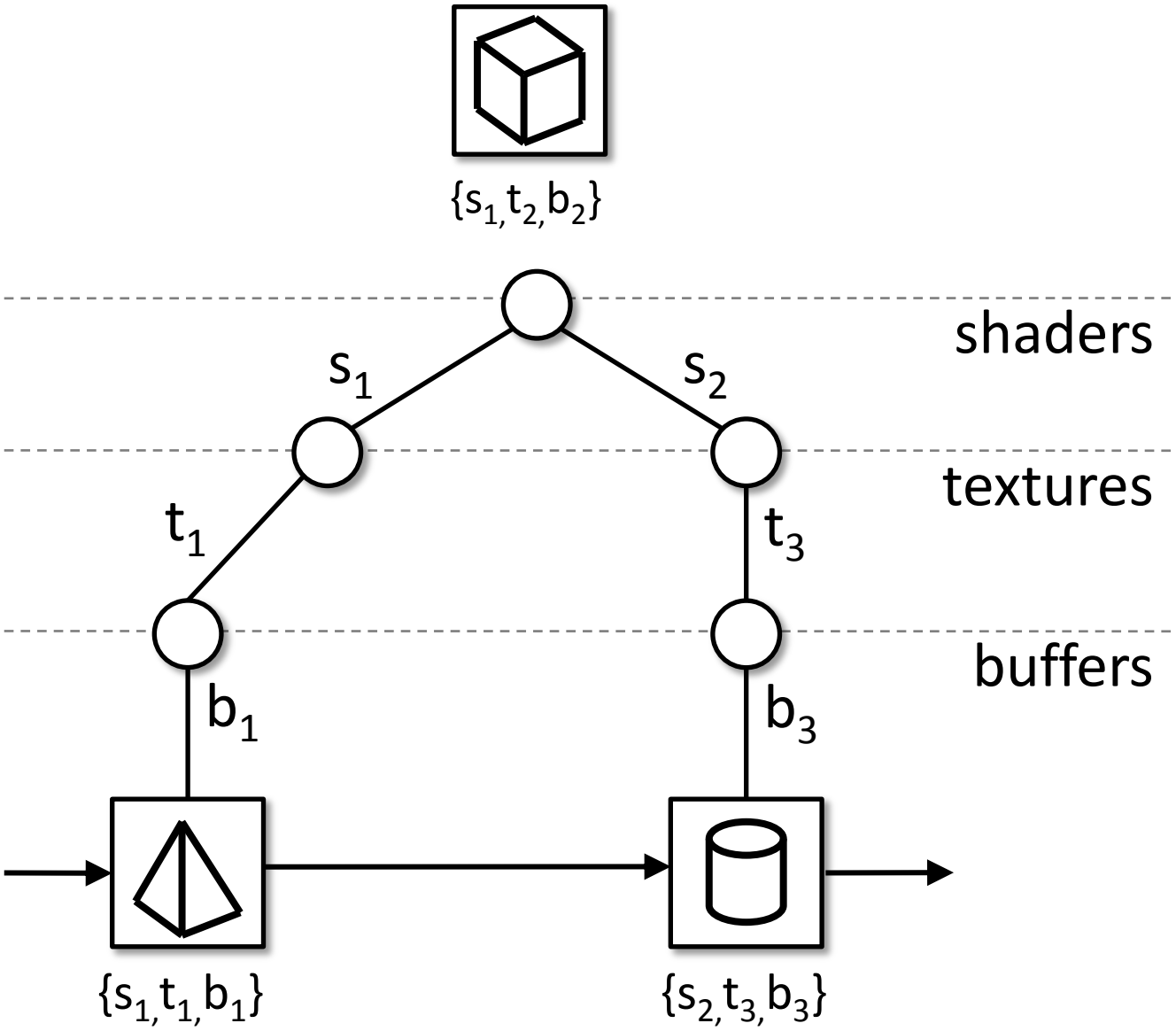
Optimal order

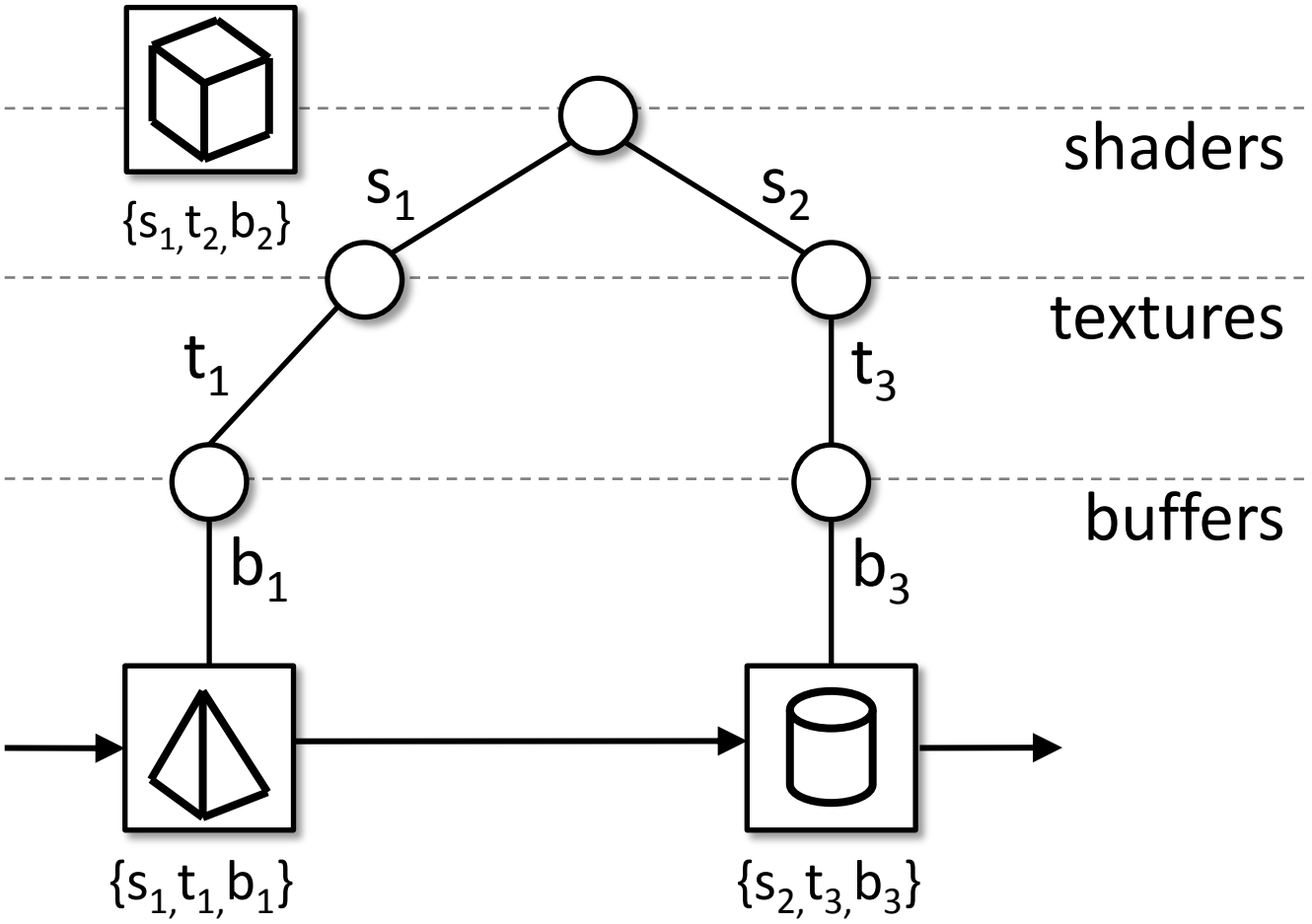
- Costs for all transitions (shaders, textures, etc.)
- Minimize total cost

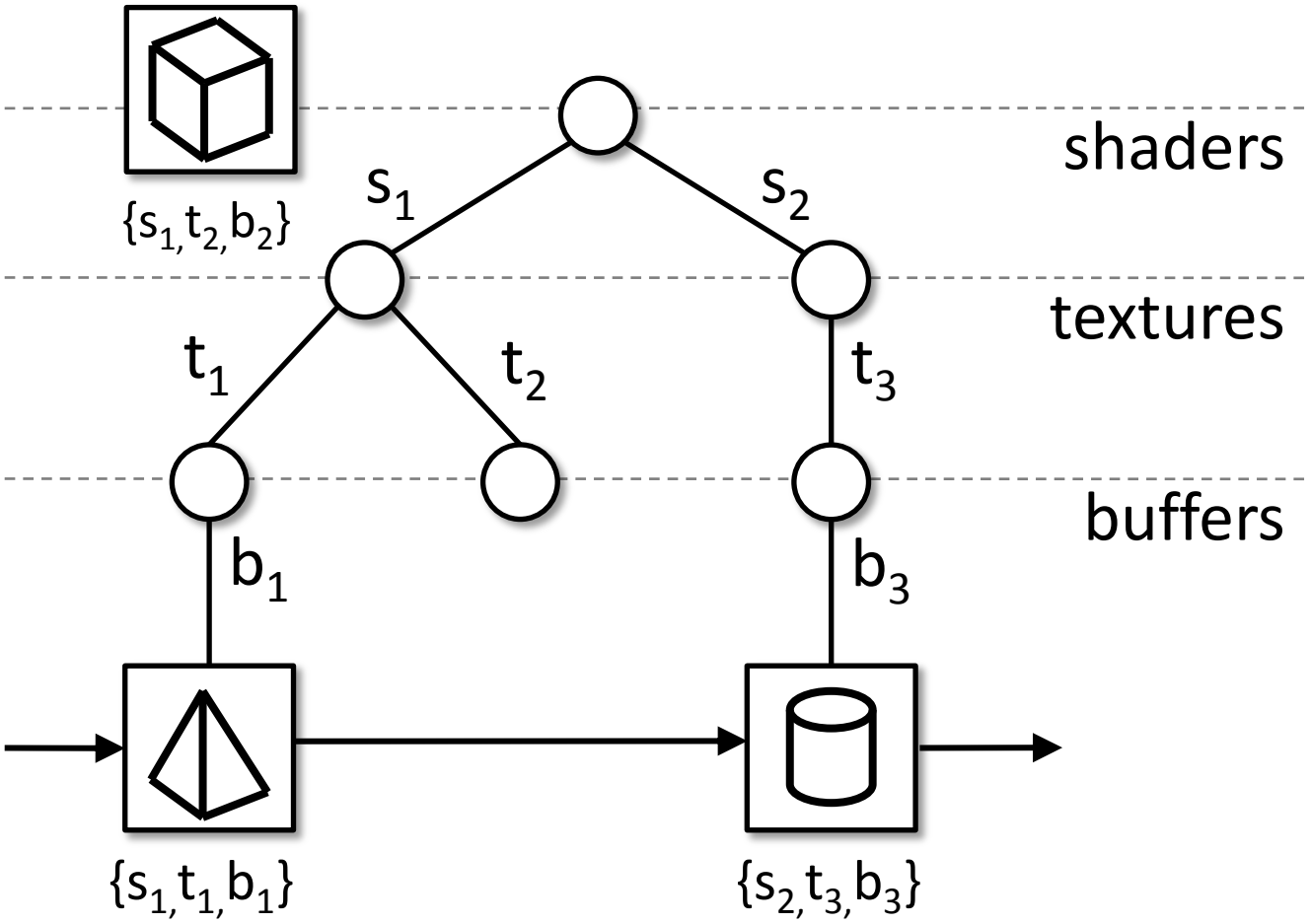
Incremental changes?

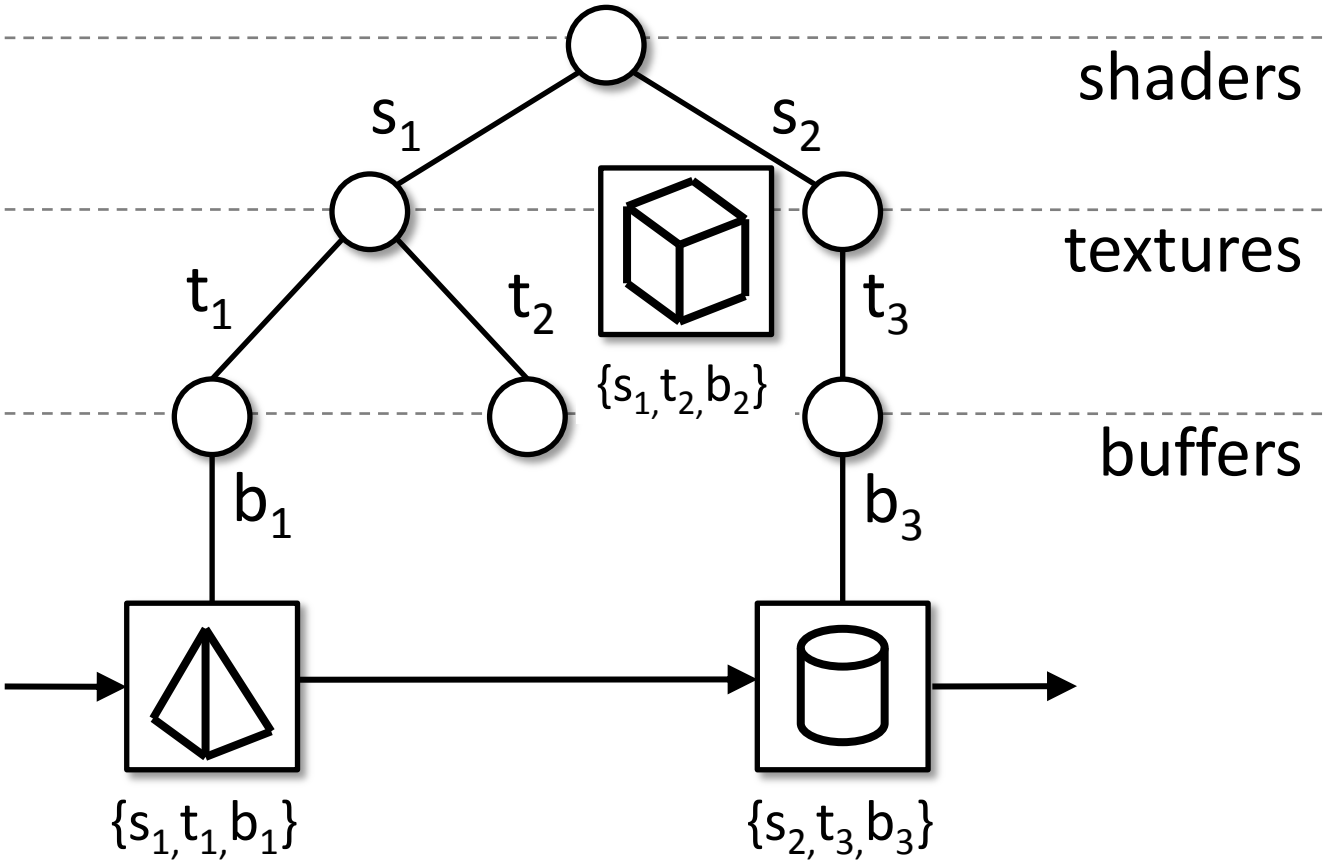
- Cannot maintain optimality
- State-trie turns out to be effective
- Easy to update

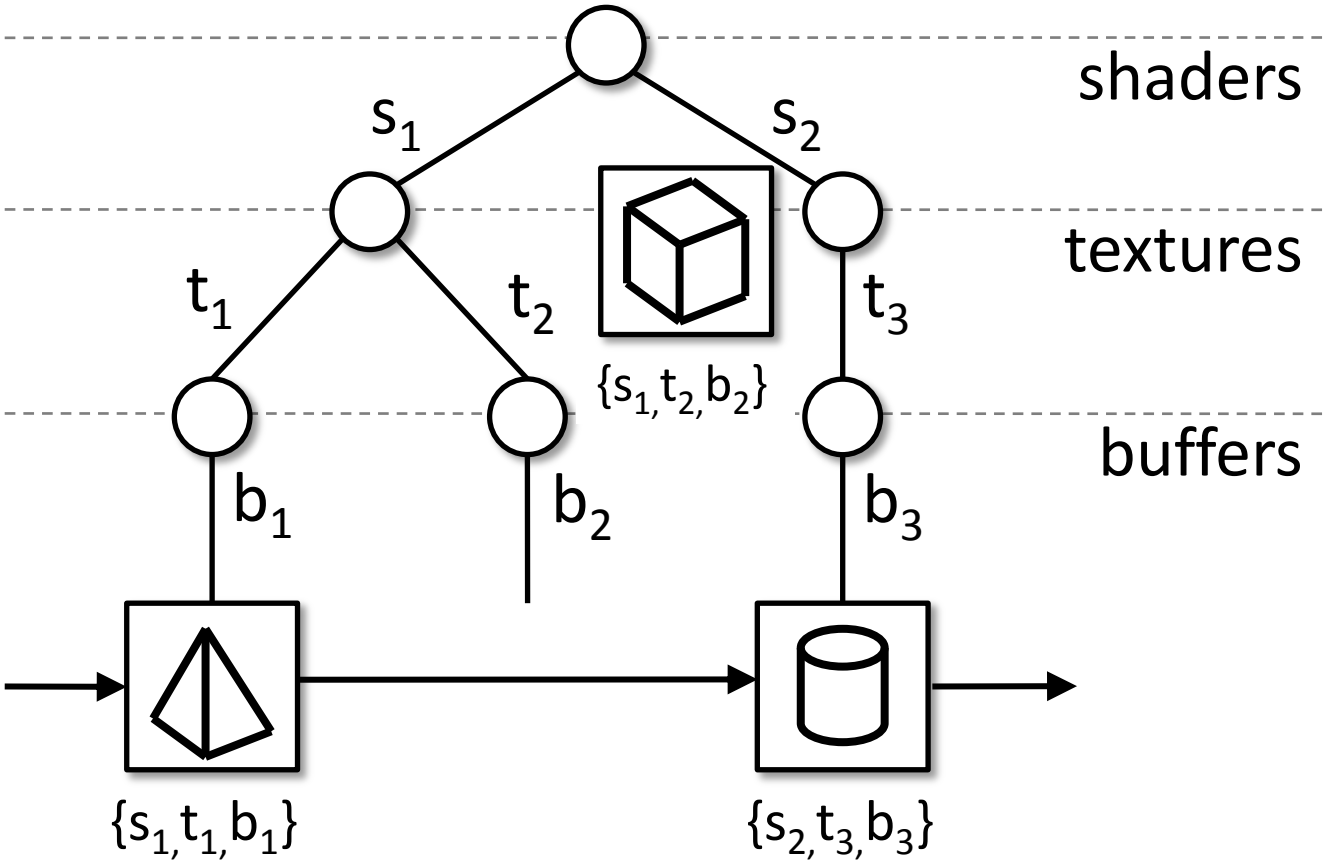

shaders
textures
buffers
s_1
s_2
t_1
t_3
b_1
b_3
$\{s_1, t_1, b_1\}$
$\{s_2, t_3, b_3\}$

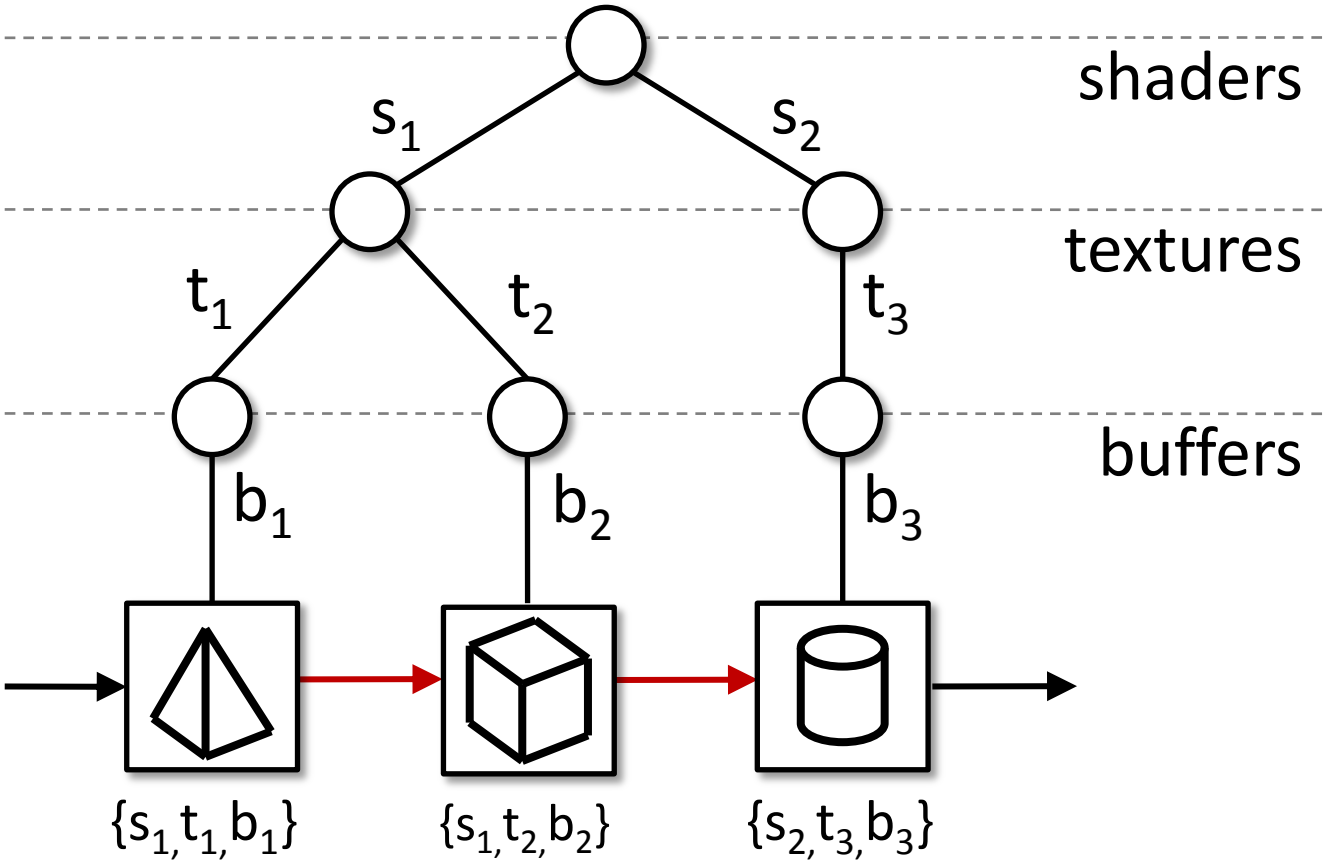

{s1,t2,b2}
shaders
s1
s2
textures
t1
t3
buffers
b1
b3
{s1,t1,b1}
{s2,t3,b3}


shaders
textures
buffers
s_1
s_2
t_1
t_3
b_1
b_3
$\{s_1, t_2, b_2\}$
$\{s_1, t_1, b_1\}$
$\{s_2, t_3, b_3\}$

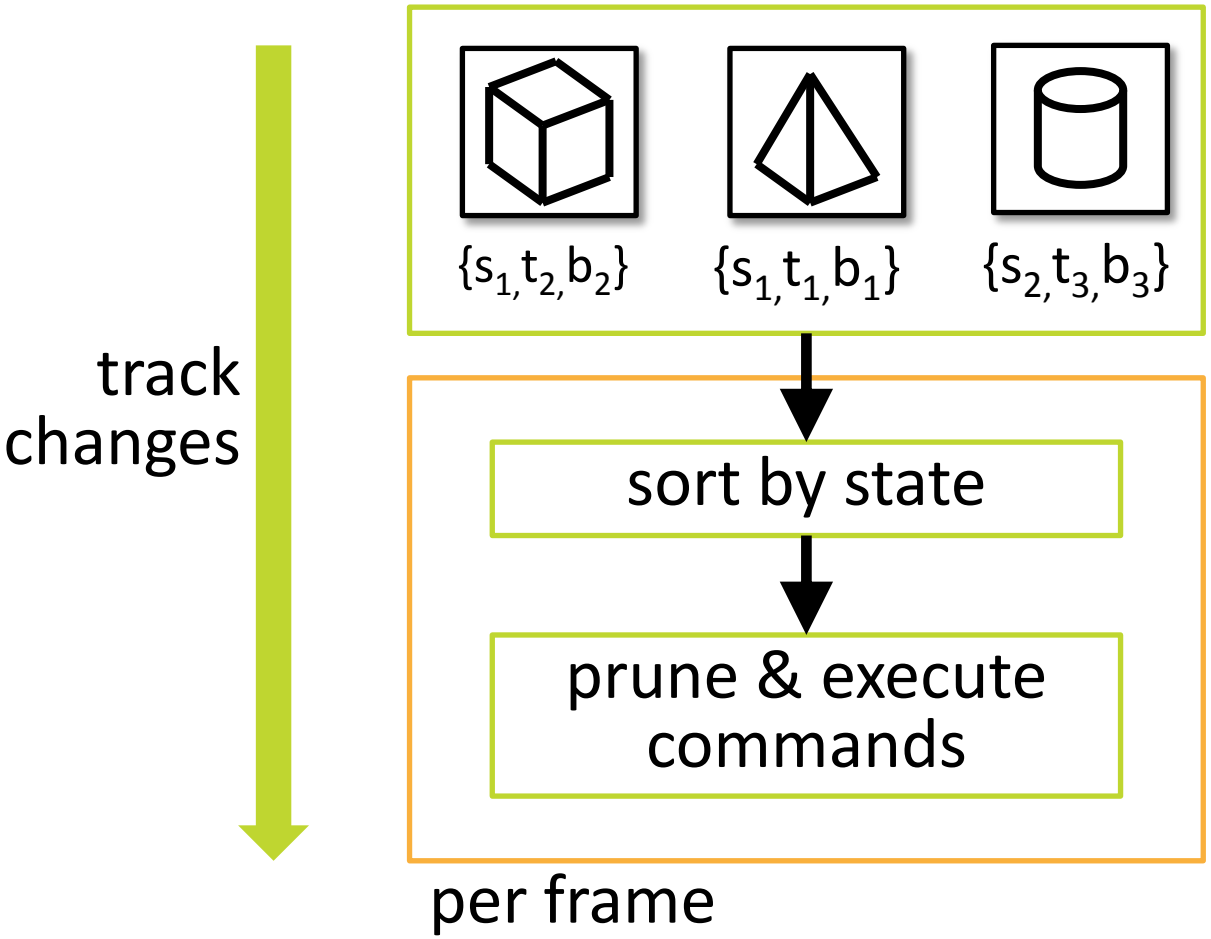

shaders
textures
buffers
s_1
s_2
t_1
t_2
t_3
b_1
b_3
$\{s_1, t_2, b_2\}$
$\{s_1, t_1, b_1\}$
$\{s_2, t_3, b_3\}$


shaders
textures
buffers
s_1
s_2
t_1
t_2
t_3
b_1
b_3
$\{s_1, t_2, b_2\}$
$\{s_1, t_1, b_1\}$
$\{s_2, t_3, b_3\}$

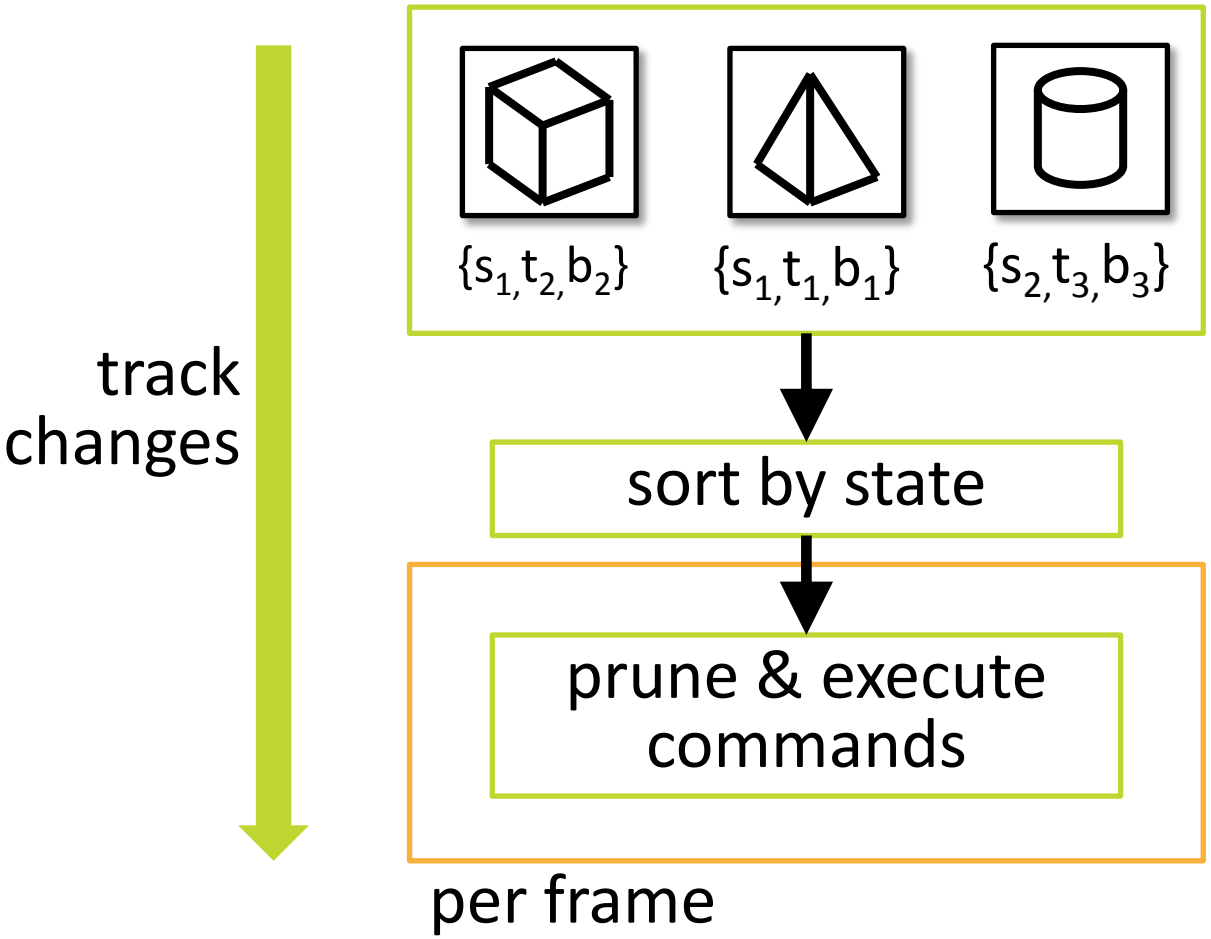

shaders
textures
buffers
s_1
s_2
t_1
t_2
t_3
b_1
b_2
b_3
$\{s_1, t_2, b_2\}$
$\{s_1, t_1, b_1\}$
$\{s_2, t_3, b_3\}$


shaders
textures
buffers
s_1
s_2
t_1
t_2
t_3
b_1
b_2
b_3
$\{s_1, t_1, b_1\}$
$\{s_1, t_2, b_2\}$
$\{s_2, t_3, b_3\}$

Reduce runtime costs



Reduce runtime costs



Success:
less stuff at runtime!

Idea: Graphics commands are data too

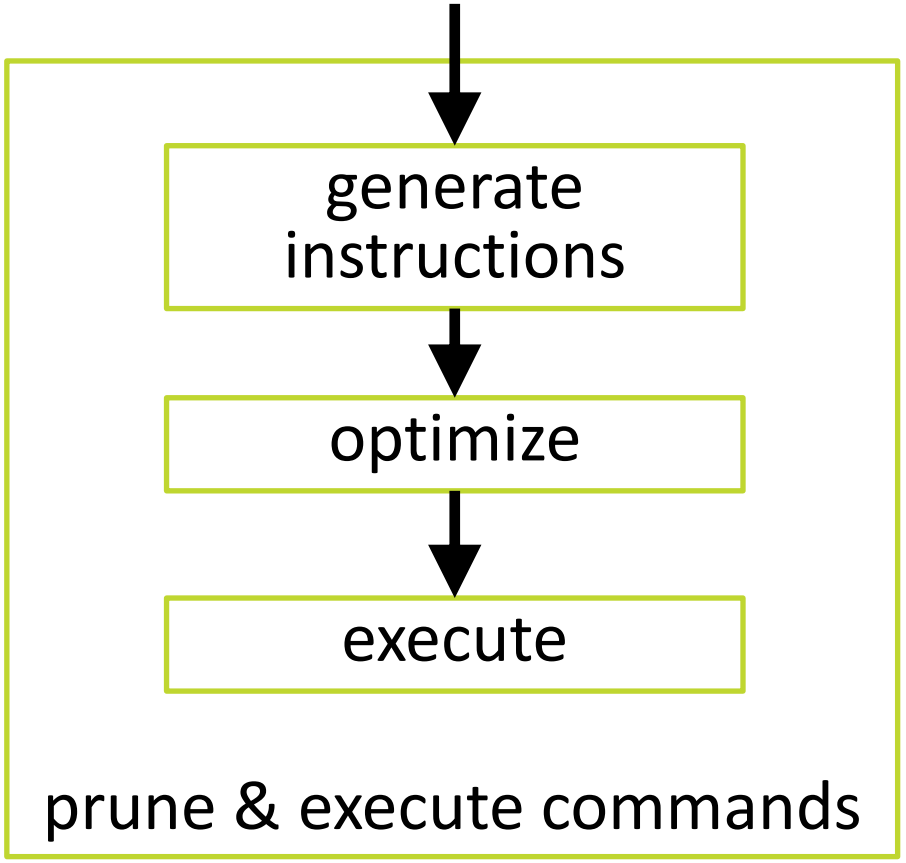
Instructions describe graphics commands

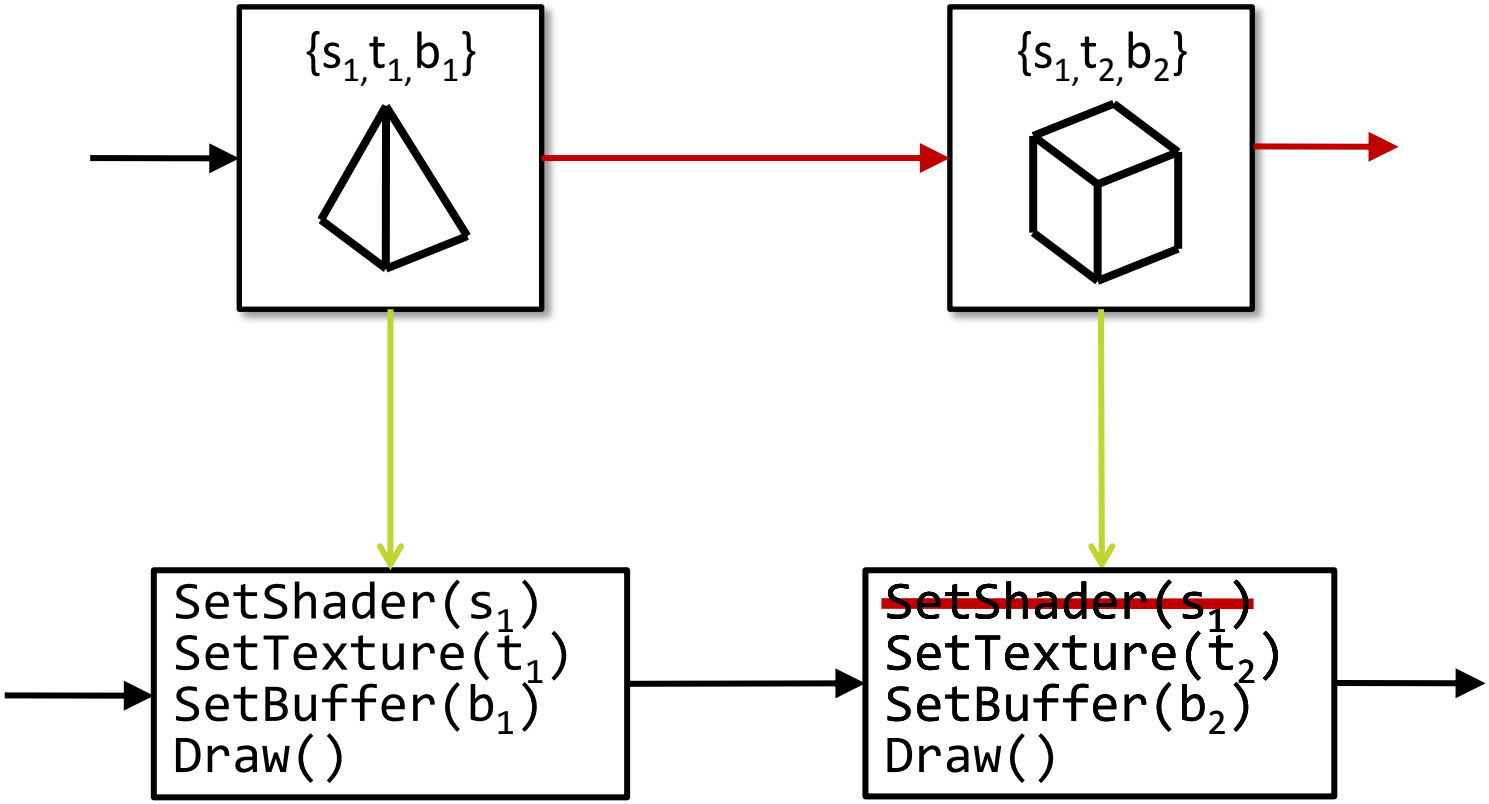
Decouple creation and execution

Examples

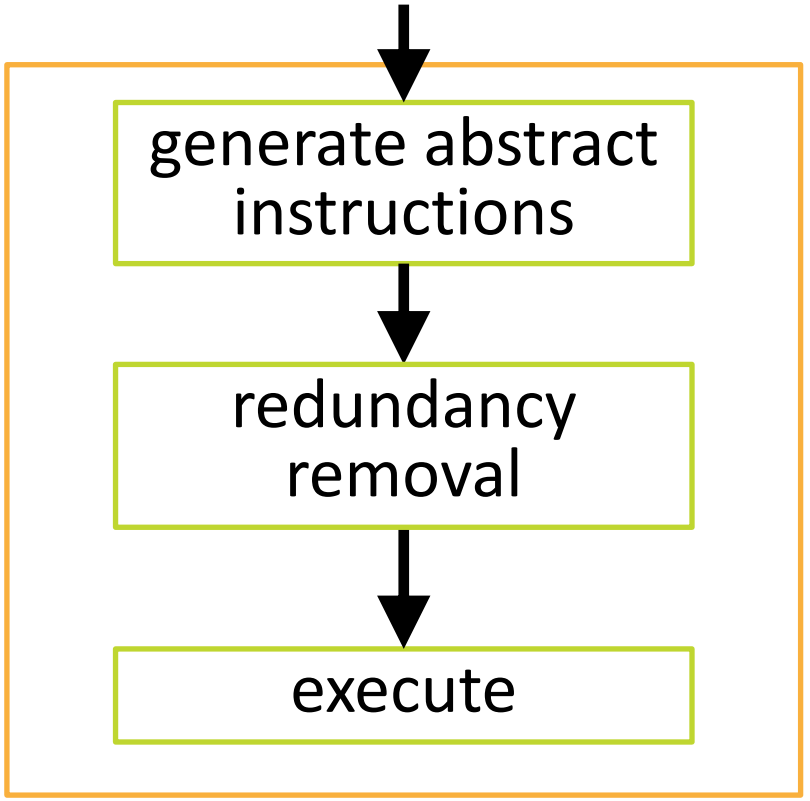
- SetShaderInstruction(5)
- DrawInstruction(Lines, 0, 100)
- …

Idea: Instructions are data too



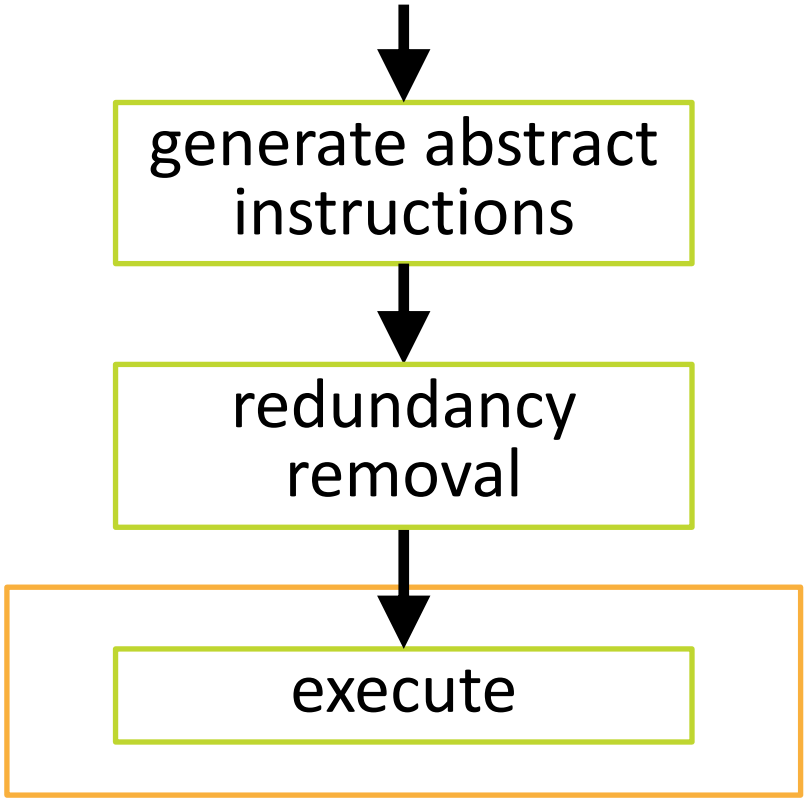

{s1,t1,b1}
{s1,t2,b2}
SetShader(s1)
SetTexture(t1)
SetBuffer(b1)
Draw()
SetShader(s1)
SetTexture(t2)
SetBuffer(b2)
Draw()

Reduce runtime costs



per frame

Reduce runtime costs



Success:
less stuff at runtime!

But...

How to efficiently execute optimized instructions?

Interpreter techniques

Switch interpreter

- Managed / Unmanaged
- Employed by Wörister et al. 2013

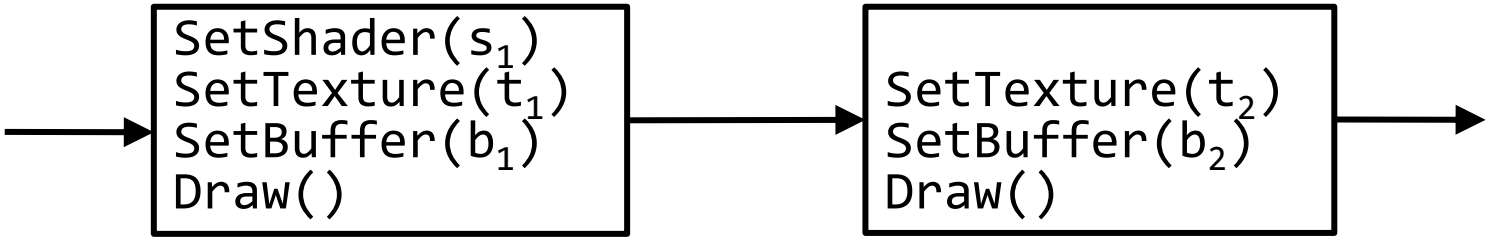
Indirect threaded code

- Array of function pointers

Direct threaded [Bell 1973]

- Compile the whole list of fragments
- Baseline for performance (changes are expensive)

No overhead! Native compilation



Just-in-time compilation

Executable memory

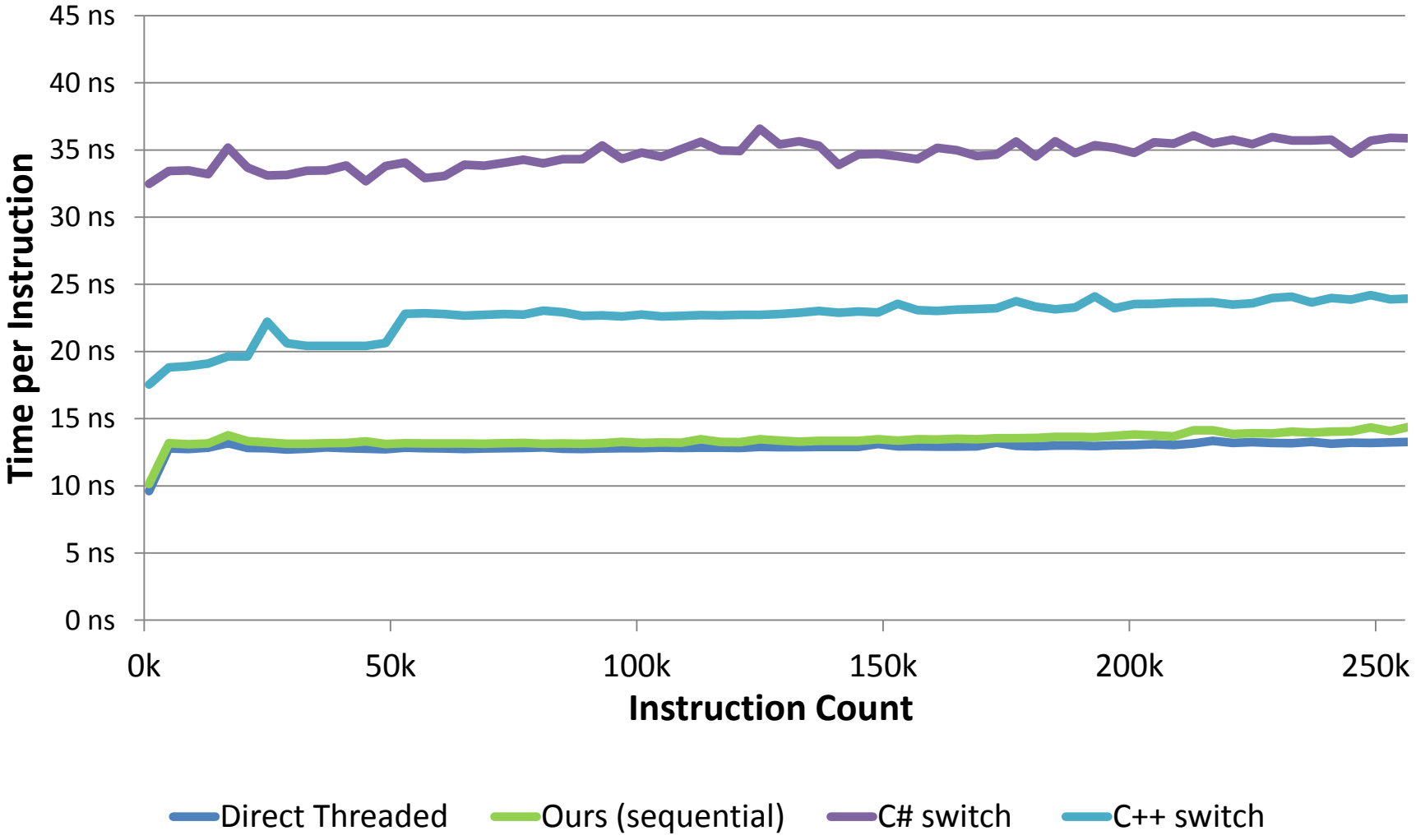
0xEAF1

```
MOV EXC, 0xb44
MOV RAX, 0x7f
CALL RAX
JMP 0xFEED
```

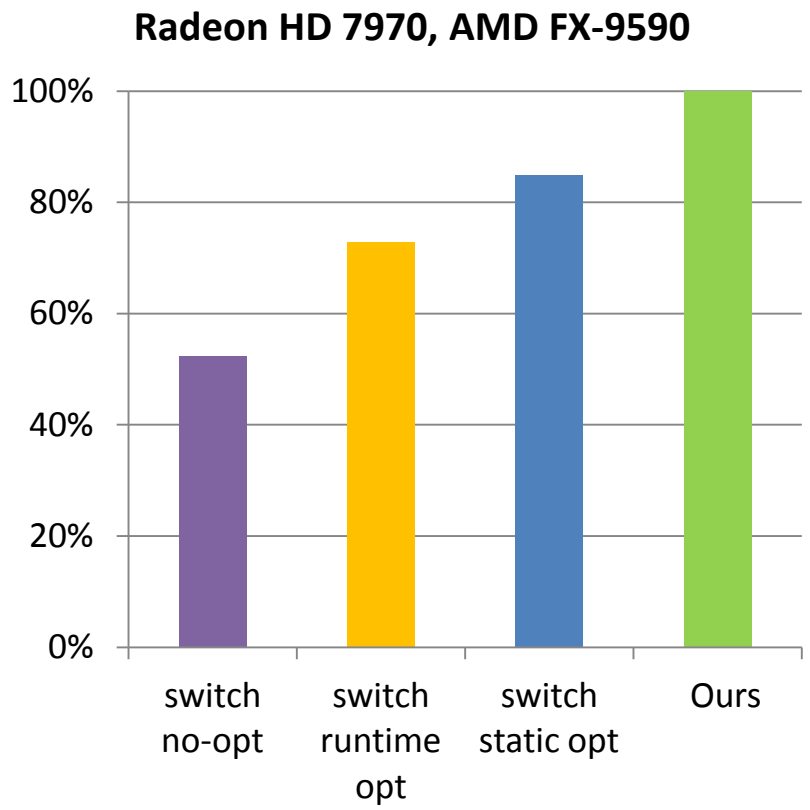
0xFEED

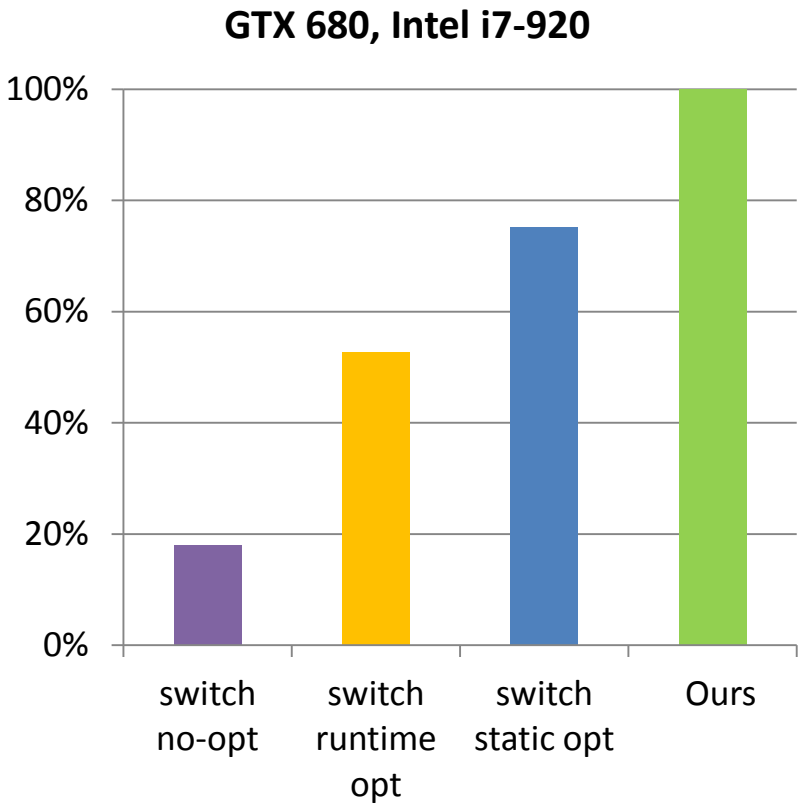
```
MOV EXC, 0xe44
MOV RAX, 0x5a
CALL RAX
JMP 0xDEAD
```

Benchmark: fake driver



Benchmark: real driver

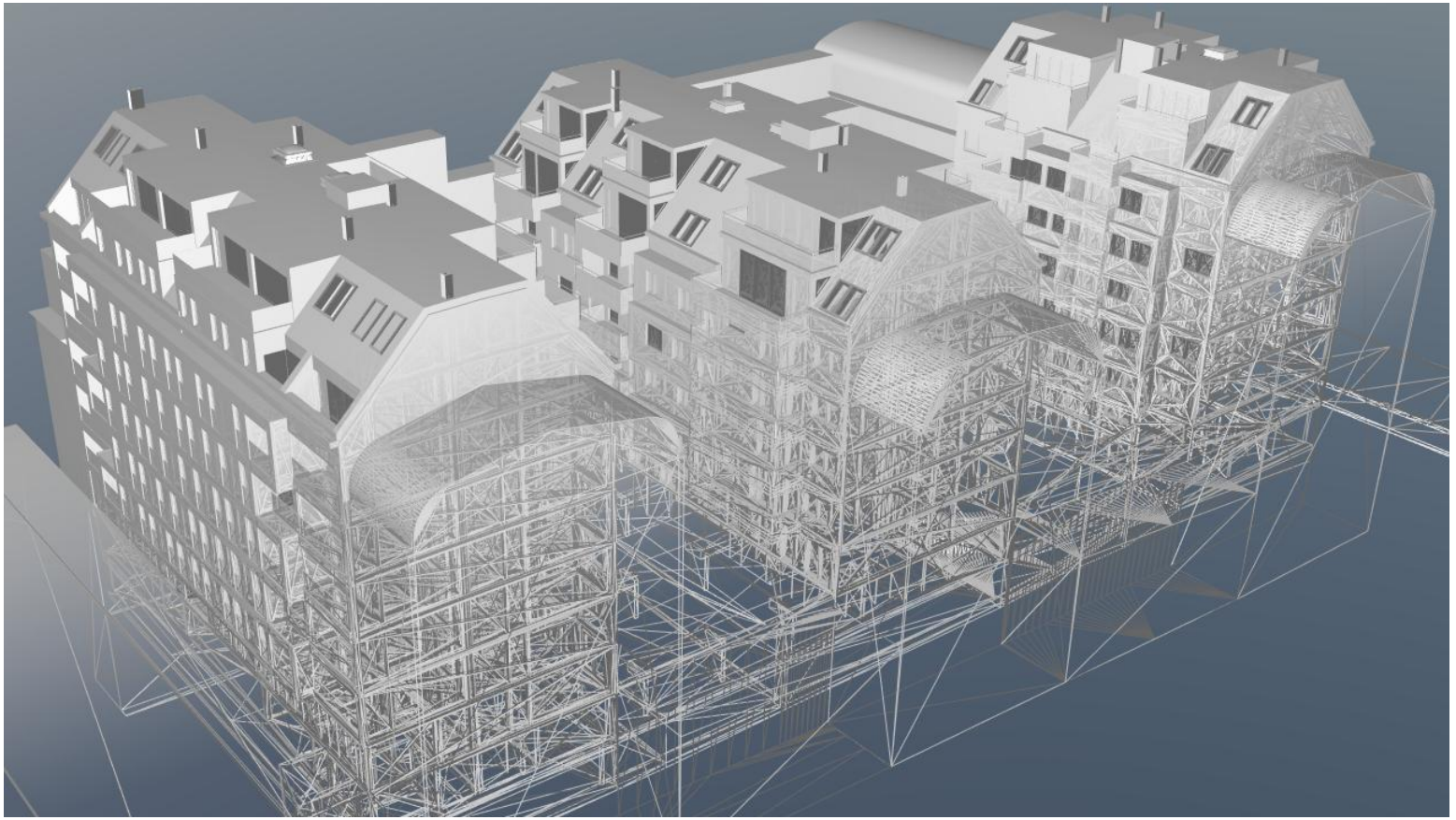




(Average relative framerate)

Benchmark: other engines

Architecture (7022 objects)

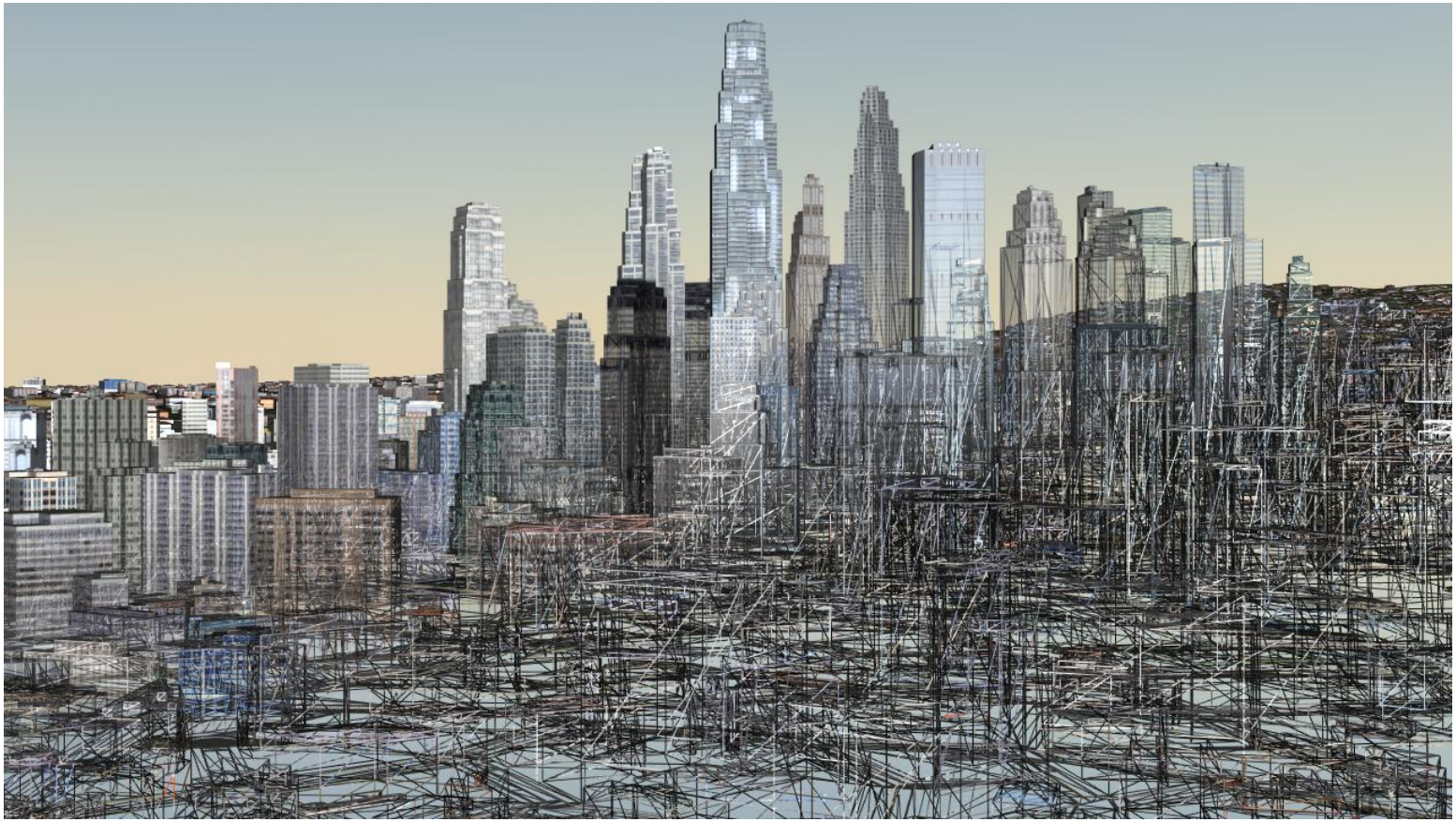


Benchmark: other engines

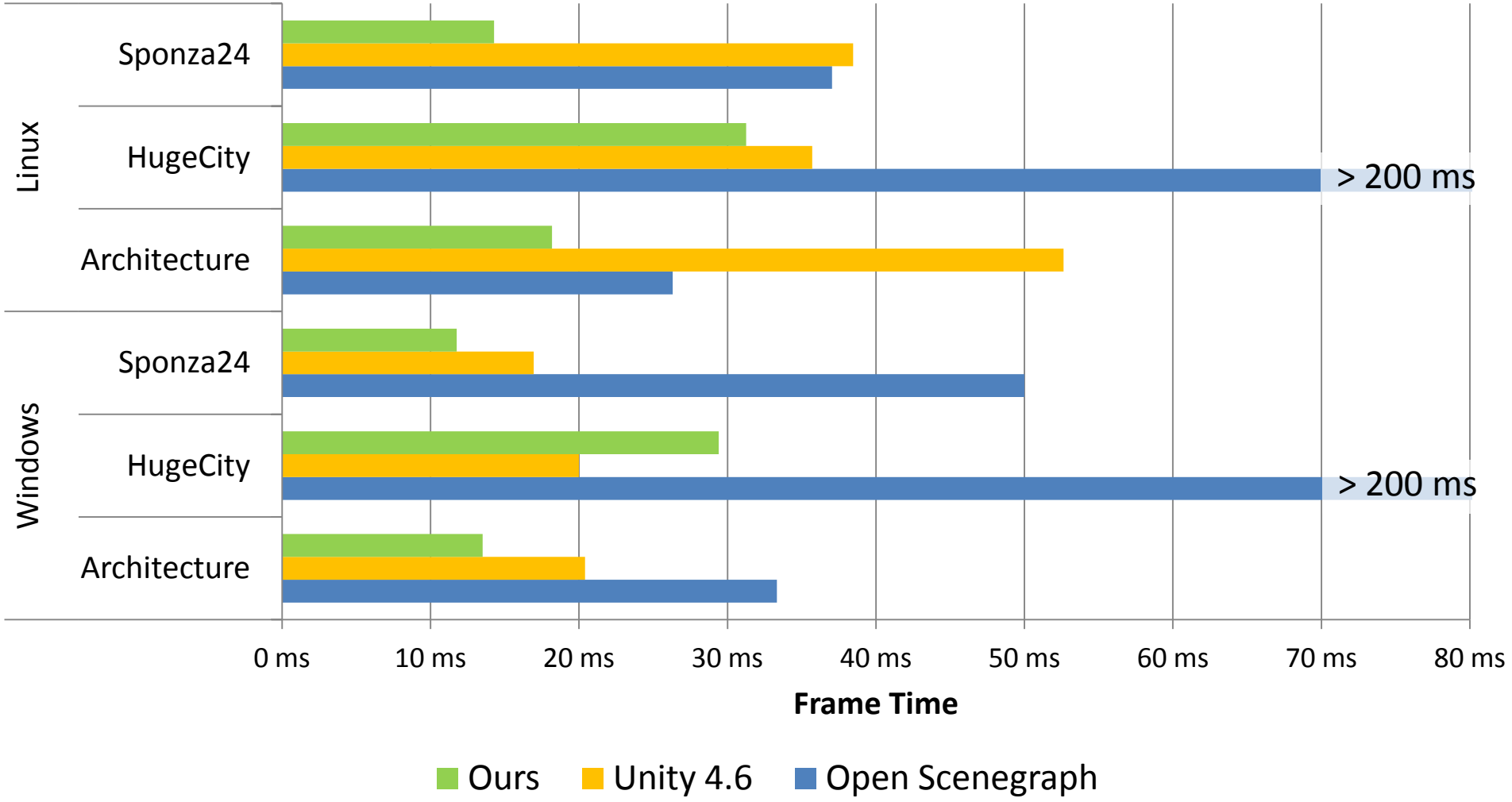
Sponza24 (9408 objects)

Benchmark: other engines

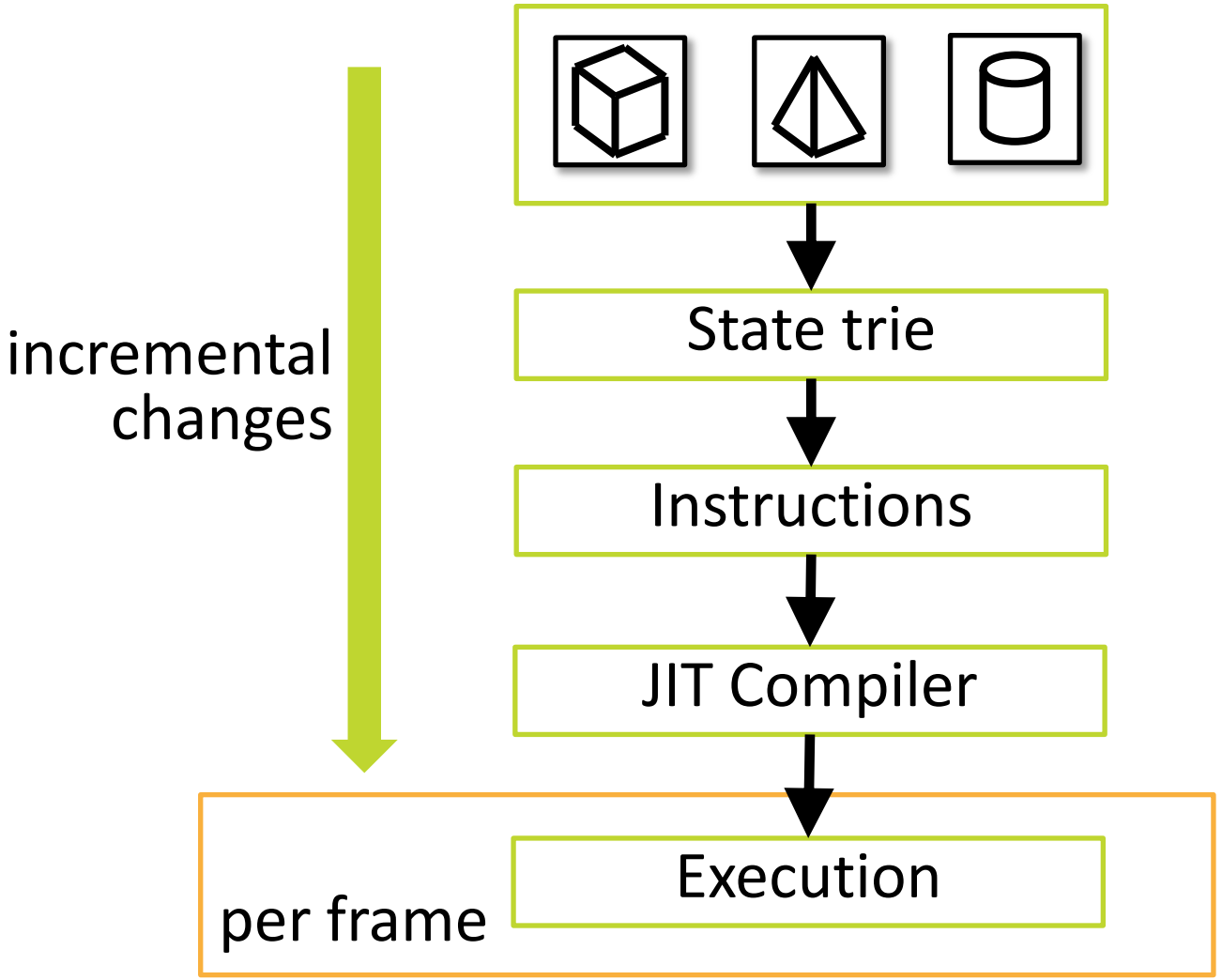
HugeCity (6580 objects)



Benchmark: other engines



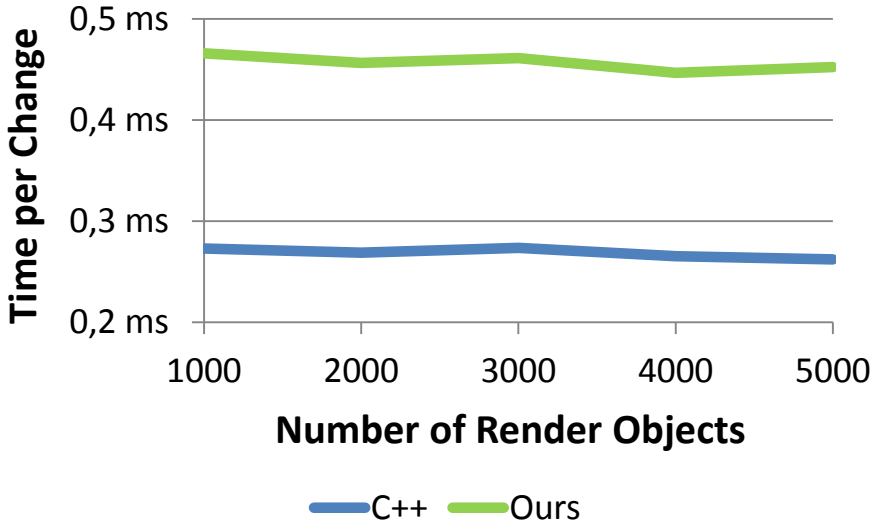
What we achieved



Costs & Outlook

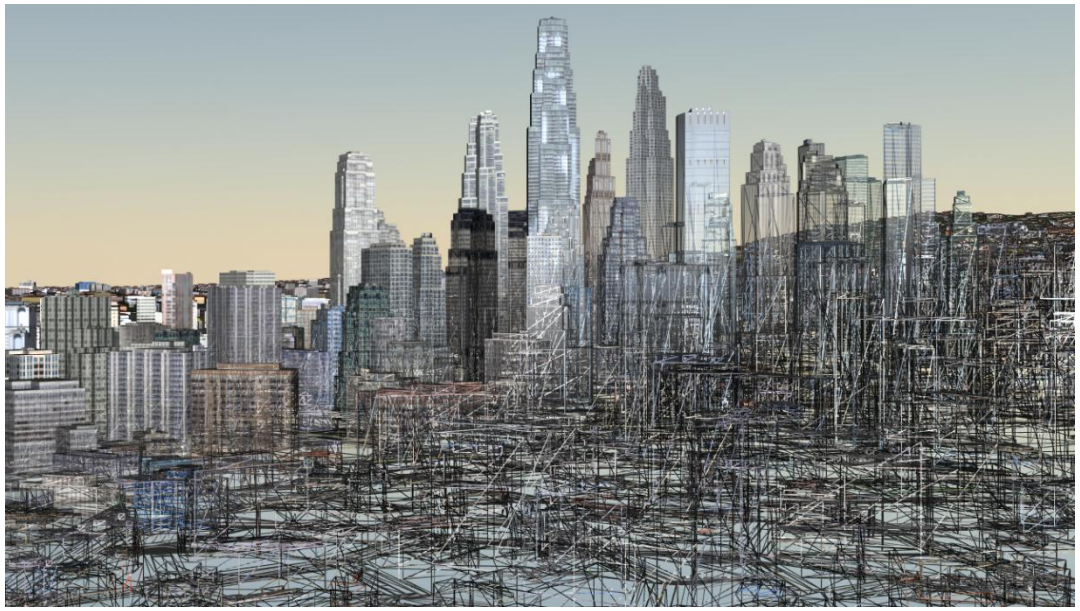
Costs

- Compilation overhead
  still more than 2000 changes/s
- Platform dependent compiler



Outlook

- NV_command_lists, Mantle,
  DirectX 12, etc.
- SceneGraph frontend
- Driver integration anyone? ☺
- Order dependent rendering



OPEN SOURCE

Amazing applications and research prototypes are powered by the Aardvark visual computing platform.

A selected set of Aardvark libraries is open source and available on https://github.com/vrvis.

CONTACT

Visit our website: http://www.vrvis.at/projects/aardvark

Email Stefan Maierhofer (Aardvark Project Manager): sm@vrvis.at

Thank you for your attention!

Please visit us at

http://www.VRVis.at/

vrvis