Tile Pair-Based Adaptive Multi-Rate Stereo Shading

Yazhen Yuan Rui Wang Hujun Bao

State Key Lab of CAD&CG
Zhejiang University, Hangzhou, China

Real-time Rendering in industry

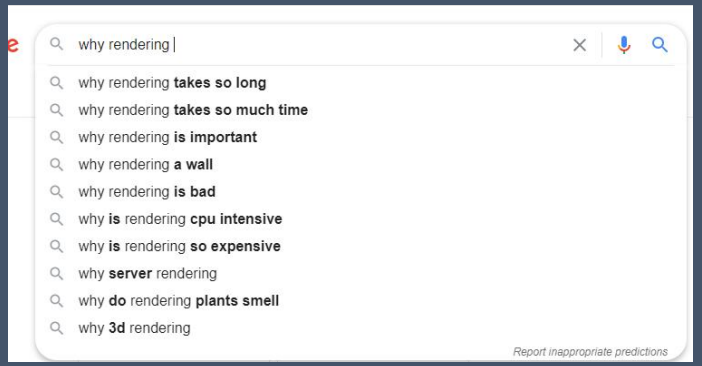
Applications

Engine

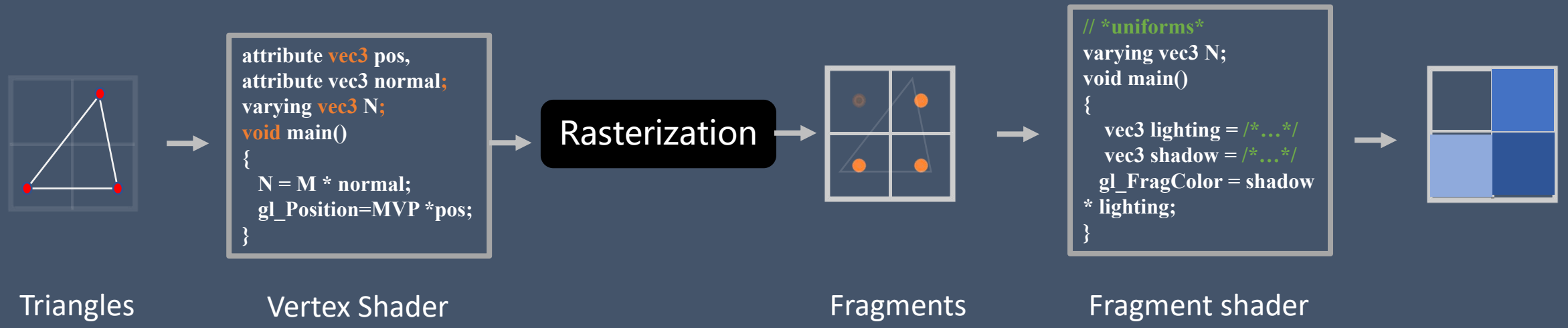
Platform

API

Real-time rendering pipeline



Traditional Real-time rendering pipeline



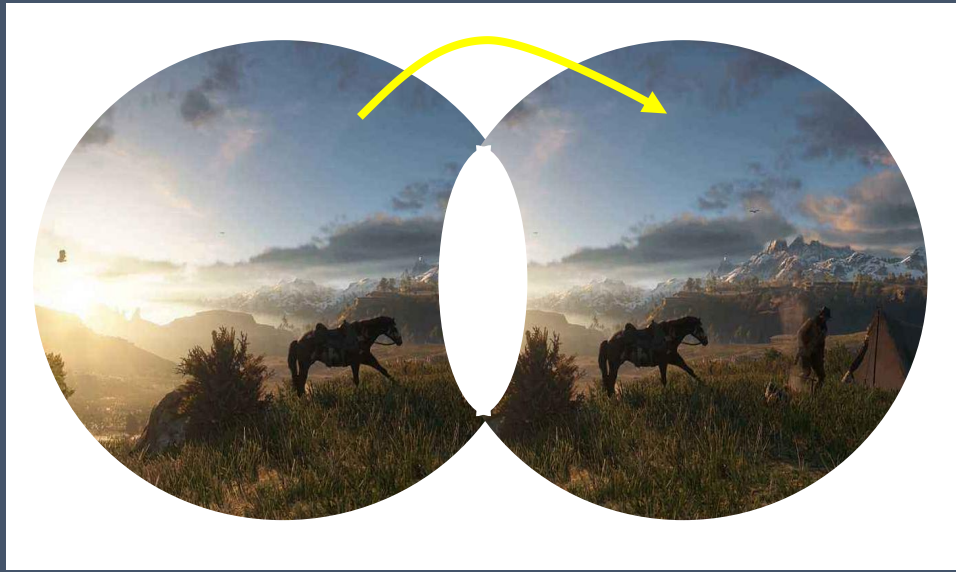
Rendering For VR/AR is Challenging

- Stereo Shading
  - 2x shading work
- Higher FPS
  - 90hz and more
- Mobile!!!

Optimizations in Real-time rendering pipeline

- Utilize existed rendering pipeline/shader stage
  - More efficient shader
  - Multiple Pass:
    - Deferred rendering, pre-z.
- Optimize rendering pipeline
  - Tile-based rendering
  - Variable rate shading
  - Multi-Res shading
- Extend or invent new rendering pipeline
  - Multi-rate shading
    - He et al 2014, AMFS 2014
  - RTX

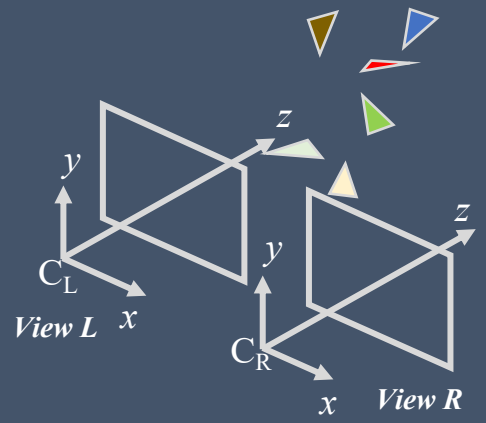
Previous Optimizations in Stereo Rendering



- Extending pipeline with single pass
  - [De sorbier et al. 2014]
  - Single pass stereo, NVIDIA
  - Not reuse shading
- Project shading among views
  - [Fu et al. 1996]
  - [Fhel 2004]
  - [Bulbul et al.2010]
  - Shading incoherence
  - Suffered from holes

Our Goal

-- A new dedicated rendering pipeline for VR/AR



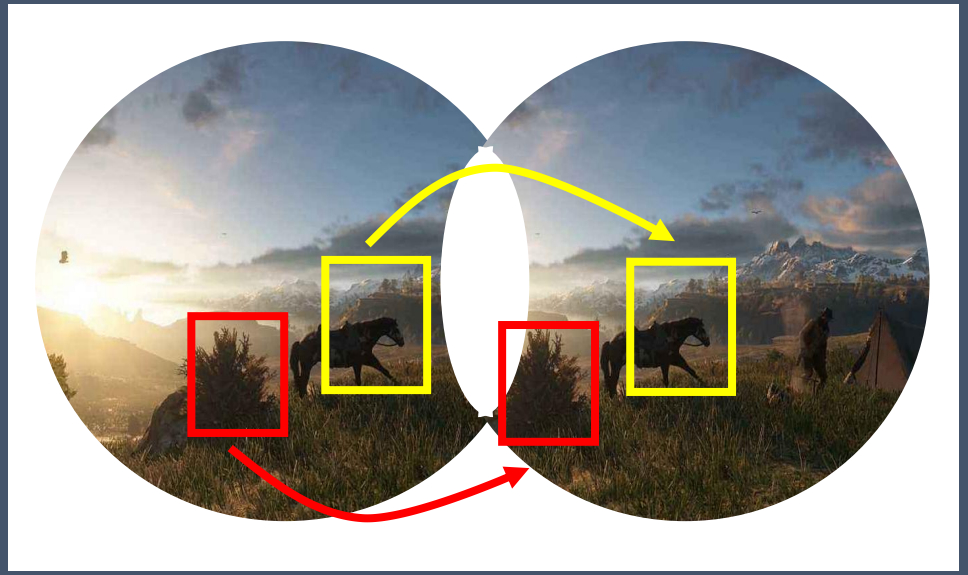
- Designed for mobile GPU
- Single pass

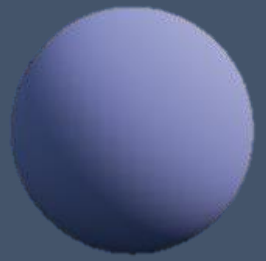
Reduce computation cost

- Less I/O access
- Less shading cost

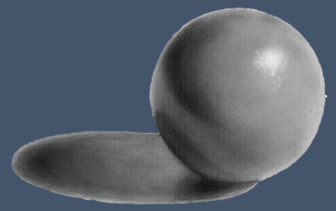
Motivation #1

--Reuse low rate shading





Diffuse



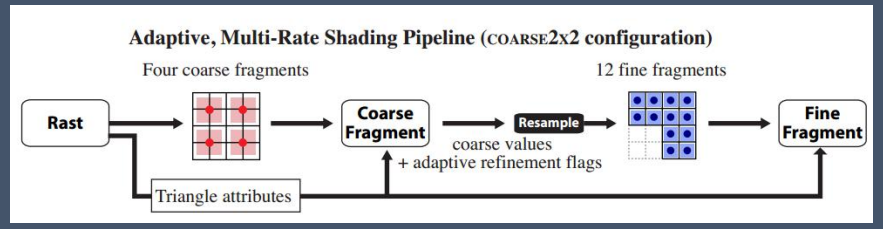
Shadow

Specular

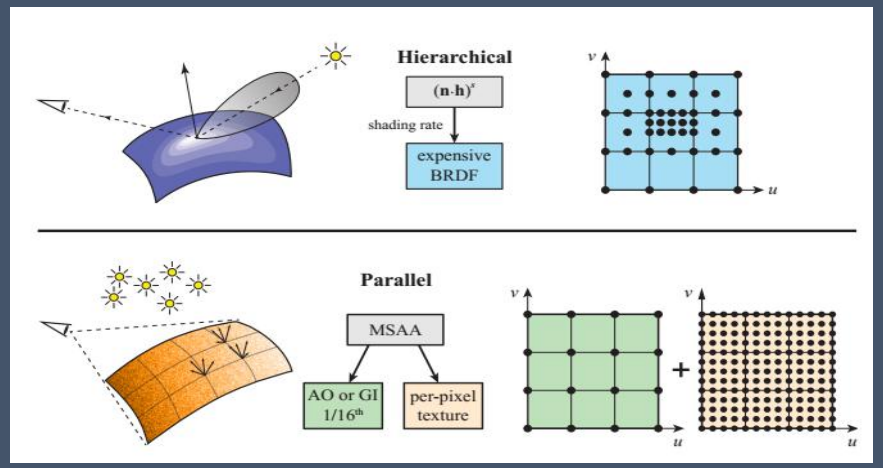
(far from reflection low freq. region)

Motivation #1
--Reuse low rate shading

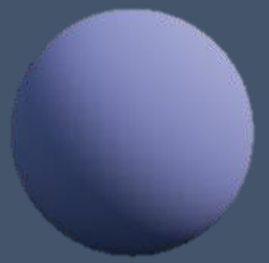
Previous adaptive multi-rate shading pipelines



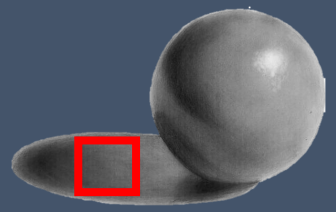
He et al. 2014



Clarberg et al. 2014



Diffuse



Shadow
(umbra region)

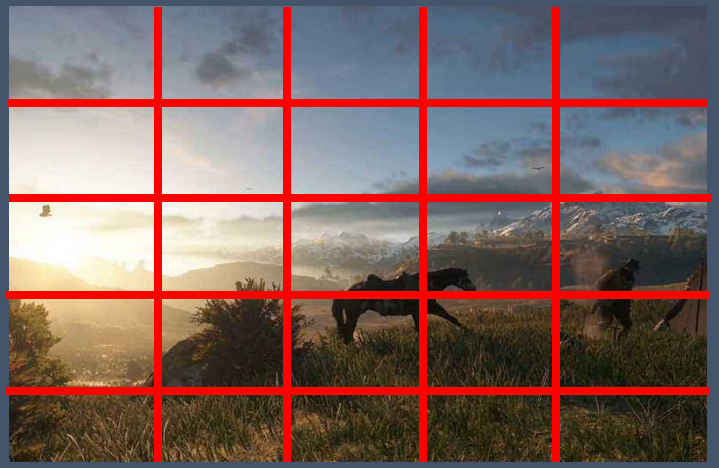
Specular

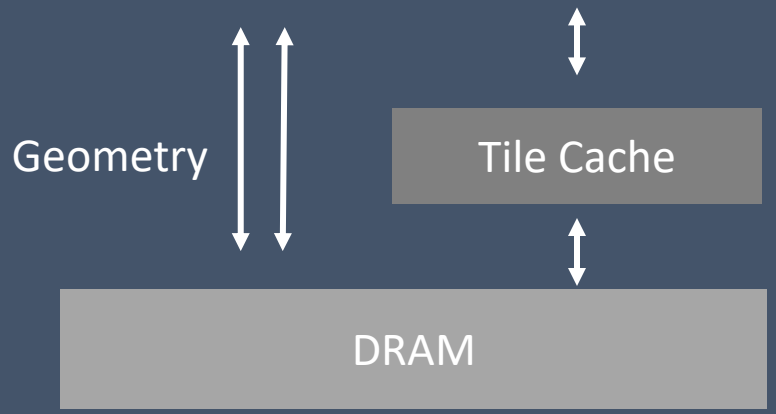
(far from reflection
low freq. region)

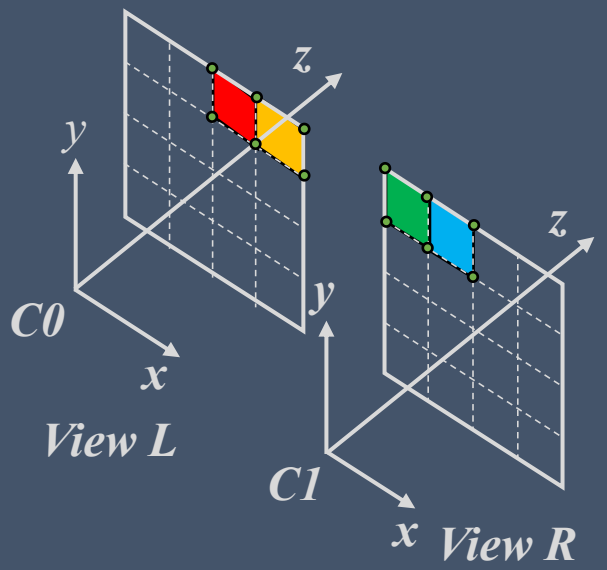
Motivation #2

-- break the gap of mobile cache

Tiled Based Rendering

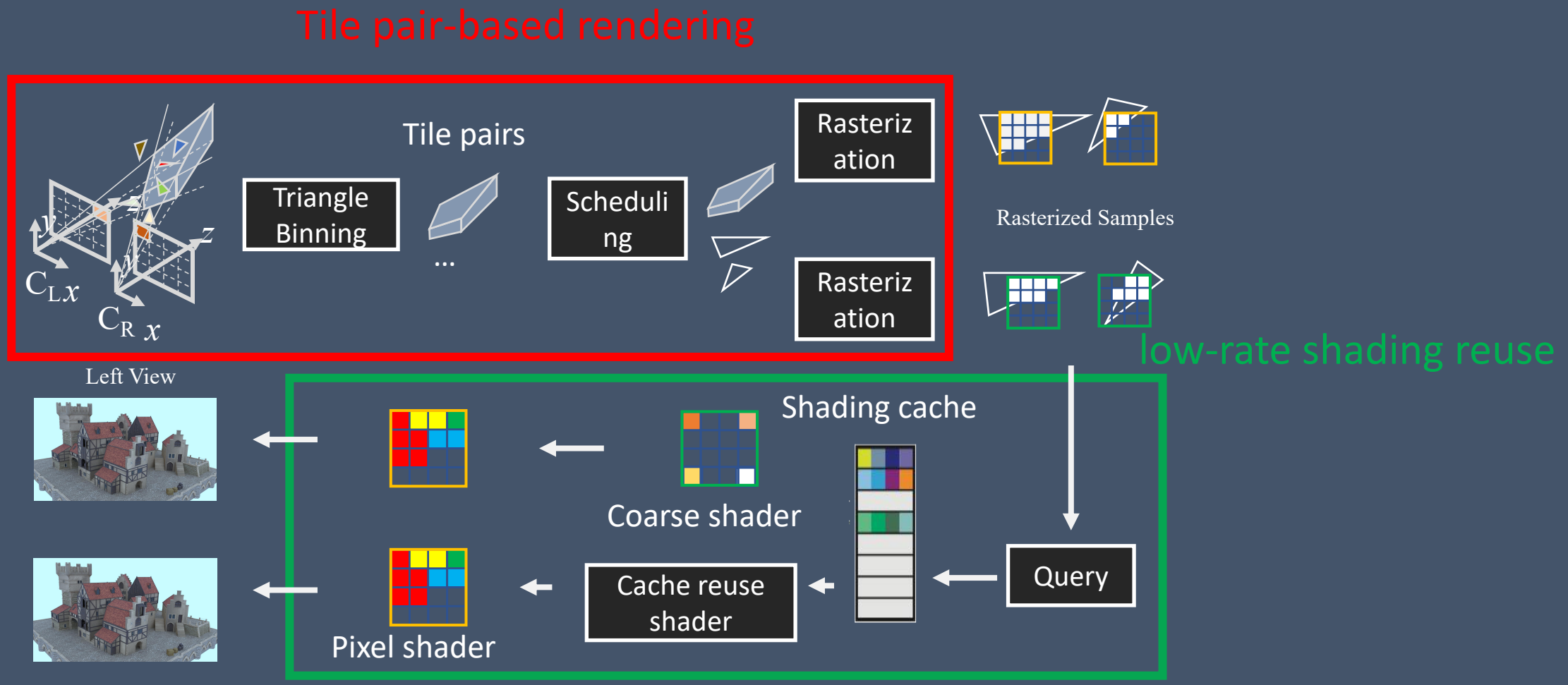






Avoid 2x scheduling and shading in VR rendering

Our Framework



Our Framework

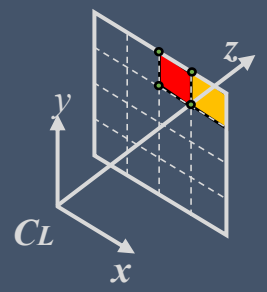
- Exploit data locality through **tile pair** structure
  - Tile pair triangle bin
  - Tile pair scheduling
- Multi-rate stereo view shading reuse
  - Shading cache and query
  - Cache reuse shader
- Evaluation

Our Framework

- Exploit data locality through **tile pair** structure
  - Tile pair triangle bin
  - Tile pair scheduling
- Multi-rate stereo view shading reuse
  - Shading cache and query
  - Cache reuse shader
- Evaluation

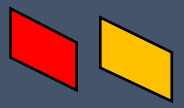
Tile Pair-based Rendering

-- Data locality through tile



Triangle Binning

Tiles

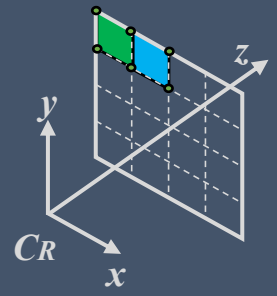


Scheduling

Rasterization

Shading

Left View



Triangle Binning

Scheduling

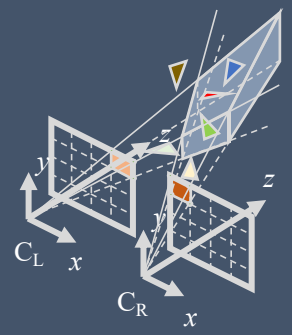
Rasterization

Shading

Right View

Tile Pair-based Rendering

-- Data locality through tile pair



Triangle Binning

Tile pairs

...

Scheduling

Rasterization

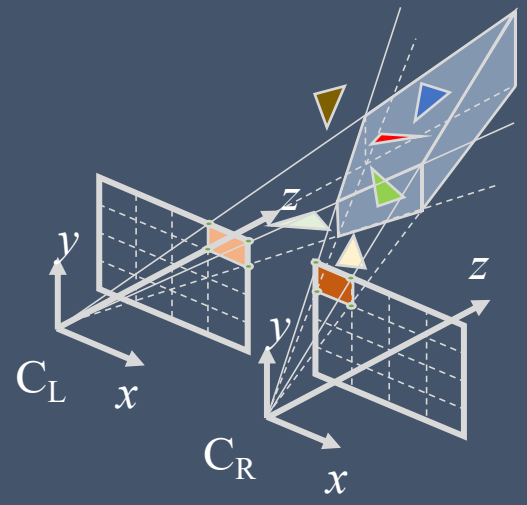
Shading

Left View

Rasterization

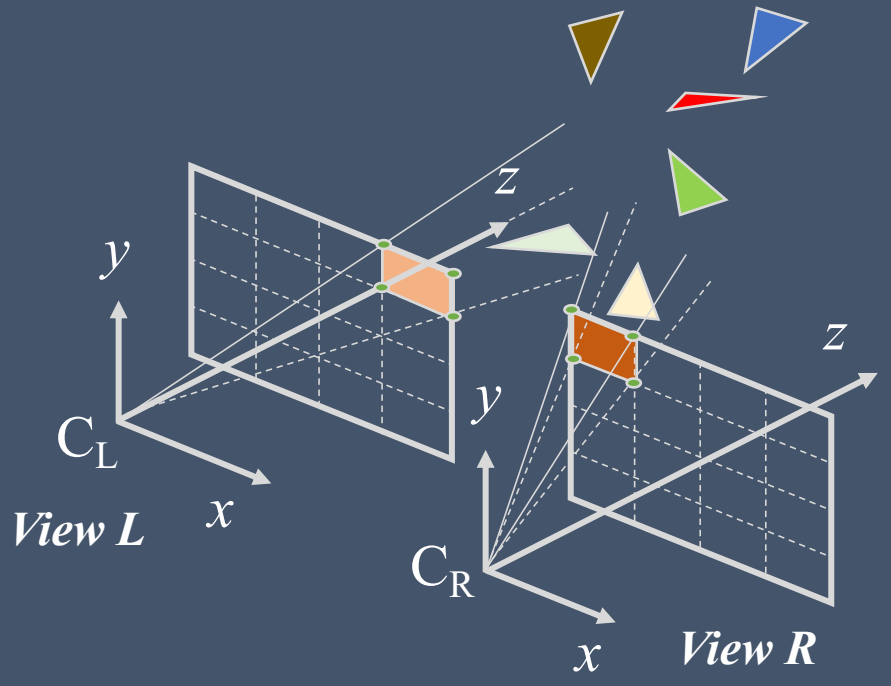
Shading

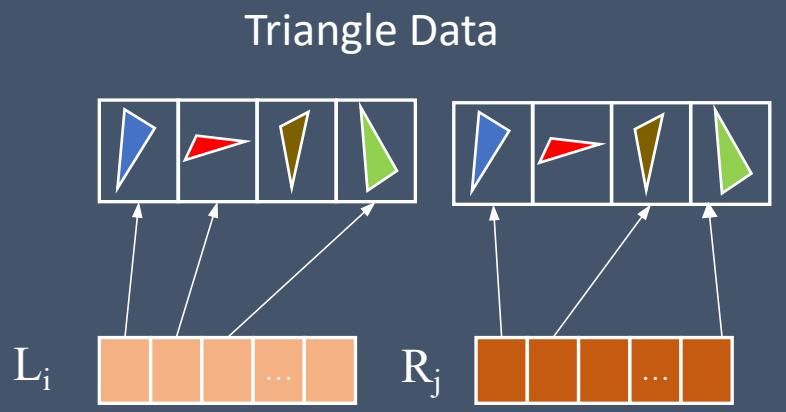
Right View



Tile Pair-based Rendering

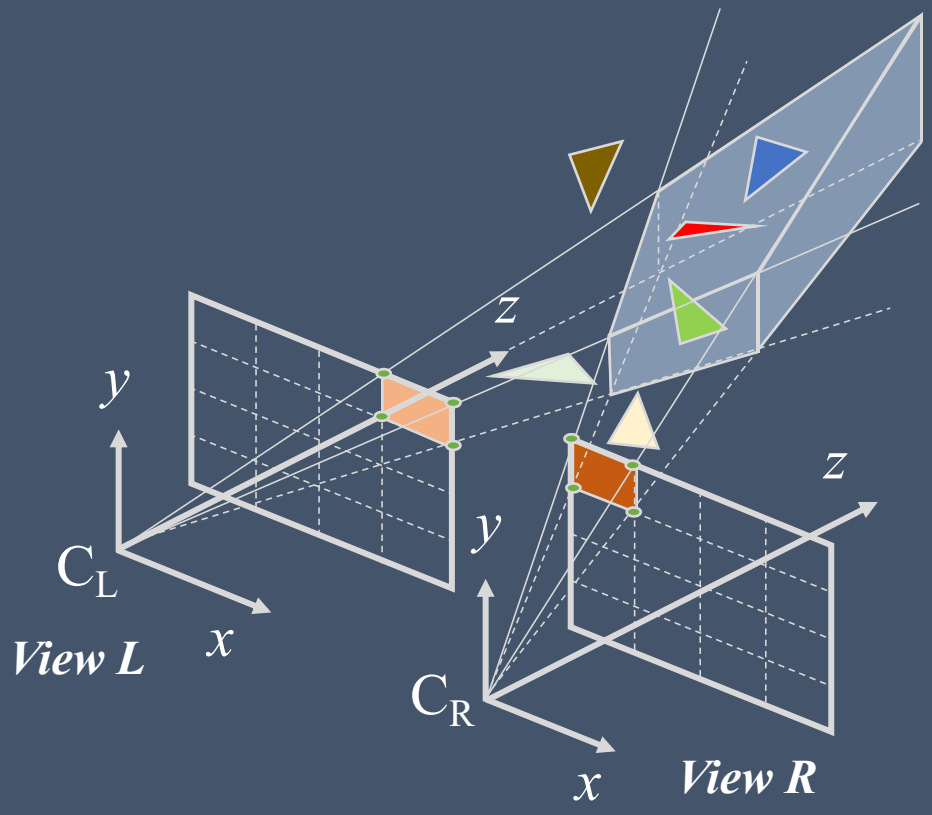
-- Triangle tile list

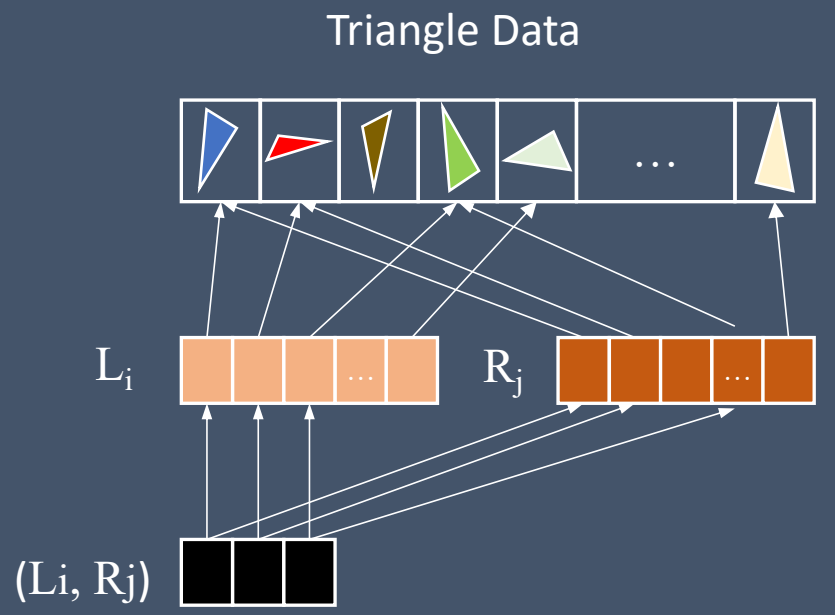




Tile Pair-based Rendering

--Triangle tile pair list



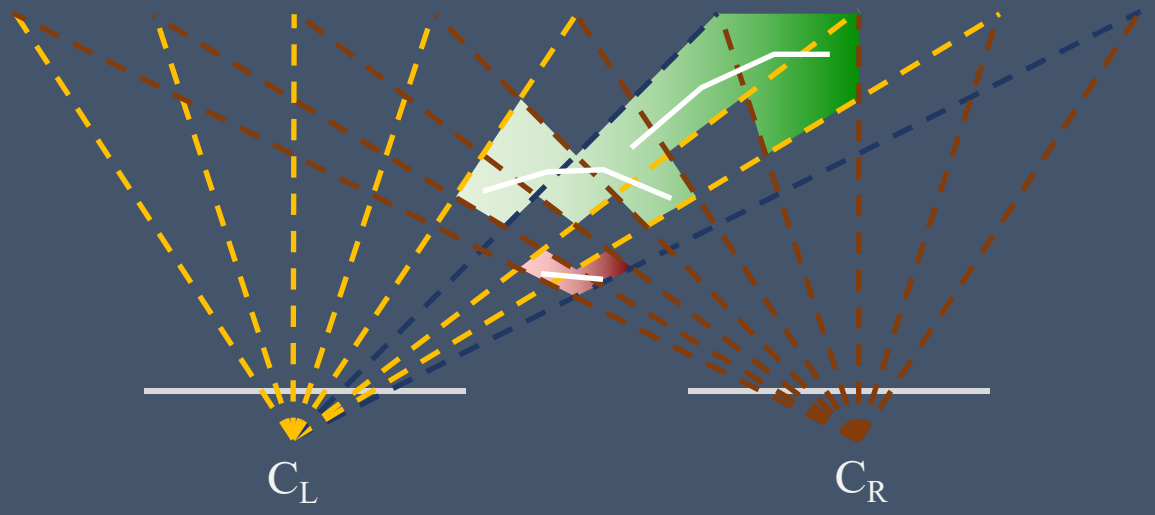


Triangle tile pair list

Tile Pair-based Rendering

-- Scheduling tile pairs

View L	…	L_6	L_7	L_8	L_9	L_{10}	L_{11}	L_{12}
View R	R_1	R_2	R_3	R_4	R_5	R_6	R_7	…

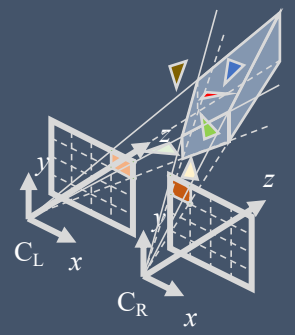


Sequence 1: $(L_{11}, R_1) \rightarrow (L_{12}, R_1) \rightarrow (L_{12}, R_2) \rightarrow (L_{11}, R_2) \rightarrow$ End

Sequence 2: $(L_9, R_2) \rightarrow (L_{10}, R_2) \rightarrow (L_{10}, R_3) \rightarrow (L_9, R_3) \rightarrow (L_{11}, R_3) \rightarrow (L_{11}, R_4)$

$\rightarrow (L_{10}, R_4) \rightarrow (L_{10}, R_5) \rightarrow (L_{11}, R_5) \rightarrow (L_{11}, R_6) \rightarrow (L_{10}, R_6) \rightarrow$ End

Tile Pair-based Rendering
--How to reuse?



Triangle Binning

Tile pairs

...

Scheduling

Rasterization

Rasterization

Shading

Shading

Left View

Right View

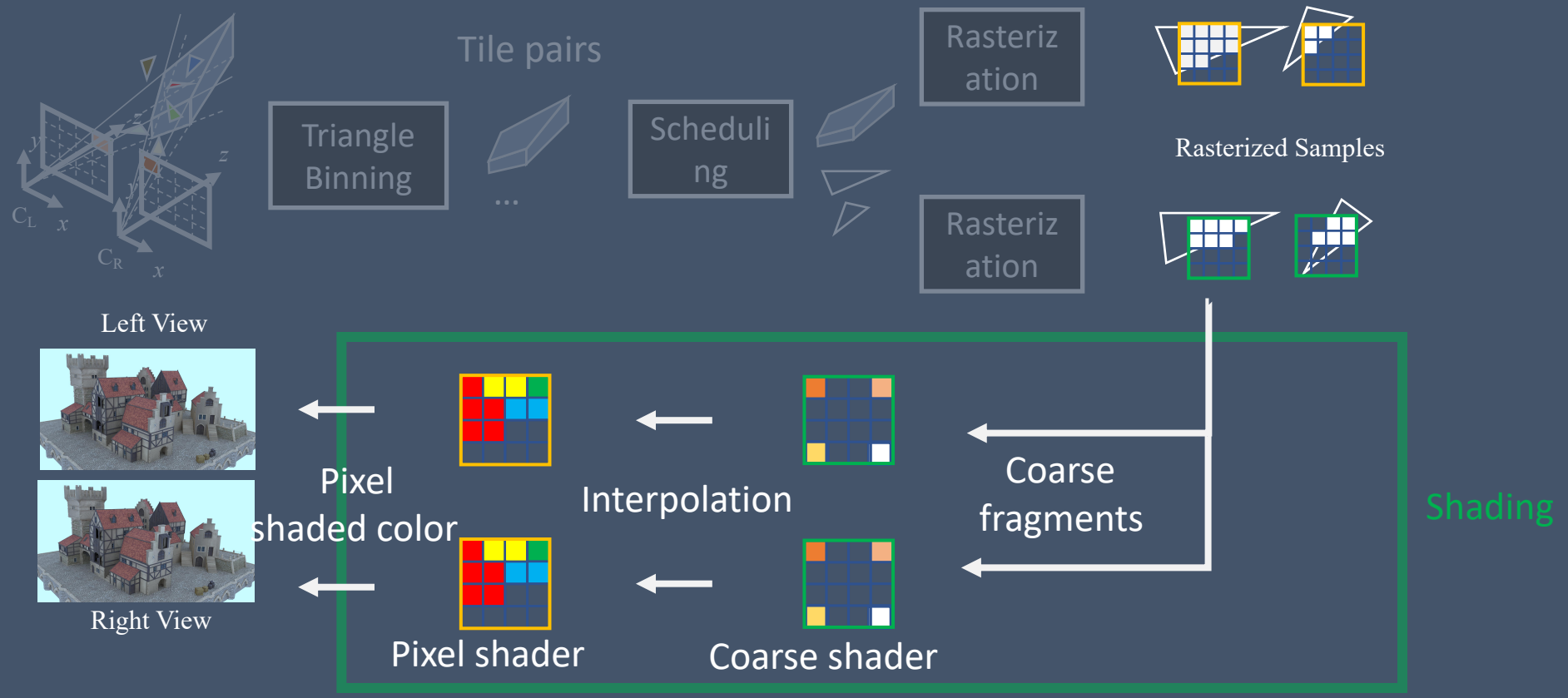
How to resue shading between views

Our Framework

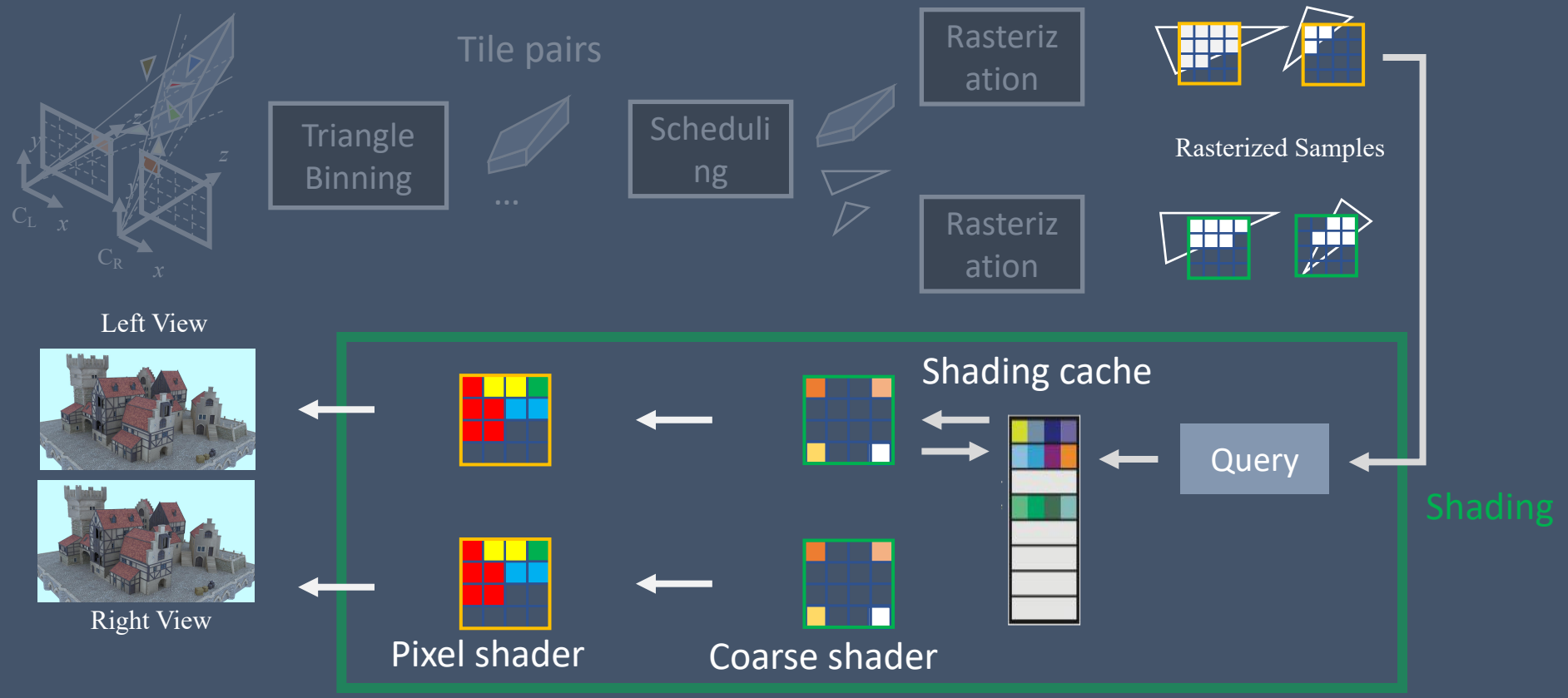
- Exploit data locality through **tile pair** structure
  - Tile pair triangle bin
  - Tile pair scheduling
- Multi-rate stereo view shading reuse
  - Shading cache and query
  - Cache reuse shader
- Evaluation

Our Shading Pipeline

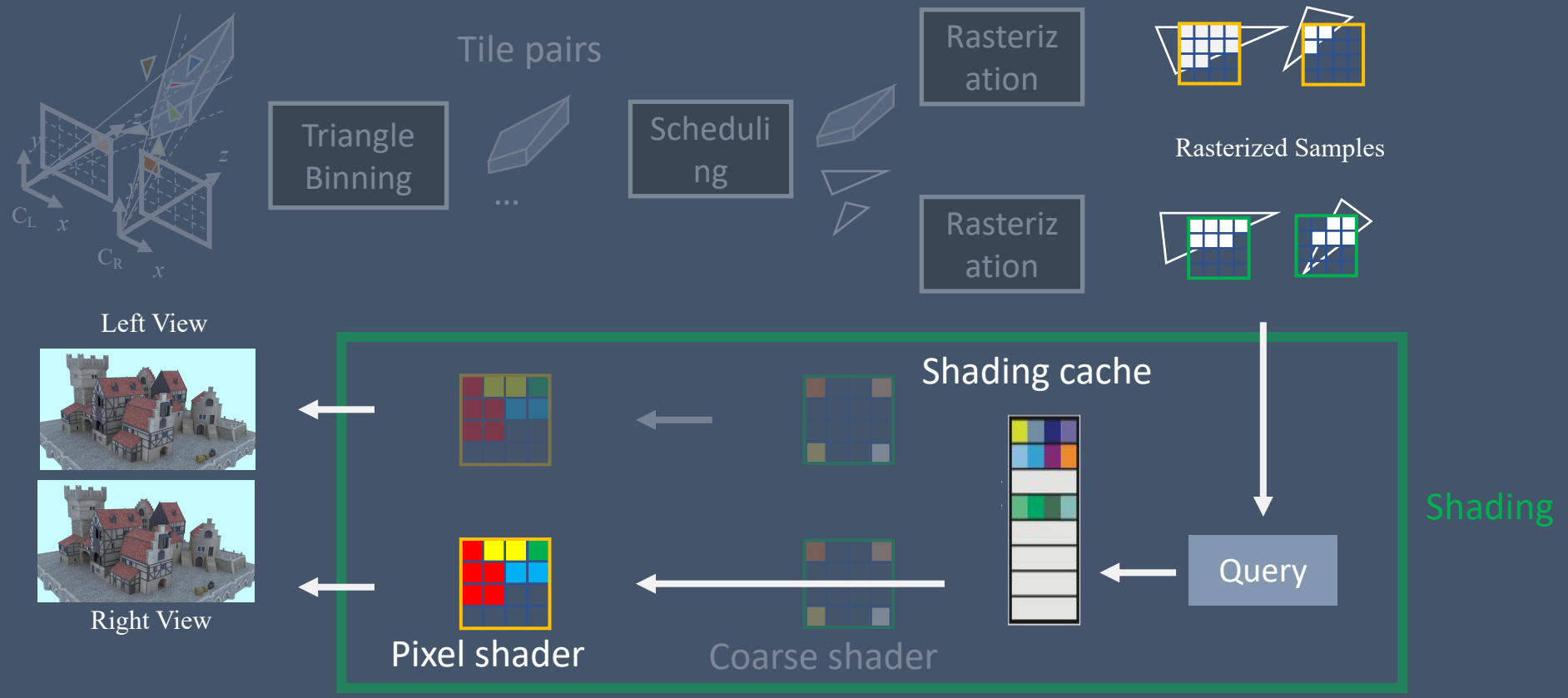
-- Multi-rate shading [He et al. 2014]



Our Shading Pipeline
-- Multi-rate stereo shading



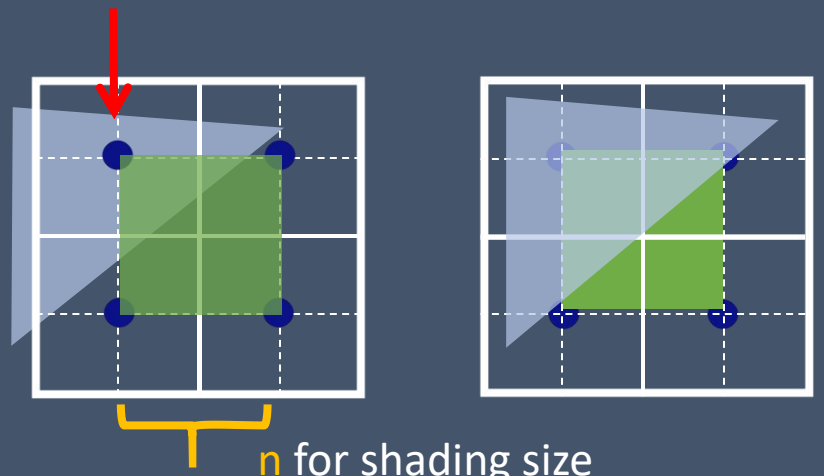
Our Shading Pipeline
-- Multi-rate stereo shading



Our Shading Pipeline
-- Shading key query

$$key = h(q, n, v)$$

q for screen position



n for shading size

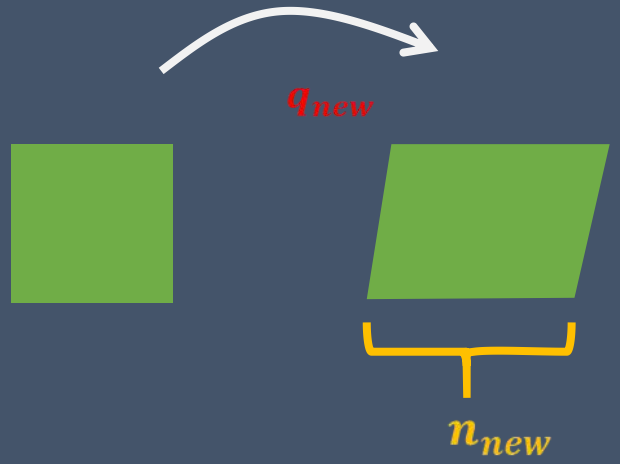
Left view

Right view

v for left or right view

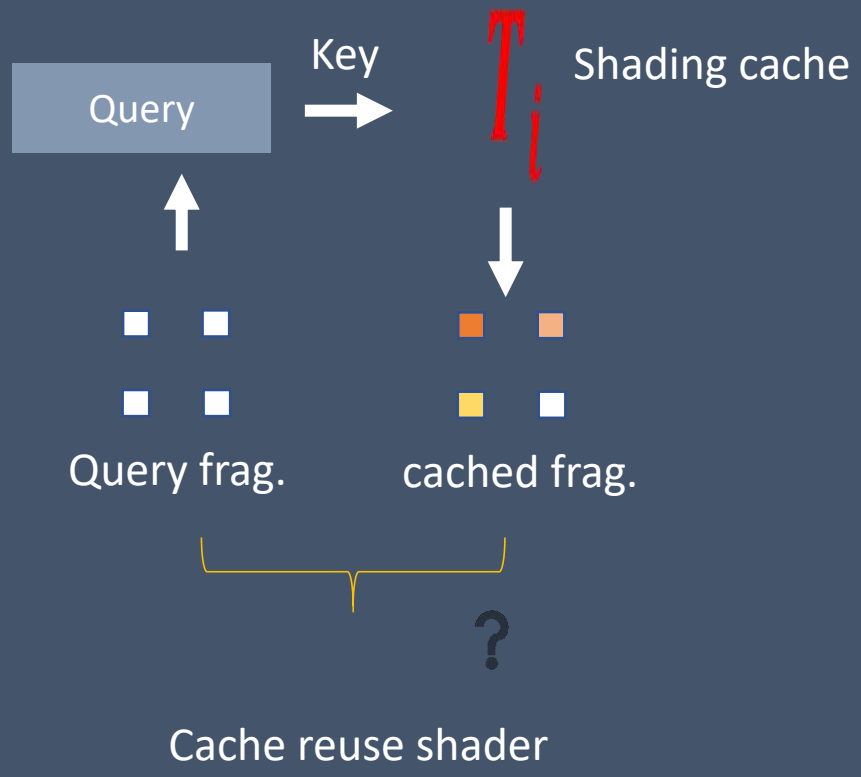
In case query failed,
project to query another view

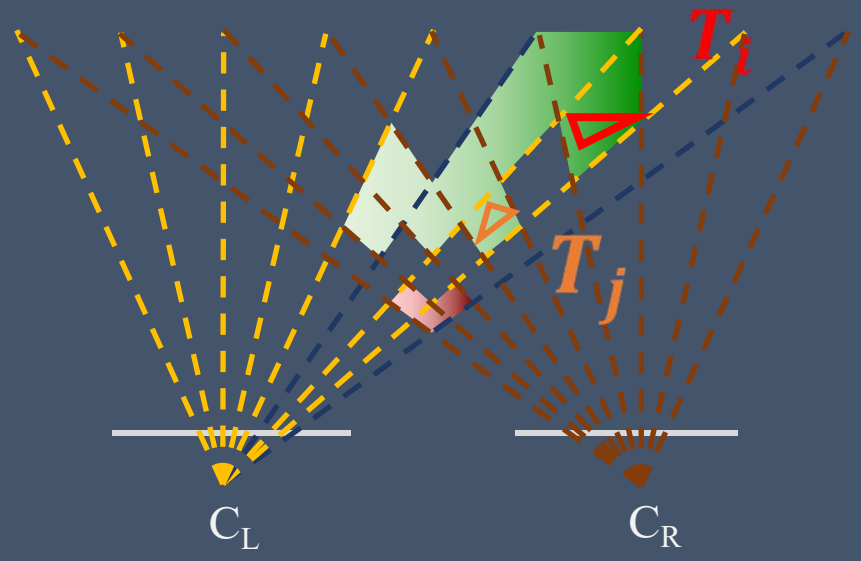
Project to another view



$$key_{new} = h(q_{new}, n_{new}, !v)$$

Our Shading Pipeline
--Cache reuse shader

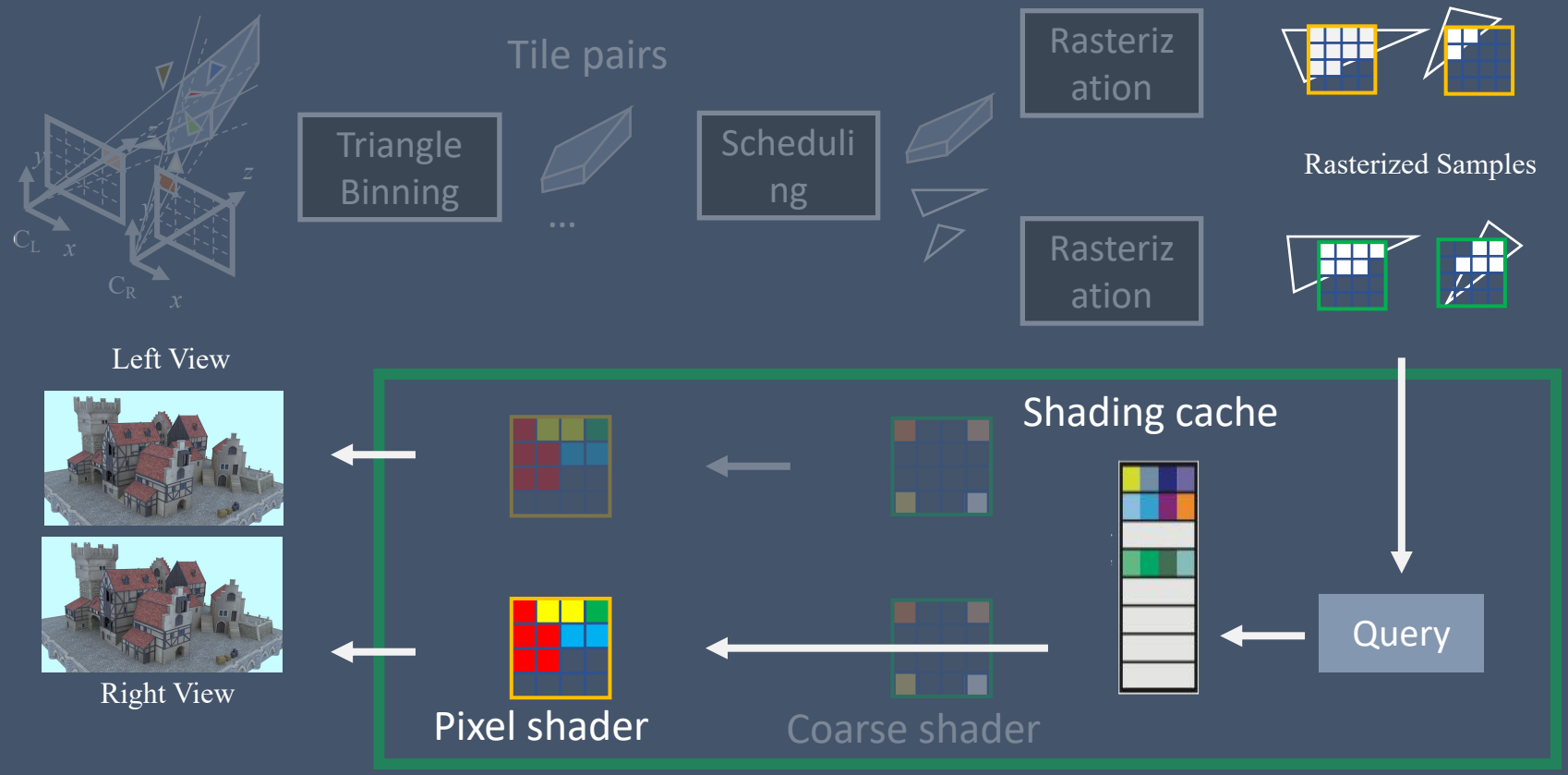




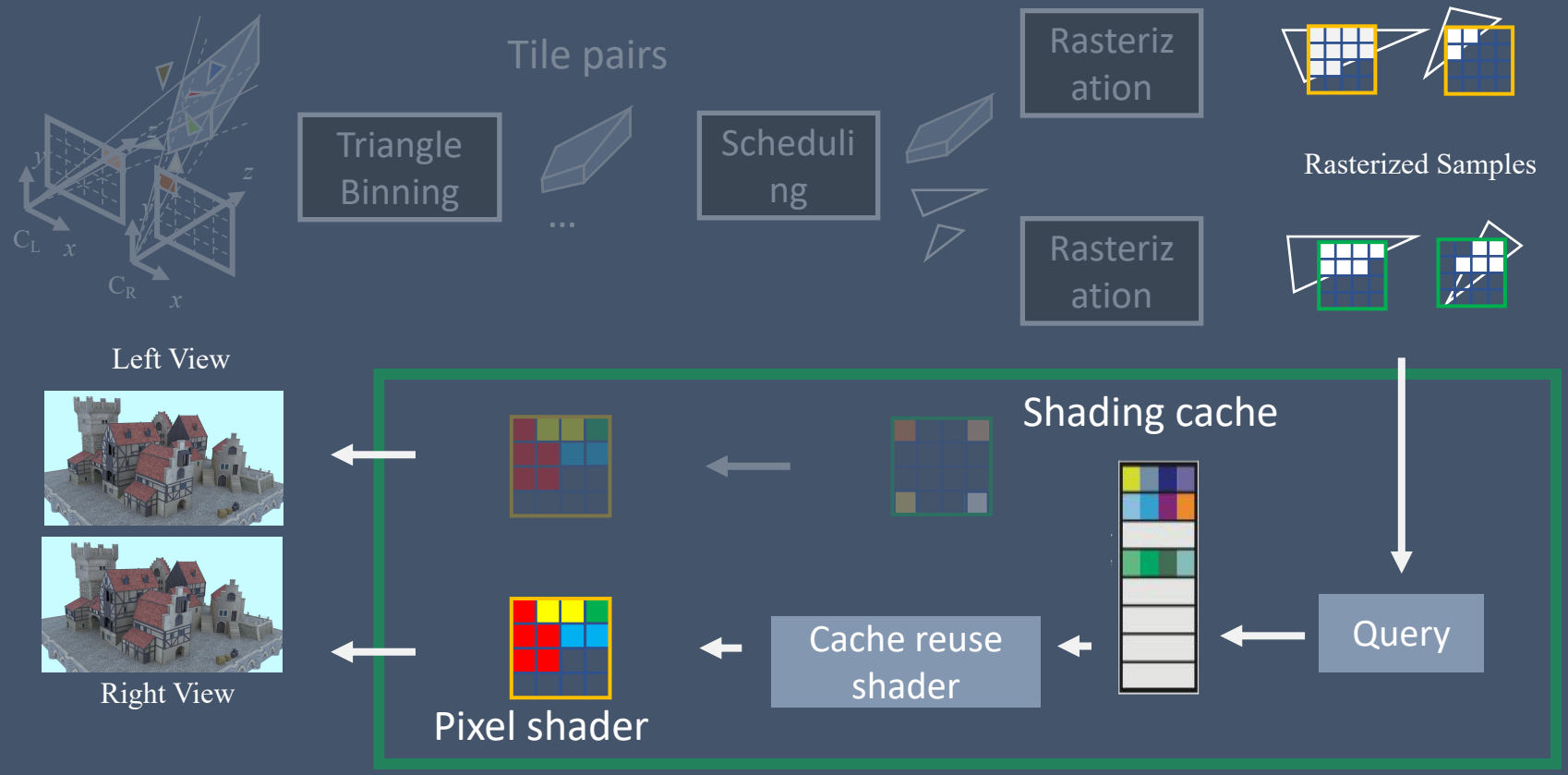
A simple example shader

```
bool reuse(frag query, frag cached){
    float3 dist = in.pos – cached.pos;
    return len(dist) < THRESHOLD;
}
```

Our Shading Pipeline
--Cache reuse shader



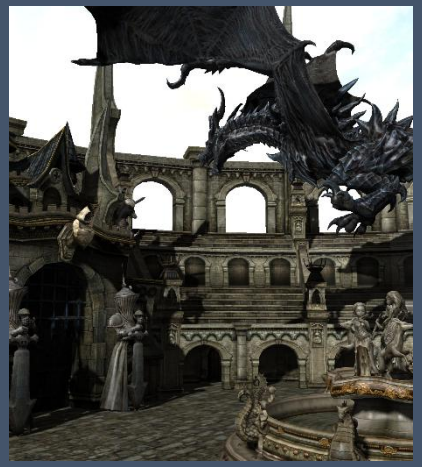
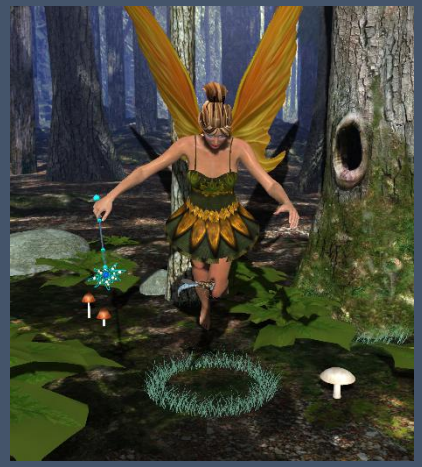
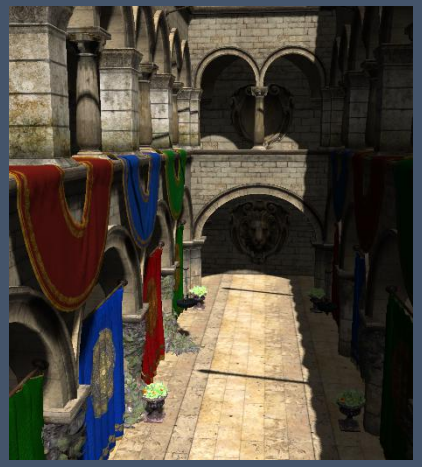
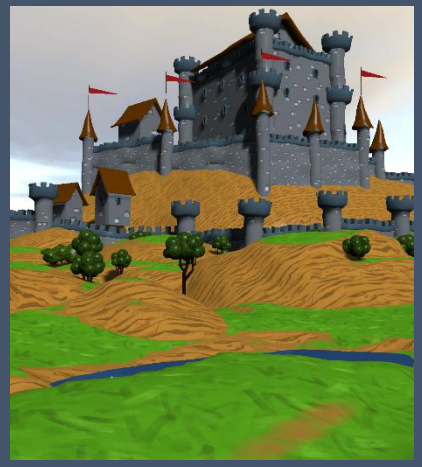
Our Shading Pipeline
--Cache reuse shader



Our Framework

- Exploit data locality through **tile pair** structure
  - Tile pair triangle bin
  - Tile pair scheduling
- Multi-rate stereo view shading reuse
  - Shading cache and query
  - Cache reuse shader
- Evaluation

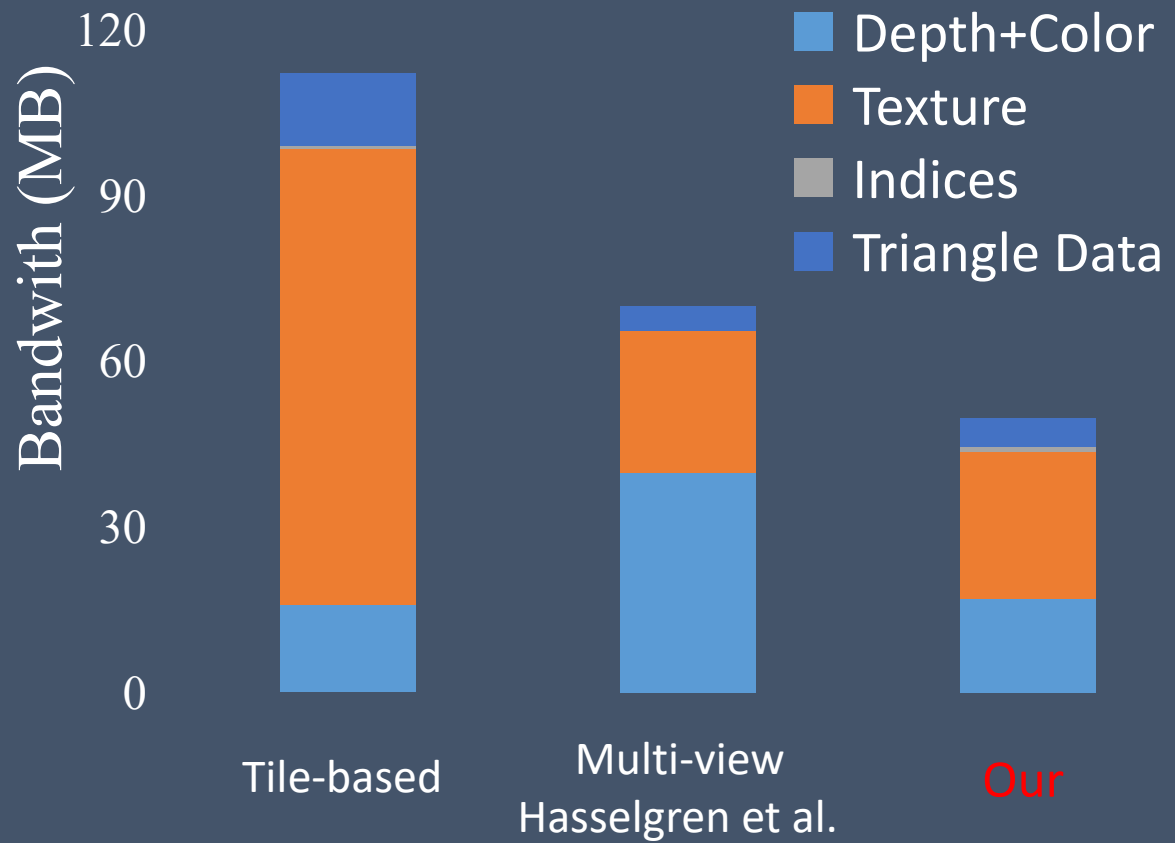
Evaluation

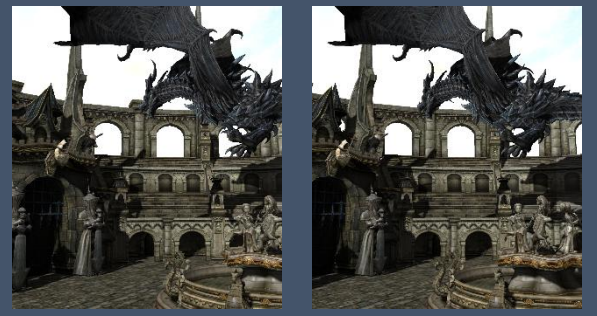





- Use software GPU simulator
  - on-chip bandwidth
  - per-pixel shading inst.
- Various scene configurations
  - Triangle size
  - Triangle area
  - Tessellation level
  - Tile size
  - Cache size

Bandwidth

Extensively use L1 cache (vertex, texture, depth/color).





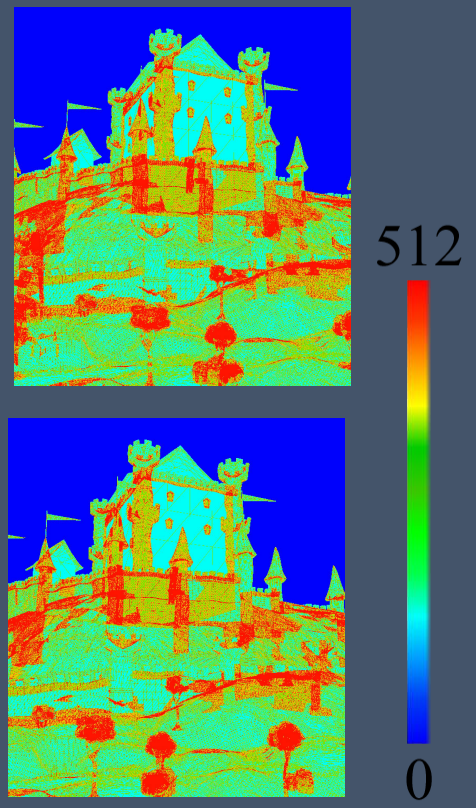
16*16 pixel^2
tile configurations

Please refer to our paper for more results!

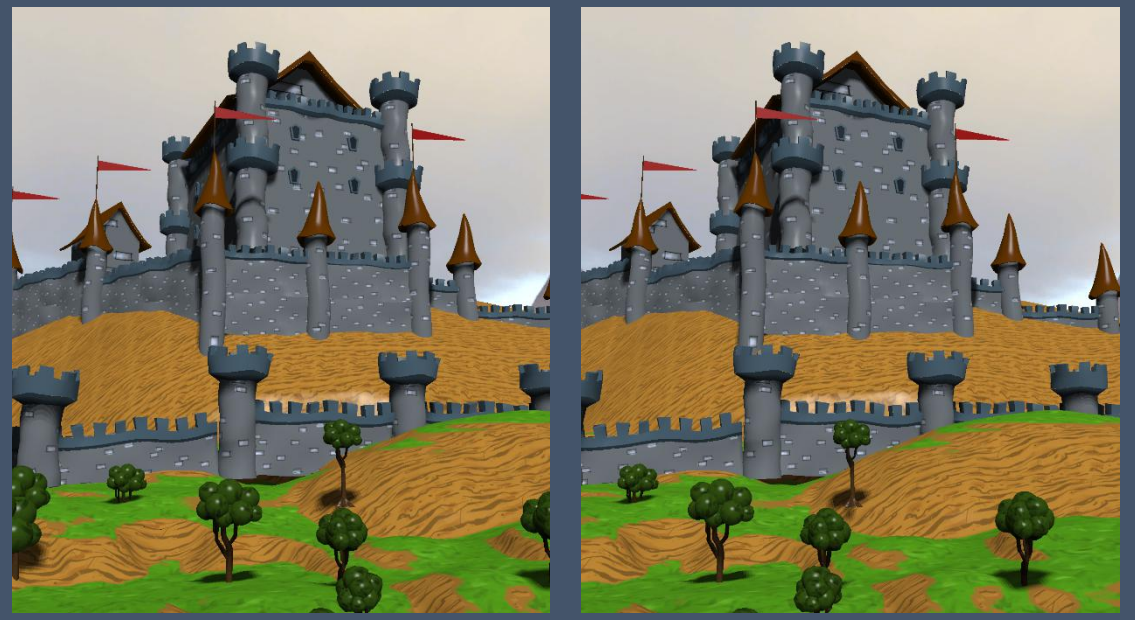
Shading Rates

Left

Right



Per-pixel shading 100%

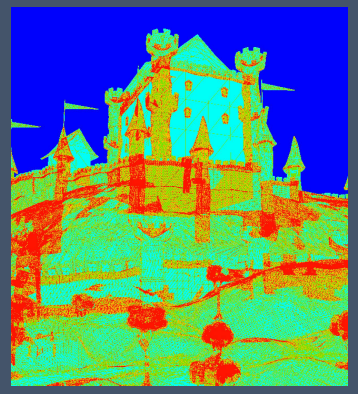


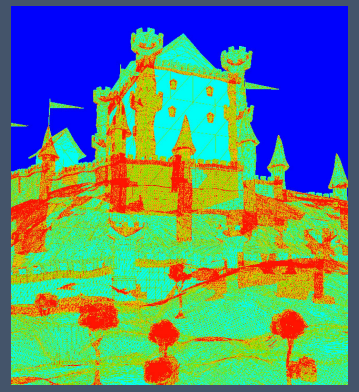
Castle is a tessellated scene
17 pixel^2/ per triangle.

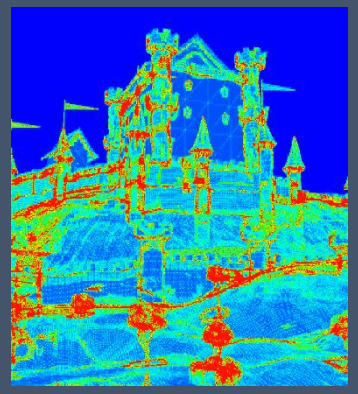
Shading Rates

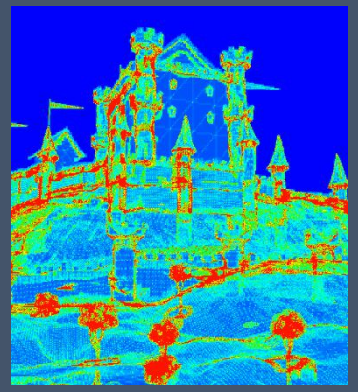
Left

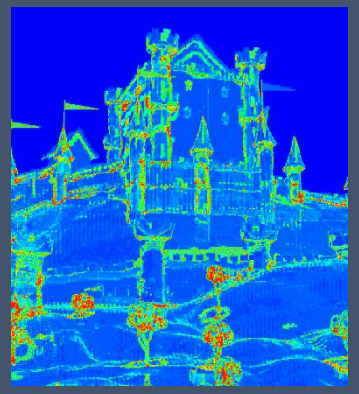
Right

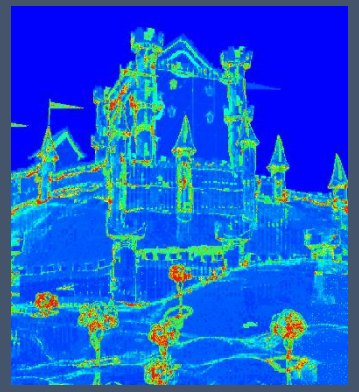












512

0

Per-pixel shading 100%

Multi-rate
He et al. 61%

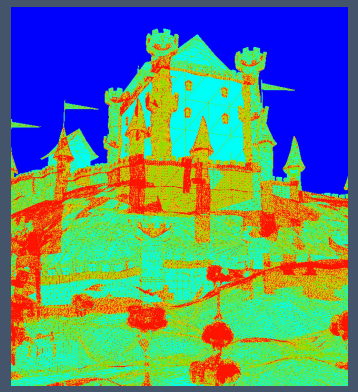
Our 27.6%

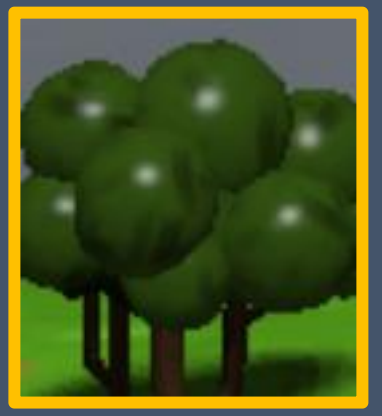
Image Quality

Shading image

Final Rendered image

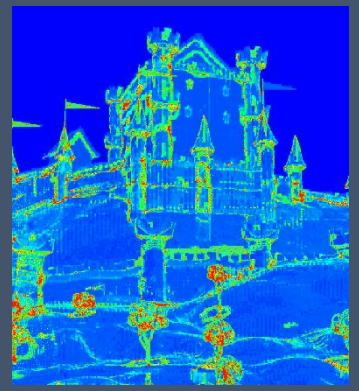
Per-pixel



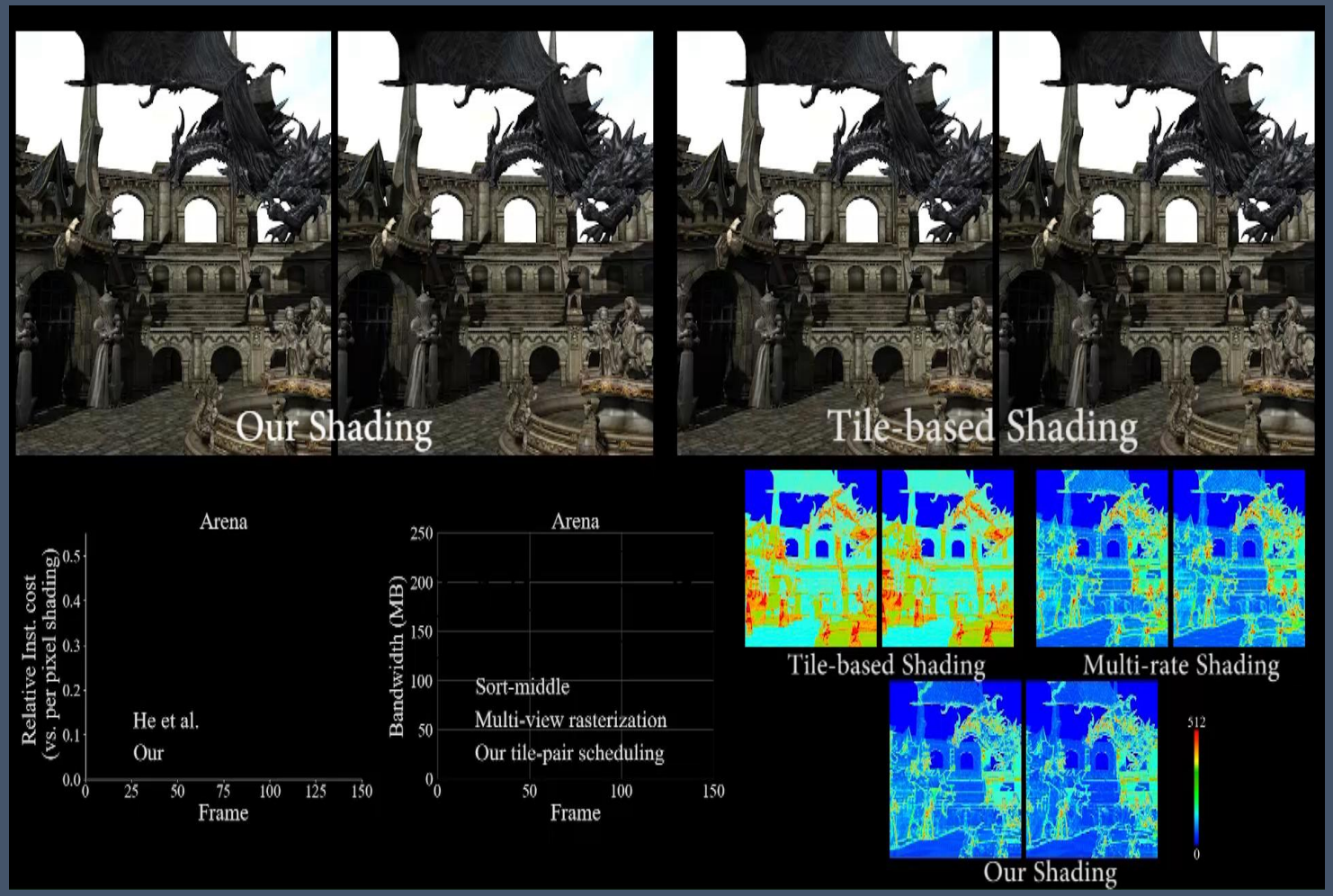


Our

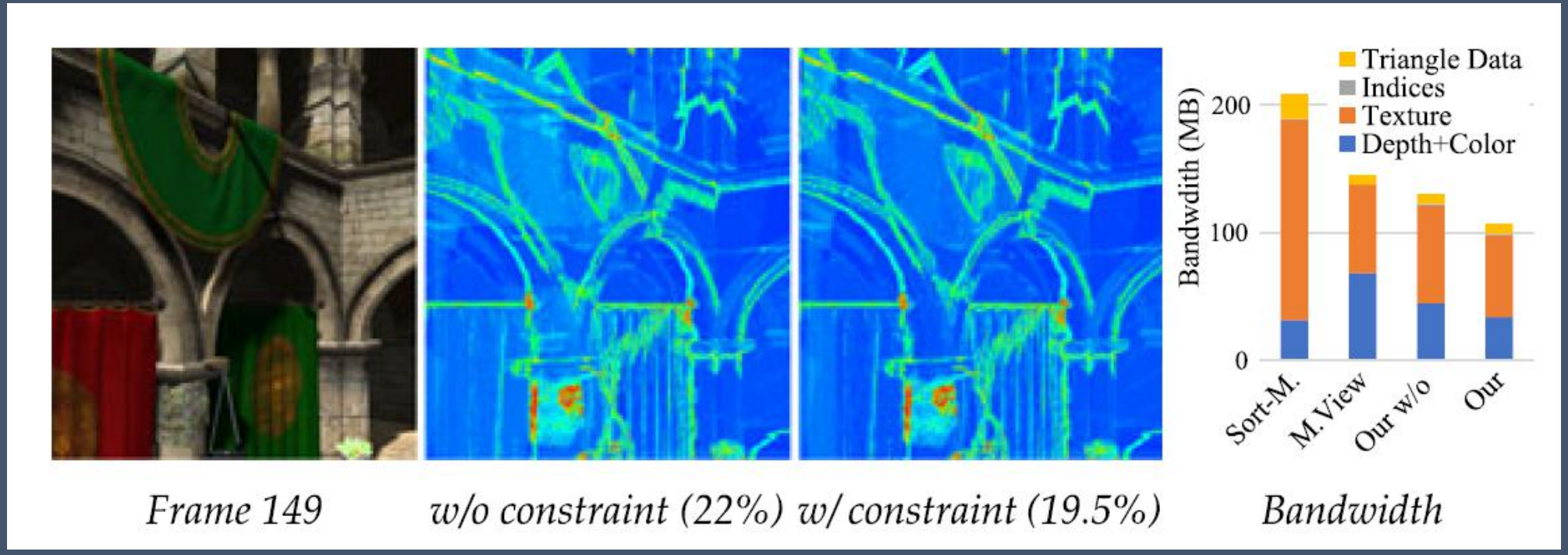
SSIM 99.98%



Animation


Our Shading
Tile-based Shading
Arena
Relative Inst. cost (vs. per pixel shading)
0.0
0.1
0.2
0.3
0.4
0.5
He et al.
Our
0
25
50
75
100
125
150
Frame
Arena
Bandwidth (MB)
0
50
100
150
200
250
Sort-middle
Multi-view rasterization
Our tile-pair scheduling
0
50
100
150
Frame
Tile-based Shading
Multi-rate Shading
512
0
Our Shading

Schedule


Triangle Data
Indices
Texture
Depth+Color
Bandwdith (MB)
200
100
0
Sort-M.
M.View
Our w/o
Our
Frame 149
w/o constraint (22%)
w/ constraint (19.5%)
Bandwidth

Conclusions

- Tile pair structure naturally maintains the data locality.
- It’s beneficial to replace costly shader computations with cheap cache reuse shader.

More Results Can Be Found In The Paper