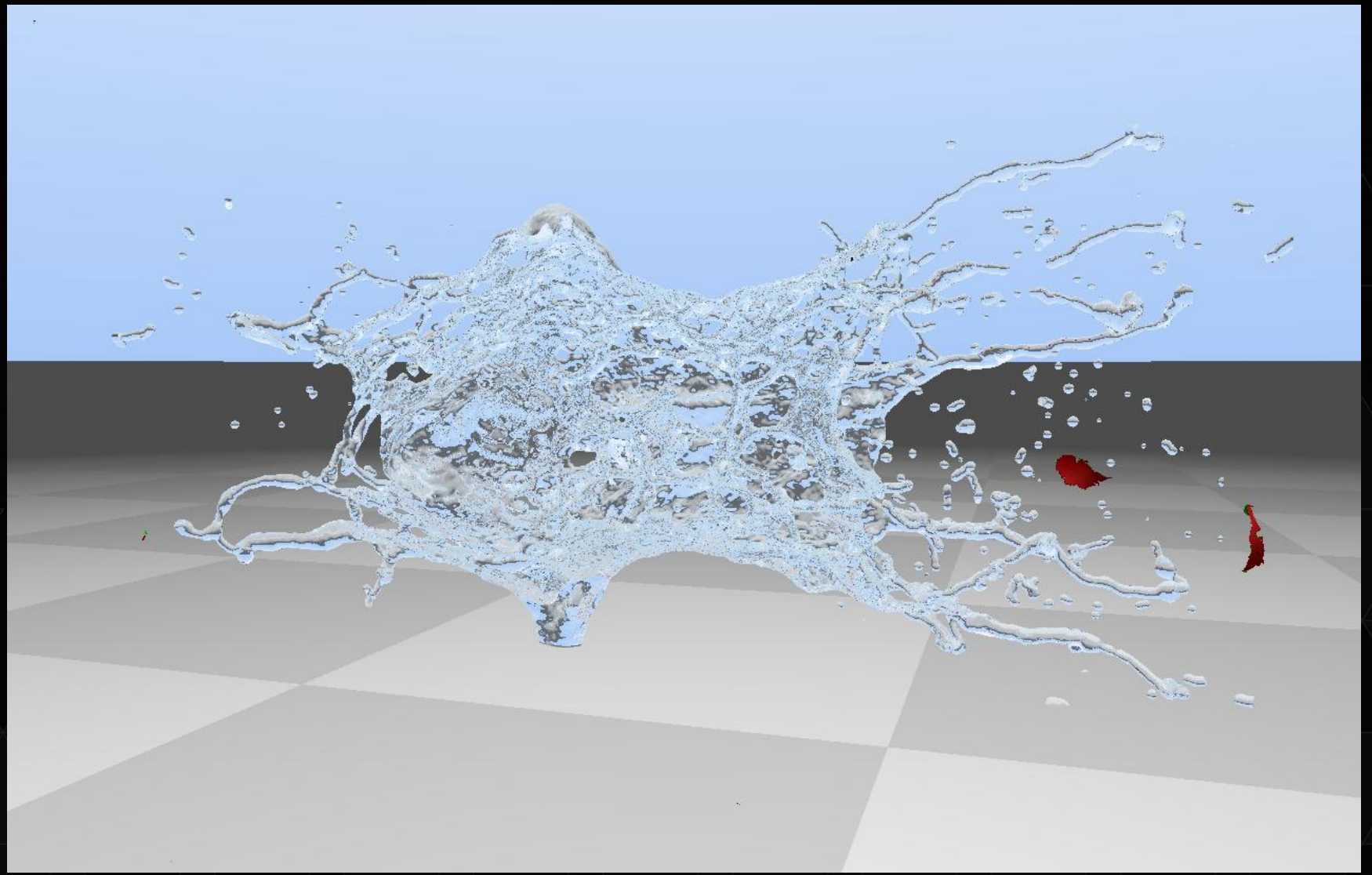
GPU TECHNOLOGY CONFERENCE

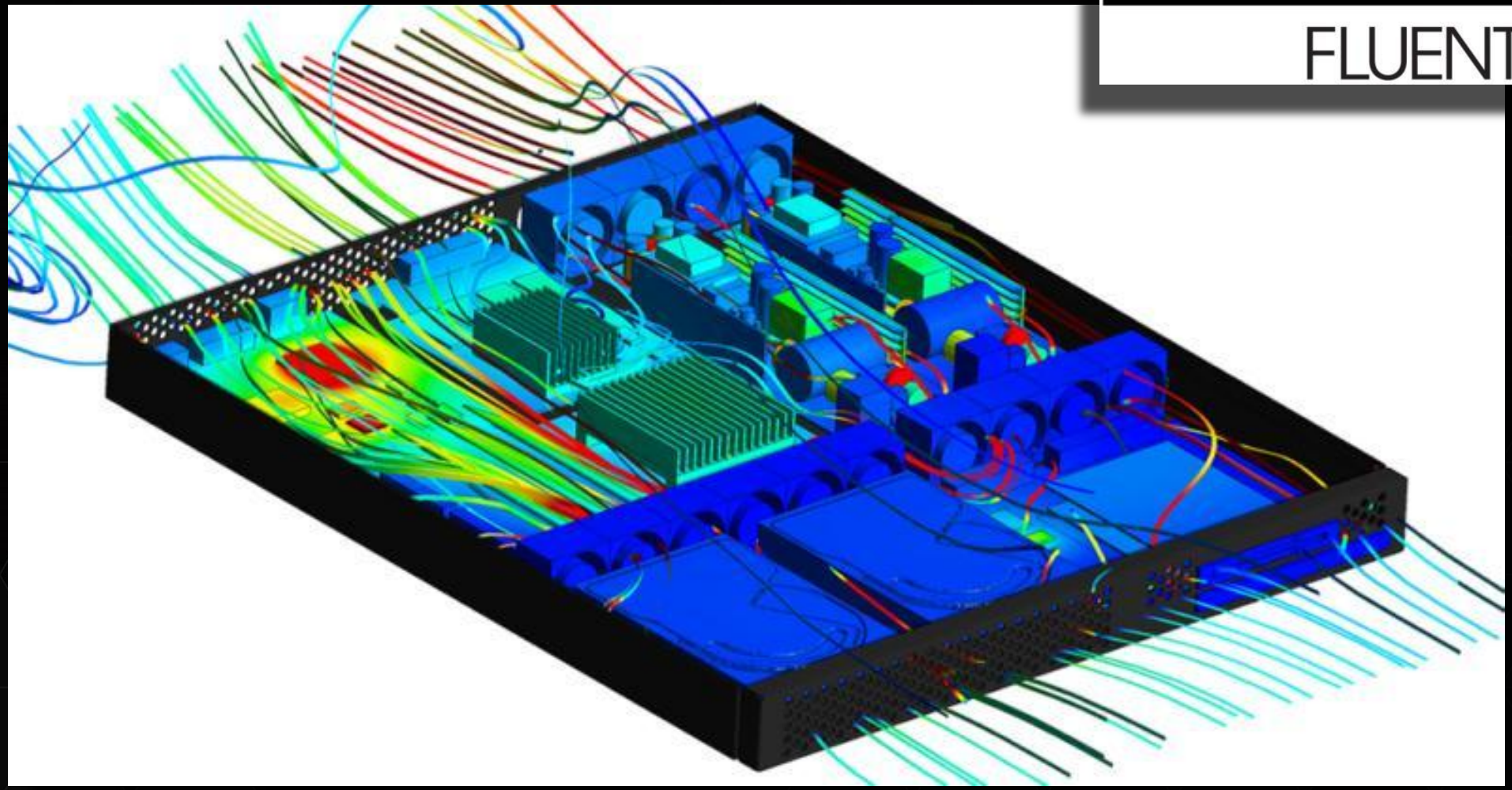
ADVANCES IN OPTIX

DAVID K. MCALLISTER, PH.D.

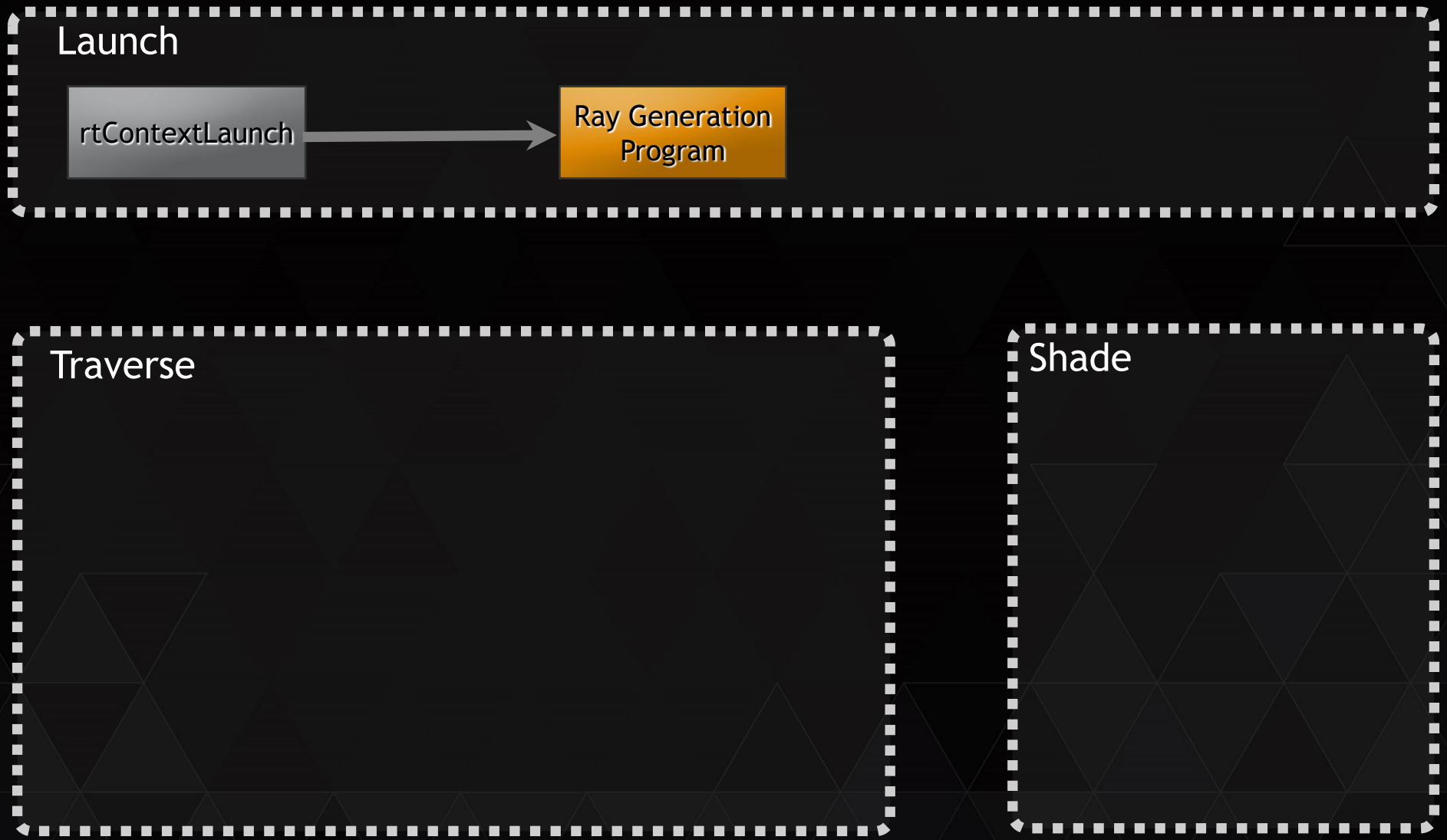
OPTIX MANAGER



ANSYS® FLUENT®

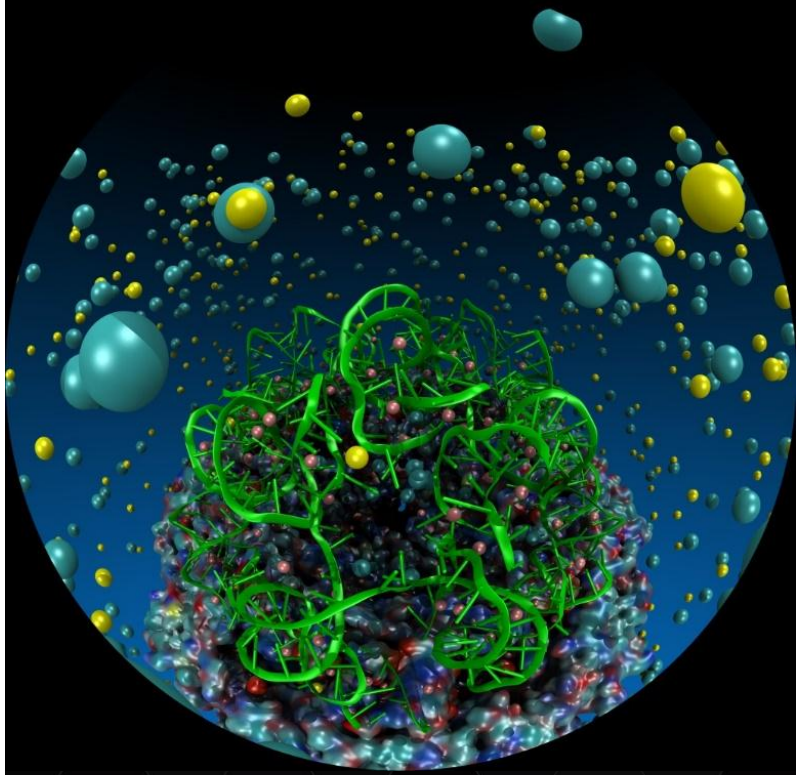


OPTIX EXECUTION MODEL

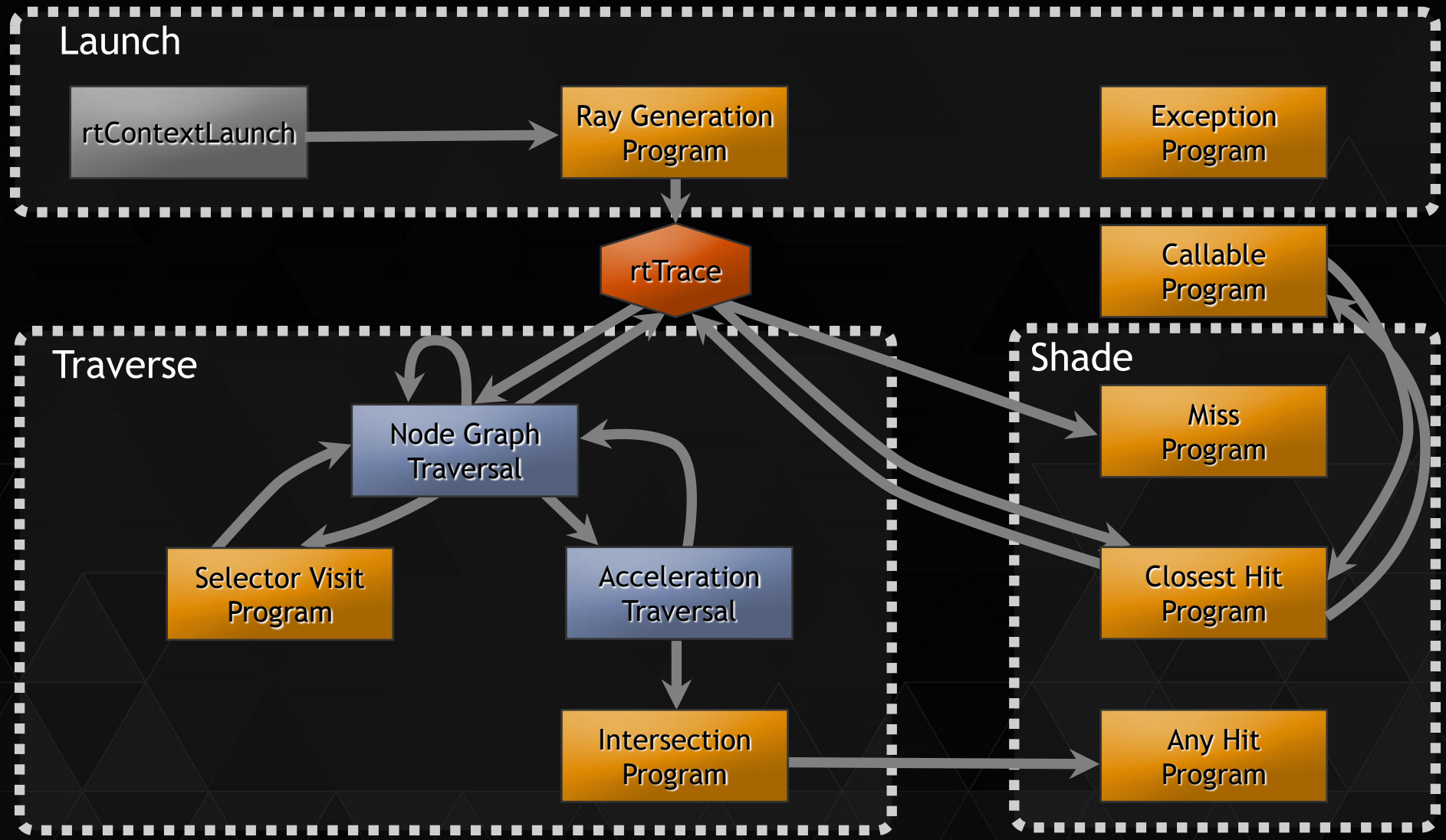


SAMPLE DEVICE CODE

```
RT_PROGRAM void dome_camera()
{
    size_t2 screen = output_buffer.size();

    float2 d = make_float2(launch_index) / make_float2(screen)
             * make_float2(2.0f, 2.0f) - make_float2(1.0f, 1.0f);
    float3 angle = make_float3(d.x, d.y, sqrtf(1.0f - (d.x*d.x + d.y*d.y)));
    float3 ray_origin = eye;
    float3 ray_direction = normalize(angle.x*normalize(U) +
                                     angle.y*normalize(V) +
                                     angle.z*normalize(W));

    optix::Ray ray(ray_origin, ray_direction, radiance_ray_type, scene_epsilon);

    PerRayData_radiance prd;
    prd.importance = 1.f;
    prd.depth = 0;

    rtTrace(top_object, ray, prd);

    output_buffer[launch_index] = make_color(prd.result);
}
```

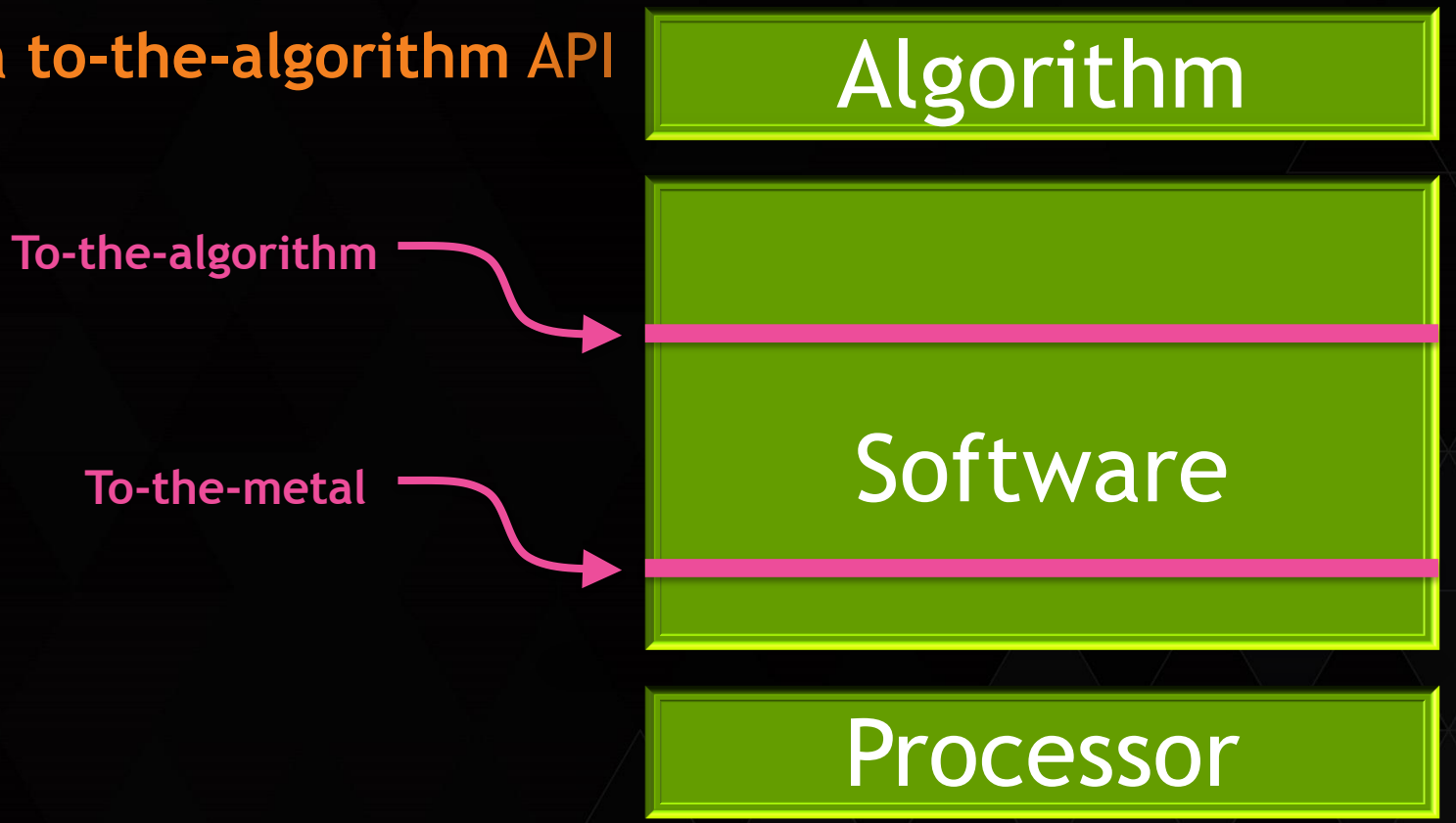


OPTIX EXECUTION MODEL



OPTIX ENCAPSULATES THE ALGORITHM

- OptiX is a **to-the-algorithm** API



GOLDENROD

MAJOR ARCHITECTURAL RENOVATION

- LLVM-based OptiX compiler

- Better GPU ray tracing performance
- More fluid interactive rendering
- Better multi-GPU scaling
- More efficient complex node graphs
- Additional input languages
- CPU backend

UNIFIED VIRTUAL MEMORY

- Merges CPU and GPU memory spaces
- Full read/write access from both processors
- Eliminates GPU memory footprint barrier
- Coming in Pascal architecture (2016)

OPTIX 3.7

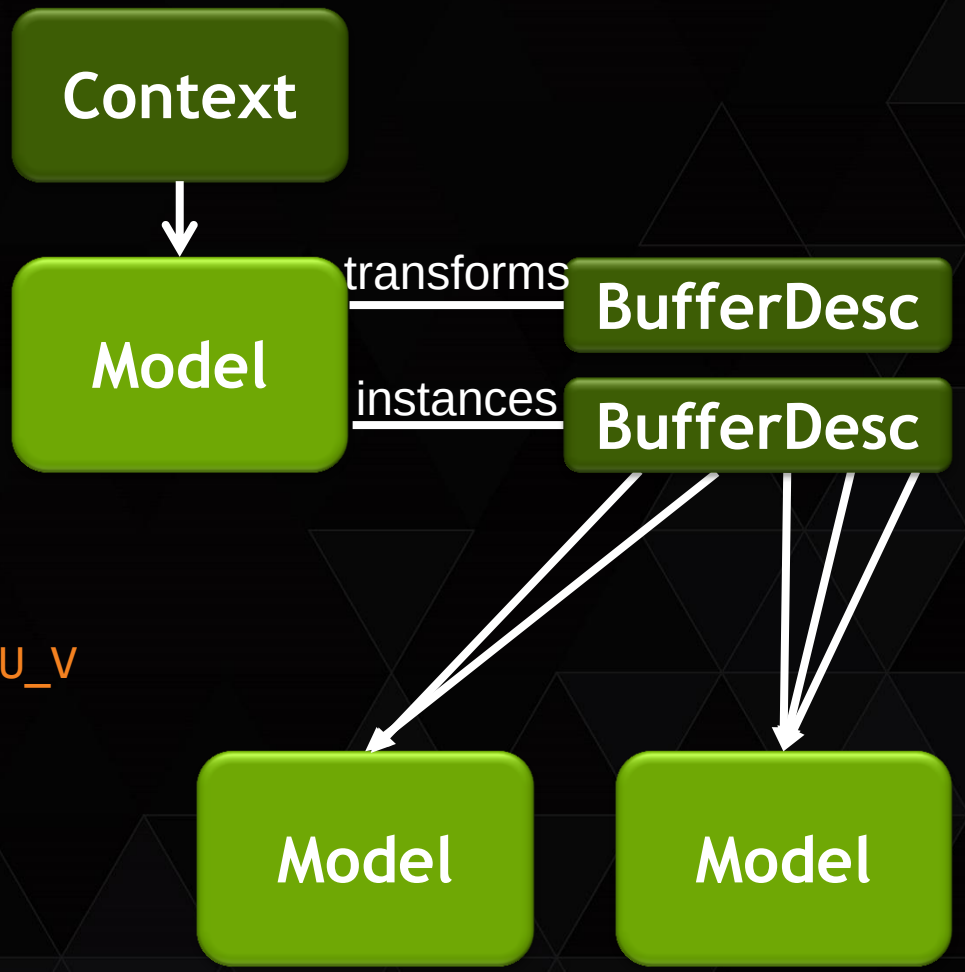
OPTIX PRIME

- Specialized for ray tracing
- Latest algorithms from NVIDIA Research
  - ray tracing kernels
  - Treelet Reordering BVH (TRBVH)
- Support for asynchronous computation
- CPU support

- No programing model support for shading
- No support for Quadro VCA
- No support for dynamic materials
- Triangles only
- No ability to target different architectures

INSTANCING IN PRIME

- A model is a set of instances:
  - `RTP_BUFFER_FORMAT_INSTANCE_MODEL`
  - `RTP_BUFFER_FORMAT_TRANSFORM_FLOAT4x3`
- New API call
  - `rtpModelSetInstances`
- Hit result formats
  - `RTP_BUFFER_FORMAT_HIT_T_TRIID_INSTID`
  - `RTP_BUFFER_FORMAT_HIT_T_TRIID_INSTID_U_V`

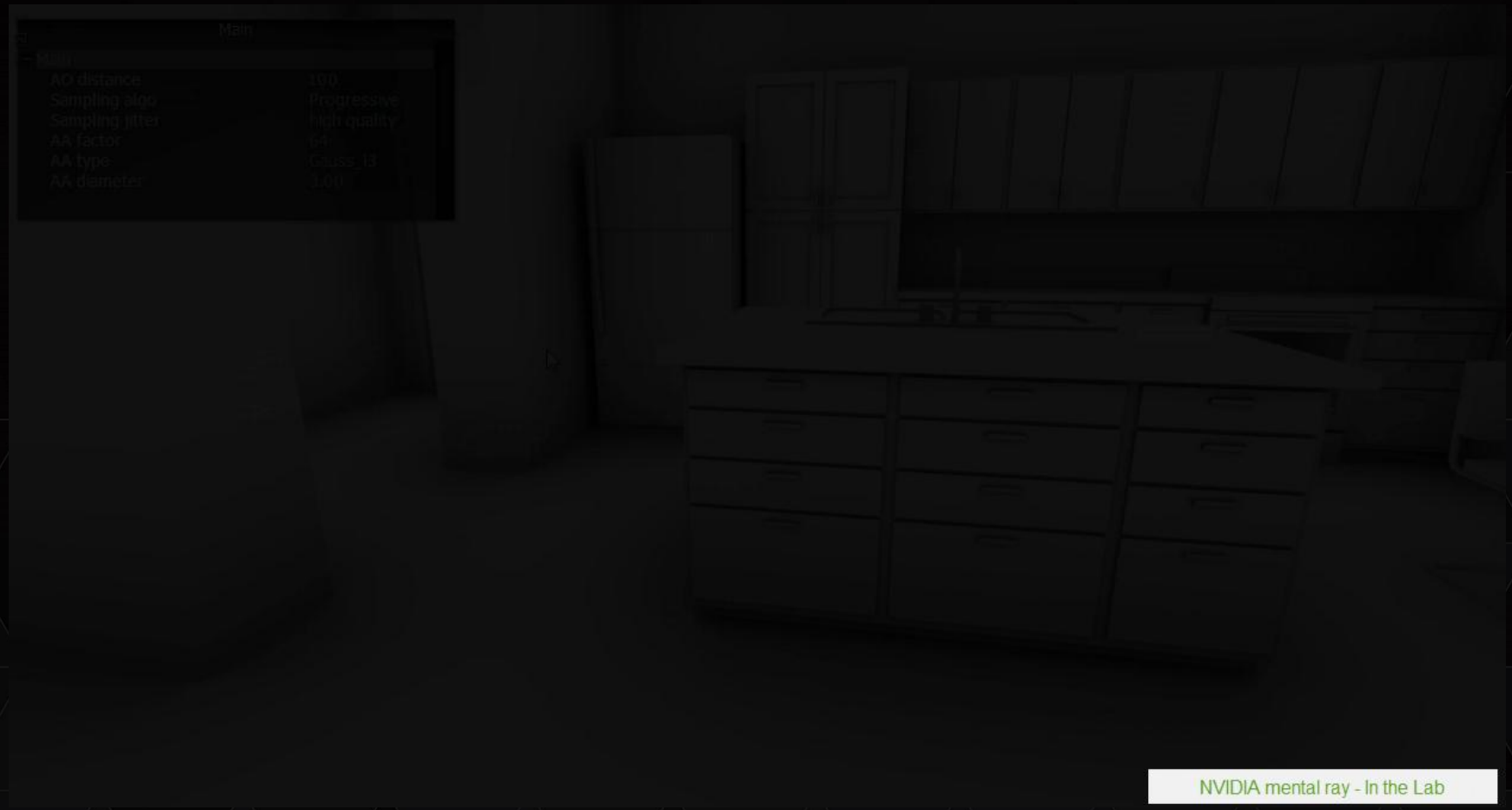


INSTANCING IN PRIME

```
std::vector<instInfo_t> instanceData;
std::vector<RTPmodel> instanceList;
std::vector<SimpleMatrix4x3> transformList;
createInstances(numInstances, models, instanceList, transformList, instanceData);

RTPbufferdesc instances, transforms;
rtpBufferDescCreate(context, RTP_BUFFER_FORMAT_INSTANCE_MODEL, RTP_BUFFER_TYPE_HOST, &instanceList[0], &instances);
rtpBufferDescSetRange(instances, 0, instanceList.size());

rtpBufferDescCreate(context, RTP_BUFFER_FORMAT_TRANSFORM_FLOAT4x3, RTP_BUFFER_TYPE_HOST, &transformList[0], &transforms);
rtpBufferDescSetRange(transforms, 0, transformList.size());

RTPmodel scene;
rtpModelCreate(context, &scene);
rtpModelSetInstances(scene, instances, transforms);
```

GPU

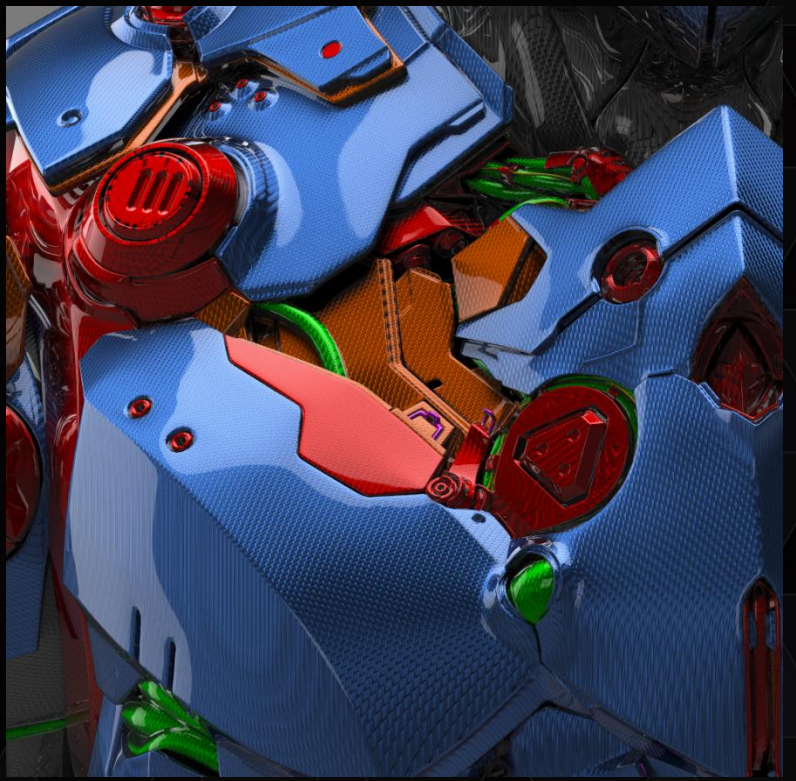
OPTIX PRIME IN MENTAL RAY 3.12



OPTIX 3.8

PROGRESSIVE API

- Render all subframes in a single API call
- Encapsulate even more of the algorithm



STREAM BUFFERS

```
RTbuffer output_buffer, stream_buffer;
rtBufferCreate(context, RT_BUFFER_OUTPUT, &output_buffer);
rtBufferCreate(context, RT_BUFFER_PROGRESSIVE_STREAM, &stream_buffer);

rtBufferSetSize2D(output_buffer, width, height);
rtBufferSetSize2D(stream_buffer, width, height);
rtBufferSetFormat(output_buffer, RT_FORMAT_FLOAT4);
rtBufferSetFormat(stream_buffer, RT_FORMAT_UNSIGNED_BYTE4);

rtBufferBindProgressiveStream(stream_buffer, output_buffer);
```

PROGRESSIVE API

```
rtContextLaunchProgressive2D(context, width, height, num_subframes);

while(!finished) {
  int ready;
  rtBufferGetProgressiveUpdateReady(stream_buffer, &ready, 0, 0);

  if(ready) {
    rtBufferMap(stream_buffer, &data);
    display(data);
    rtBufferUnmap(stream_buffer);
  }

  if(scene_changed()) {
    // Update OptiX state
    rtVariableSet(...);
  }

  rtContextLaunchProgressive2D(context, width, height, num_subframes);
}
```

PROGRESSIVE API (DEVICE)

```
rtDeclareVariable(unsigned int, subframe_idx, rtSubframeIndex, );

unsigned int seed = rand_seed(launch_index, frame, subframe_idx);
```

Quadro VCA

Under the Hood

GPUs	8 x M6000-VCA GPUs
GPU Memory	12 GB per GPU
CUDA Cores	23,040
CPU Cores	20 Physical
System Memory	256 GB
Storage	4 x 512GB SSD
Network	2 x 1GigE 2 x 10GigE (SFP+) 1 x InfiniBand
Installed Software	Iray IQ + Cent OS Linux + VCA Cluster Manager
U.S. MSRP	$50,000

OptiX App

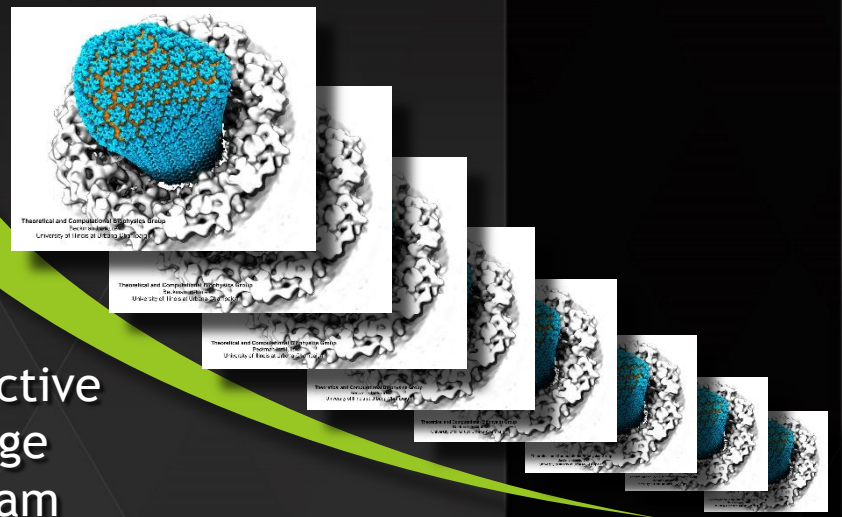
Incremental Updates

Ethernet or Internet

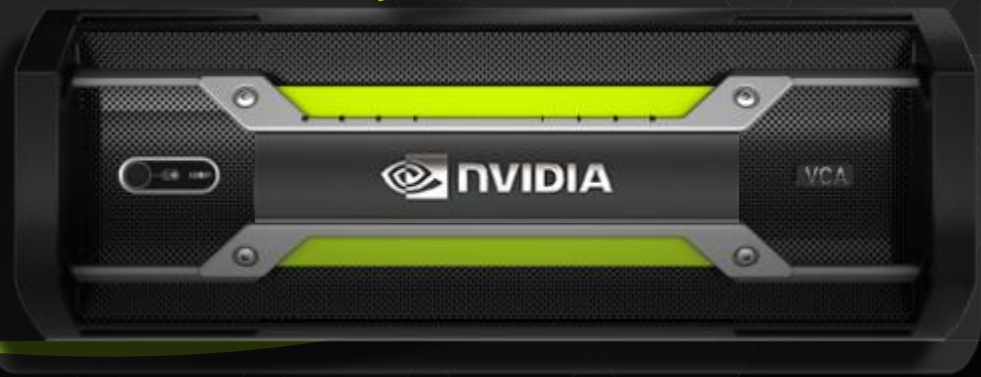
Custom OptiX Applications
All Processing on VCA

OptiX Leveraging
Same Infrastructure as Iray
(using DiCE)

Minimal Work
within the OptiX App



Interactive Image Stream

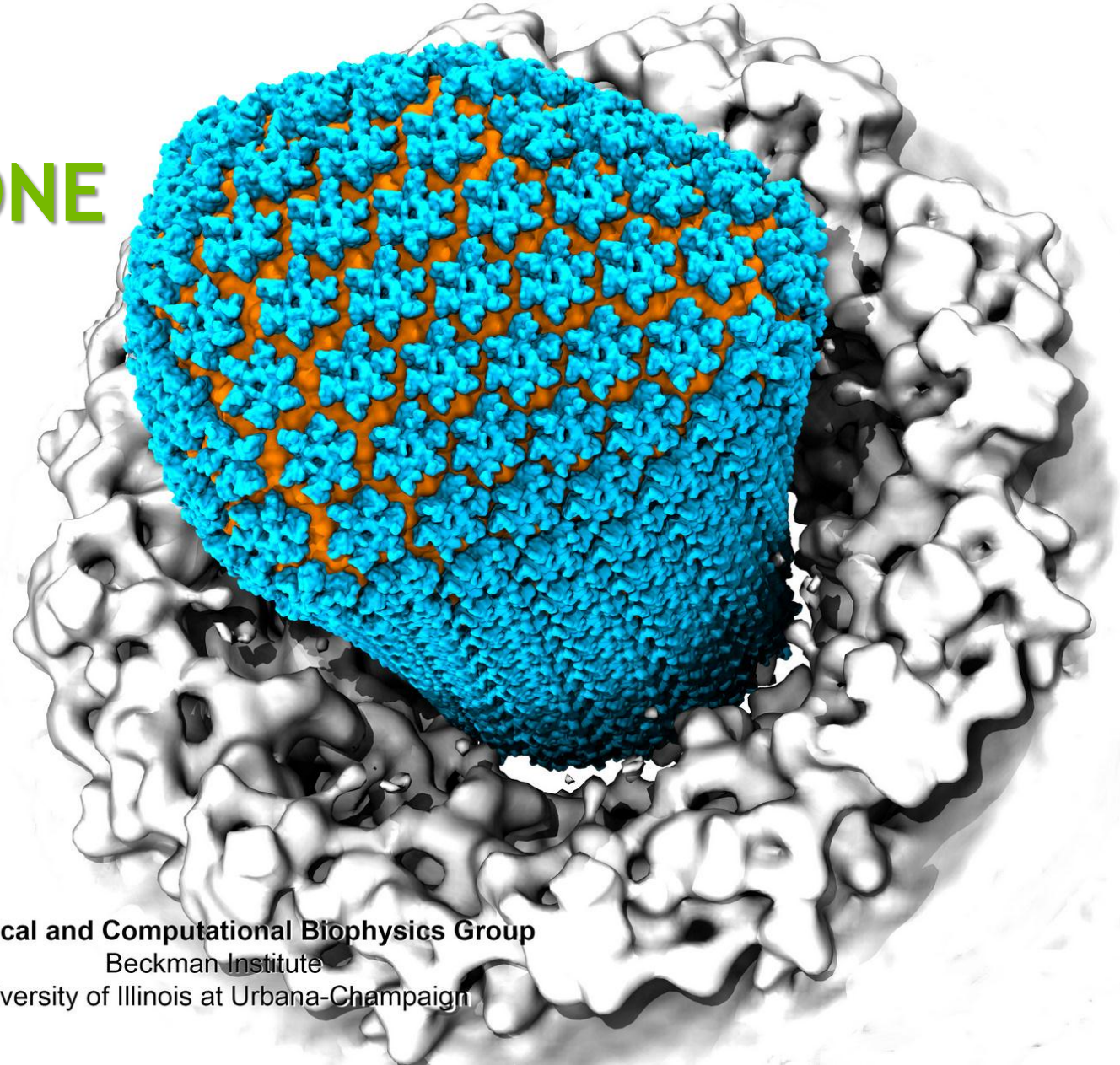


CONNECTION API

```
RTremotedevice rdev;
rtRemoteDeviceCreate("url", "user", "password", &rdev));

unsigned int num_configs;
rtRemoteDeviceGetAttribute(rdev, RT_REMOTEDEVICE_ATTRIBUTE_NUM_CONFIGURATIONS,
sizeof(unsigned int), &num_configs);

int vca_config_index = chooseConfig(num_configs);

rtRemoteDeviceReserve(rdev, vca_num_nodes, vca_config_index);

int ready;
do {
  rtRemoteDeviceGetAttribute(*rdev, RT_REMOTEDEVICE_ATTRIBUTE_STATUS, sizeof(int), &ready);
  if(ready != RT_REMOTEDEVICE_STATUS_READY) sleep(10);
} while(ready != RT_REMOTEDEVICE_STATUS_READY);

rtContextCreate(context);
rtContextSetRemoteDevice(*context, rdev));
```

GPU TECHNOLOGY CONFERENCE

JOHN STONE



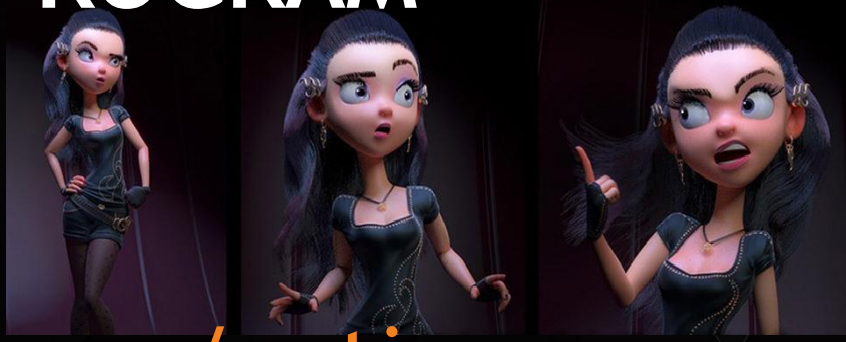
Theoretical and Computational Biophysics Group
Beckman Institute
University of Illinois at Urbana-Champaign

QUADRO VCA ON DEMAND

- Easily enable VCA in your OptiX application.
- Anyone can get time on a VCA.
- Coming soon.

REGISTERED DEVELOPER PROGRAM

- Access latest OptiX version
- Access private beta releases
- Tighter communication with OptiX developers
- https://developer.nvidia.com/optix



GPU TECHNOLOGY CONFERENCE

MORE OPTIX TALKS

Session	Title	Day	Start	End	Room	Speaker
S5659	Accelerating Mountain Bike Development with Optimized Design Visualization	Tuesday	13:30	13:55	LL21A	Geoff Casey
S5188	FurryBall RT: New OptiX Core and 30x Speed Up	Tuesday	15:00	15:25	LL21D	Jan Tománek
S5643	Advanced Rendering Solutions from NVIDIA	Tuesday	15:30	16:20	LL21E	Phillip Miller
S5622	Dekko: A Framework for Real-Time Preview for VFX	Wednesday	9:30	9:55	LL21D	Damien Fagnou
S5644	Flexible Cluster Rendering with NVIDIA VCA	Wednesday	10:00	10:50	LL21E	Phillip Miller
S5541	CATIA Live Rendering Iray and NVIDIA VCA	Wednesday	10:00	10:50	LL21A	Pierre Maheut
S5409	Custom Iray Applications and MDL for Consistent Visual Appearance	Wednesday	14:00	14:50	Ll21E	Dave Hutchinson
S5246	Innovations in OptiX	Wednesday	15:00	15:50	LL21E	David McAllister
S5628	Simulation-Based CGI for Automotive Applications	Wednesday	16:00	16:25	LL21A	Benoit Deschamps
S5386	VMD: Publication-Quality Ray Tracing of Molecular Graphics with OptiX	Thursday	9:00	9:25	LL21E	John Stone
S5416	Accelerad: Daylight Simulation for Architectural Spaces Using GPU Ray Tracing	Thursday	14:00	14:25	LL21E	Nathaniel Jones
S5210	GPU-Accelerated Spectral Caustic Rendering of Homogeneous Caustic Objects	Thursday	14:30	14:55	LL21E	Budianto Tandianus

JOIN THE CONVERSATION

#GTC15