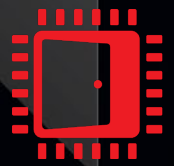
AMD
together we advance_

Ray Tracing Large Dynamic Game Worlds on AMD GPUs

Holger Gruen and Josh Barczak - AMD



AMD
GPUOpen

Public

Agenda

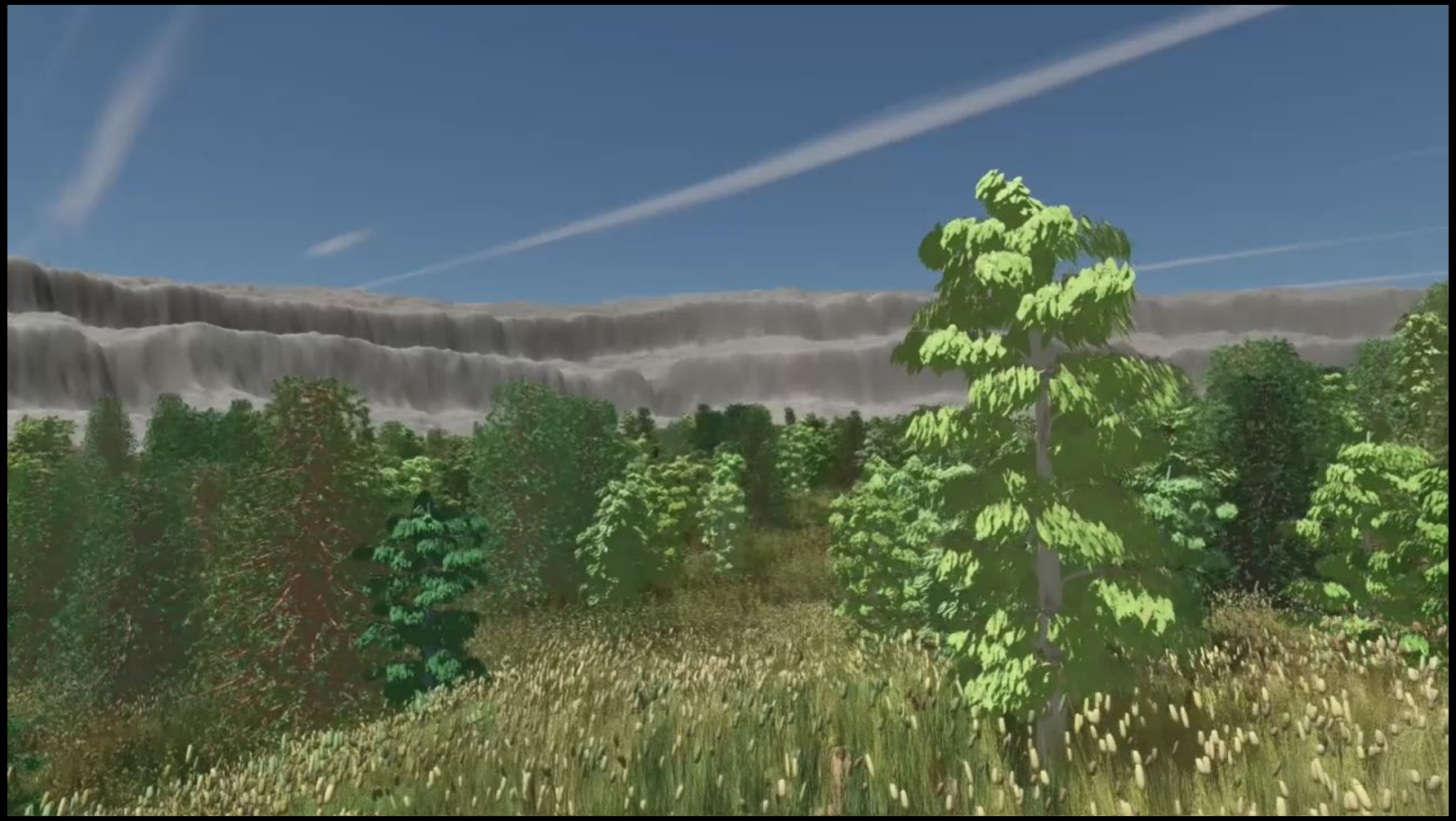
- Tetrahedral Cage Driven Animations
- Multi-Level Acceleration Structures
- Dense Geometry Format

Public

AMD
together we advance_

Tetrahedral Cage Driven Animations

Public

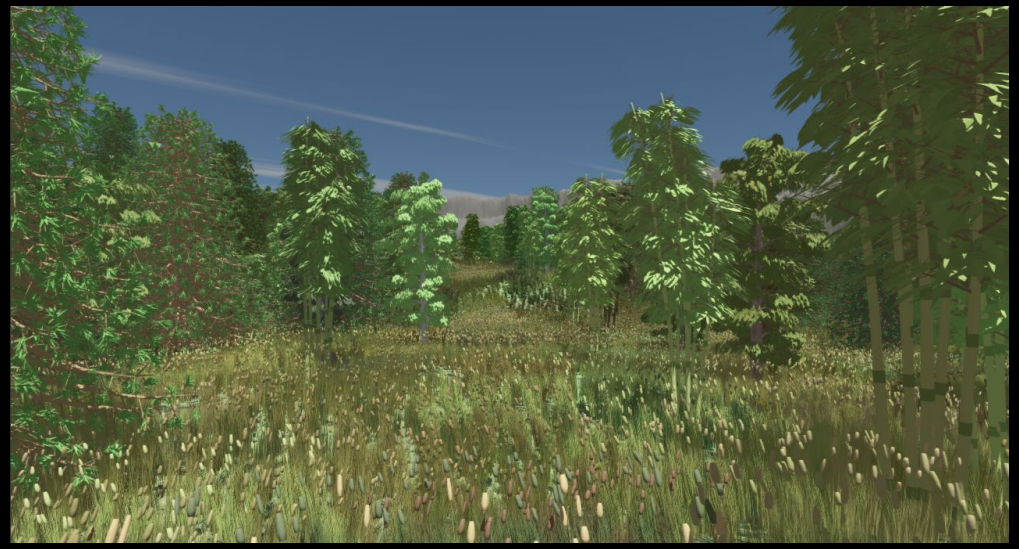


>40k independently animated plants ray traced with primary + shadow rays on RX 9070 XT @ 1080p @ 60+ FPS
Focus is on independent animation and geometric complexity (3 bil. tris w/o and ~500mio with LOD selection)
1.7 GB of BVH – memory for independent BLAS-objects/CLAS-objects prohibitively high on RX 9070 XT

Public

Using classic real-time ray tracing techniques for this scene is hard

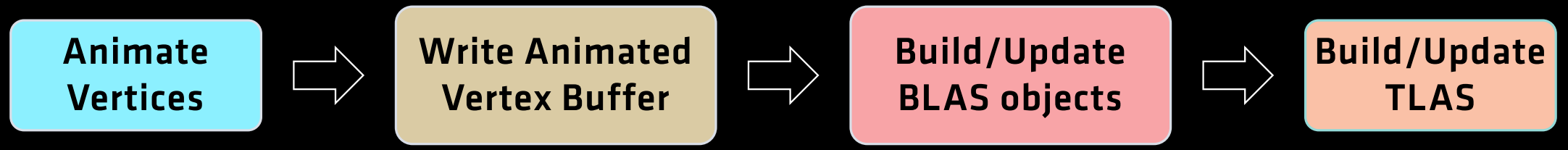
- ~3 billion animated triangles at the highest LOD
- ~40k **independently** animated plants
  - Assume locally varying wind vectors
- Solution space for ray tracing such a scene was limited
  - Staggered updates + aggressive LOD
  - Animate low LOD parts of the world rigidly
- Memory reqs for BVH structures can be prohibitive
- Prohibitively high BVH build/update time
- Rasterization works ok with the right LOD techniques



Public

Ray tracing independently animated objects is expensive

Steps for every independently (non-rigidly) animated object:



Vertex Animation

- Running time scales with vertex count.

Vertex Animation

- Temporary animated geometry must be stored.

BLAS Build

- **Build time** and **memory** scale with triangle count.

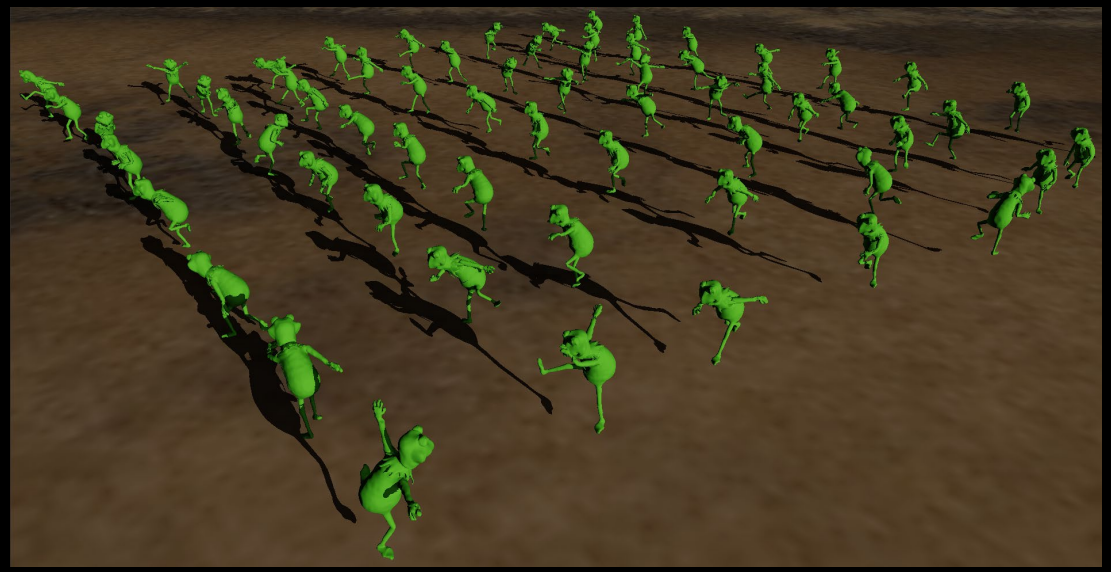
TLAS Build

- The top-level structure can only be finalized after BLAS updates finish.

DXR 2.0: Independently animated objects still need to animate, store and update unique CLAS sets

Public

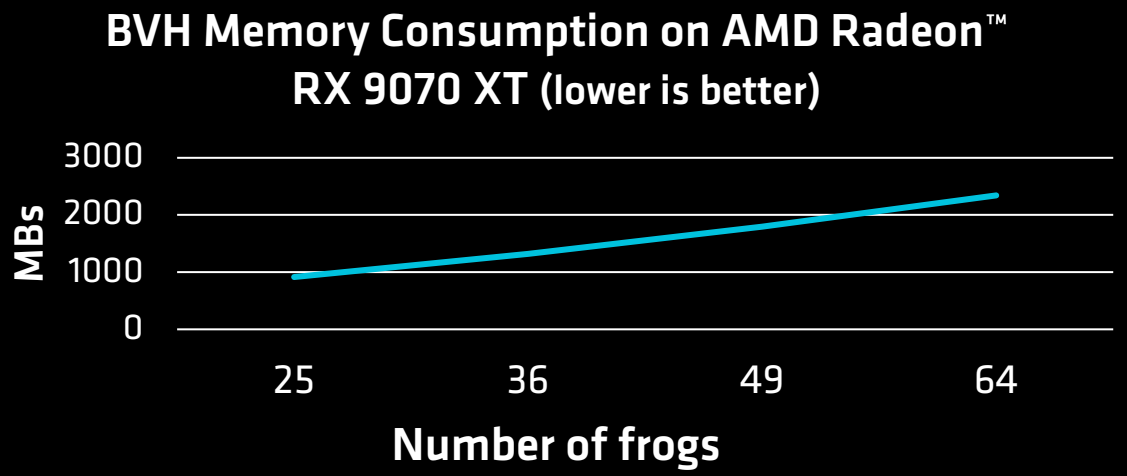
Test case: Memory consumption gets out of hand quickly!

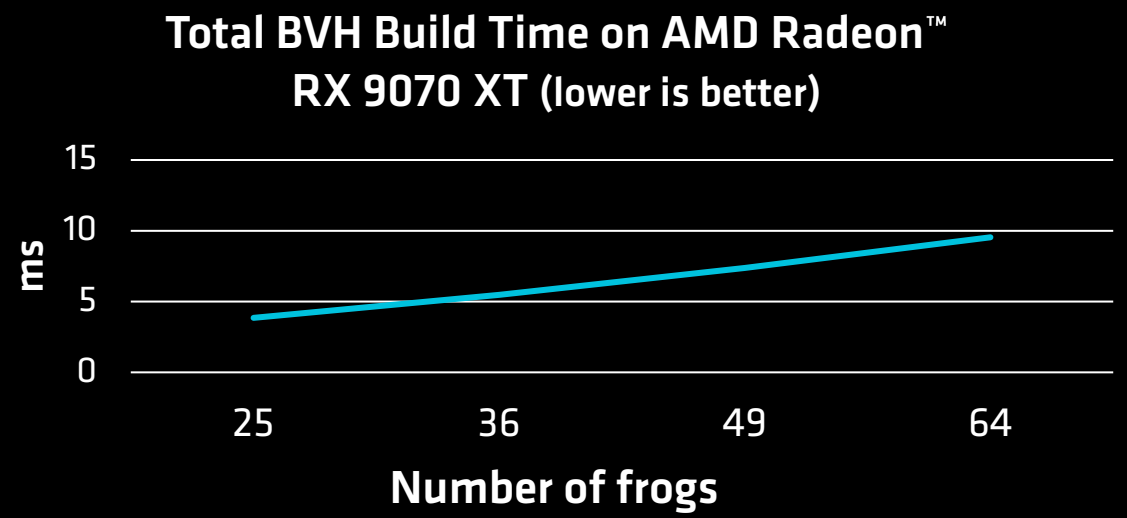


249k vertices per frog

408k triangles per frog

64 unique objects ~26M triangles





Public

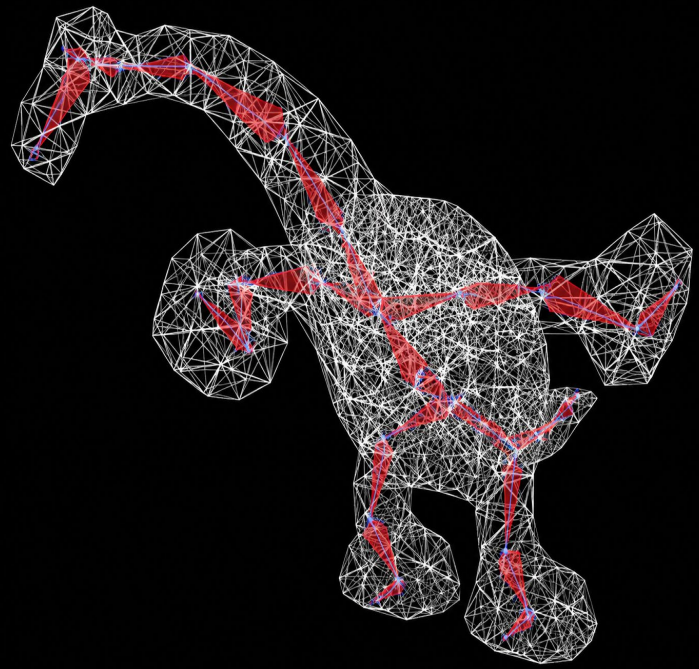
Can we do better? Can tetrahedral cages help?

- Tetrahedral cages have been used to deform objects before
- See e.g. HPG2025

High-Performance Graphics (2025)
A. Knoll and C. Peters (Editors)

Real-time rendering of animated meshless representations

Pacôme Luton and Thibault Tricard
INRIA / LJK (CNRS, UGA, INP-G), France



Public

Animate a coarse proxy, not every vertex of the complex geometry

- **Animate tetrahedral cage instead of a dense set of vertices** (Luton et. al HPG 2025) =>
  Animate only the vertices of the 'small' cage
- **Decouple BVH complexity from geometry complexity** (of the animated object)
  => Lower BVH build time and memory footprint
- **Decouple animation complexity from geometric complexity**
  => Lower animation cost

Why Tetrahedra?

Simple piecewise-linear map between animated cage and rest-pose space; Allows to use hardware instance XFs to deform rays

Quality/Performance Knob

Fewer tetrahedra are cheaper; more tetrahedra give a better approximation of the deformation.

Public

Offline Preprocessing: Build a tetrahedral-cage (simple version)

Fill Rest-Pose BBox With Voxels

Delete Empty Voxels

Split Remaining Voxels Into Tetrahedra

Delete Empty Tetrahedra

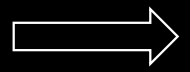
Clip Mesh Into Rest-Pose mini-Meshes



Public

Once during application startup



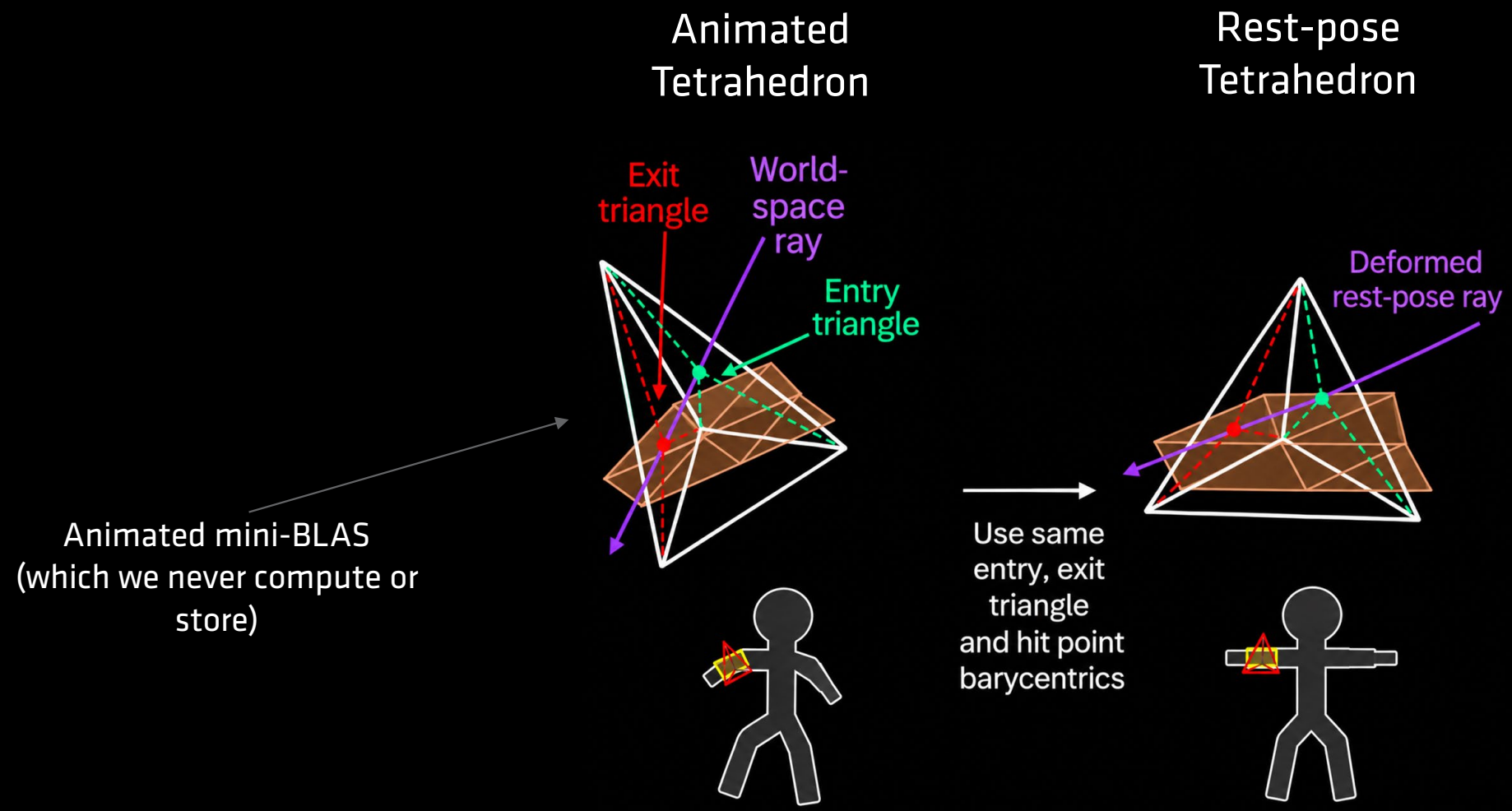


Build <u>one</u> static reusable mini-BLAS object from each static clipped mini-mesh

Make sure to build these using TRACE_FAST and compact them

Public

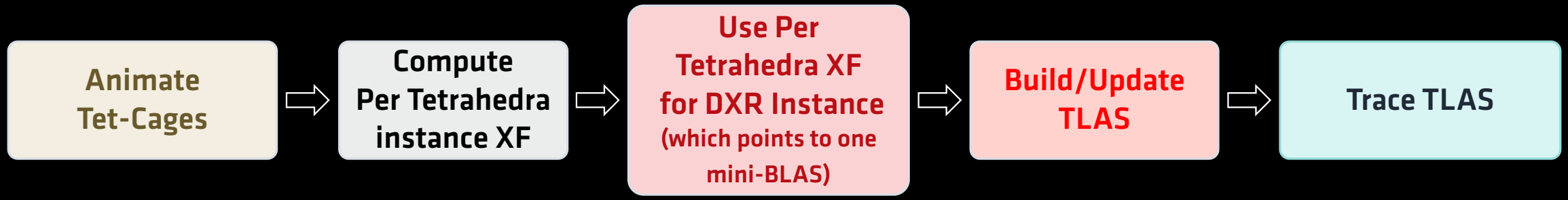
Hang on, how do we ray trace the animated object then?



Transform the ray instead of transforming the geometry!!

Public

Per frame animation pipeline



Ray deformation can be mapped to **DXR Instance transforms** => **Full Hardware acceleration**

$$B=\begin{pmatrix} v1.x-v0.x & v2.x-v0.x & v3.x-v0.x & 0 \\ v1.y-v0.y & v2.y-v0.y & v3.y-v0.y & 0 \\ v1.z-v0.z & v2.z-v0.z & v3.z-v0.z & 0 \end{pmatrix}$$ static tetrahedron (precomputed)

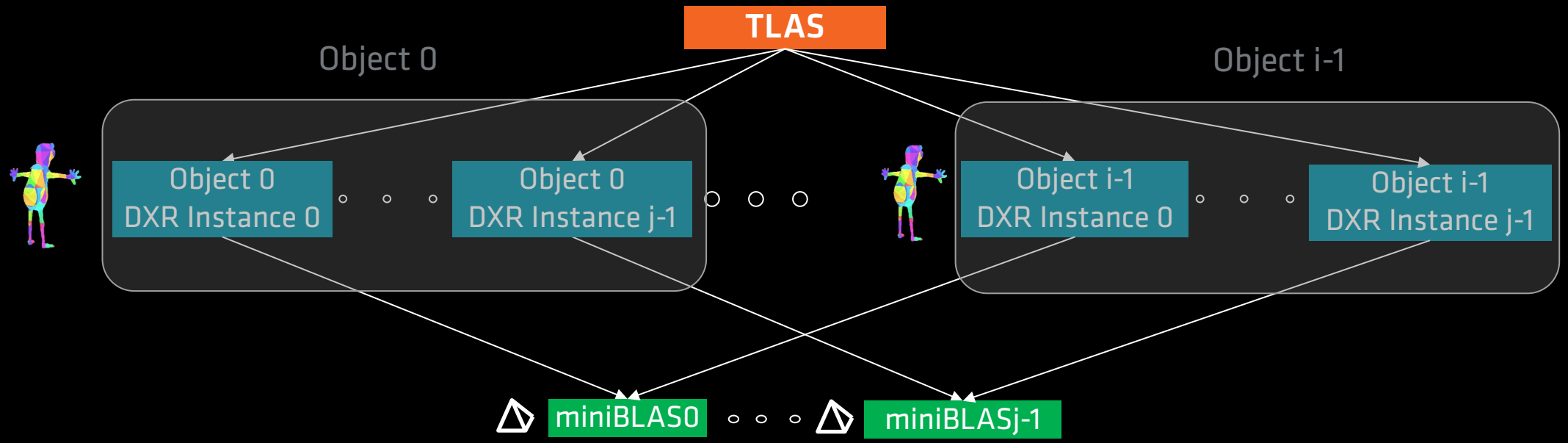
$$A=\begin{pmatrix} va1.x-va0.x & va2.x-va0.x & va3.x-va0.x & va0.x \\ va1.y-va0.y & va2.y-va0.y & va3.y-va0.y & va0.y \\ va1.z-va0.z & va2.z-va0.z & va3.z-va0.z & va0.z \end{pmatrix}$$ animated tetrahedron

$\Rightarrow$ DXR instance XF for a given tetrahedron is $Inst2World \cdot A \cdot B^{-1}$

The paper contains all the details ...

Public

DXR 1.0 BVH layout



All tetrahedron-cage animated instances of an object reuse the same set of mini BLAS objects!

Public

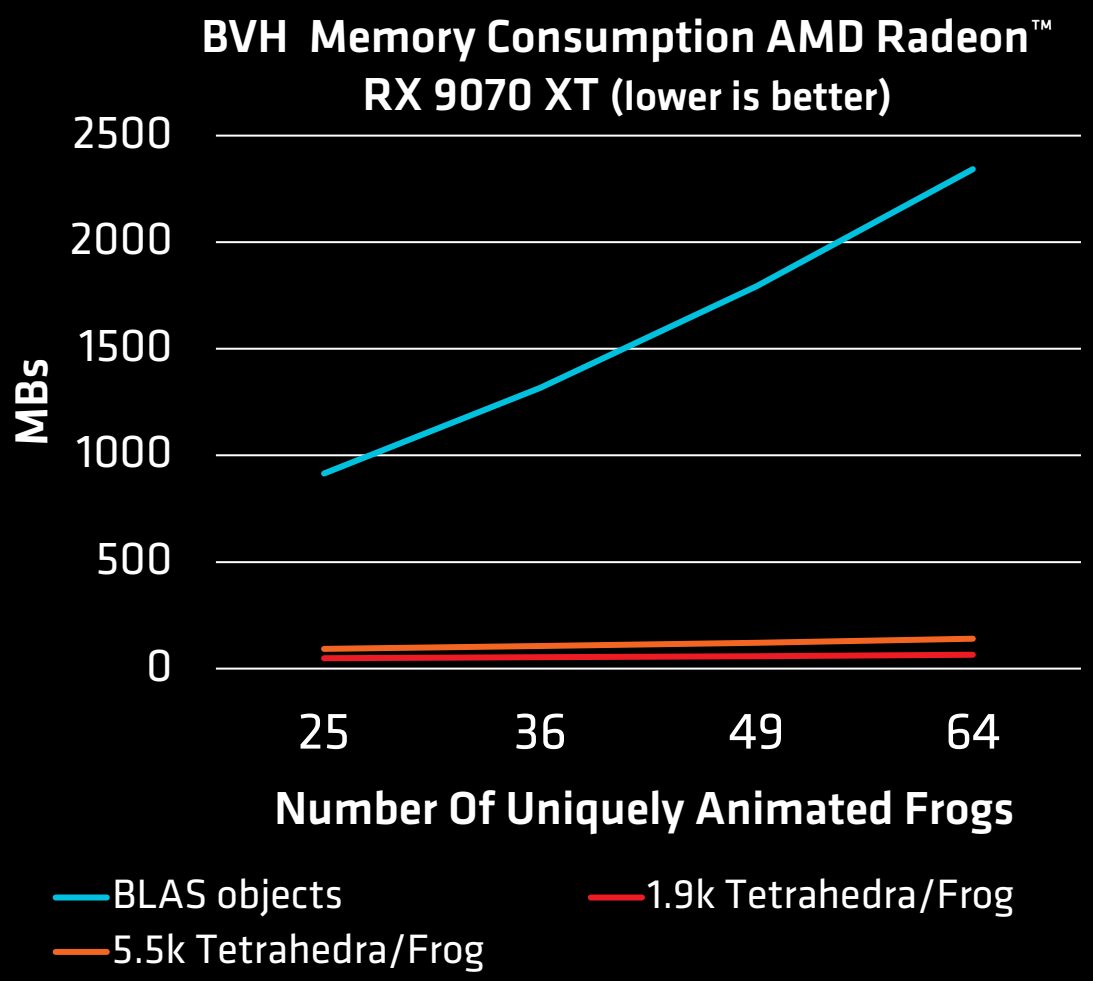
A frog animated using a skinned set of tetrahedra

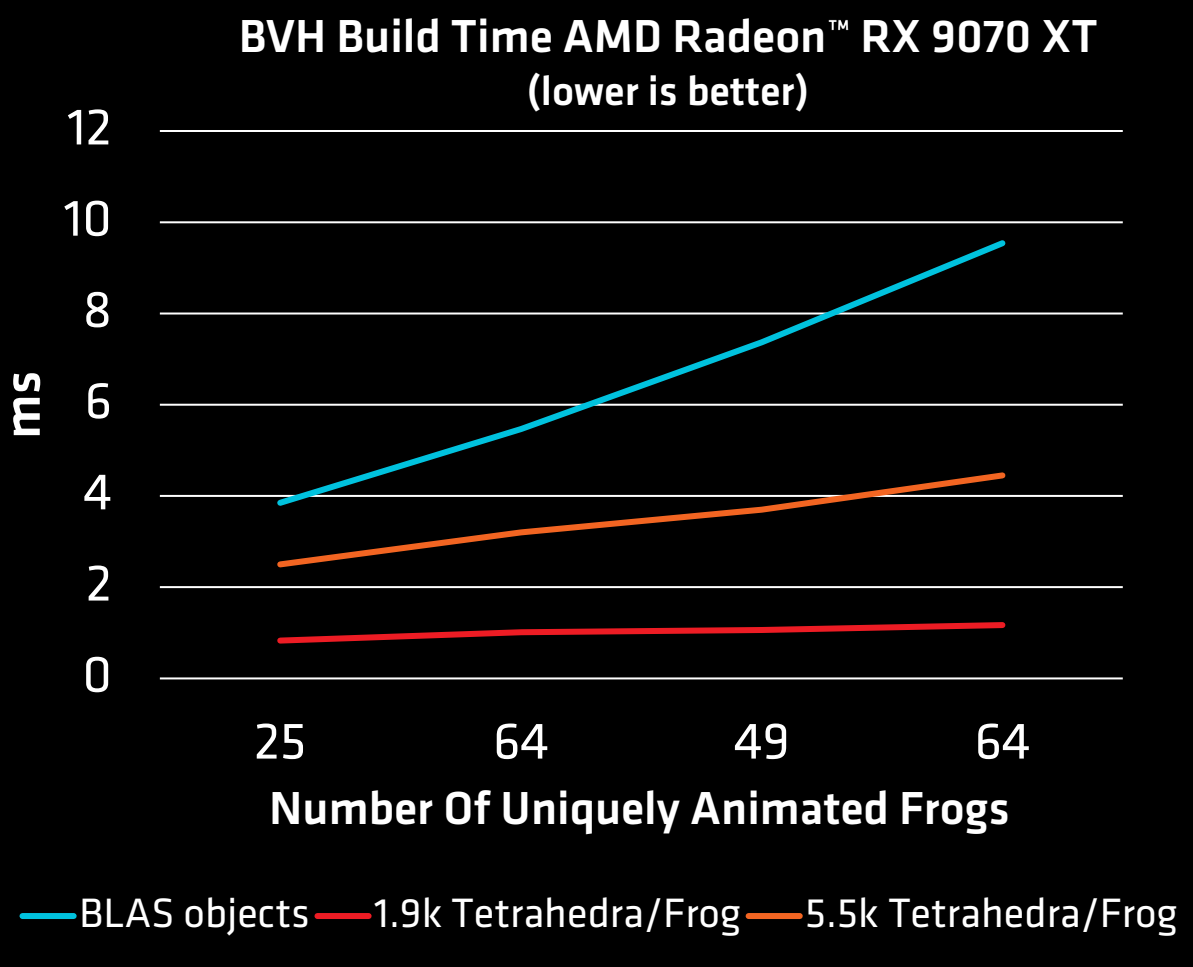


Public

AMD
together we advance_

New BVH memory consumption and build time

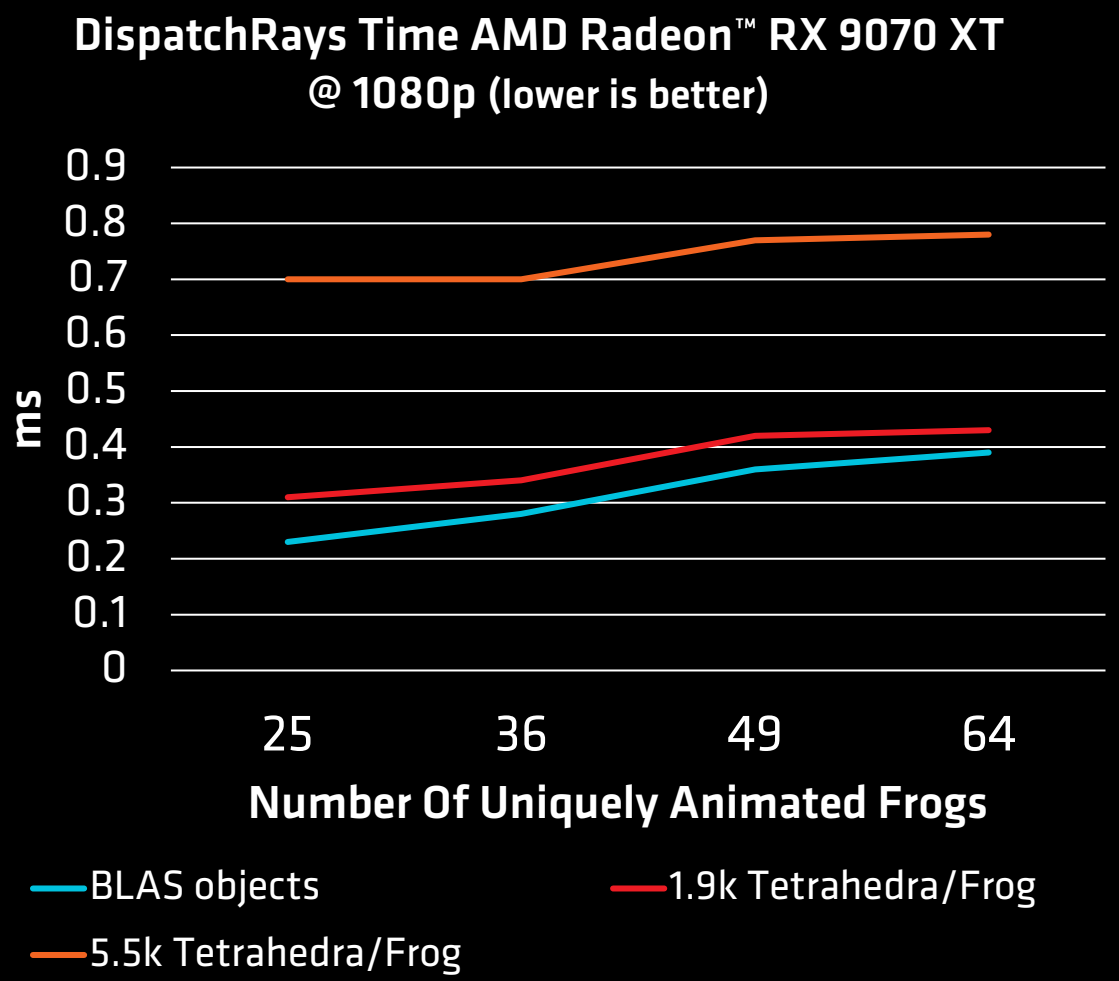


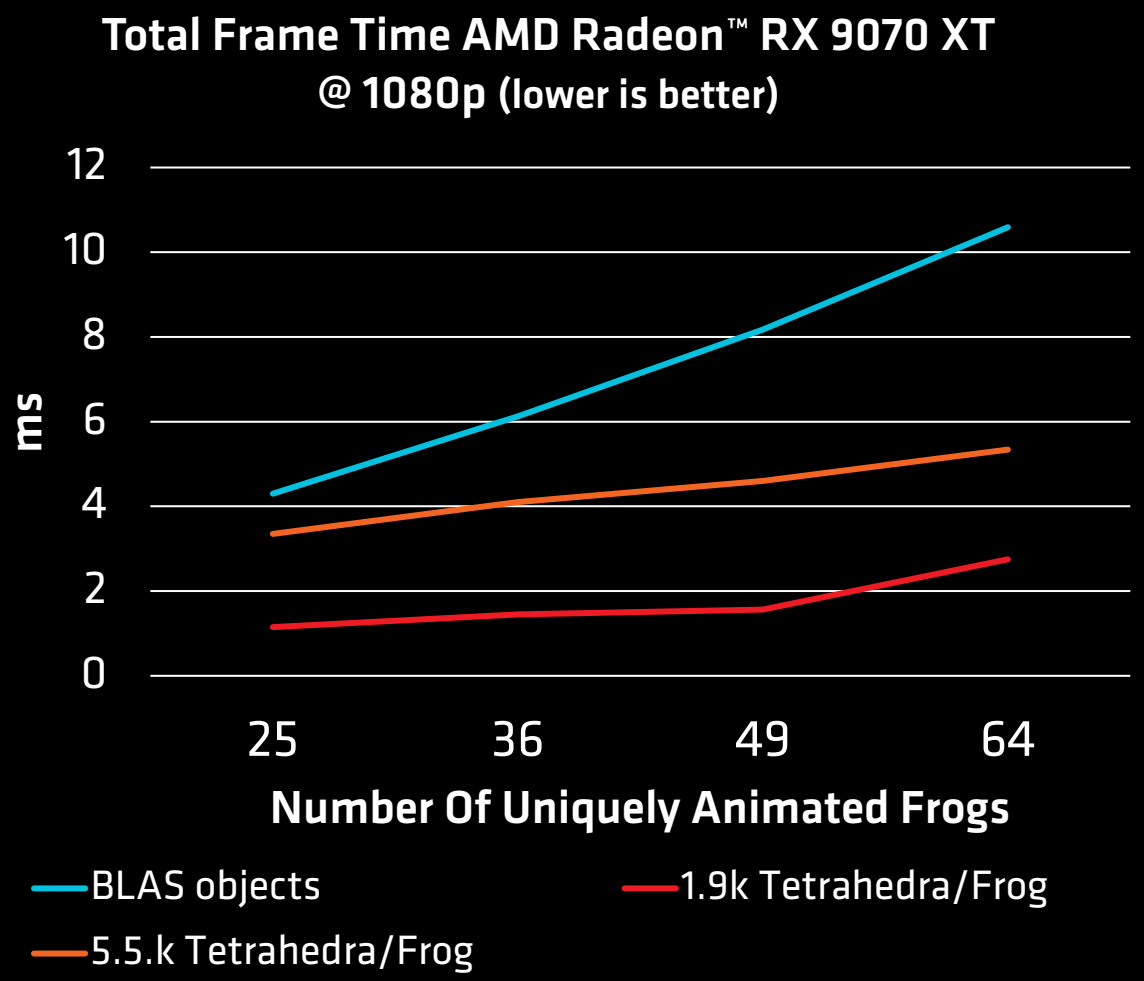


Public

AMD
together we advance_

DispatchRays() time (2 Rays Per Pixel) and total frame time





Public

Other cage animated objects



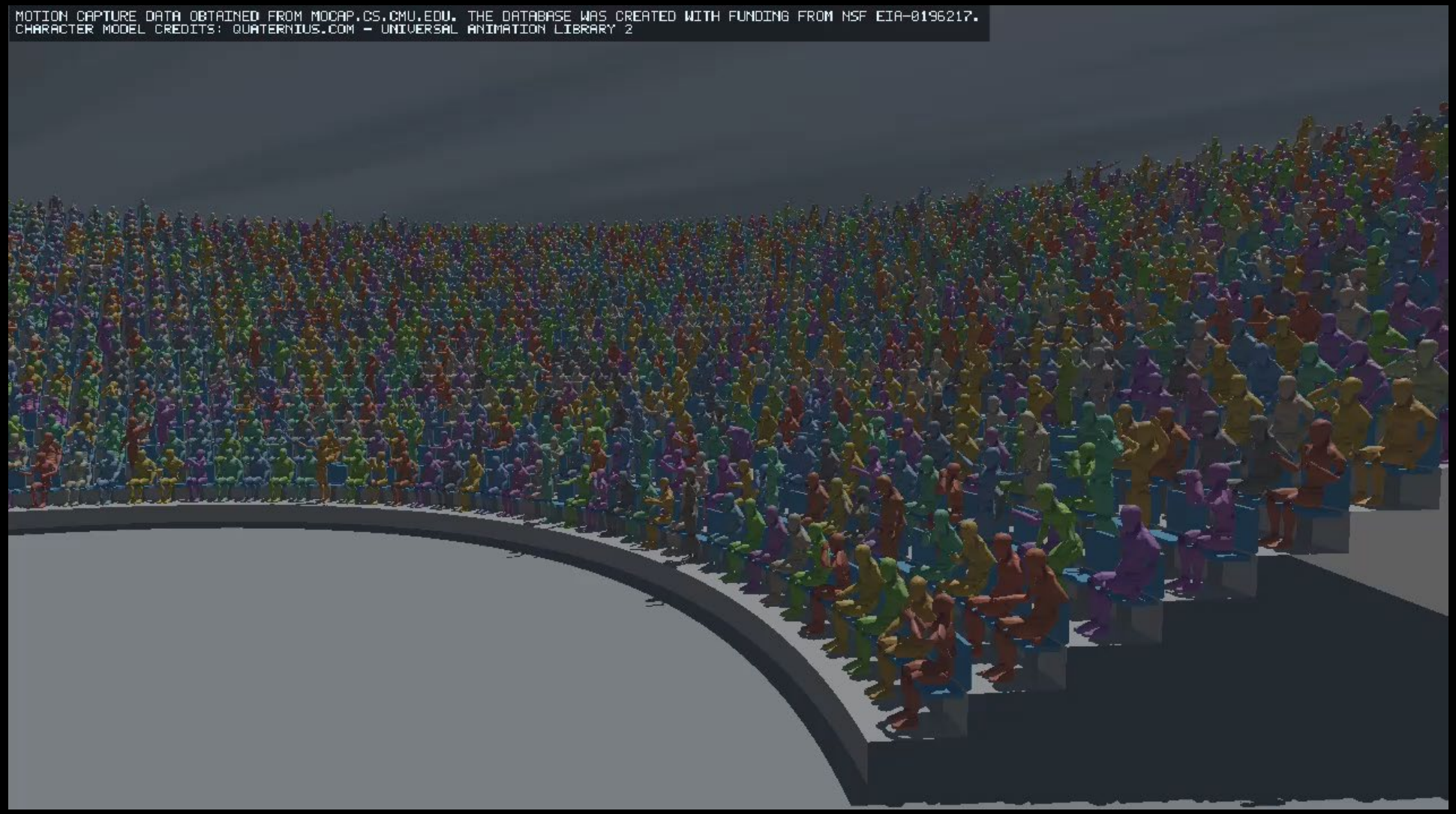
Public

3000 characters with 25k triangles each on a laptop PC



Public

2500 spectators with 25k triangles each on a laptop PC



Public

Conclusions and caveats

- Use it to optimize BVH build time and memory for complex animated geometry
  - We eliminate BLAS builds and instead have more complex TLAS builds
  - P-TLAS/MLAS can help with higher TLAS instance counts
- The method generates approximate macro-scale animations
  - Can save animation time and/or memory over uniquely animated CLAS objects
- A different ray traverses each mini-mesh
  - Watertightness issues are overcome through a ‚dirty' trick – see paper
- You need to precompute a tetrahedral cage for your objects
  - This includes creating clipped mini meshes
- You can rasterize tetrahedra-deformed geometry
  - Can be more expensive than non-clipped geometry
- Does not combine easily with continuous LOD techniques like Nanite (yet)

Public

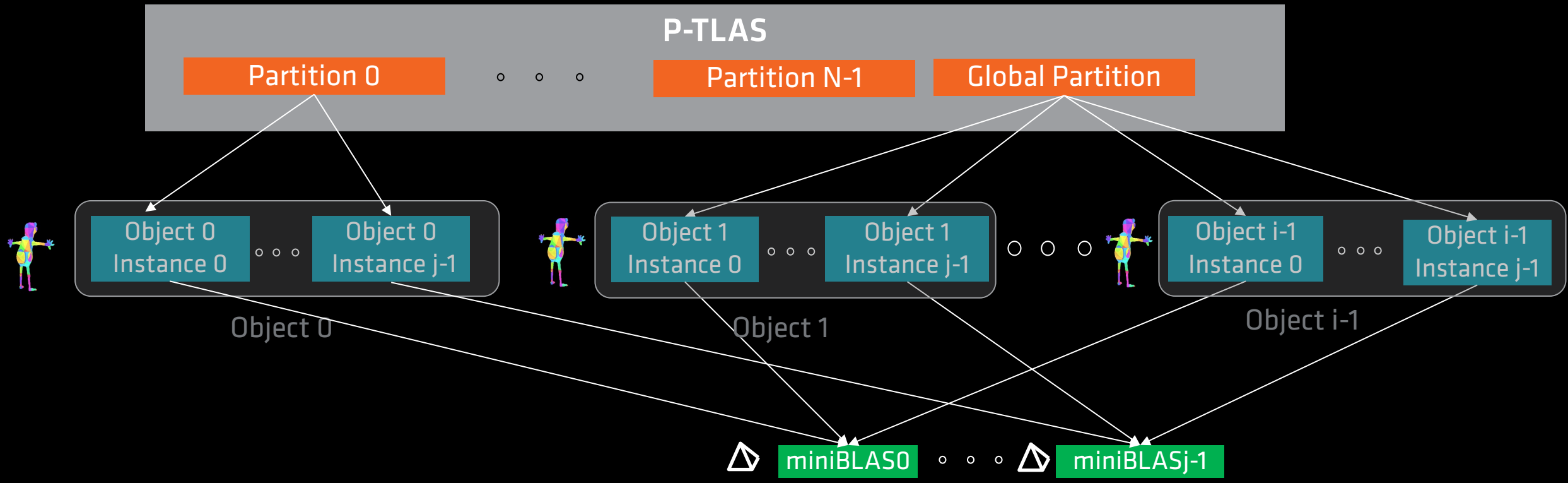
Our research source code will be available soon

- Header only C++ API for building tetrahedral cages
  - Support for skinned and articulated objects
  - Support for keyframe animations
  - Preliminary GPU support for optimizing bone weights for cage vertices

- Various sample apps
  - Builder apps for skinned and keyframe animated object
  - DXR samples for skinned and keyframe tetrahedra-animated objects
    - Includes sample that does hybrid animations (part tetrahedra-animated – part skinned)
    - Includes sample that rasterizes tetrahedra-animated objects but traces shadow rays
  - Vulkan® sample apps for skinned and keyframe tetrahedra-animated objects

Public

Tetrahedral cages and DXR 2.0 P-TLAS



Public

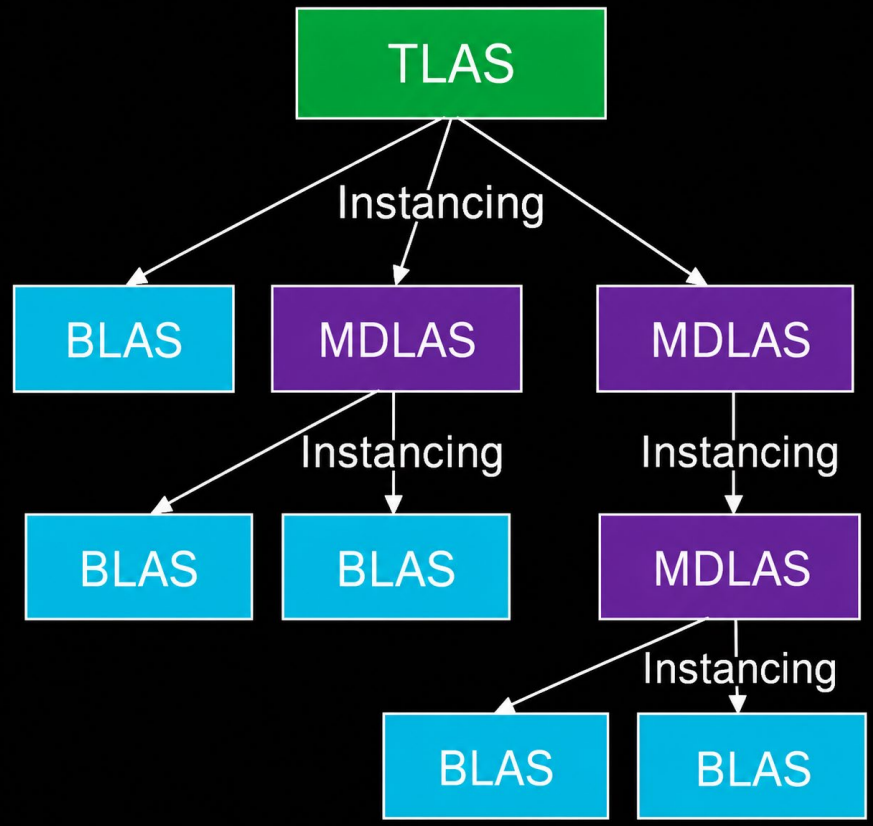
AMD
together we advance_

Multi-Level Acceleration Structures - MLAS

Public

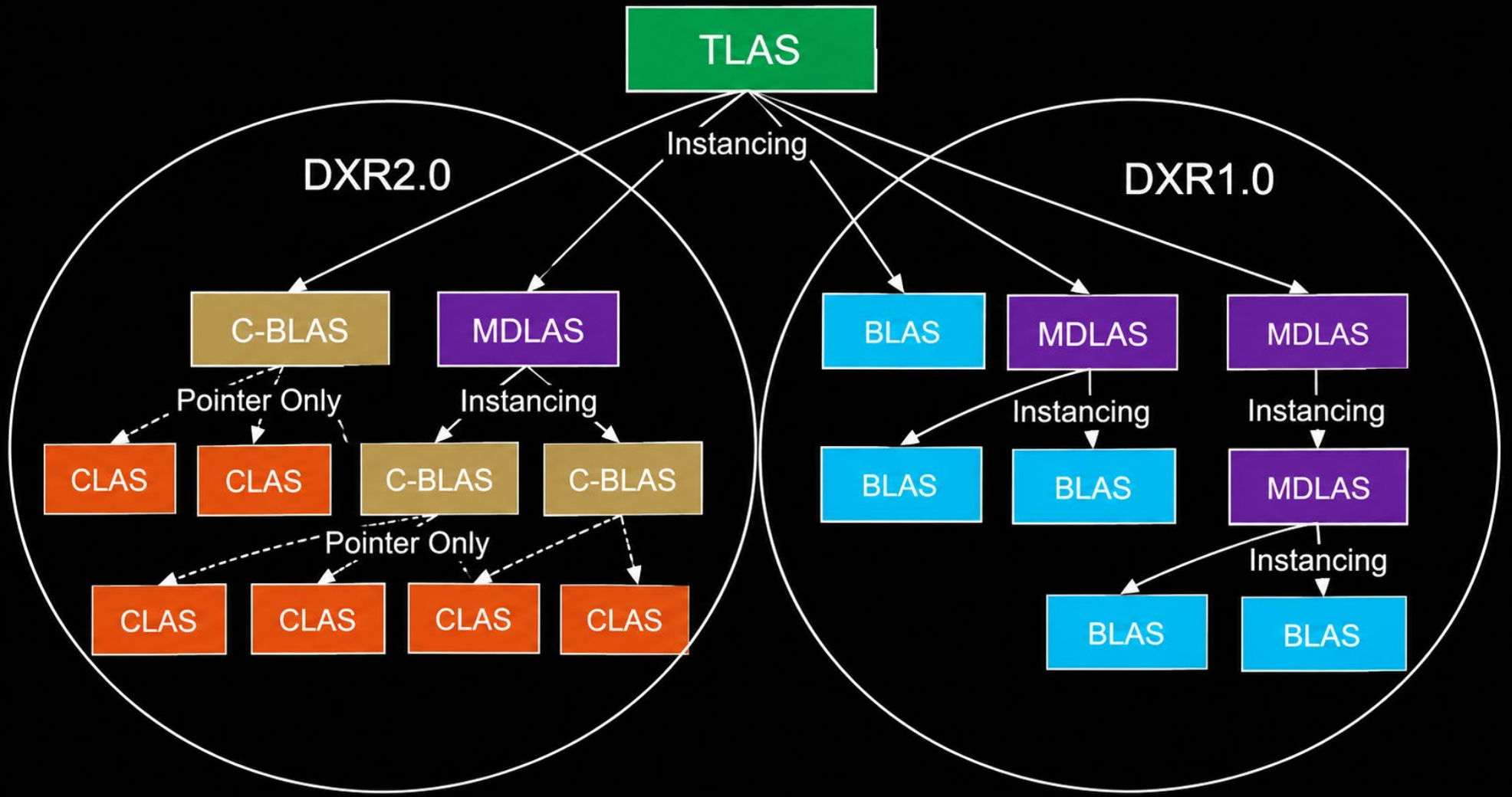
Multi-Level Acceleration Structures (MLAS-RT)

- Utilize up to 3 levels of instancing
- Save memory through multi-level reuse
  - E.g. reuse a house that is build from brick instances
- Increased BVH build speed
  - Help to keep TLAS complexity low
  - Control what part of the world to build/update
  - Use different build-flags for different parts of the world
- Trade BVH build time for traversal speed => higher FPS
- AMD Vulkan® extension just around the corner
  - to be released in a few days



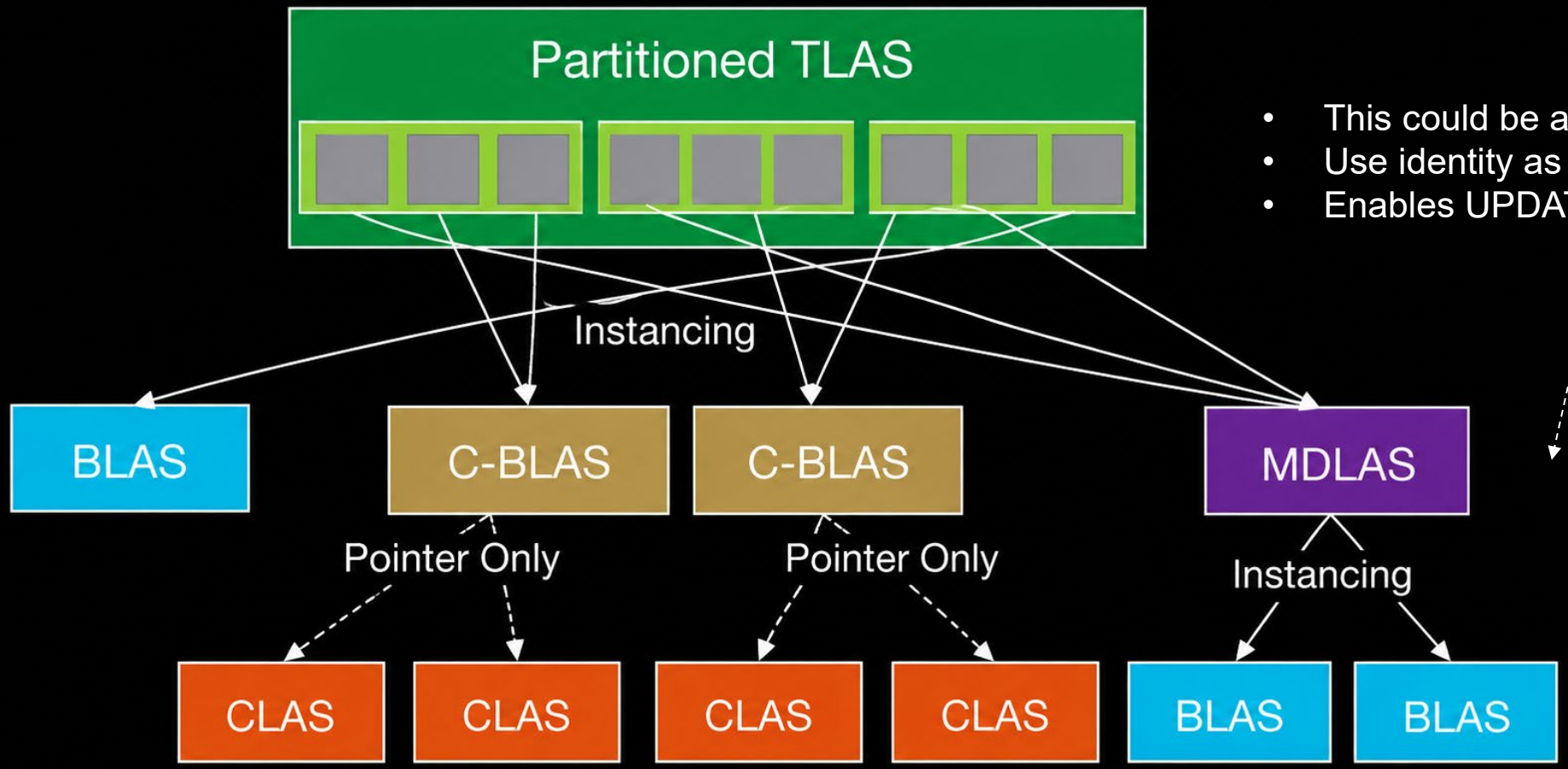
Public

Potential MLAS future: combine with DXR 2.0 C-BLAS & CLAS



Public

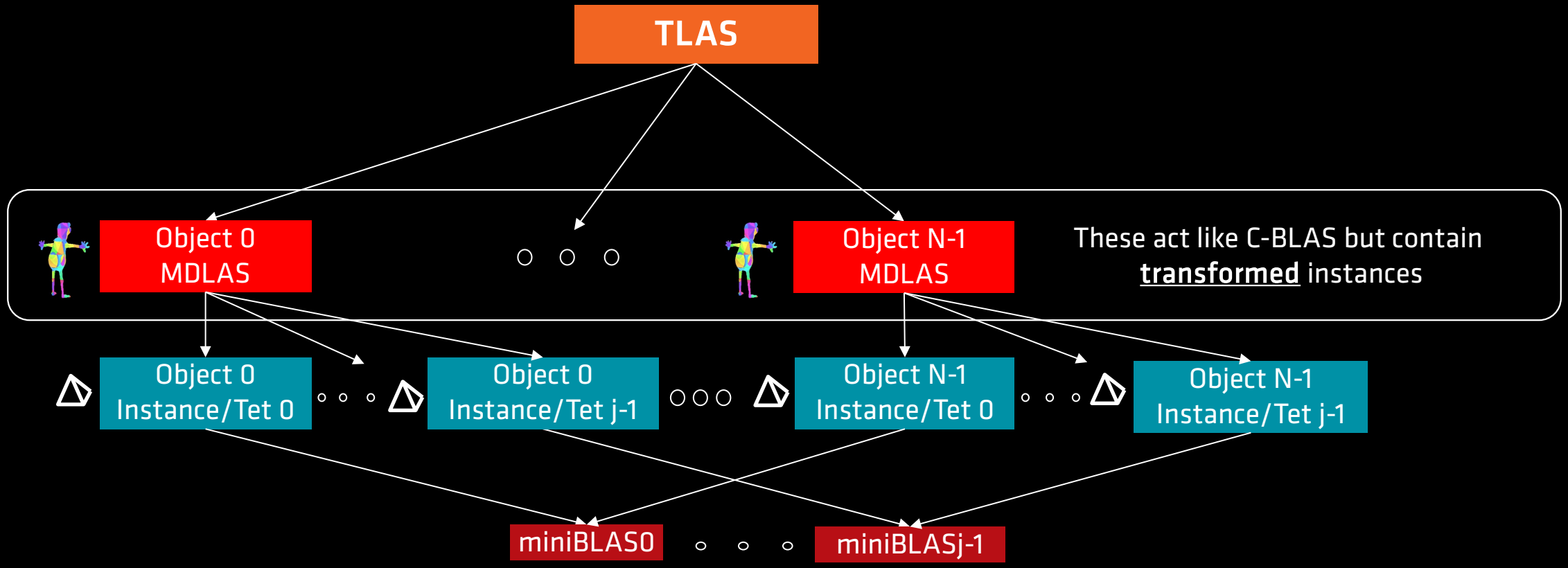
Potential MLAS future: combine with DXR 2.0 P-TLAS



- This could be a cage-animated object
- Use identity as the instance transform
- Enables UPDATE_INSTANCE

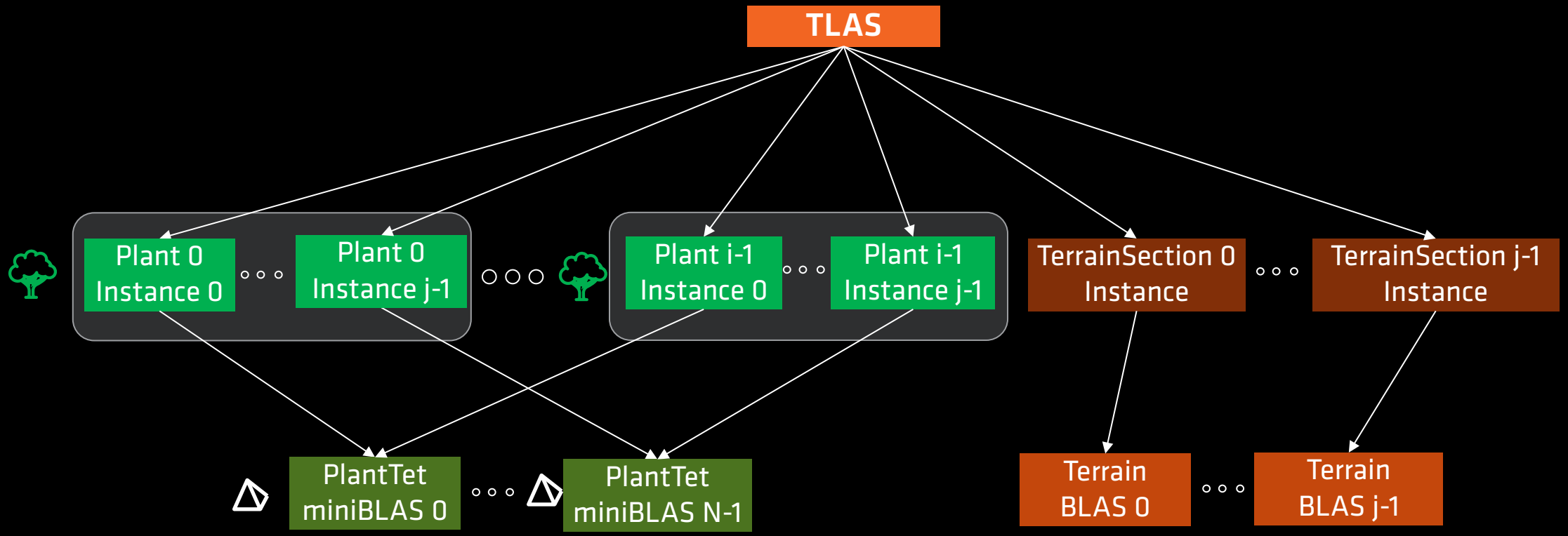
Public

MLAS And Tetrahedral Cage Animated Objects



Public

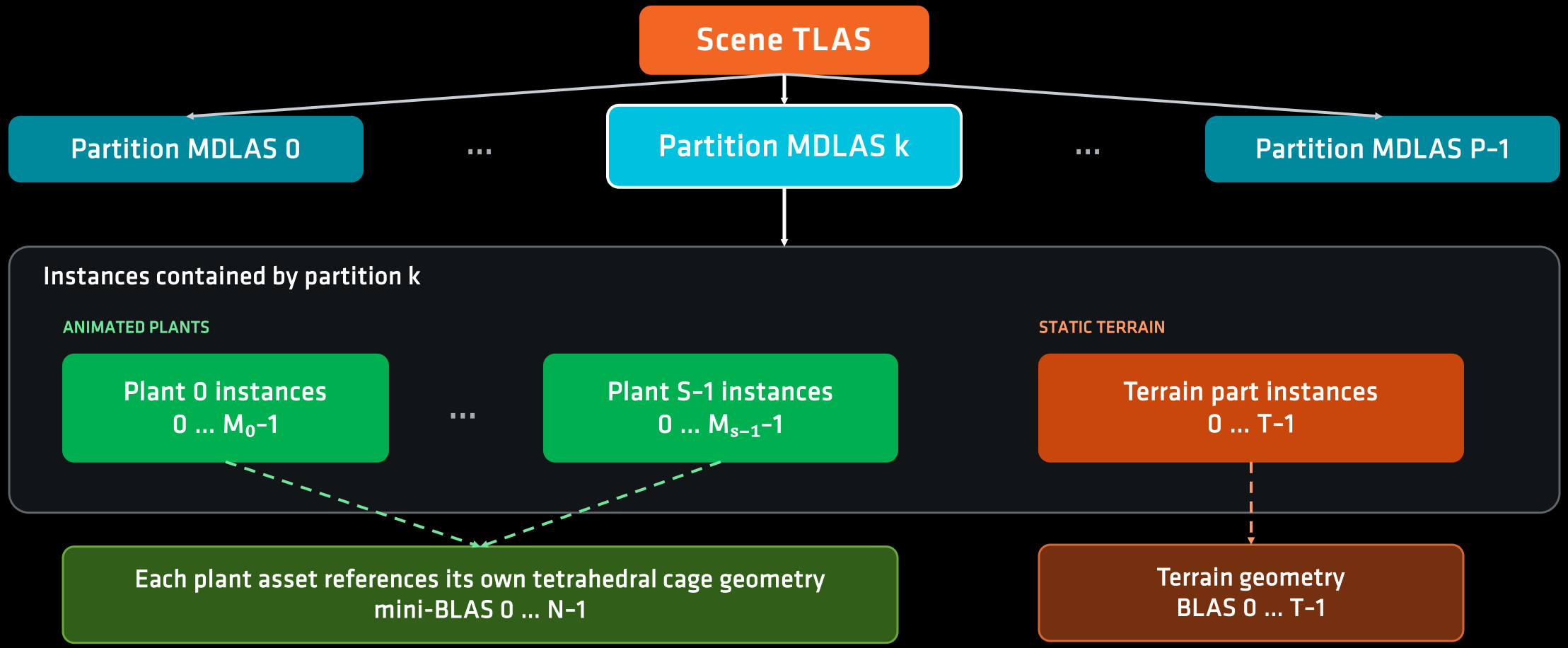
DXR 1.0 BVH of our plant animation demo



- This runs at 60+ FPS – but it is TLAS build limited!
- We need to rebuild the TLAS every frame as LOD changes happen all the time

Public

MLAS hierarchy in the plant animation demo



This runs with noticeably higher FPS and is now traversal limited!

Public

Why can MLAS generate higher FPS?

- No LOD change in a partition MDLAS => in-place update
- If LOD changes in a partition MDLAS => rebuild
  - LOD changes do not happen within all partitions every frame
- If >30% of all plants in a partition MDLAS have LOD0 => always rebuild it
  - This keeps the quality of the BVH in an OK state => traversal rate does not go down too much
- Can lead to noticeably higher FPS on average on RX 9070 XT
  - Results depend on scene complexity
- More control over what you rebuild can generate higher FPS

Public

MLAS or P-TLAS for Tetrahedra Cages?

- Tetrahedra Instances can be put into a MDLAS or a P-TLAS partition
  - You can do both depending on what works best
- Tetrahedron-Instances need to use WRITE_INSTANCE as the Xf changes
- You can put all Tetrahedra instances of an object into a MDLAS
  - This MDLAS can now be updated in-place in isolation or rebuilt
- A MDLAS with all Tetrahedra-instances of an object can be in a P-TLAS partition
  - If you use an identity transform for that MDLAS => you can use UPDATE_INSTANCE

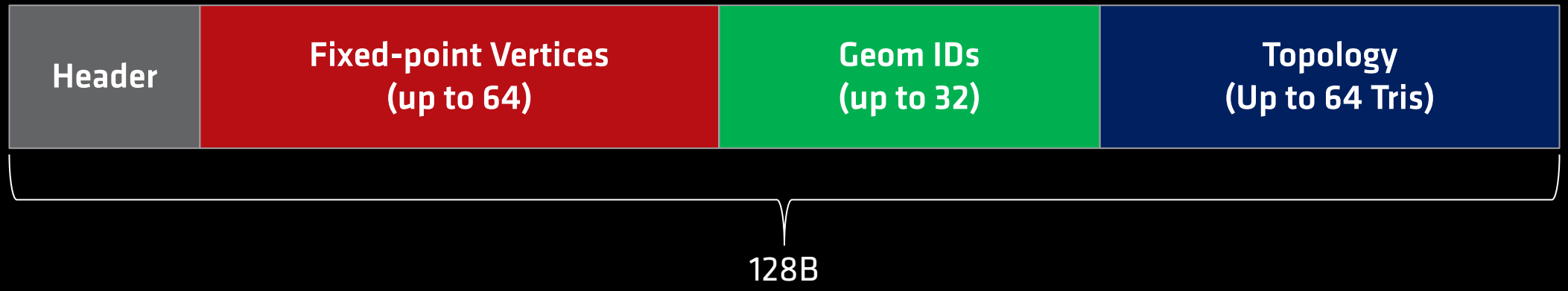
Public

AMD
together we advance_

Dense Geometry Format - DGF

Public

DGF: Block-compressed geometry



- Standardized block layout built offline
  - Aggressive, lossy compression
  - SAH-Optimized block formation

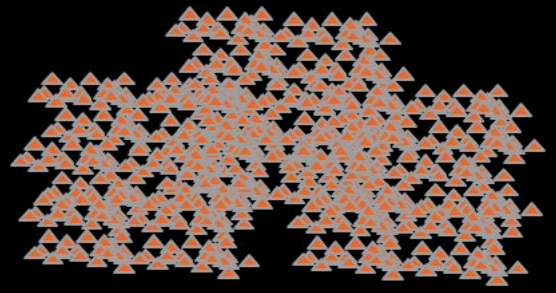
- Future AMD Architectures will have native HW support
  - SW Solutions available today
  - Cross-vendor Vulkan® extension in development

DGF is to geometry as DXT is to texture

Public

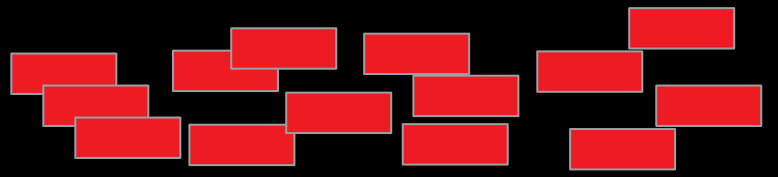
DGF: Block-compressed geometry

25M Triangles
(~18B/Tri)



Offline Encode

~700K SAH-Efficient Blocks
(~4B/Tri)



BuildRTAS

High Working Set
Expensive BLAS Build

BuildRTAS

Low Working Set
Cheaper BLAS Build

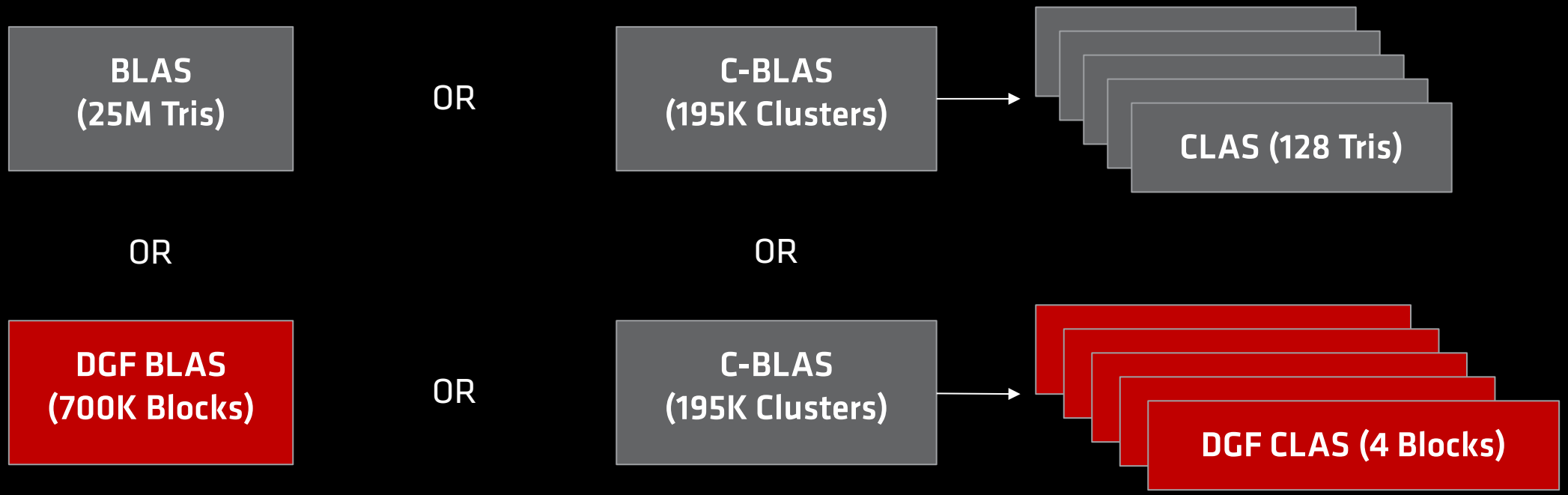
Public

DGF: Benefits

- Today (current HW and drivers)
  - Smaller disk footprints and download sizes
  - Smaller memory footprints for raster passes, RT hit shading
    - DGF blocks can replace VB/IB at some performance cost
- Future (native HW support)
  - Faster RTAS builds
  - Smaller RT {B|C}LAS
    - Compacted and uncompacted
  - Faster ray traversal
    - Smaller RTAS are good for the cache
    - Offline block construction improves SAH
  - HW-accelerated shader decode

Public

DGF is CLAS-compatible



Public

Fine-grained quality control

b ➔ User-specified target precision (in bits)



Uncompressed
(18 B/Tri)



b=16
(4.2 B/Tri)



b=14
(3.4 B/Tri)

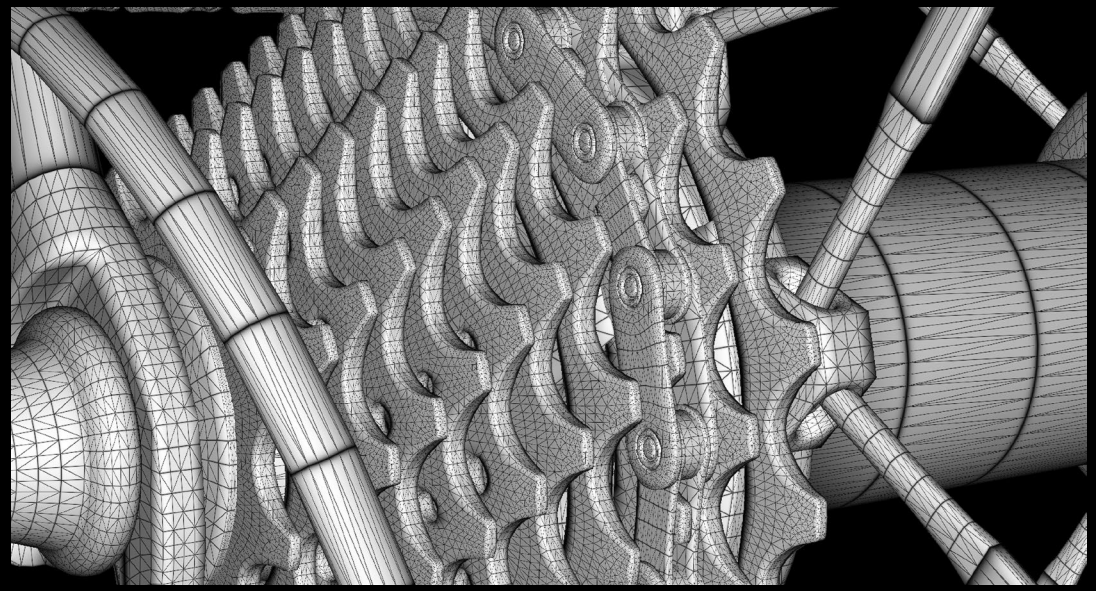


b=12
(2.9 B/Tri)

Public

Preserves mesh topology

- No dependency on complex remeshing/resampling
  - In practice, these are slow, invasive and difficult for artists to control
- Preserves important features of the original mesh
  - Sharp corners
  - Silhouettes
  - UV chart boundaries
  - “Locked” verts/edges for LOD

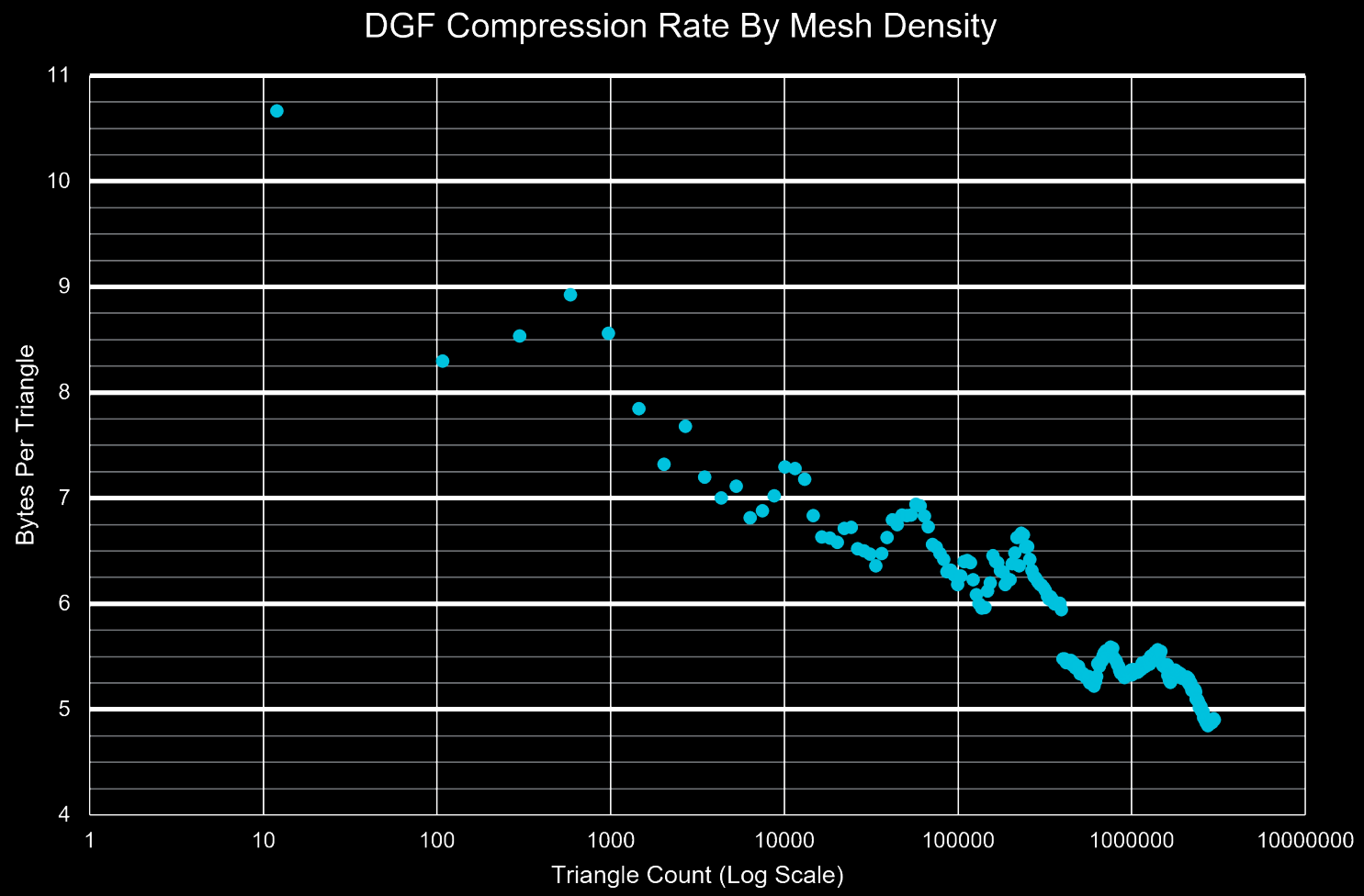


b=16
3.73 B/tri

Public

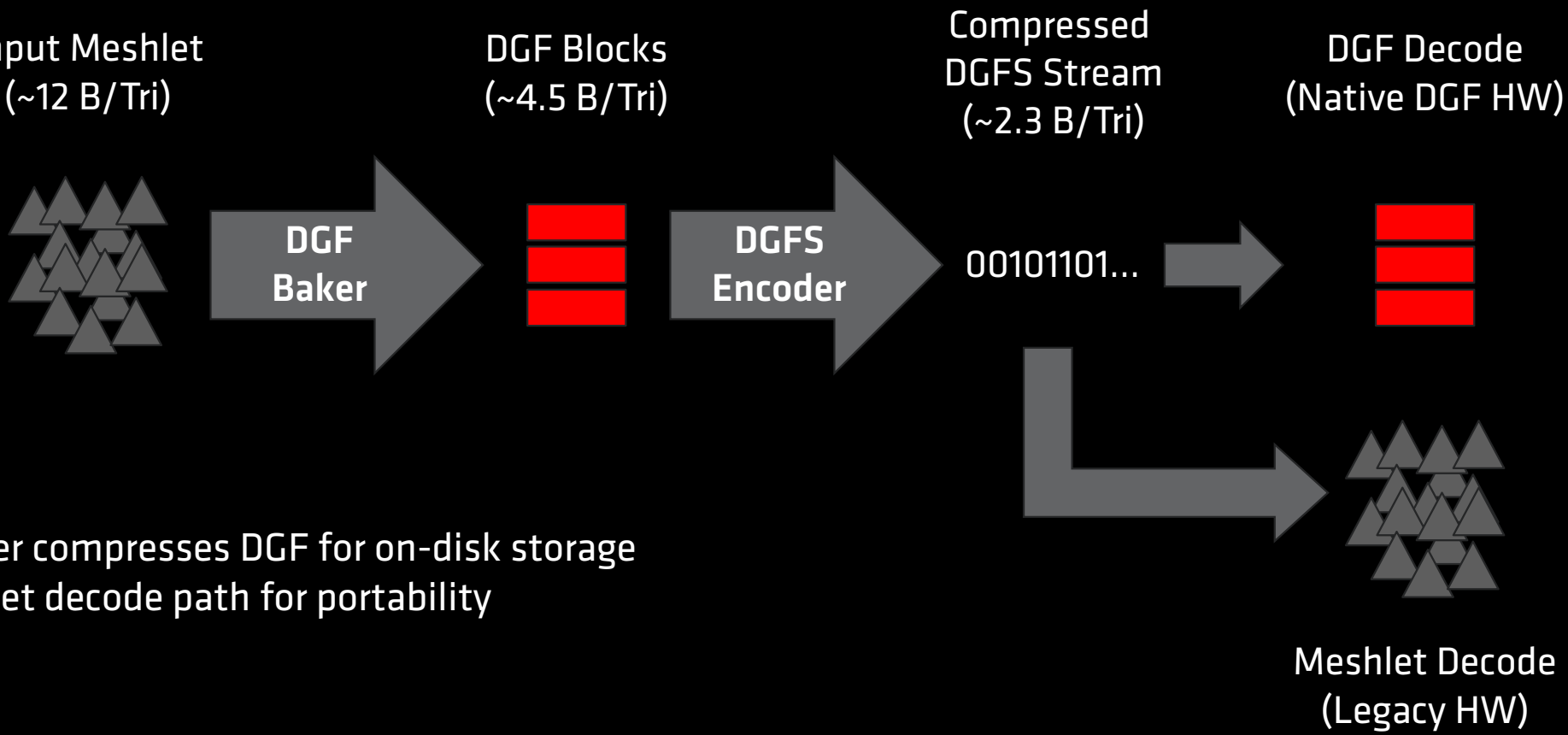
DGF compression rates

- Synthetic benchmark on a tessellated, rotated unit cube
  - Assumes b=16
- More triangles ➔ More profit
- Compare these data points with indexed meshes:
  - 18B/Tri (F32/U32)
  - ~12B/Tri (F32 Meshlet)
  - ~8B/Tri (F16 Meshlet)



Public

DGF SuperCompression (DGFS)



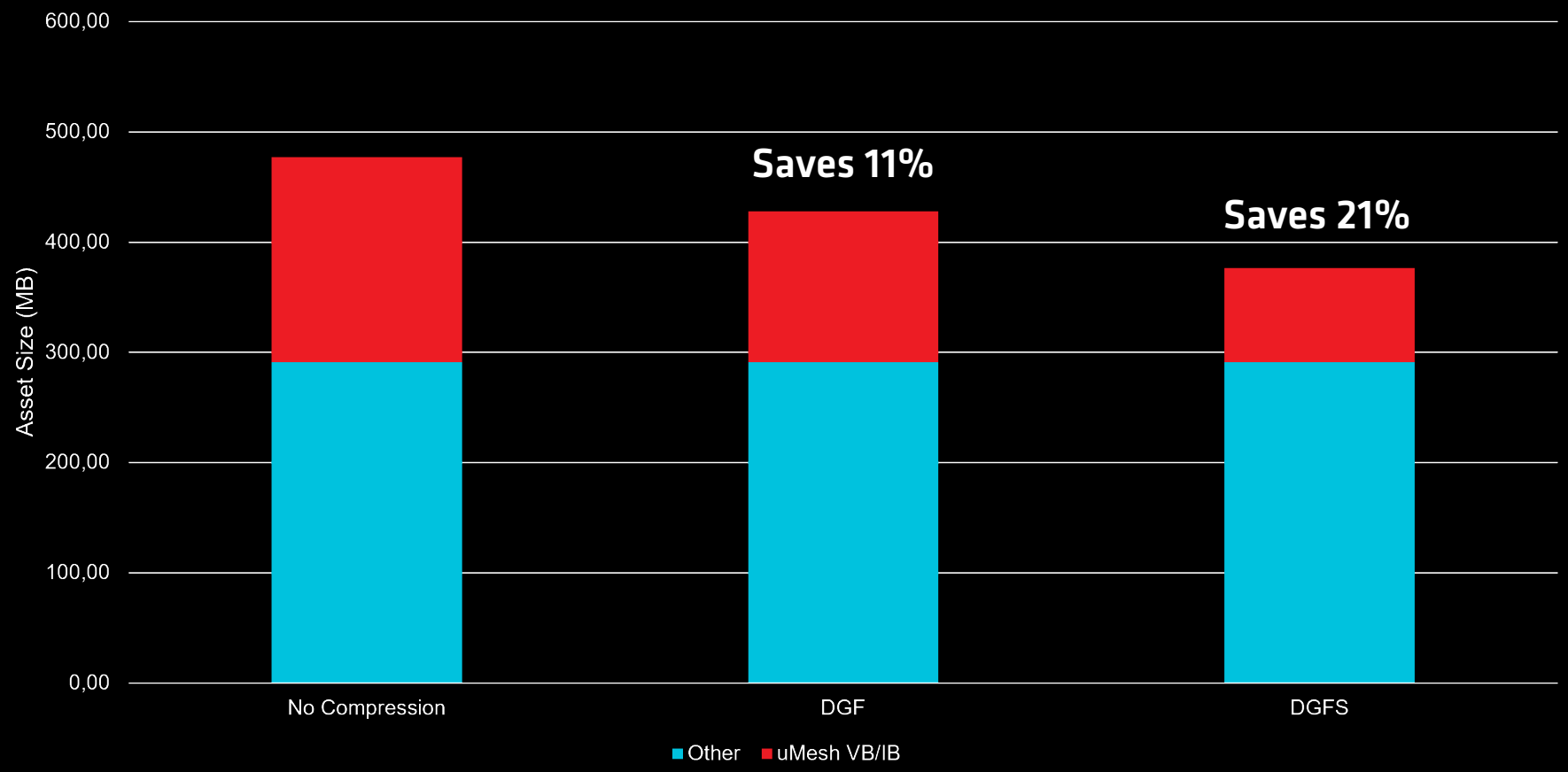
- Further compresses DGF for on-disk storage
- Meshlet decode path for portability

Public

AMD
together we advance_

DGF and Tet Cages

- DGF is ideal for static geometry… and our Tet BLAS are static ☺



Public

AMD
together we advance_

COPYRIGHT AND DISCLAIMER

© 2026 Advanced Micro Devices, Inc. All rights reserved.

AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

Vulkan and the Vulkan logo are registered trademarks of the Khronos Group Inc.

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate releases, for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

THIS INFORMATION IS PROVIDED 'AS IS." AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES


AMD