Temporally Dense Ray Tracing

Pontus Andersson, 2019-07-08

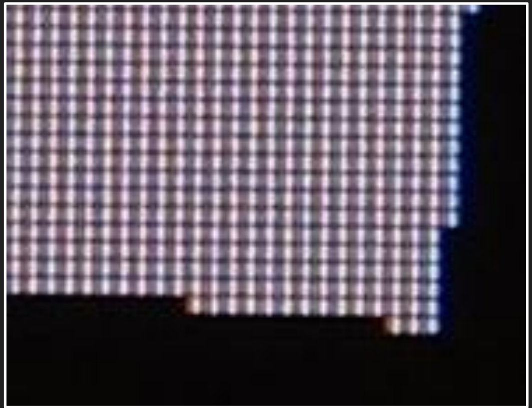
with Jim Nilsson, Marco Salvi, Josef Spjut, and Tomas Akenine-Möller

NVIDIA

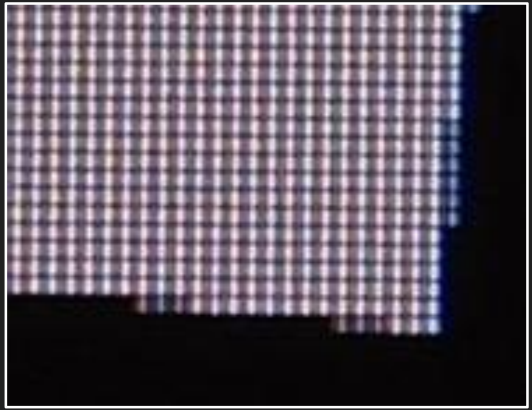
Motivation

Higher frame rates results in

- Improved accuracy and target tracking in games
- Motion blur reduction
- Less perceived aliasing and noise



60 Hz



120 Hz

240 Hz

An Algorithm for Faster Rendering

Goal: ~4x speedup at iso-quality

1920x1080 on an NVIDIA RTX 2080 Ti:

290 FPS
(581k tri, 3 lights)

310 FPS
(1M tri, 15 lights)

260 FPS
(192k tri, 3 lights)

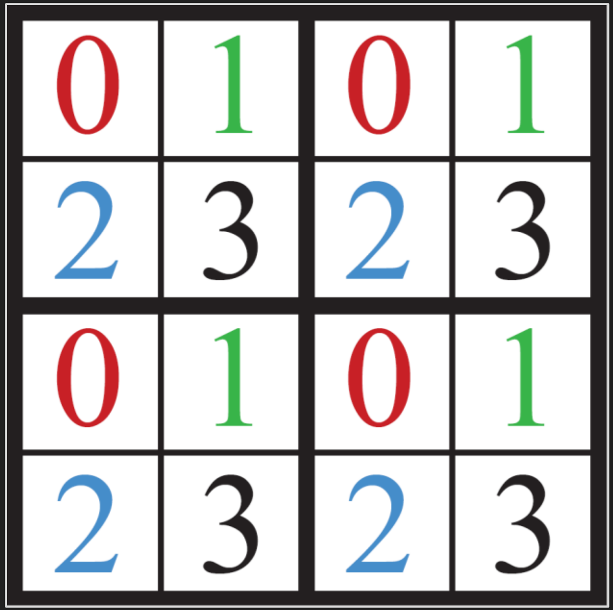
An Algorithm for Faster Rendering

Terminology

Full frame: The full resolution output frame

Tile: 2 × 2 pixels

Subframe: Collection of pixels with the same ordering number



An Algorithm for Faster Rendering

Terminology

Full frame: The full resolution output frame

Tile: 2 × 2 pixels

Subframe: Collection of pixels with the same ordering number

0	1	0	1
2	3	2	3
0	1	0	1
2	3	2	3

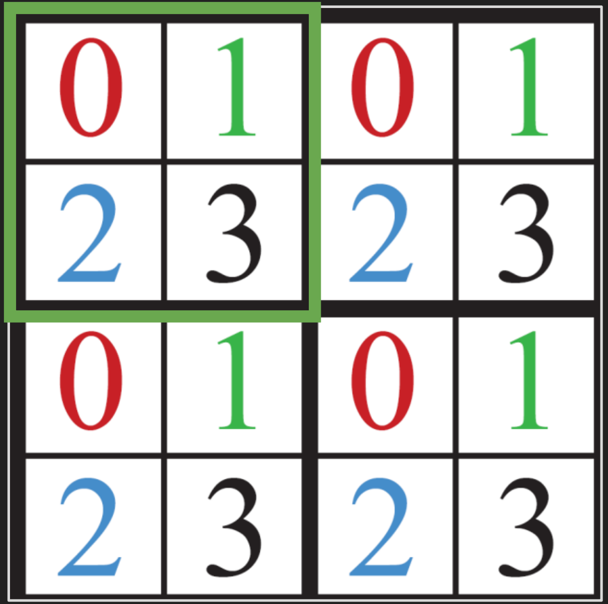
An Algorithm for Faster Rendering

Terminology

Full frame: The full resolution output frame

Tile: 2 × 2 pixels

Subframe: Collection of pixels with the same ordering number



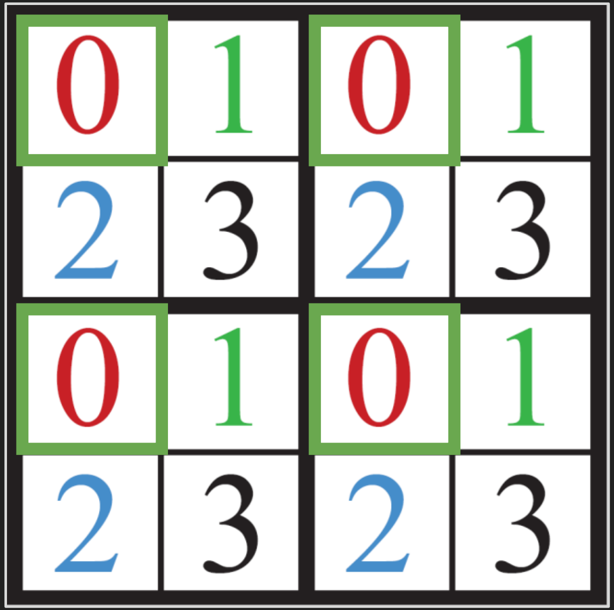
An Algorithm for Faster Rendering

Terminology

Full frame: The full resolution output frame

Tile: 2×2 pixels

Subframe: Collection of pixels with the same ordering number



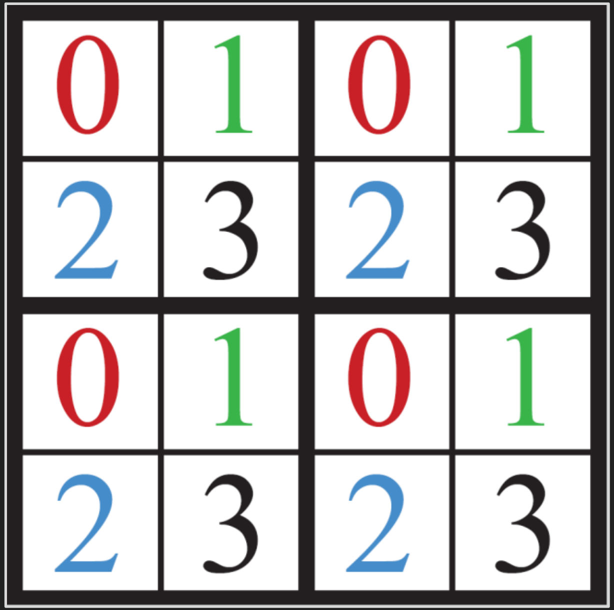
An Algorithm for Faster Rendering

Our Approach – First Step (not new)

Raytrace the pixels of one subframe per iteration

→ ¼ samples per pixel in the full frame

Pixels in the latest subframe are ***new***, the others are ***old***



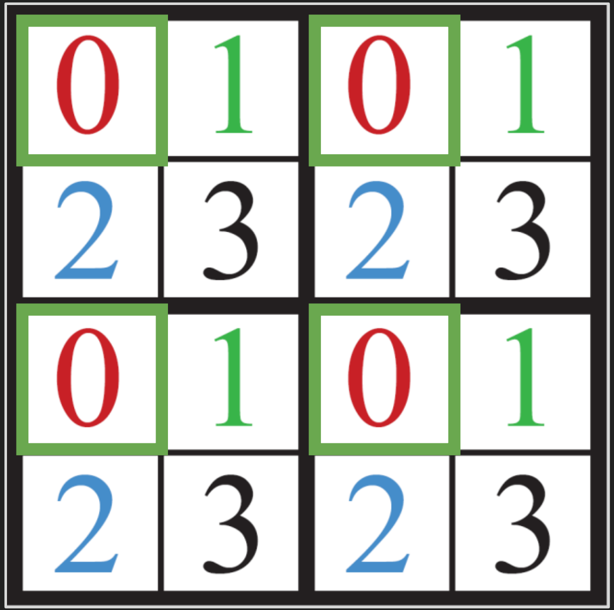
An Algorithm for Faster Rendering

Our Approach – First Step (not new)

Raytrace the pixels of one subframe per iteration

→ ¼ samples per pixel in the full frame

Pixels in the latest subframe are ***new***, the others are ***old***



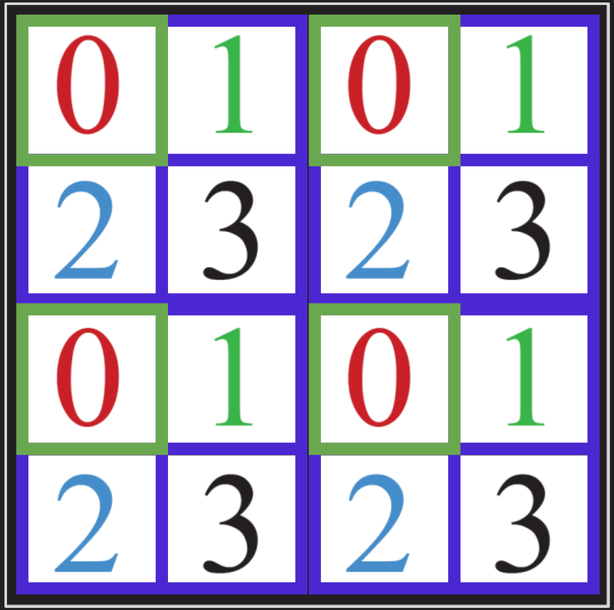
An Algorithm for Faster Rendering

Our Approach – First Step (not new)

Raytrace the pixels of one subframe per iteration

→ ¼ samples per pixel in the full frame

Pixels in the latest subframe are ***new***, the others are ***old***



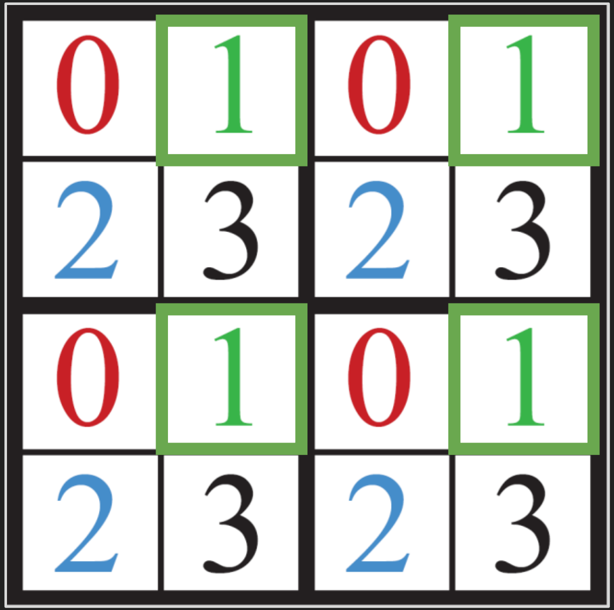
An Algorithm for Faster Rendering

Our Approach – First Step (not new)

Raytrace the pixels of one subframe per iteration

→ ¼ samples per pixel in the full frame

Pixels in the latest subframe are ***new***, the others are ***old***



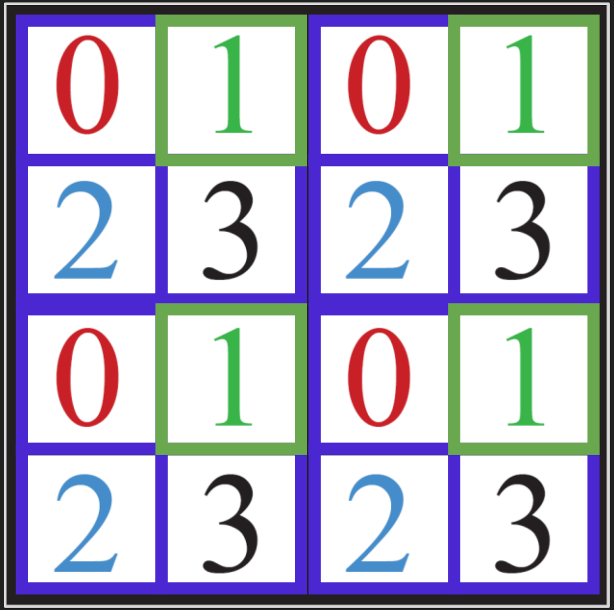
An Algorithm for Faster Rendering

Our Approach – First Step (not new)

Raytrace the pixels of one subframe per iteration

→ ¼ samples per pixel in the full frame

Pixels in the latest subframe are ***new***, the others are ***old***



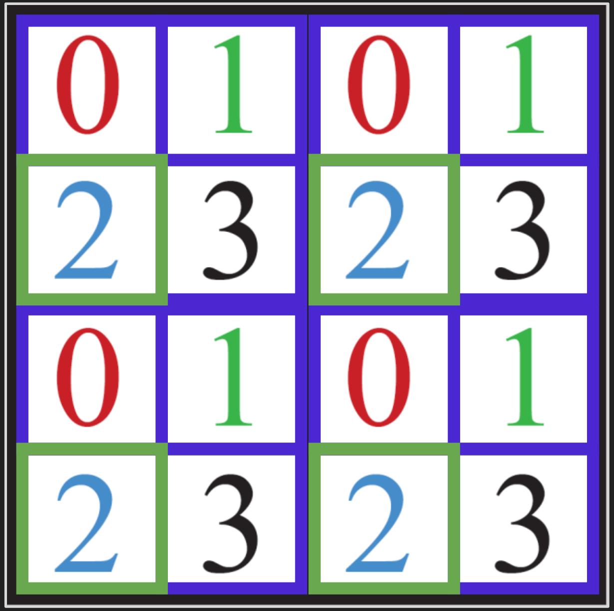
An Algorithm for Faster Rendering

Our Approach – First Step (not new)

Raytrace the pixels of one subframe per iteration

→ ¼ samples per pixel in the full frame

Pixels in the latest subframe are ***new***, the others are ***old***



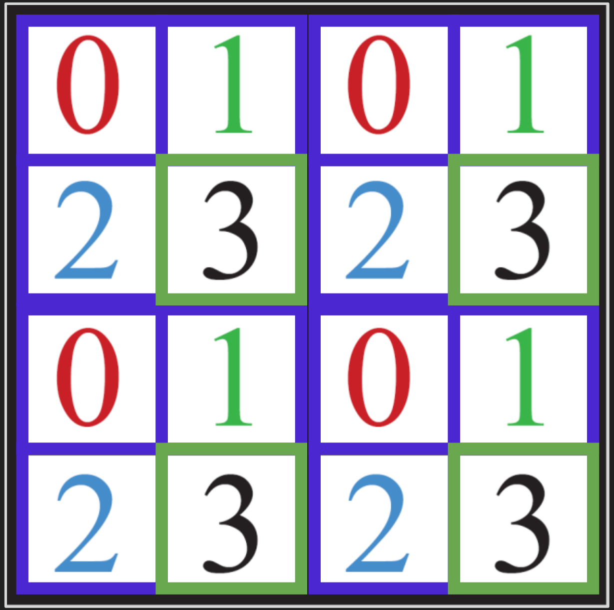
An Algorithm for Faster Rendering

Our Approach – First Step (not new)

Raytrace the pixels of one subframe per iteration

→ ¼ samples per pixel in the full frame

Pixels in the latest subframe are ***new***, the others are ***old***



An Algorithm for Faster Rendering

Our Approach – First Step (not new)

¼ samples per pixel per frame

Works fine when no movement

Disturbing during motion,
even at 240 FPS

Improving the Perceptual Experience

Our Approach

Process all pixels in the final frame

Consider static/dynamic pixels separately

If *static*: average

If *dynamic*:

- Copy **new** pixels from subframe to full frame
- Reproject and clamp **old** pixels, similar to TAA without exponential average

Improving the Perceptual Experience

Dynamic, Old Pixels – Clamp Footprint

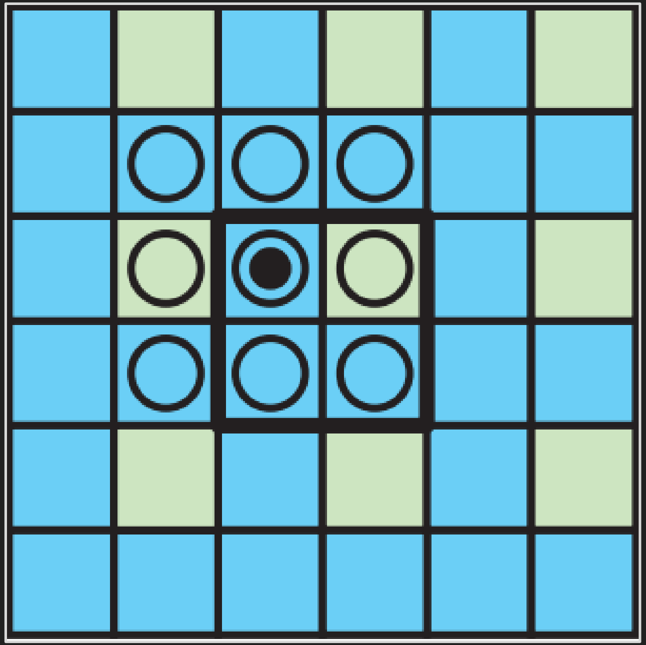
Reproject and clamp old pixels

Old pixels are blue, new pixels are green

Current pixel marked with bullet,
pixels used in footprint marked with circles

Three alternatives:

- Immediate neighbors



Improving the Perceptual Experience

Dynamic, Old Pixels – Clamp Footprint

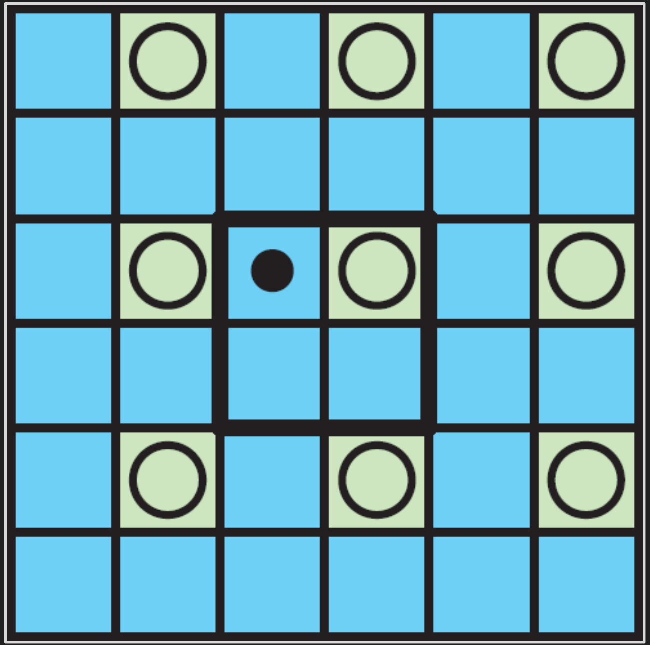
Reproject and clamp old pixels

Old pixels are blue, new pixels are green

Current pixel marked with bullet,
pixels used in footprint marked with circles

Three alternatives:

- Immediate neighbors
- 3x3 area in newest subframe



Improving the Perceptual Experience

Dynamic, Old Pixels – Clamp Footprint

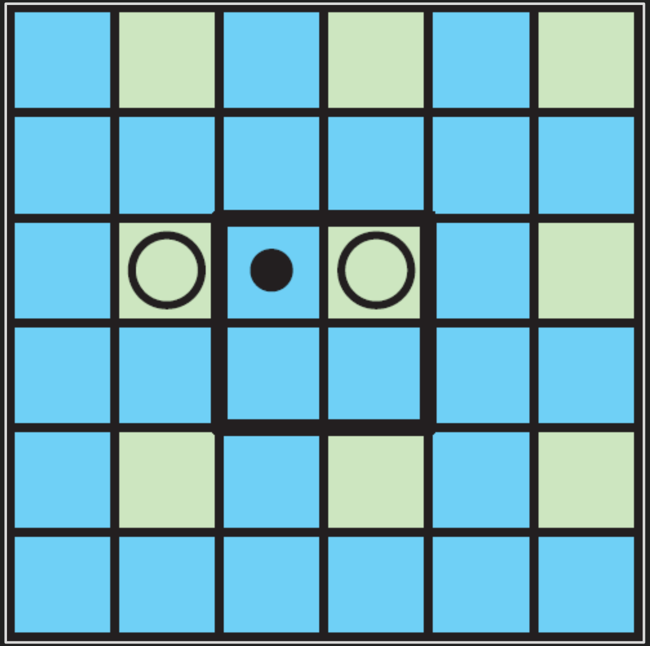
Reproject and clamp old pixels

Old pixels are blue, new pixels are green

Current pixel marked with bullet,
pixels used in footprint marked with circles

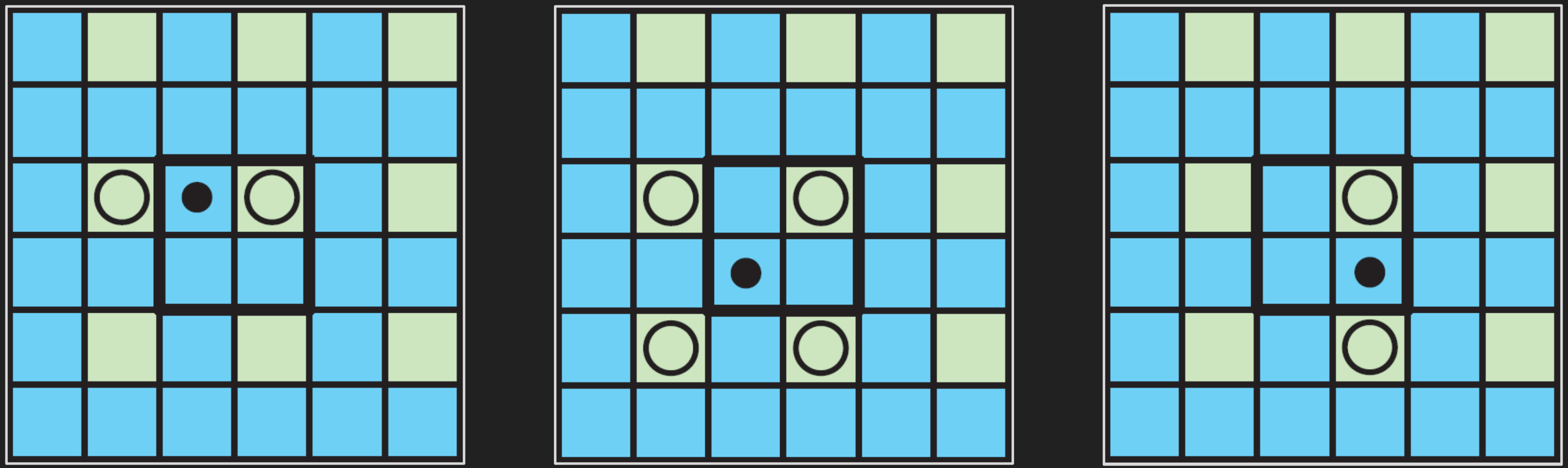
Three alternatives:

- Immediate neighbors
- 3x3 area in newest subframe
- Ours: immediate *and* new neighbors



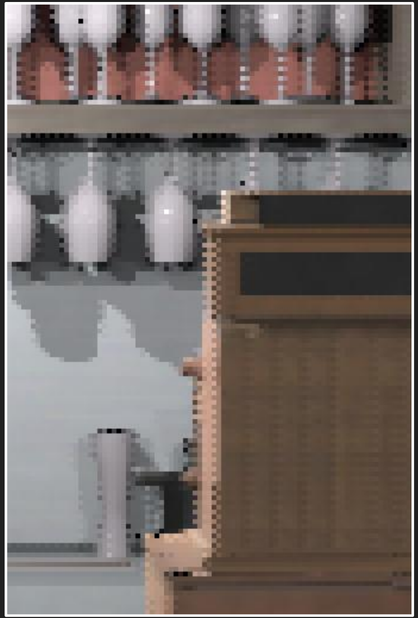
Improving the Perceptual Experience

Dynamic, Old Pixels – Clamp Footprint

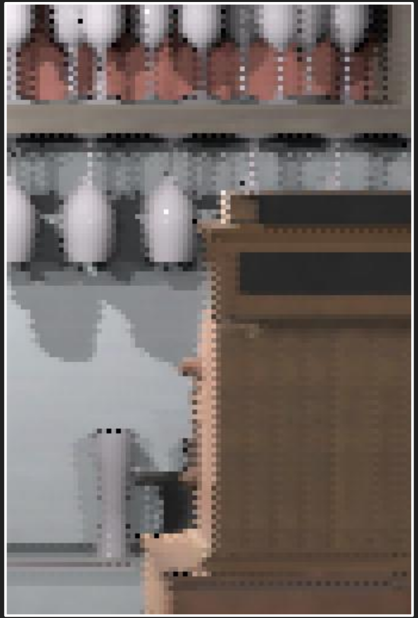


Improving the Perceptual Experience

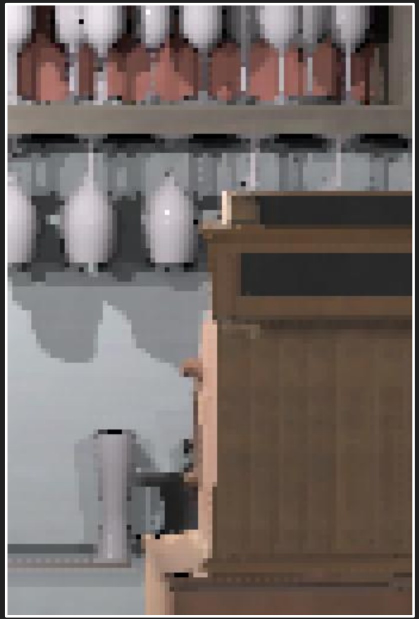
Dynamic, Old Pixels – Clamp Footprint



Immediate neighbors



3x3 area in newest subframe



Ours

Adaptive Pixel Ordering Selection

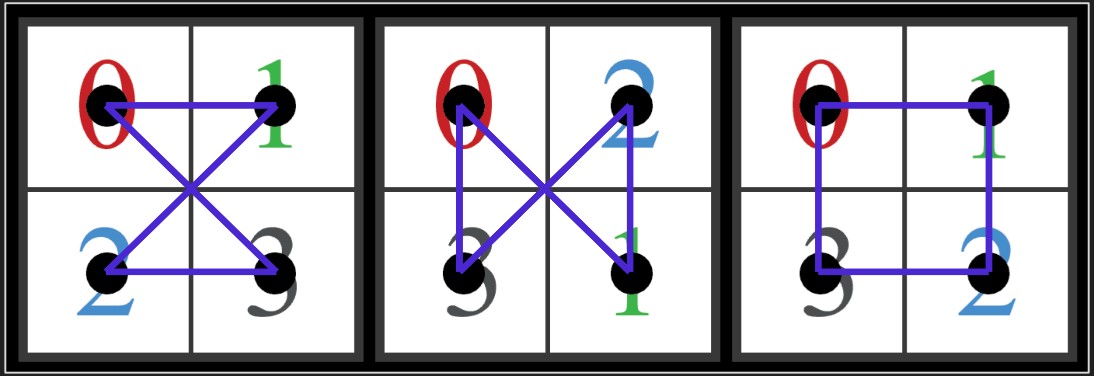
Pixel Ordering Makes all the Difference!

4! = 24 possible pixel orderings

Three basic shapes:

- **Hourglass**
- **Bowtie**
- **Square** (no benefit)

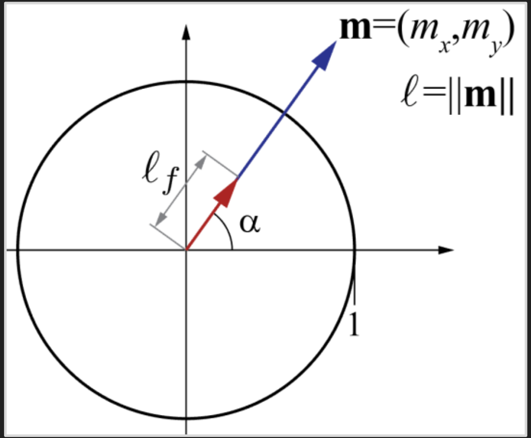
Our most interesting result!



Hourglass Bowtie Square

Adaptive Pixel Ordering Selection

Our Approach



Decision is a function of motion and pixel neighborhood contents

We narrow it down to pixel motion:

- direction α and
- length's fraction part ℓ_f

Horizontal motion ($\alpha = 0$):

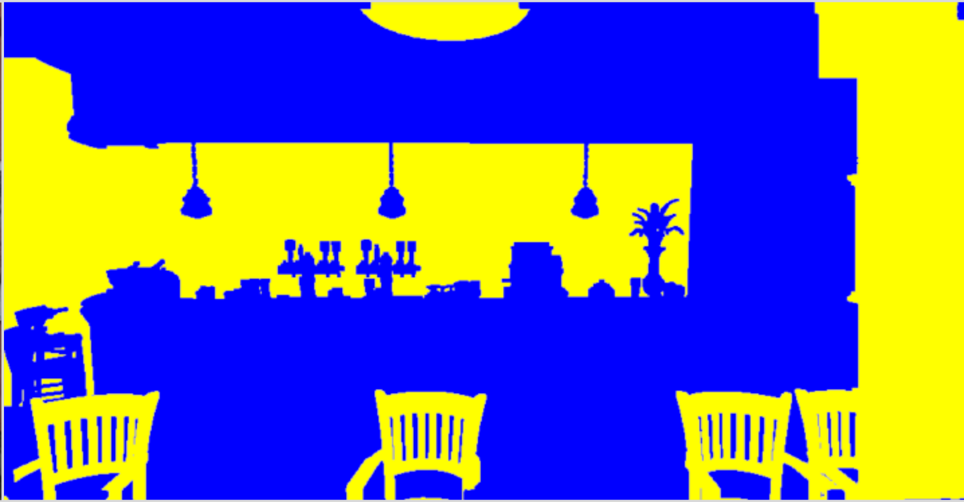
$$\begin{cases} \text{hourglass} & \text{if } \ell_f \in (0.25, 0.75), \\ \text{bowtie} & \text{otherwise,} \end{cases}$$

Vertical motion ($\alpha = \pi/2$):

$$\begin{cases} \text{hourglass} & \text{if } \ell_f \in [0, 0.25] \cup [0.75, 1], \\ \text{bowtie} & \text{otherwise.} \end{cases}$$

Adaptive Pixel Ordering Selection

Pixel Ordering Makes all the Difference!



Horizontal motion ($\alpha = 0$):

$$\begin{cases} \text{hourglass} & \text{if } \ell_f \in (0.25, 0.75), \\ \text{bowtie} & \text{otherwise,} \end{cases}$$

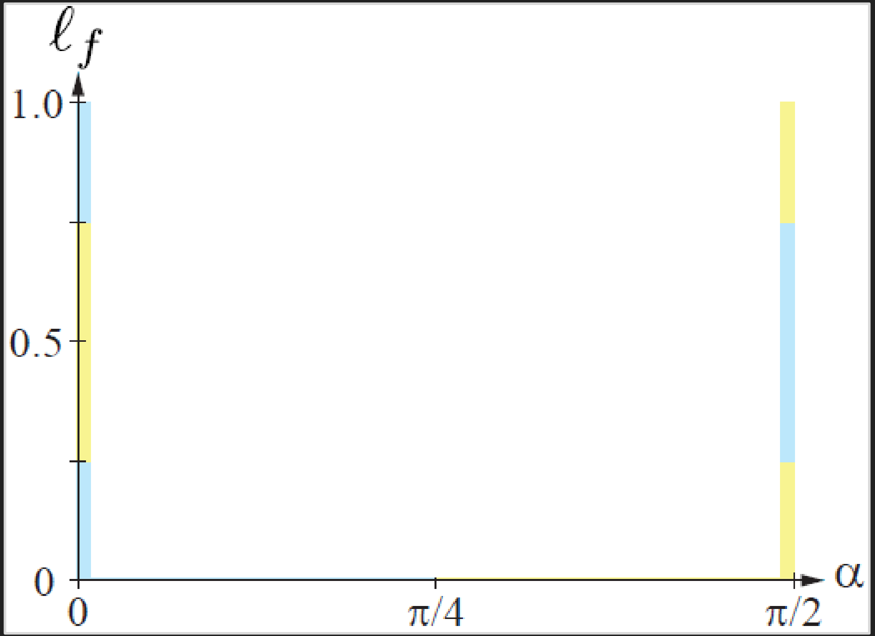
Pixel ordering decision –
yellow = hourglass,
blue = bowtie

Adaptive Pixel Ordering Selection

Our Approach

How about $\alpha \in (0, \pi/2)$?

Transition between the two extremes



yellow = hourglass, blue = bowtie

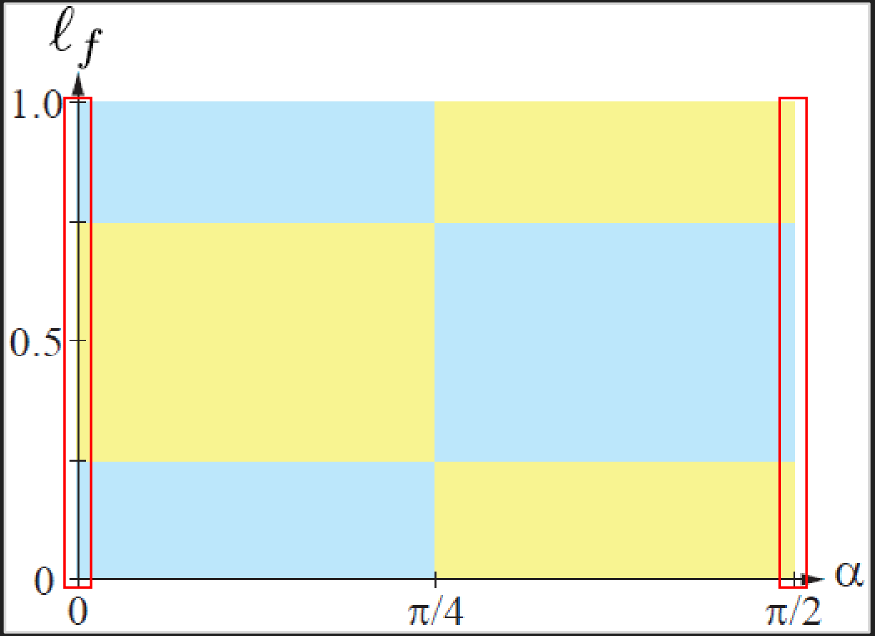
Adaptive Pixel Ordering Selection

Our Approach

How about $\alpha \in (0, \pi/2)$?

Transition between the two extremes

- Dominant motion direction



yellow = hourglass, blue = bowtie

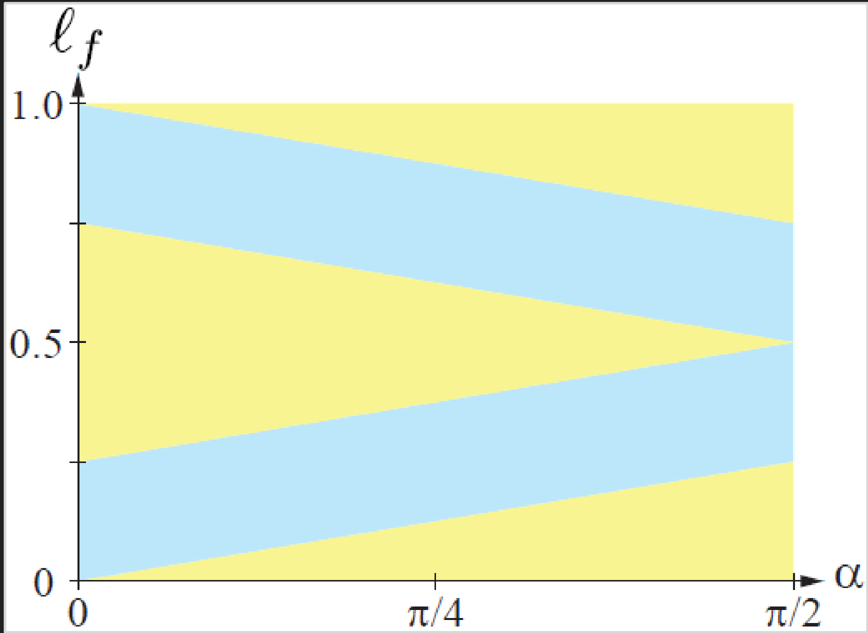
Adaptive Pixel Ordering Selection

Our Approach

How about $\alpha \in (0, \pi/2)$?

Transition between the two extremes

- Dominant motion direction
- Continuous transition #1



yellow = hourglass, blue = bowtie

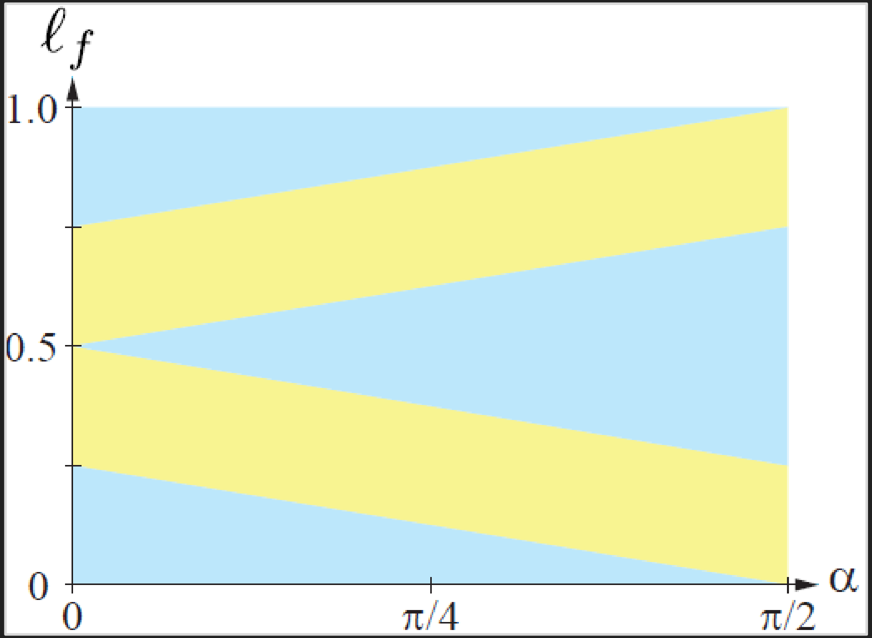
Adaptive Pixel Ordering Selection

Our Approach

How about *$\alpha \in (0, \pi/2)$*?

Transition between the two extremes

- Dominant motion direction
- Continuous transition #1
- Continuous transition #2



yellow = hourglass, blue = bowtie

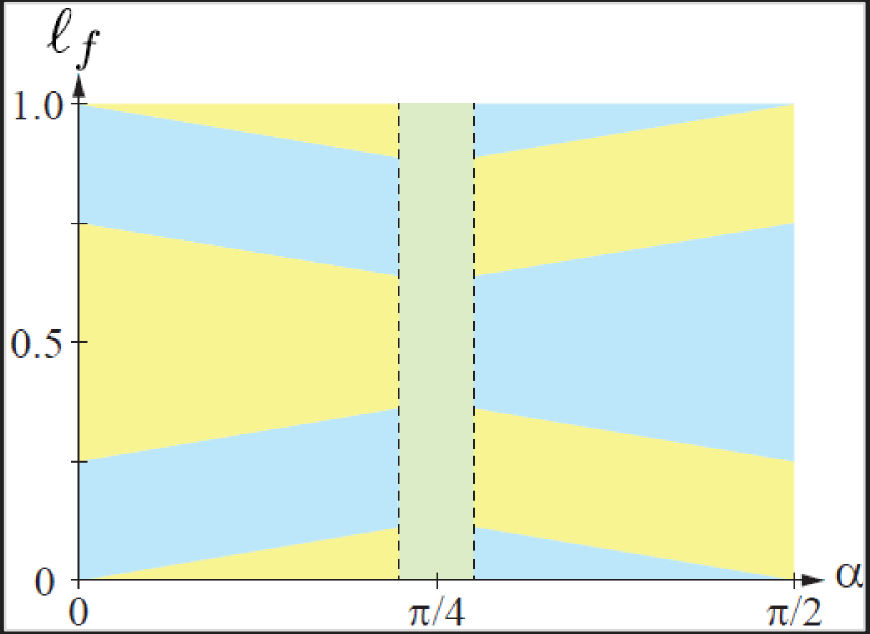
Adaptive Pixel Ordering Selection

Our Approach

How about *$\alpha \in (0, \pi/2)$*?

Transition between the two extremes

- Dominant motion direction
- Continuous transition #1
- Continuous transition #2
- Ours: Combo of #1 & #2, w/ random
  Why random? Removes structure



yellow = hourglass, blue = bowtie
green = random

Results

General

Similar or better than 1 spp @ 60 FPS

User study: 94% of subjects preferred our technique over 1 spp @ 60 FPS

TAA @ 60 FPS smoother, but substantially more blurry

Results

Experience our Results

Live demo during the poster session

Experience the magic at:

240hz.org

BACKUP: Improving the Perceptual Experience

Our Approach – Static Pixels

Moving window average while static

4x SSAA

Static/Dynamic decision done *per pixel*