AIMOTIVE

Optimal scheduling in a multi-GPU environment

2019/07/11

Agenda

- Simulation for self-driving
  - Limitations of game engines
- Synthesizing camera images
- Sensor system and scheduling
  - CPU Scheduler
  - GPU Scheduler

Simulation for self-driving

Simulation for self-driving

- Safe operation of self-driving systems requires large-scale testing
- Huge distances must be covered in various road conditions and environments
- Limited testing possibilities in the real world
- Simulators provide great tools to satisfy these requirements

Simulation for self-driving

- Simulators must be comprehensive and robust :
  - Diversity of maps, environments, conditions and driving cultures
  - Repeatability of tests and scenarios
  - Pixel-precise deterministic rendering
  - Physical realism
  - Ready-access for self-driving developers and engineers
  - Efficient use of hardware resources, from laptops to servers

Limitations of game engines

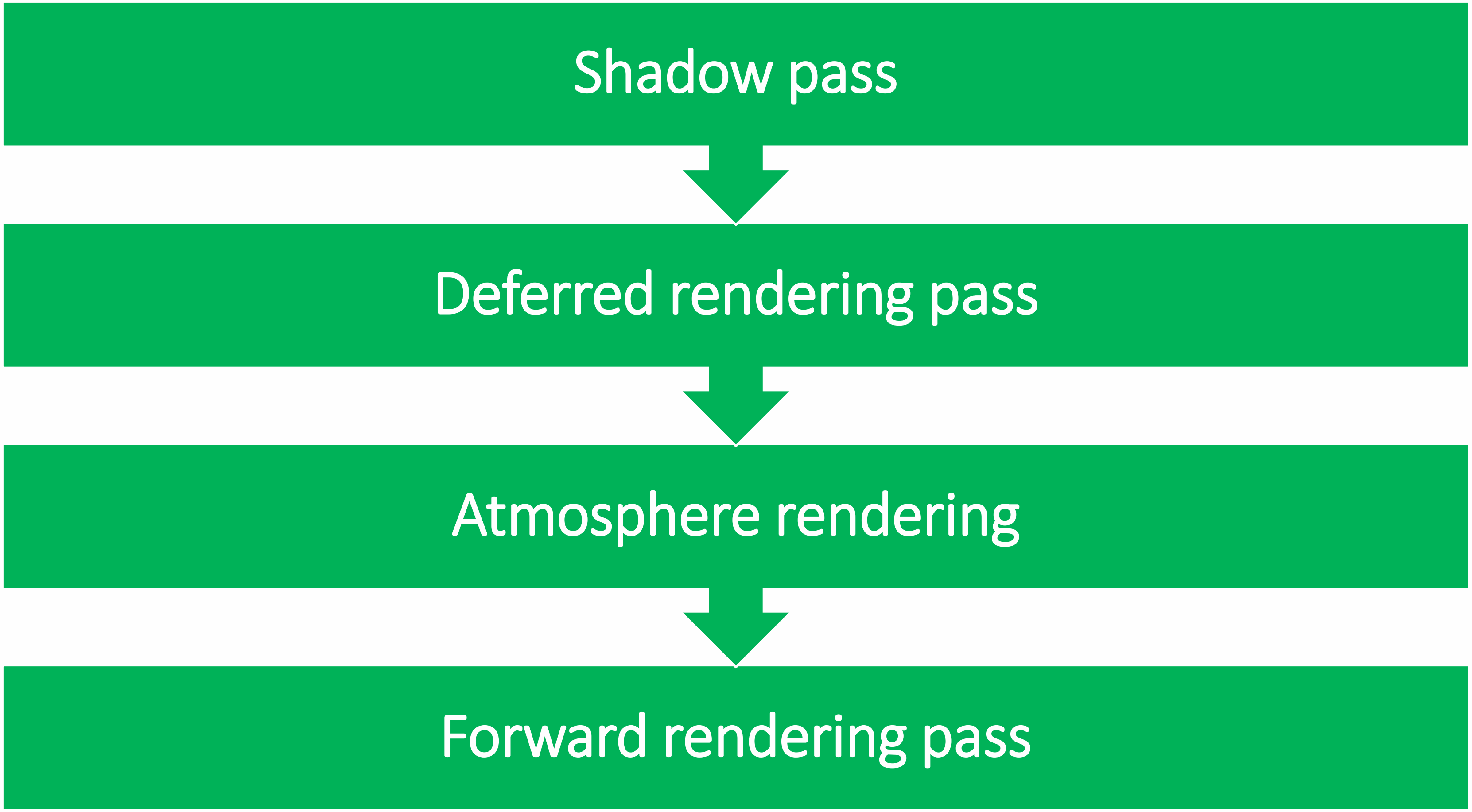
- Several problems encountered with first, game engine based simulator
  - Rendering images produced by ultra-wide and narrow camera lenses required certain modifications
  - Performance issues
  - Artifacts occurred in post-process effects
  - No support for using multiple GPUs
- These specific demands cannot be answered efficiently by game engines
- First iteration supported formulating the specifications mentioned
  - Especially for the sensor system and camera pipeline

Synthesizing camera images

Synthesizing camera images

- Problem of simulating lenses
- GPU rasterization-based rendering pipeline
  - Ray-tracing might be an option for simulating lenses in the future
- Vulkan based graphics backend
  - Multi-platform
  - Multi-GPU
- Complex rendering pipeline
  - First phase: environment capture (PBR pipeline)
  - Second phase: camera lens distortion
    - Pinhole, Fisheye

Environment capture



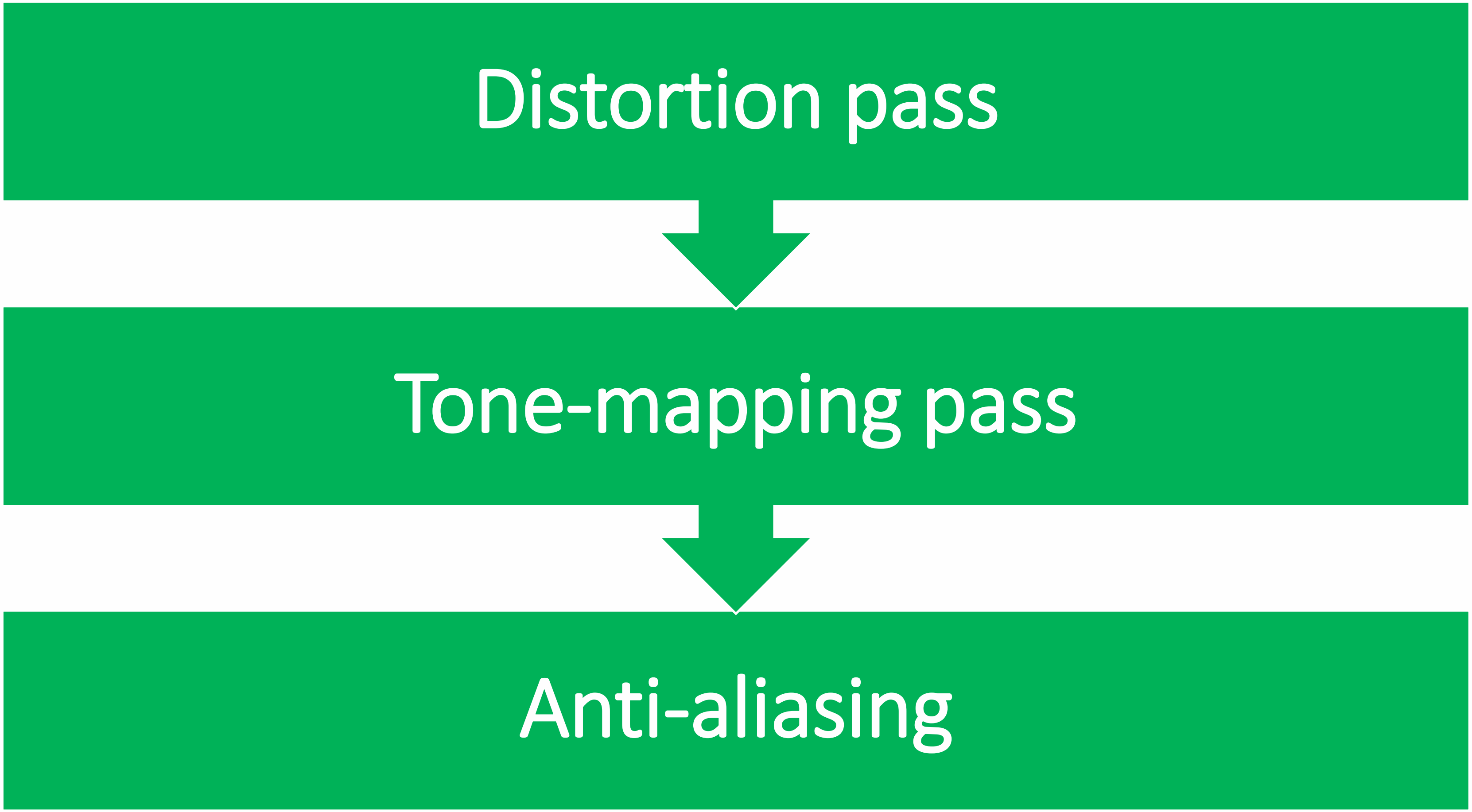
Shadow pass

Deferred rendering pass

Atmosphere rendering

Forward rendering pass

Distortion and post-process



Distortion pass

Tone-mapping pass

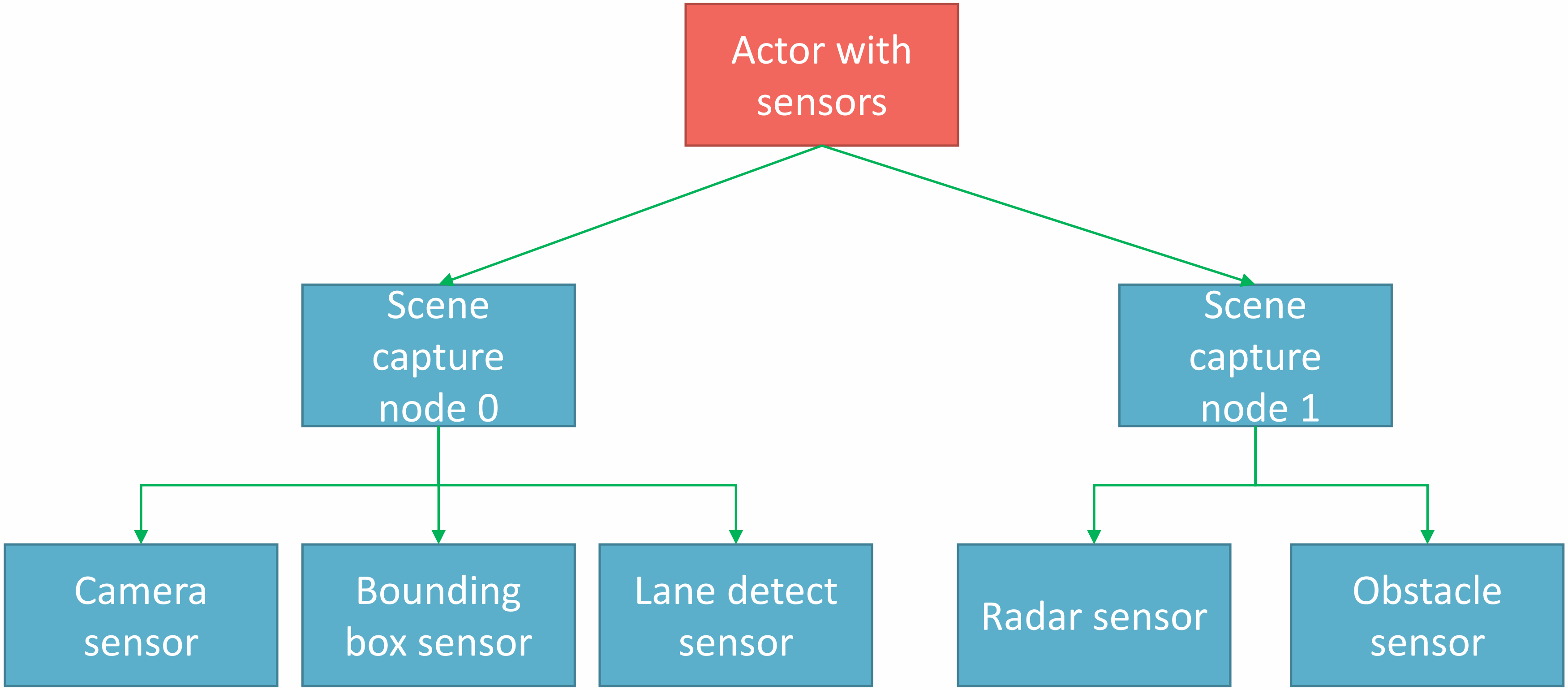
Anti-aliasing

Sensor system and scheduling

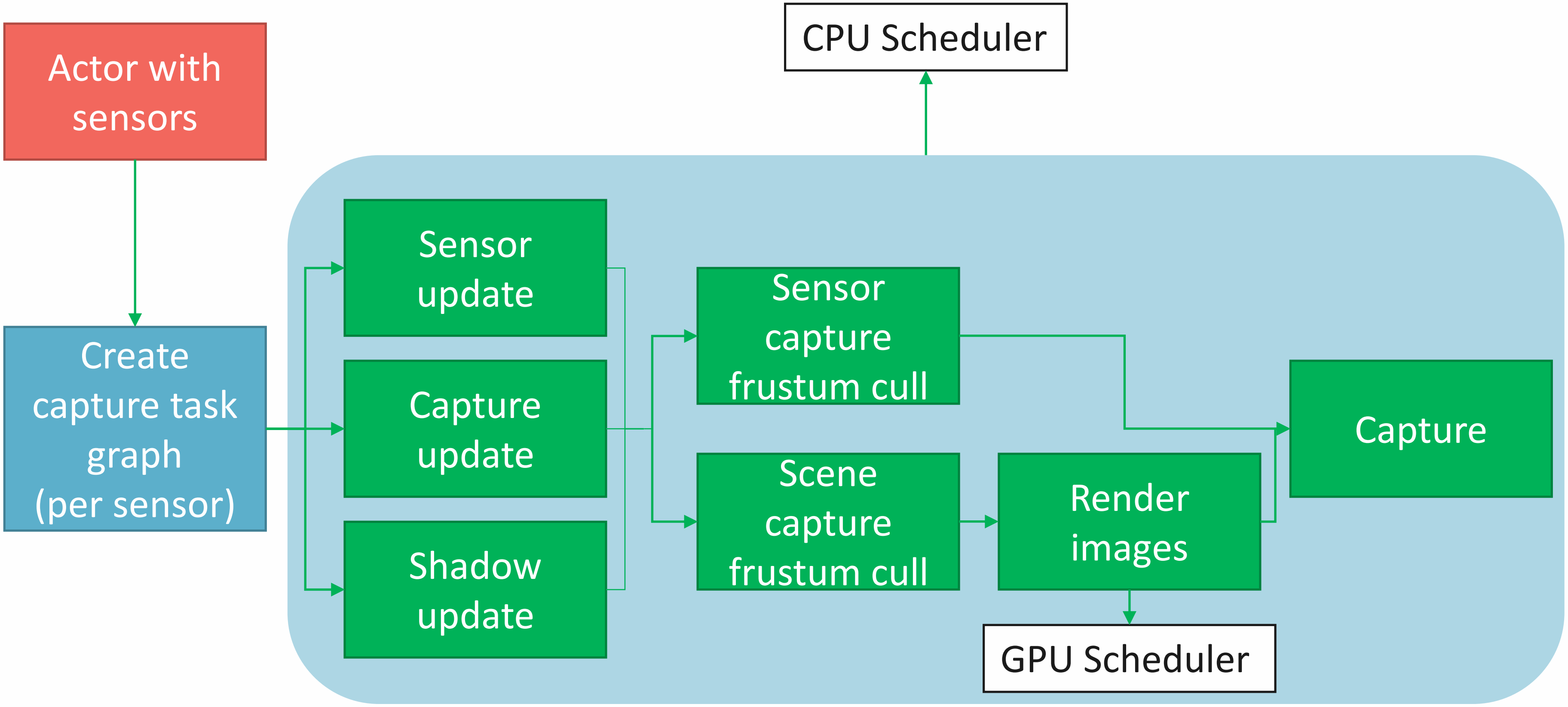
Sensor system and scheduling

- Our simulated world is a graph
  - Virtual objects
  - Hierarchy of various types of nodes (actor-, capture-, mesh-, etc. nodes)
- Basic concept of the sensor system
  - Scene capture nodes can be attached to actor nodes
  - These capture nodes provide data for their sensor nodes
- Resource management
  - Executing sensor tasks on CPU cores
  - Distributing rendering tasks among multiple GPUs
  - We implemented CPU- and GPU schedulers for this purpose

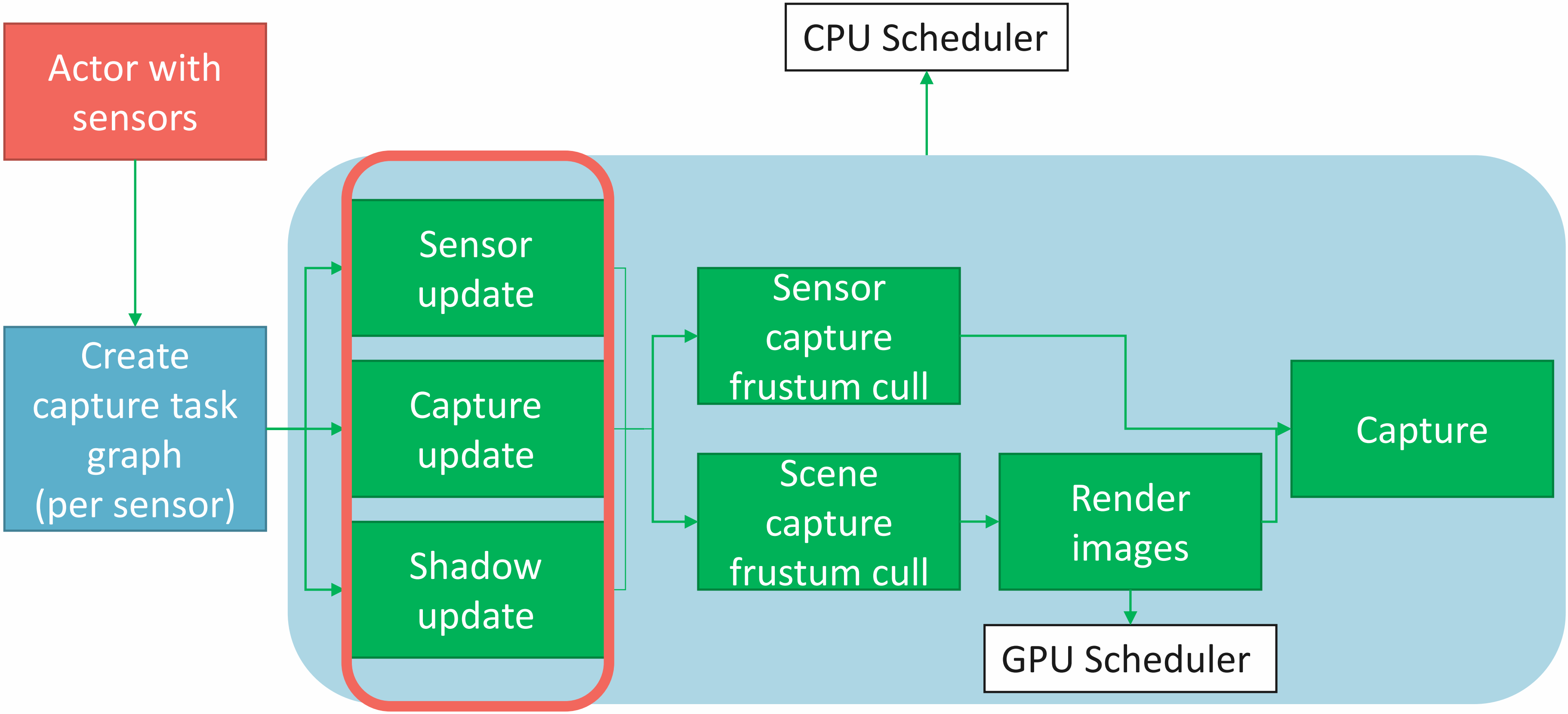
Typical sensor setup



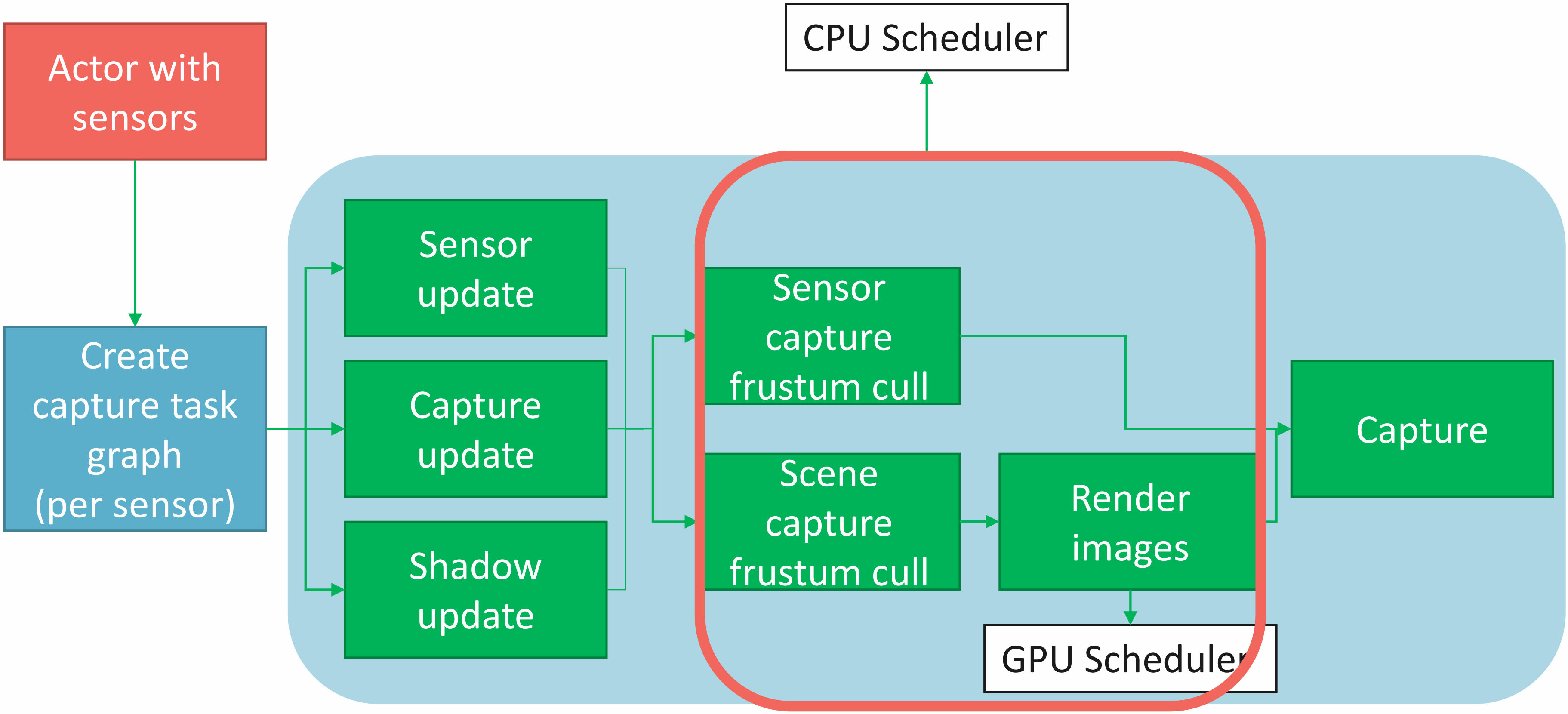
Sensor task graph



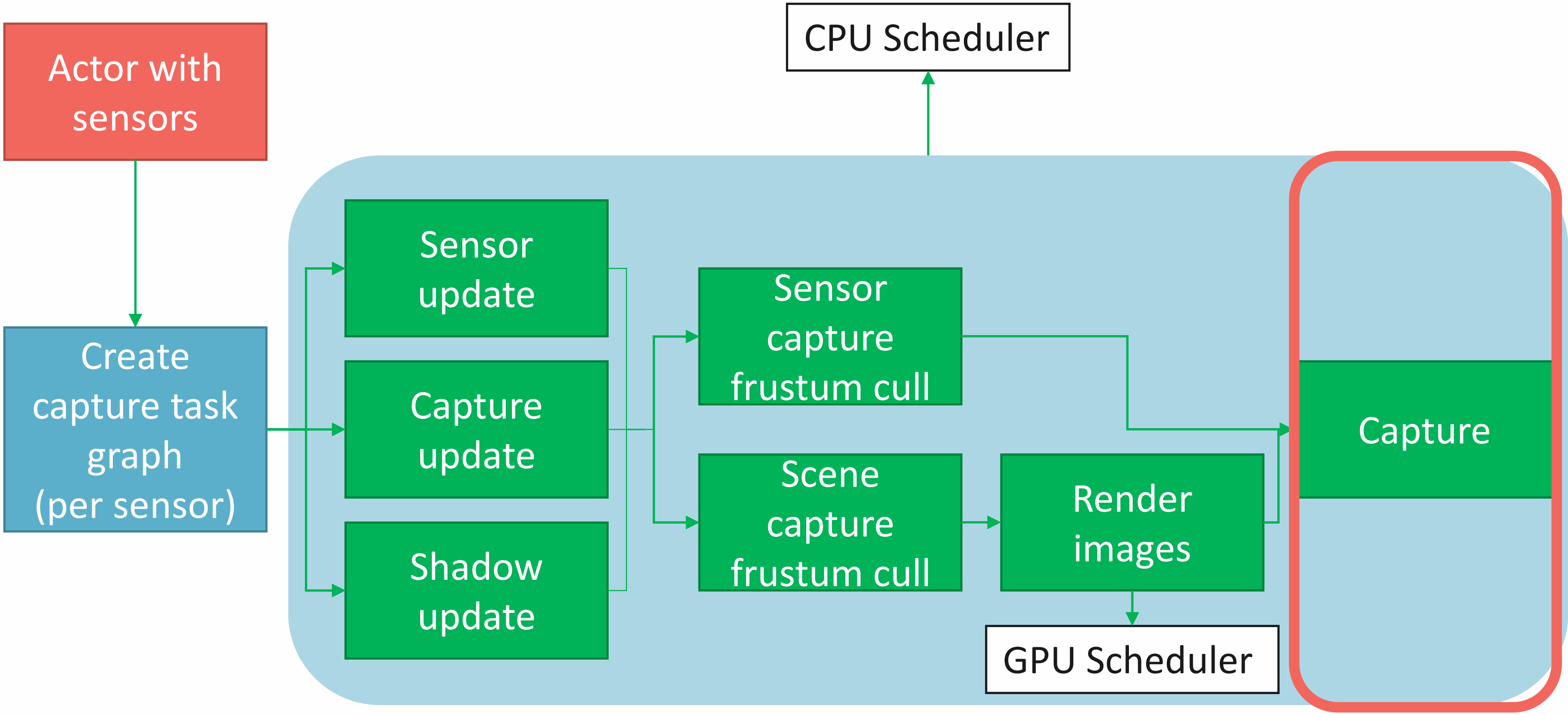
Sensor task graph



Sensor task graph



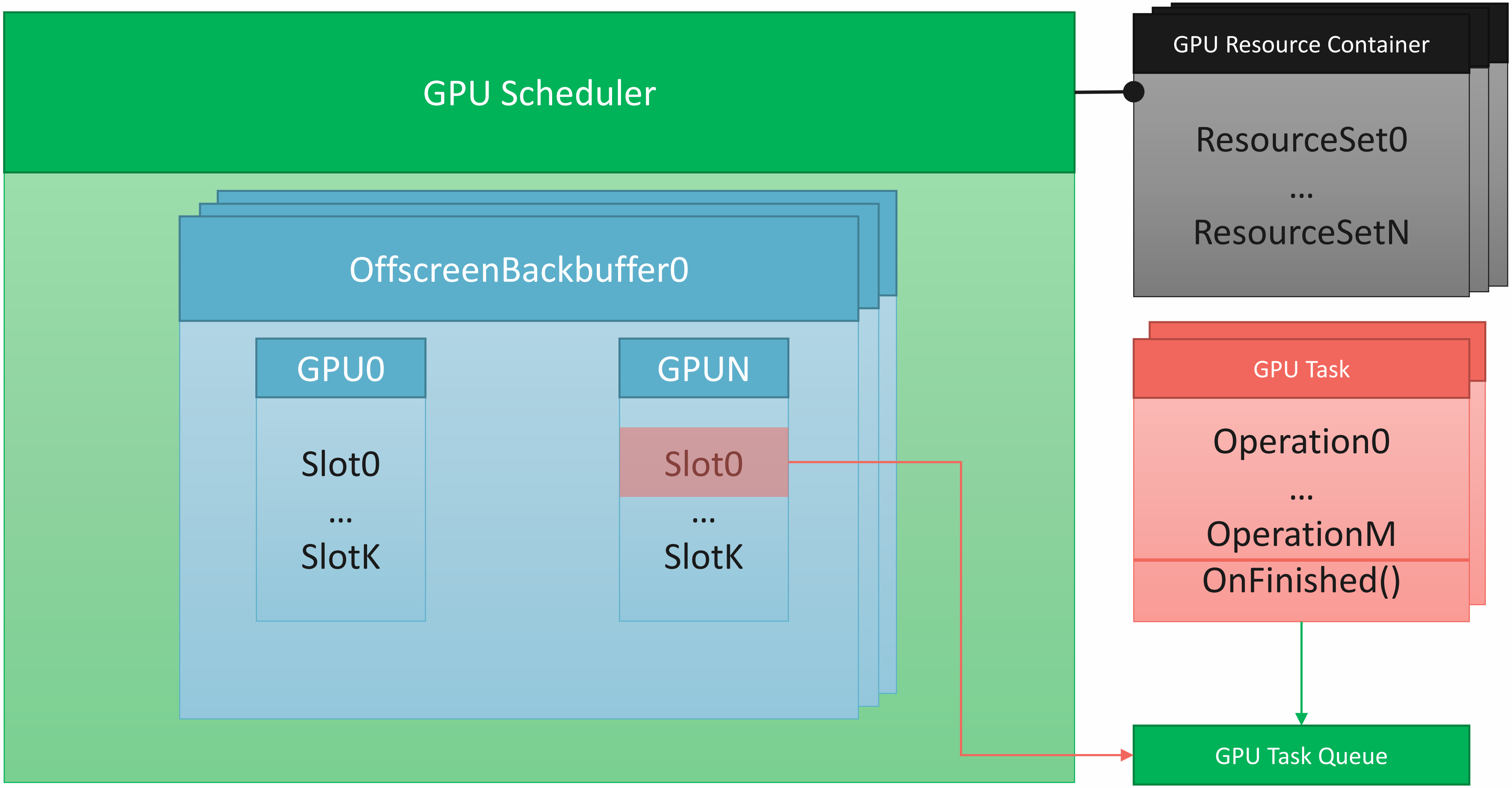
Sensor task graph



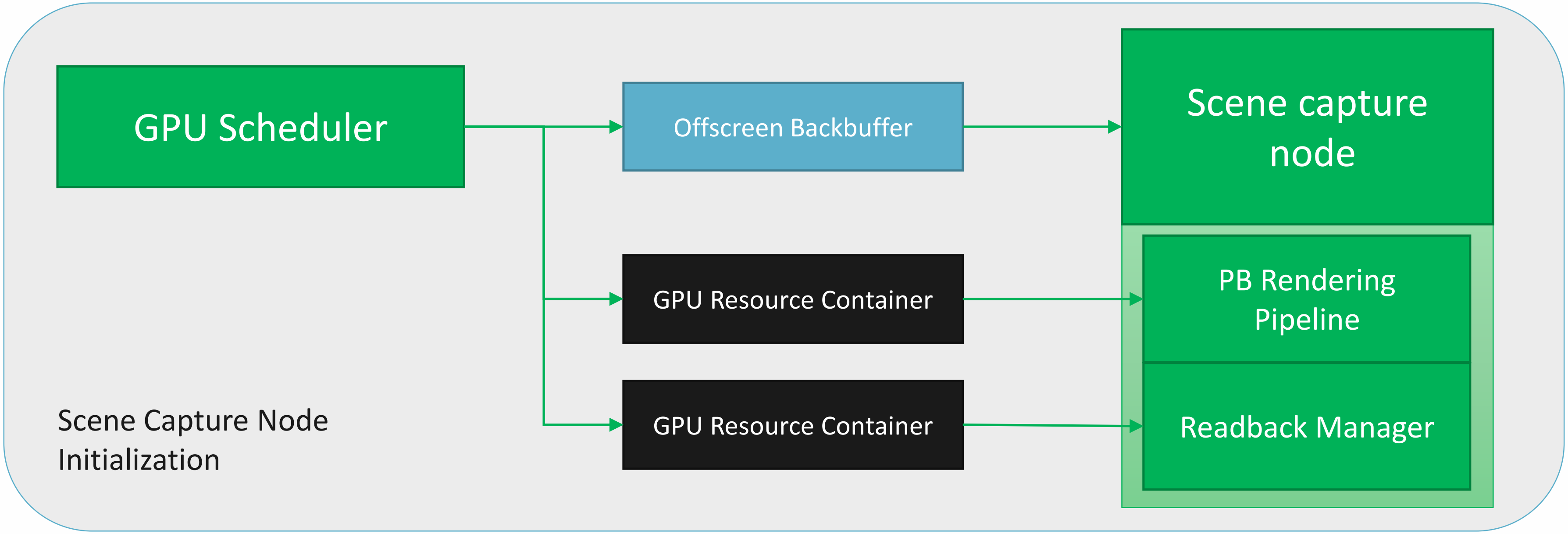
Rendering images

- Rendering
  - RGB-, Bayer images
  - Segmentation, Distance images
  - Etc.
- Two main phases
  - Executing rendering operations
  - Readback from devices
- GPU Scheduler
  - Manages GPU resources
  - Chooses a GPU slot for a task
- Definitions
  - GPU Slot – An operation buffer on a specific GPU
  - GPU Task – Subset of GPU operations in a GPU Slot
  - GPU Task Queue – Set of GPU Tasks, submission to GPU

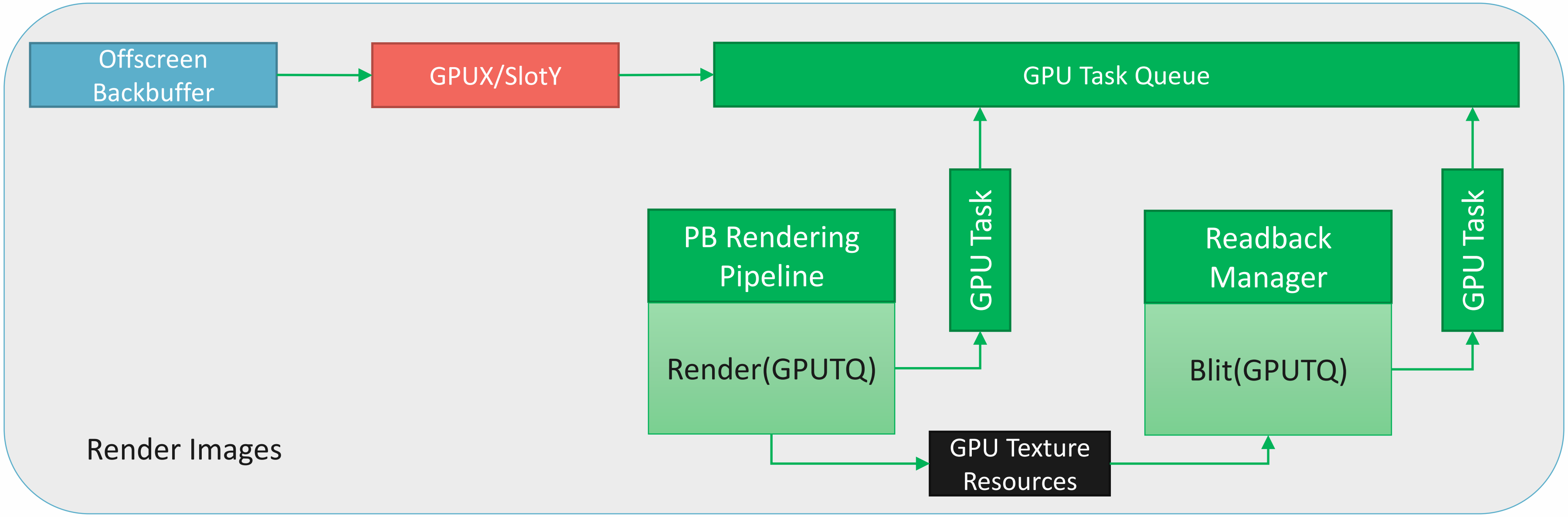
GPU Scheduler Architecture



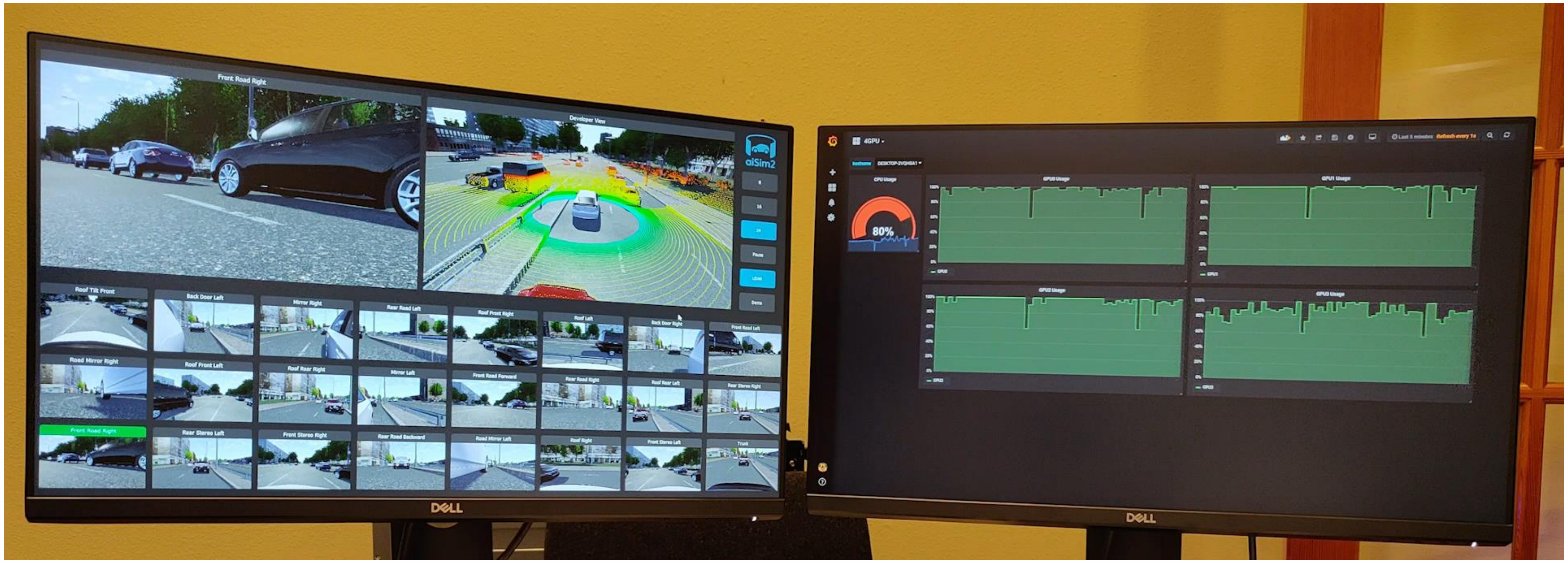
Initialization



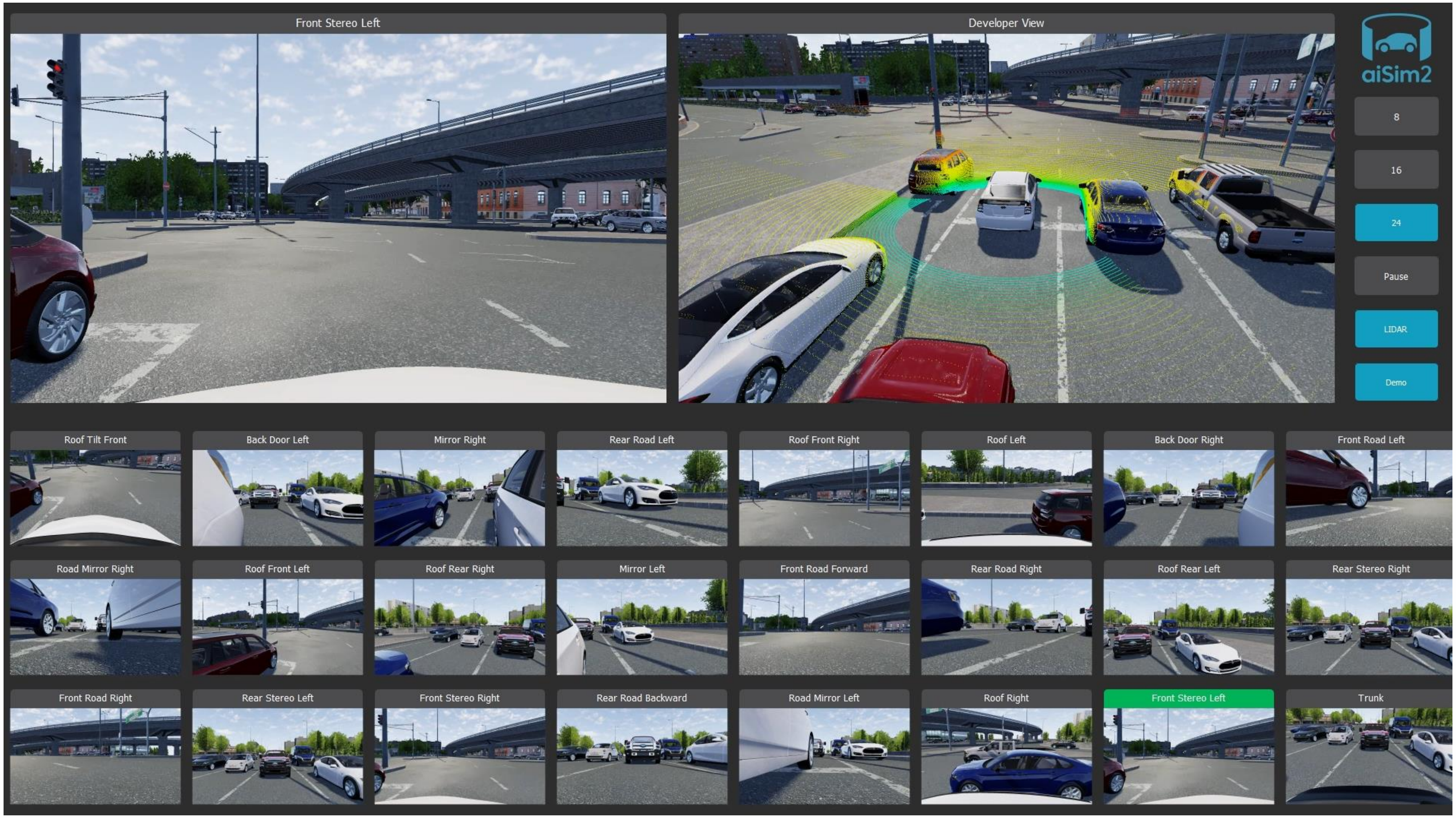
Initialization and Render Images



What we achieved?



What we achieved?



What we achieved?

AIMOTIVE

Thank you for your attention!

Do you have any questions?