Sony Computer Entertainment US Research and Development
http://www.research.scea.com

Crowd Simulation on PS3

Craig Reynolds

Game Developers Conference 2006

Crowds and Other Group Motions

- Pedestrians, urban crowds
- Armies
- Vehicle traffic
- Animal groups: flocks, herds and schools

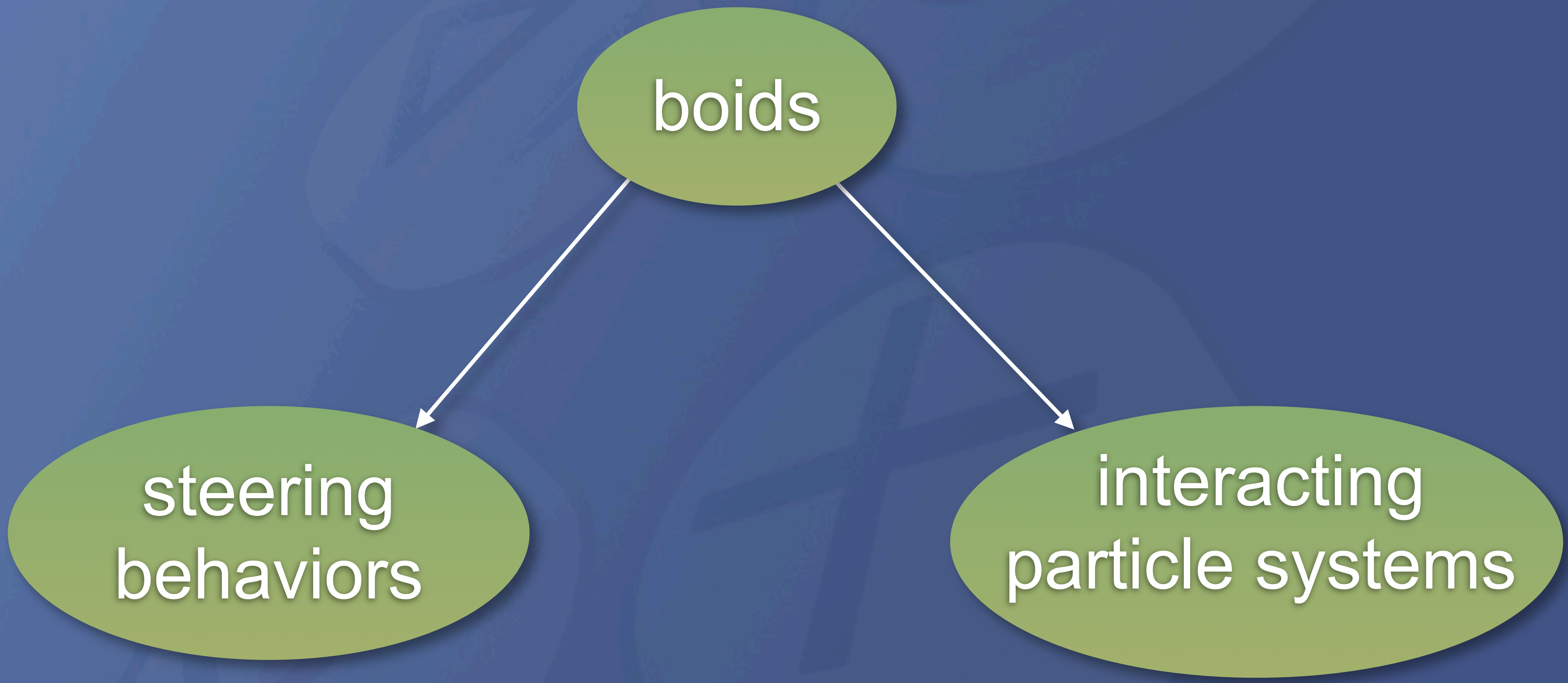
Crowd Simulation on PLAYSTATION®3

- Goal:
  - Simulate large groups of autonomous characters
- Requirements:
  - Real time: 60 frames per second
  - High performance: thousands of individuals
  - Take advantage of PS3's Cell architecture

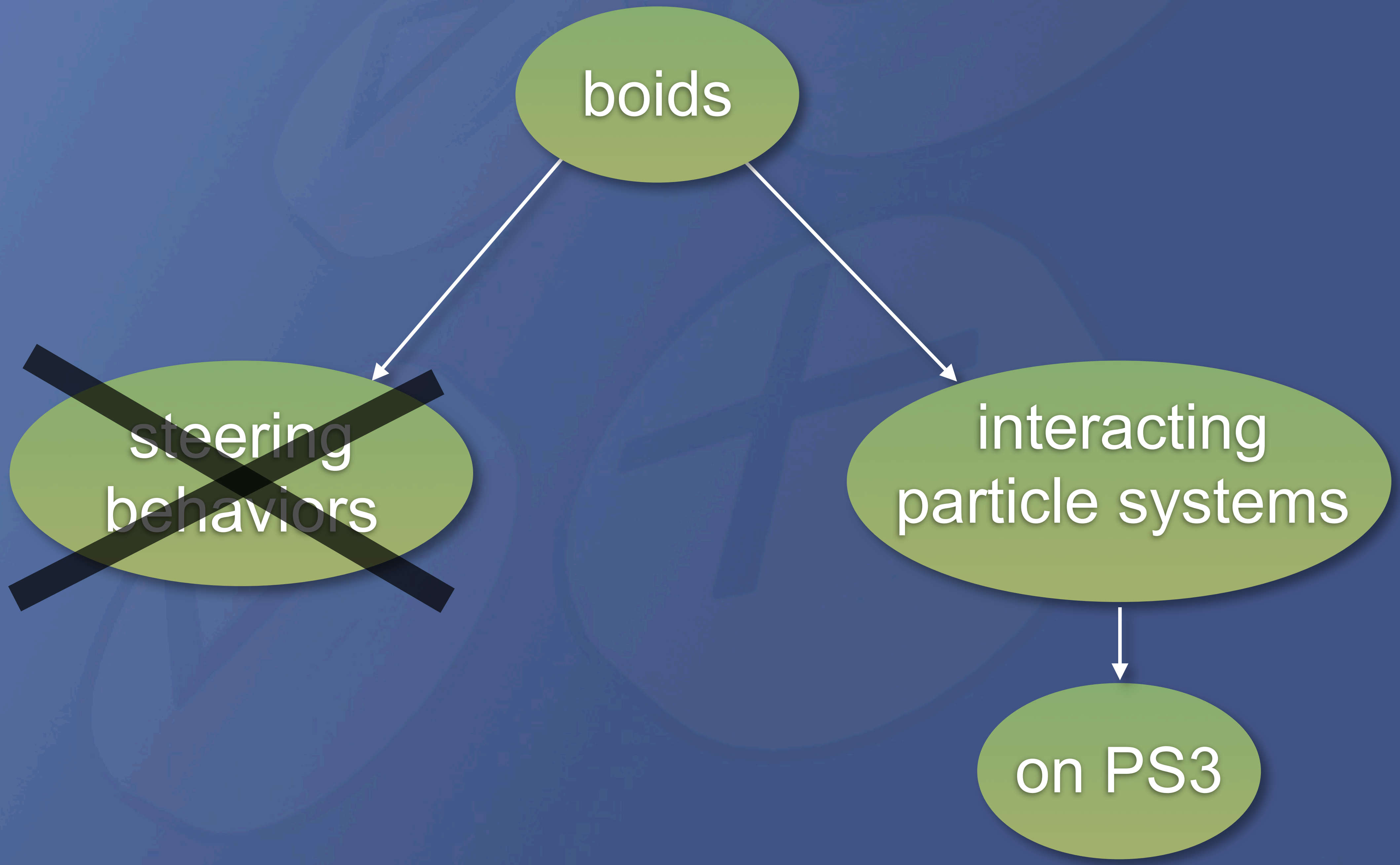
PSCrowd

- Developed for PS3's multiprocessor Cell architecture
  - Makes use of PPU, multiple SPUs and RSX GPU
- High performance:
  - Up to 10,000 simple characters at 60 fps
- Will be provided to developers as SDK sample code
  - Library
  - Demos

This presentation: related topics



This presentation: not about steering



PSCrowd: High Concept

- Subdivide space for fast proximity query
- Use same subdivision as basis of parallel execution

Keynote demo

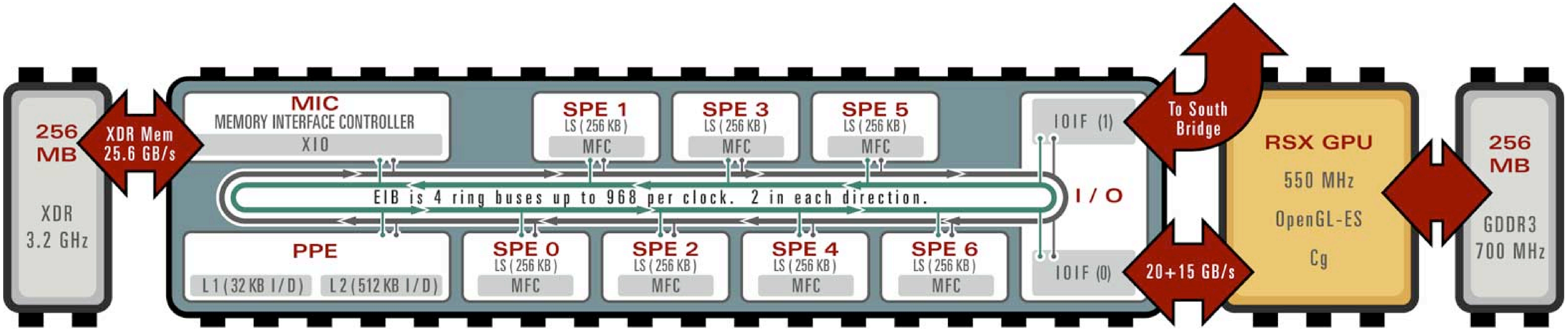
Chameleon fish demo

Queue crowd/obstacle/goals

Overview of PS3 Architecture

- 3.2 GHz clock speed
- 256 Mbyte XDR system memory
- 25.6 Gbyte/sec peak DMA rate
- Power Processor Unit (PPU) -- PowerPC CPU
- Synergistic Processor Unit (SPU)
  - 6 SPUs available to application
  - 256 Kbyte memory
- RSX GPU

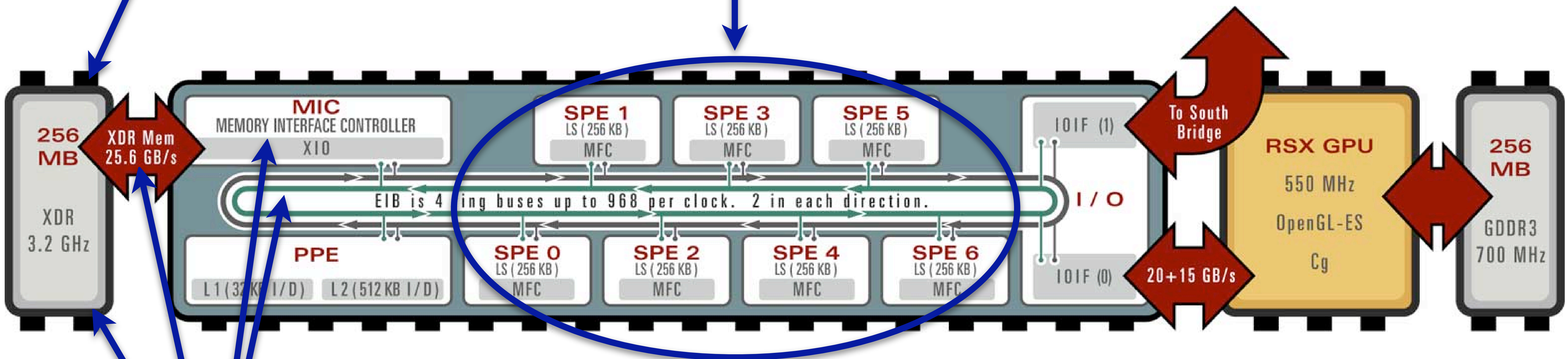
PS3 block diagram



PS3: space and speed

big XDR

small local store on SPUs



really fast DMA
(XDR, MIC, EIB)

fast SPUs

PSCrowd: Basic Concepts

- Keeps track of all individuals in the crowd
  - Sorted by position into “Buckets”
  - Provides efficient access to neighbors
- Update crowd simulation using multiple SPUs
  - Allows arbitrary behavioral model
  - Each SPU updates one Bucket (6X parallelism)
  - DMAs instance data to RSX GPU

PSCrowd Software Substrate

- PS3 SDK (libraries, tool chain, app Framework)
- PSGL graphics, based on OpenGL ES
- Cg for shaders and instancing on RSX
- OpenSteer: steering behaviors and utilities

Crowd Simulation as *Interacting* Particle Systems

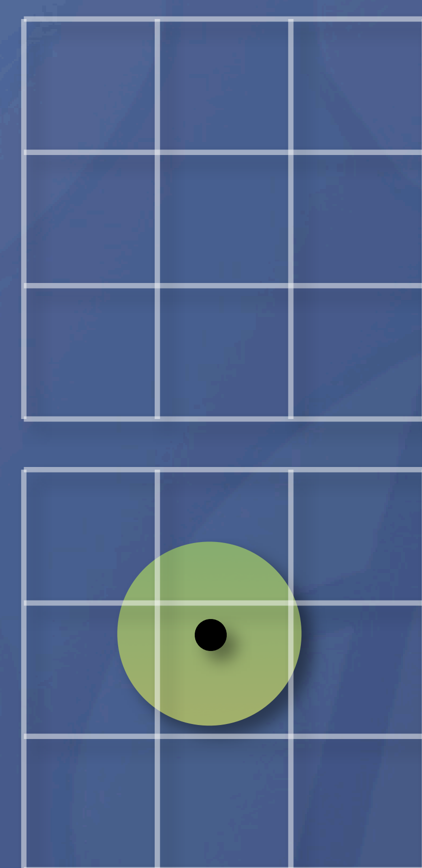
- Crowd simulation can be based on a *particle system*
- In a traditional particle system each particle has behavior and may interact with its environment
- A "crowd particle" also interacts with its *neighbors*
- Profound impact on performance:
  - Traditional particle system: $O(n)$
  - Interacting particle system: $O(n^2)$
- Large crowd populations are prohibitively expensive
- We need a fast technique for finding neighbors

Accelerating Interacting Particle Systems

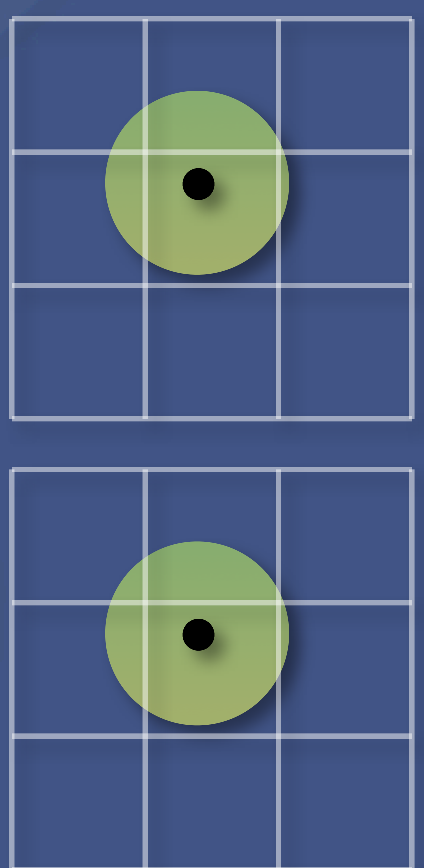
- Finite support -- behaviors based on local perception
- Spatial hashing
- Parallel execution of update computations

Accelerating Interacting Particle Systems

Finite support
(local perception)

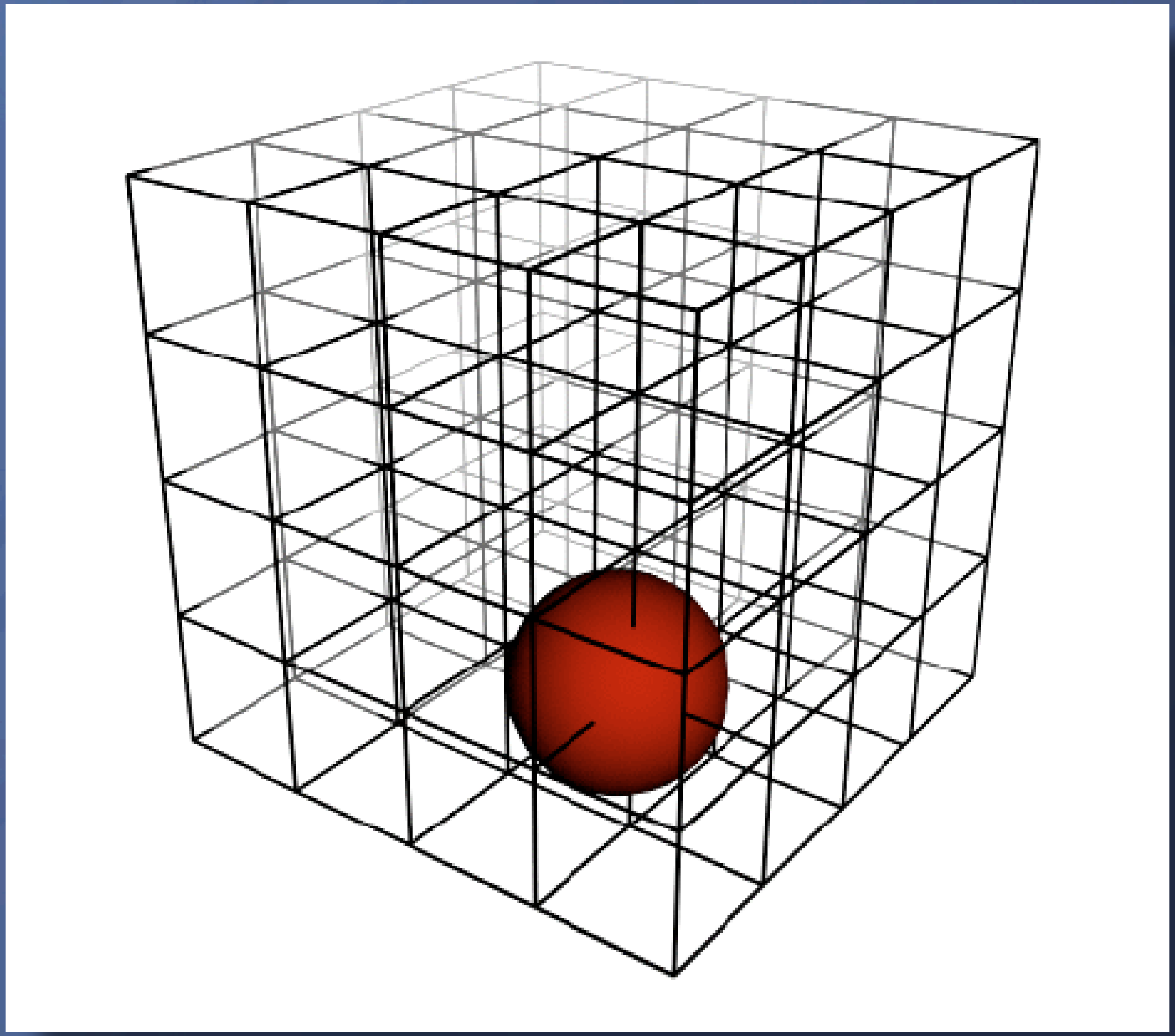


Spatial hashing



Parallel update

Spherical Neighborhood Within 3D Lattice

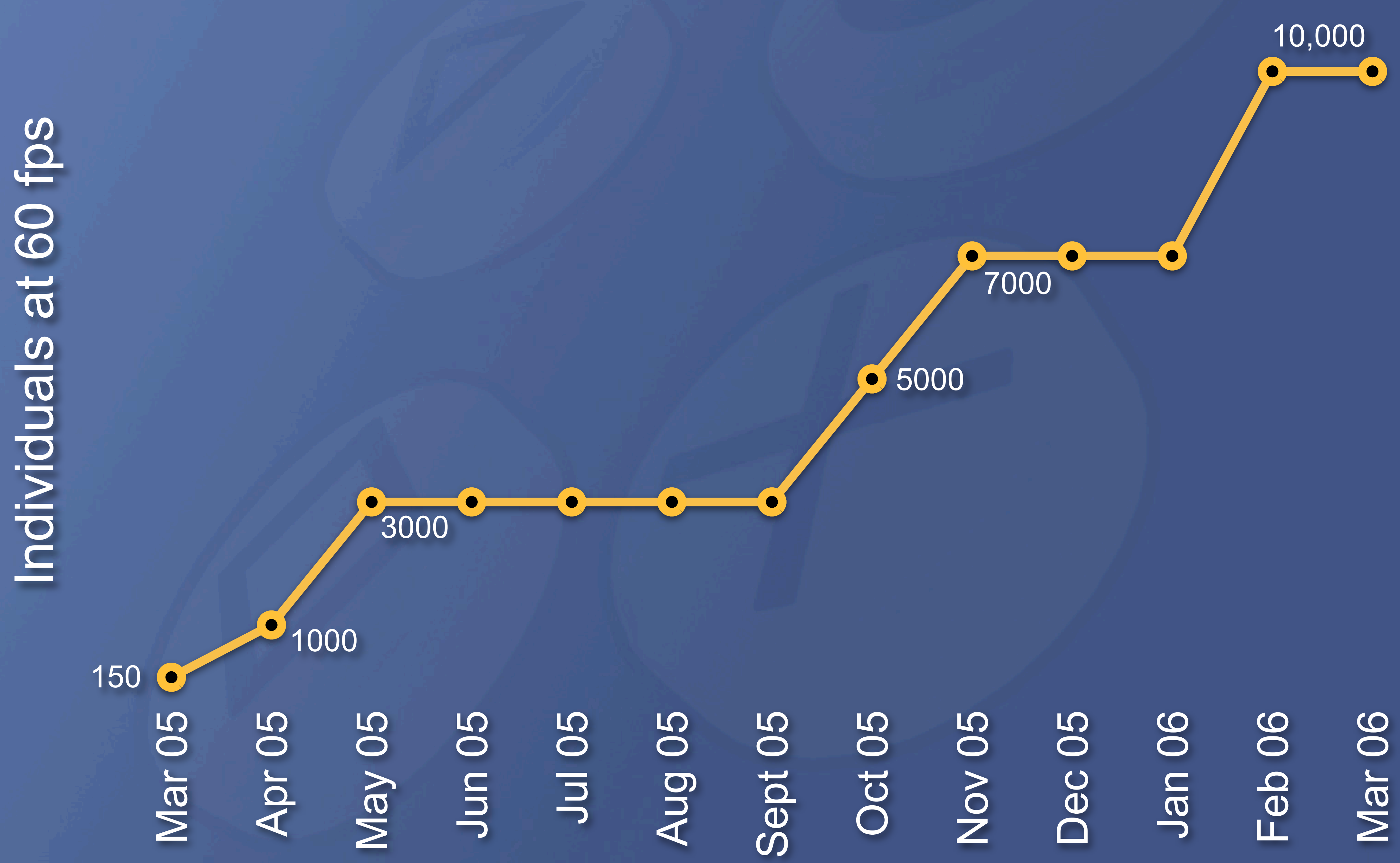


Using Spatial Subdivision to Accelerate Crowd Simulation

- Pre-sort individuals by positions
  - Break up space into smaller regions (area, volume)
  - Associate individuals with these local regions
- Find neighbors more quickly by local search
- History: listed as future work in 1987 boids paper, PS2 implementation described in 2000 PIP paper
- With multiple processors:
  - Regions are disjoint, so update them in parallel
  - Boundary conditions for perception distance

Interacting Particle Systems: Performance Timeline

Progress in PSCrowd Performance



PSCrowd C++ Library Components

- Individual
- Container classes (templates of a class based on Individual)
  - Bucket
  - Lattice
  - NearestN
- BucketUpdateParameters (BUP)

Individual class

- Represents one member of a crowd
- Base class for application-specific individuals
- Implements:
  - Basic per-agent, per-frame update
  - Various per-crowd utilities as static class functions

Individual class

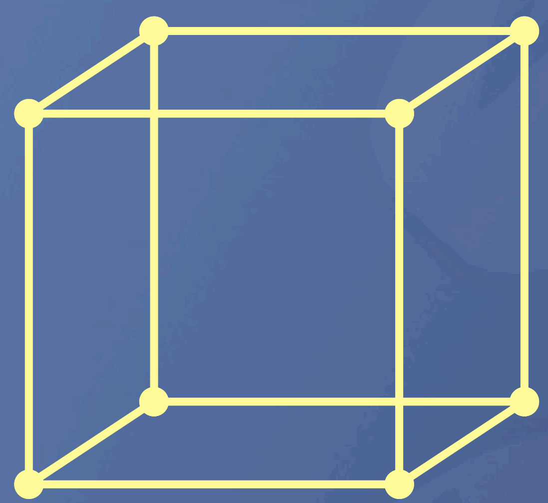
position
local x
local y
local z
speed
radius
etc...

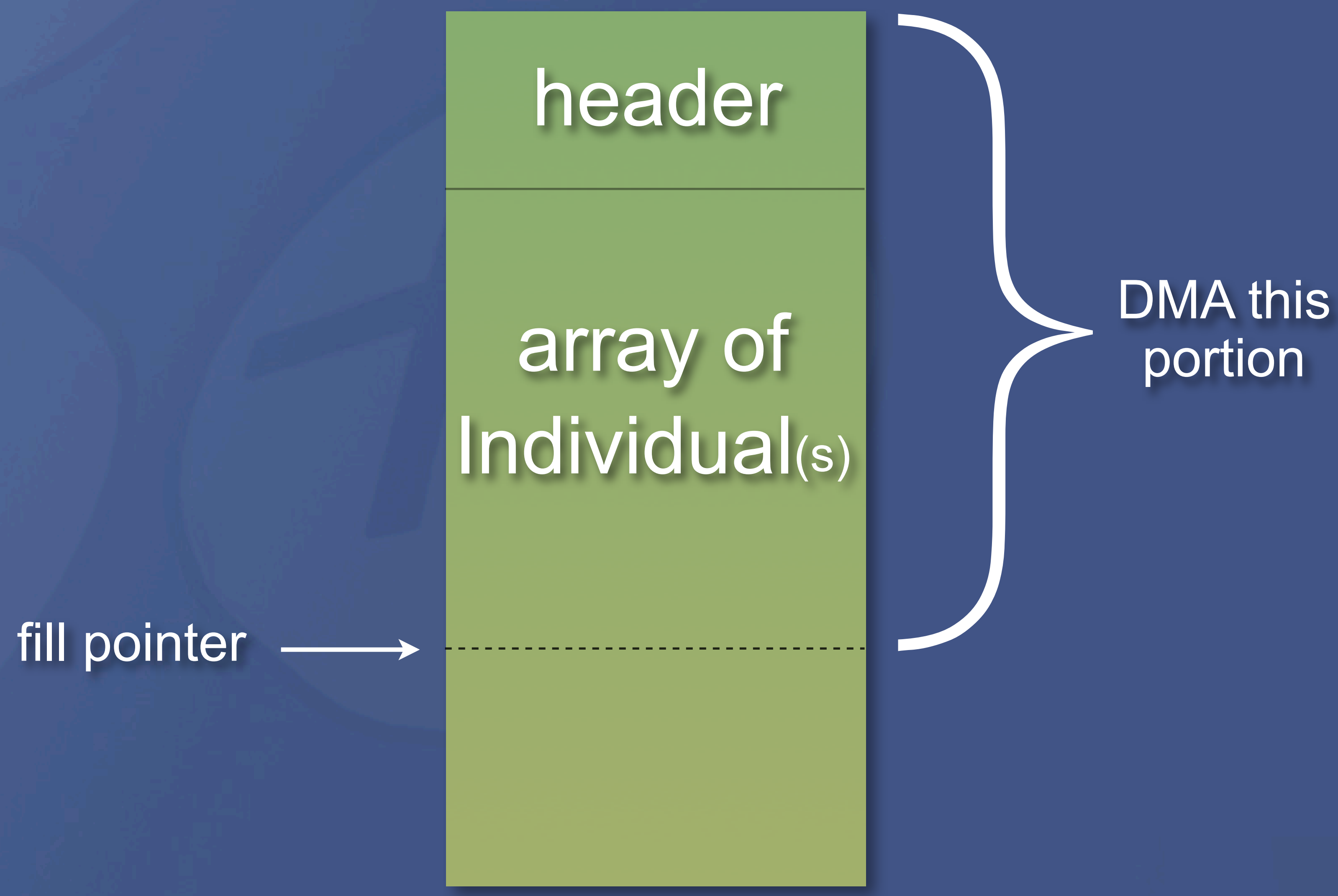
Bucket class

- Template based on class derived from Individual
- Corresponds to an axis-aligned box of 3D space
- Collection of Individuals in that box (fixed max size)
- Rebucket:
  - Once per frame (on PPU)
  - Reassign individuals who cross Bucket boundaries
    - Constant time add/delete operations.

Sony Computer Entertainment US Research and Development
http://www.research.scea.com

Bucket class

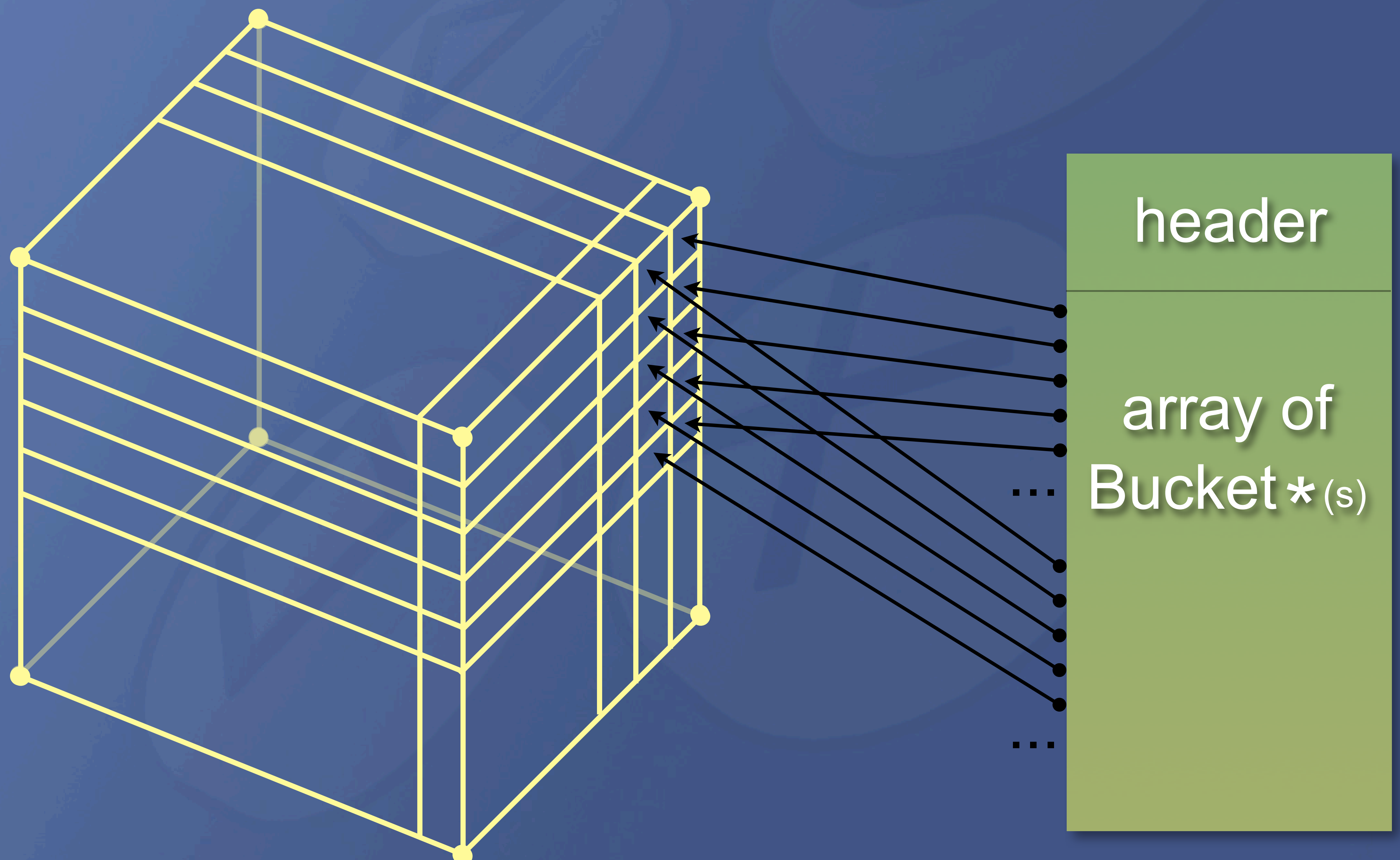




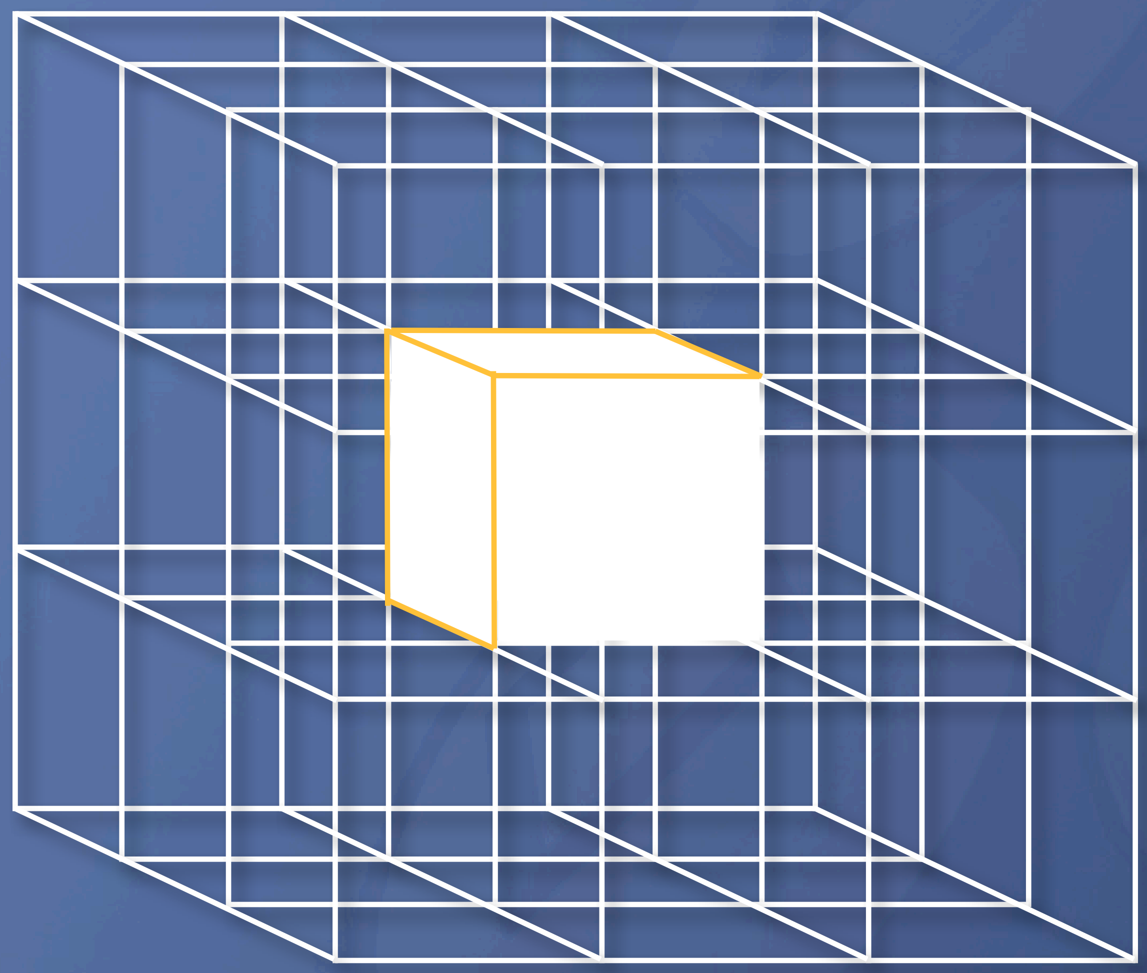
Lattice class

- Template based on class derived from Individual
- 3D array of identical Buckets
  - Contain master copies of all Individuals

Lattice



SPU Bucket update



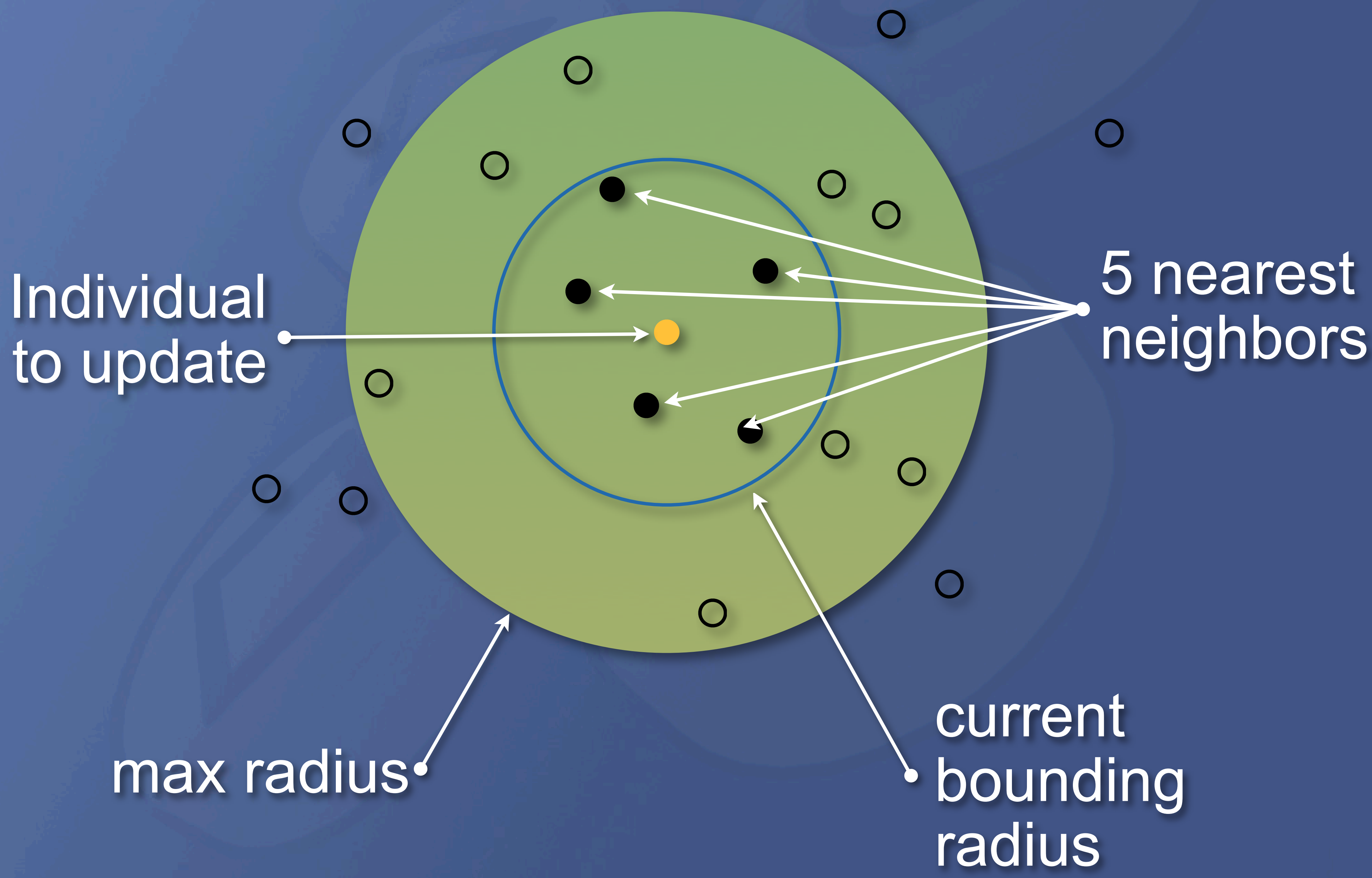
3x3x3 Bucket neighborhood

- DMA to SPU:
  - BUP (poll until ready)
  - Bucket to be updated
  - 26 “Condensed Buckets”
- Update center Bucket
  - refer to surrounding CBs
- DMA instance data to RSX
- DMA updated Bucket to PPU

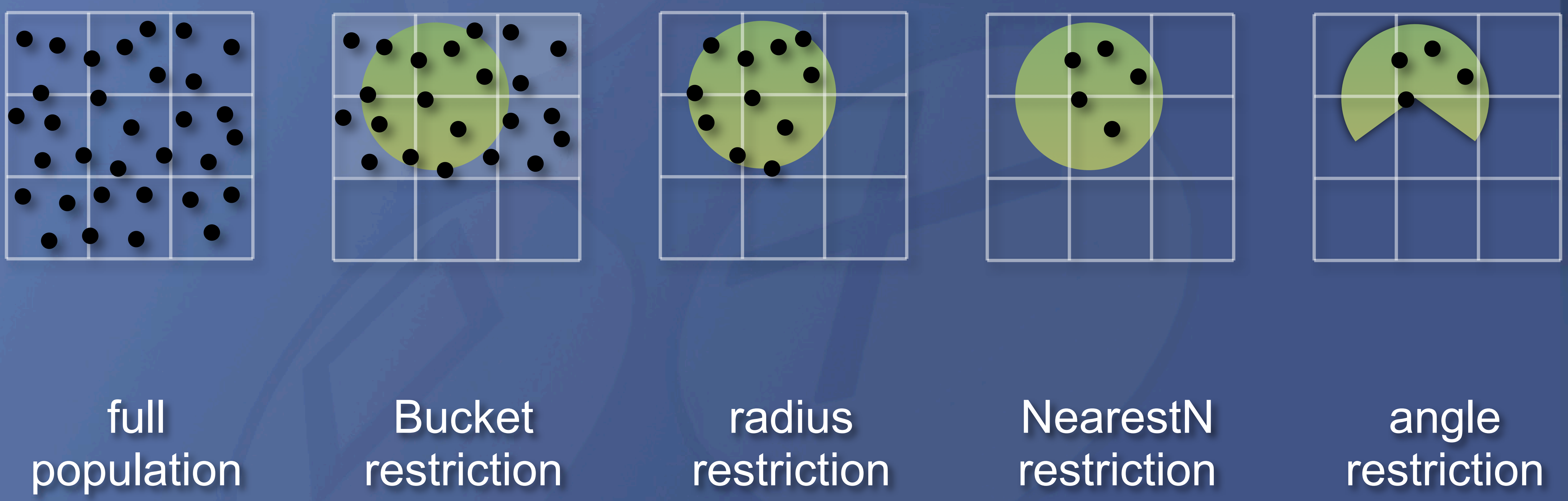
NearestN

- aka: “K nearest neighbors”
- defined by: a position, max radius and N
- applied to all Individuals within intersecting Buckets
- builds an ordered collection of the N nearest neighbors within given sphere

NearestN


Individual to update
5 nearest neighbors
max radius
current bounding radius

Neighborhood Refinement



Demonstrations

- 3 PSCrowd demos -- distributed with the software
- Demo made for Phil Harrison's GDC 2006 Keynote

PSCrowd Demonstrations

- 60 fps on prototype PS3 (CEB-2050, 3.2 GHz)
- Simple 36 triangle model, vertex animation
- 3D schooling: 7000 fish
  - Chameleon fish: *flock coloring* behavior
  - Fish species: prefer to school with their own kind
- 2D crowd: 10,000 individuals

Keynote Demonstration

- 5000 fish
- 30 fps
- 2 species
- art assets
  - textures and models: fish (3 LOD), rocks, ducks...
- procedural water
  - underwater shaders, moving surface
- COLLADA-based art path
  - digital content creation tools → PSGL graphics

Demonstrations

Behavioral Components

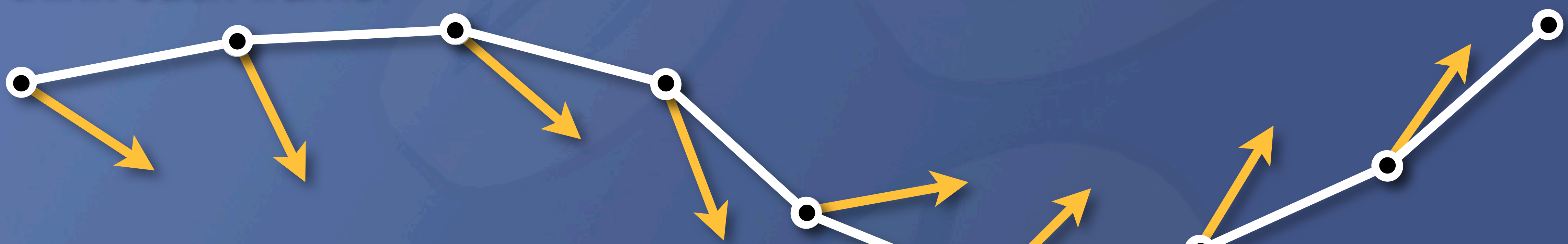
- Boids flocking behavior
  - Separation
  - Alignment
  - Cohesion
- Flock coloring
- Obstacle avoidance
- Anti-Head-on
- Leader wander
- Anti-Bucket-crowding

Behavioral Update Rate (skipThink)

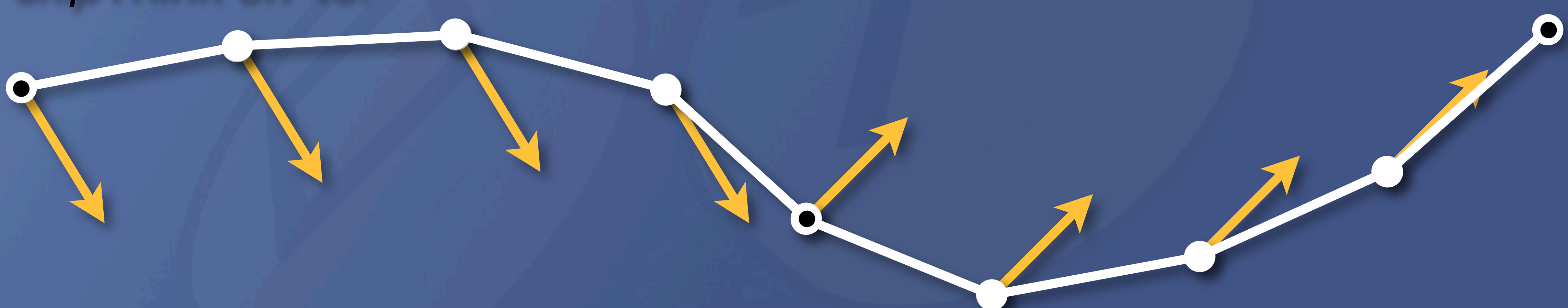
- 60 fps update for physics, animation and graphics
- Slower rate for behavioral updates
  - "on 8s" (7.5 fps) for 7000 Individuals.
    - On each frame:
      - 1/8 of Individuals *think*
      - 7/8 of Individuals *skipThink* and apply same steering force computed on the last think.
  - "on 10s" (6 fps) for 10,000 Individuals

skipThink

think each frame:

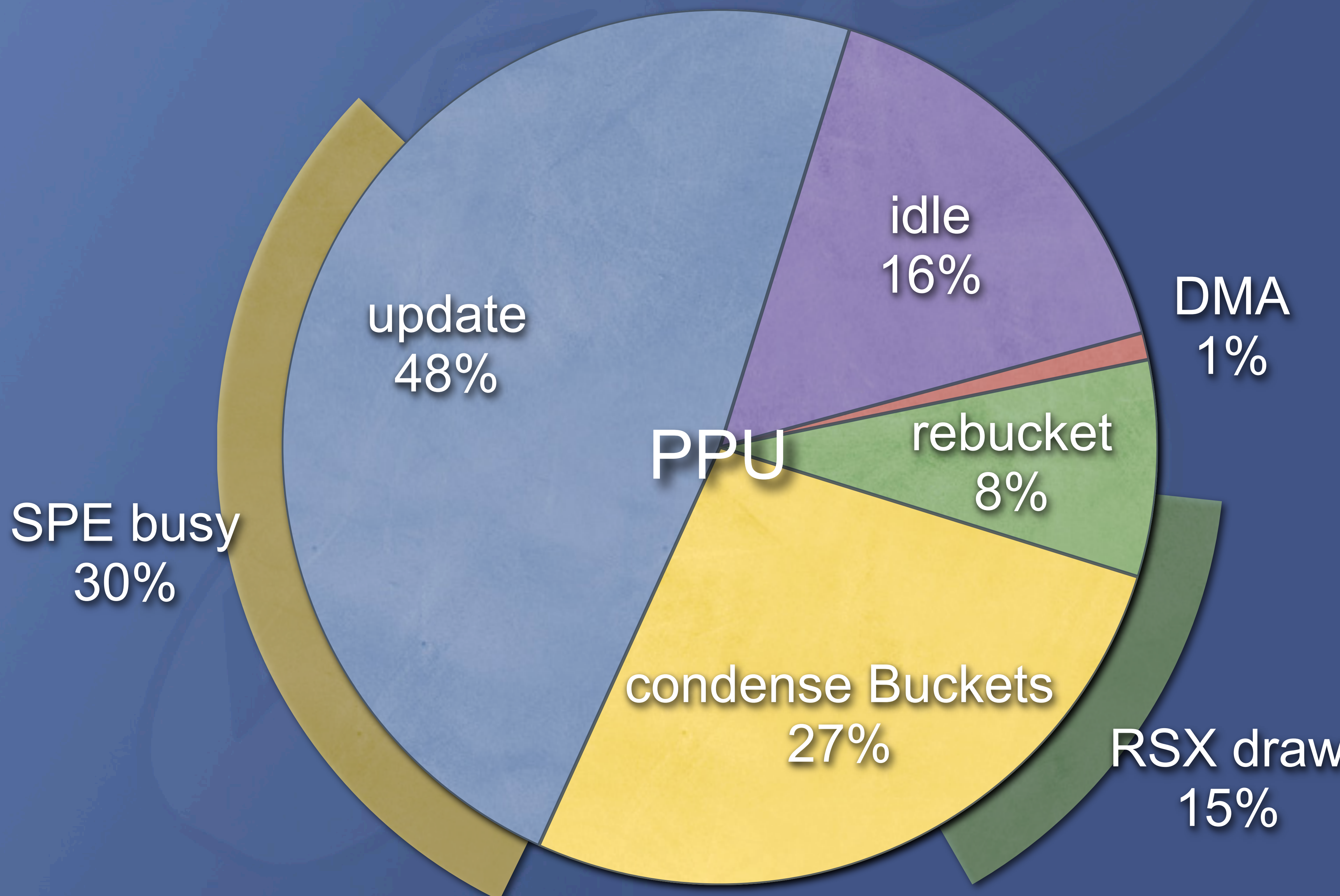


skipThink on 4s:



Overall System Utilization

("chameleon fish" demo)



Future Performance: Stewart's Number

- About 15 months ago my colleague Stewart Sargison predicted that I should be able to handle crowds of 16,000 individuals at 60 fps.
- PSCrowd can handle 10,000 today.
- Faster in the future?
  - SPU idle more than half of each frame.
  - PPU spends about half its time spoon-feeding the SPUs new Bucket assignments.

Ideas That Did Not Work

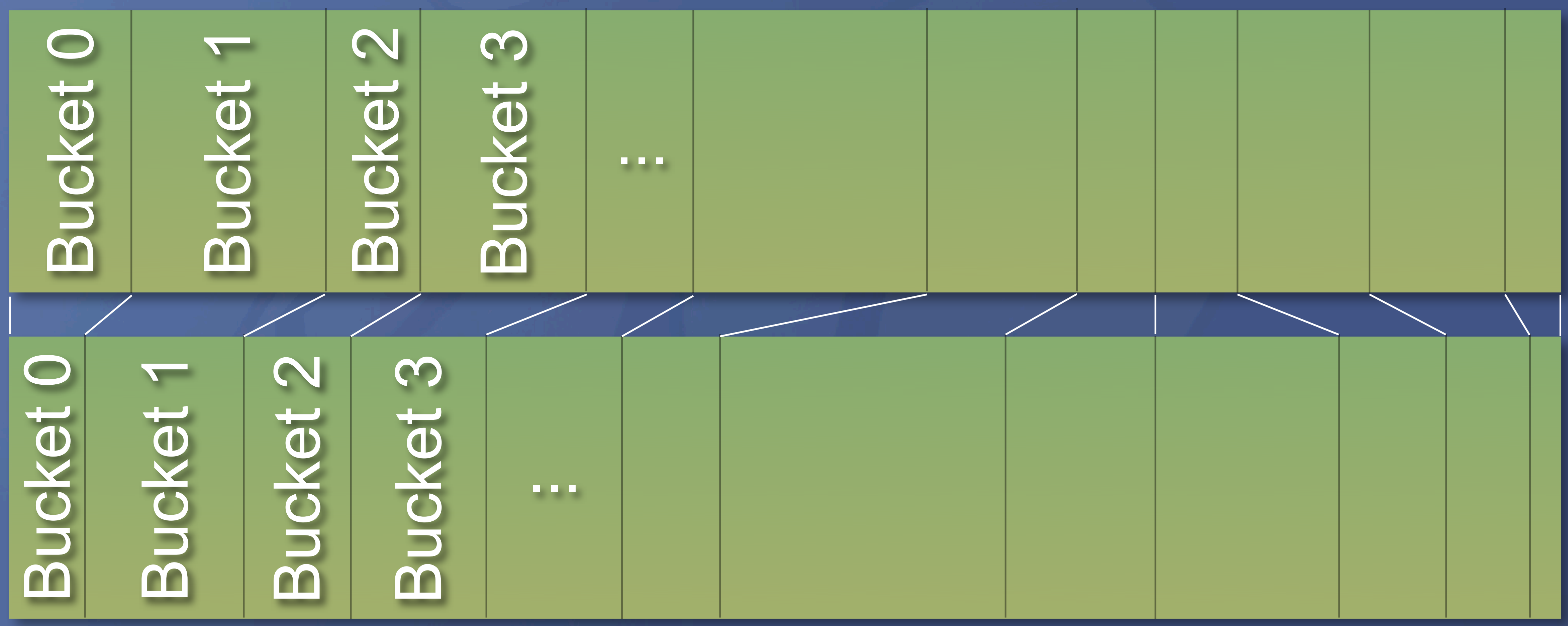
- skipThink per Bucket
  - skip *thinks* in sync on all of a Bucket's Individuals
  - mostly intended to avoid DMA
  - but DMA is so fast there was little benefit
  - problem: increased granularity of Bucket updates
- Start biggest Bucket first
  - small overhead: incremental sort of buckets by size
  - better to reduce time for all Bucket updates

Limitations and Future Work

- Large memory footprint on PPU (sparse, roughly 50X)
  - Solve by repartitioning dense Lattice into new adaptively sized Buckets for each frame?
- Bucket size and robustness:
  - Must be small to store 27 (3x3x3) on SPU
  - Fixed size (especially if small) invites overflow
  - Solve with streaming of arbitrary size Buckets?
- Weak *unaligned collision avoidance*
- No physical or kinematic non-penetration constraint
- Other kinds of spatial hashing: nav mesh, KD tree

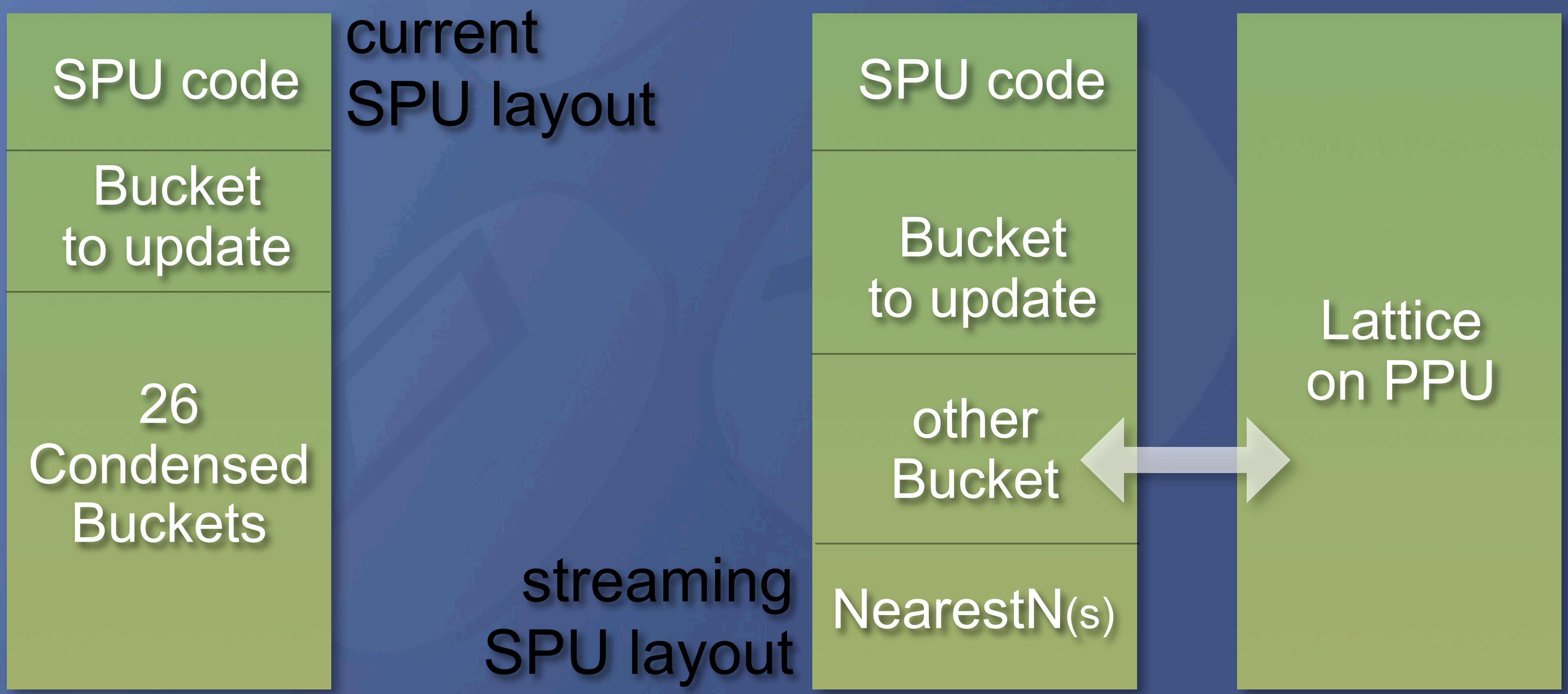
Future Work: Repartition Dense Lattice

- Large memory footprint on PPU (sparse, roughly 50X)
  - Solve by repartitioning dense Lattice into new adaptively sized Buckets for each frame?



Future Work: Streaming Buckets

- Bucket size and robustness:
  - Solve with streaming of arbitrary size Buckets?



Acknowledgments

- Sponsored by Sony Computer Entertainment
- Supported by many colleagues in Japan, Europe and here in California
- Particularly my US R&D coworkers: Gabor Nagy, Care Michaud-Wideman, Roy Hashimoto, Axel Mamode, Steven Osman, Stewart Sargison, Tanya Scovill, Trevor Smigiel, Chengdong Li, Greg Corson and Nicholas Szeto
- My boss, Director of US R&D: Dominic Mallinson

Thank you!

contacts:
http://www.research.scea.com
craig_reynolds@playstation.sony.com