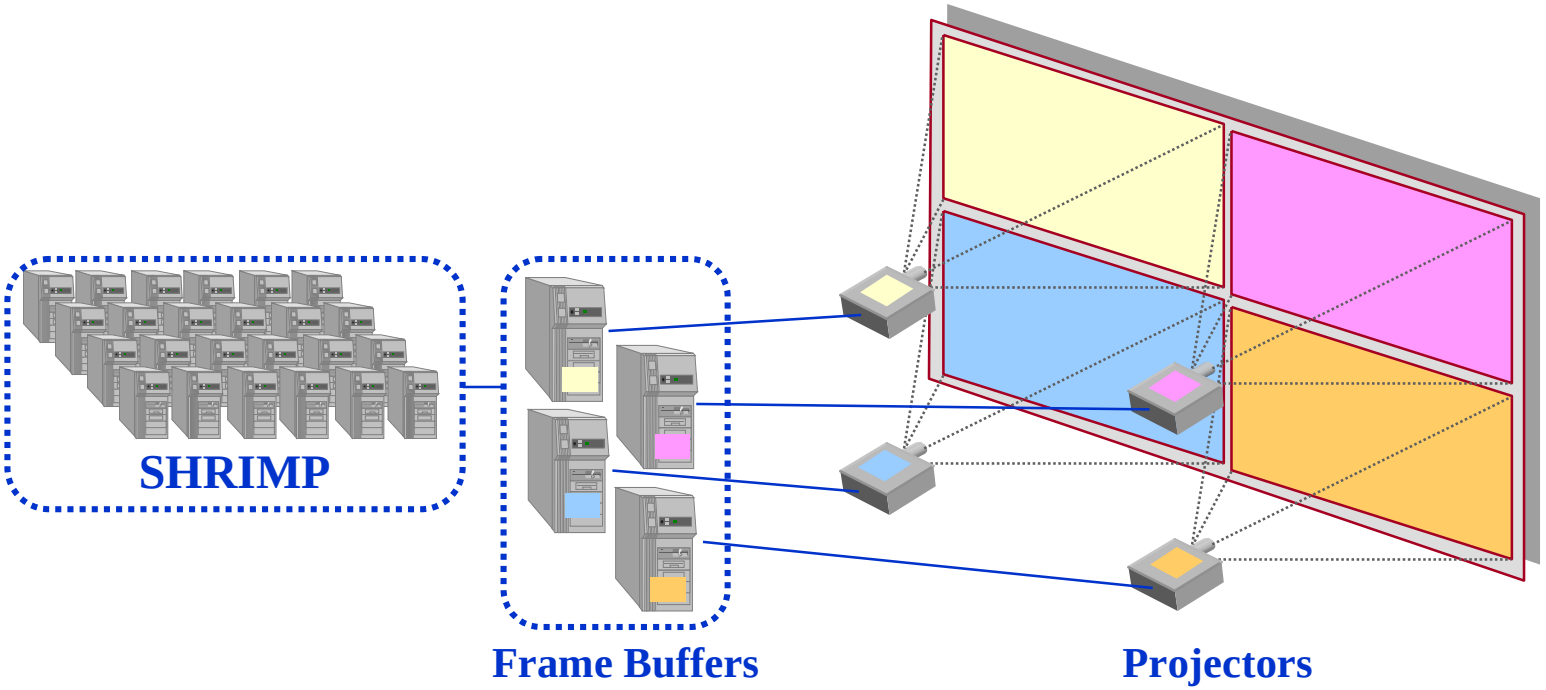
Partitioning Screen Space for Parallel Rendering

Thomas Funkhouser
JP Singh
Jiannan Zheng

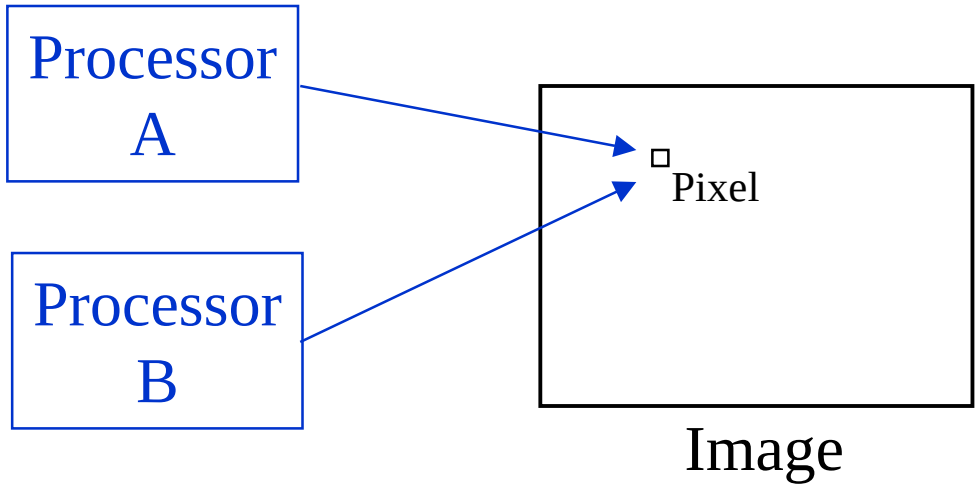
Goal

- Parallel rendering utilizing many PCs
  - Communication via a network



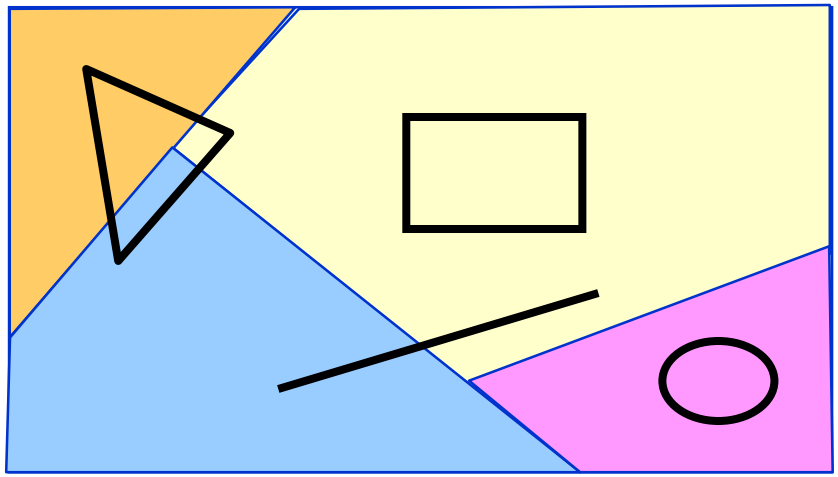
Parallel Rendering Challenge

- Basic problem:
  - Multiple rasterizers cannot write the same pixel simultaneously

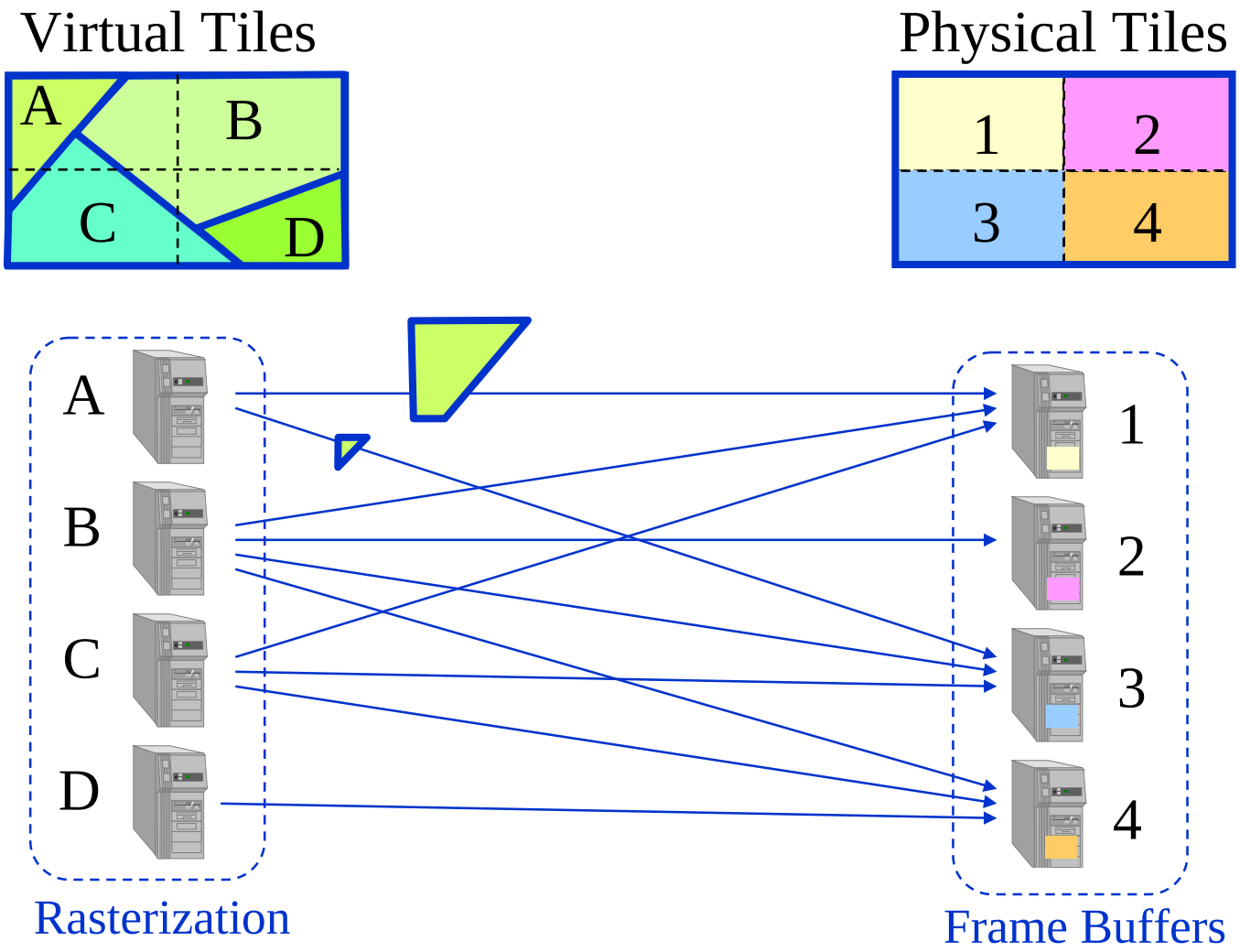


Screen Space Partitioning

- Partition screen into " tiles"
  - Can be any shape, even disjoint, but cannot overlap
  - Usually are not one-to-one with projector regions
- Render each tile on a separate processor
  - Each processor renders all primitives overlapping its tile
  - Primitives are not split at tile boundaries, and thus they may be rendered redundantly by more than one processor



Rendering with Virtual Tiles on the Wall

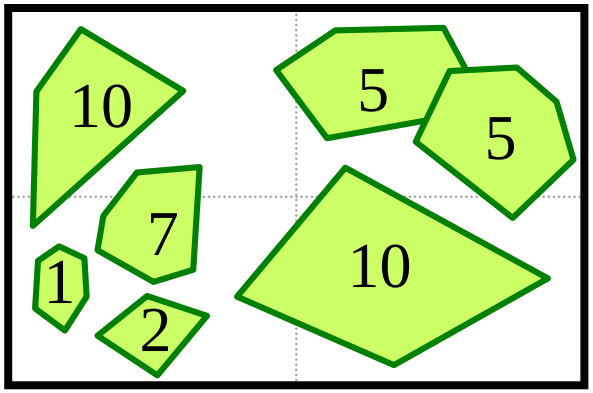


Virtual Tile Selection

- Investigate shapes and arrangements that ...
  - Partition primitives among virtual tiles evenly
    - » Complex tiles (concave regions)
  - Minimize overlap of primitives with virtual tiles
    - » Match scene geometry (non-rectilinear)
  - Sort primitives among virtual tiles rapidly
    - » Simple tiles (grids, boxes)
  - Minimize communication between processors
    - » Match physical tiles as much as possible

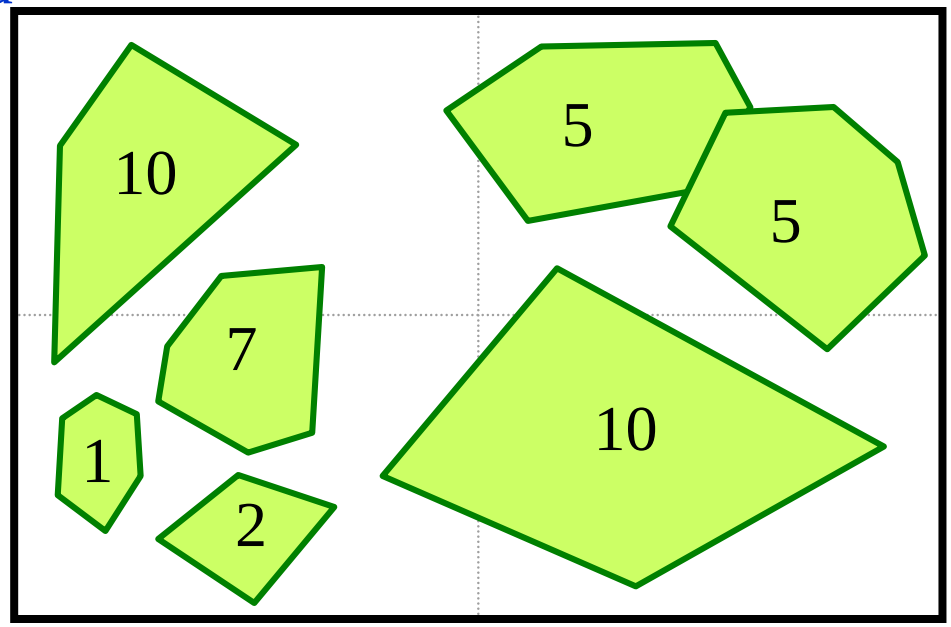
Load Balancing Problem

- Given:
  - N: Set of 2D primitives
  - P: Number of processors
- Find:
  - T: Partition of 2D space with exactly P tiles
- Minimizing:
  - F(N,T): Objective function encoding factors on previous slide



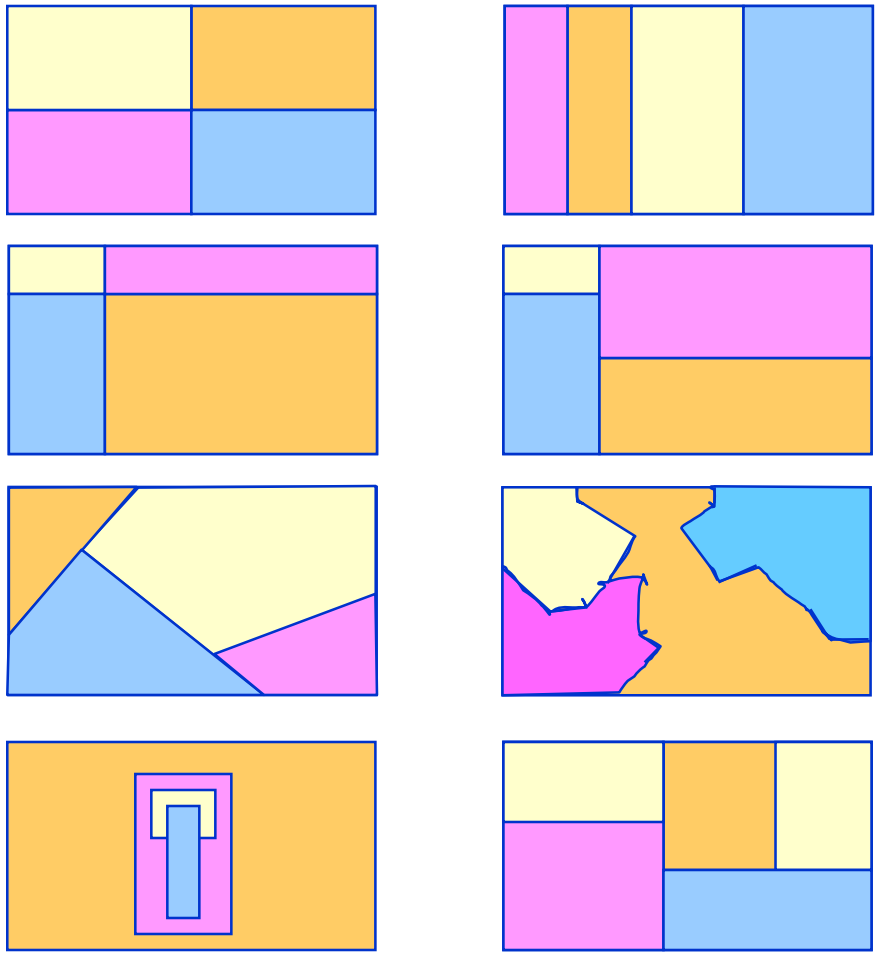
Load Balancing Problem

- Given: Set of 2D primitives with weights
- Problem: Partition 2D space into P tiles so that the overall estimated rendering time is minimized
- cumulative weight of all primitives overlapping any tile is minimized



Possible Tilings

- Boundaries
  - On grid
  - Axis-aligned
  - Linear
  - Piecewise linear
- Tiles
  - Rectangles
  - Convex
  - Concave
  - Disjoint



Approaches to Partitioning

- Start with constraints imposed by system, and adjust
  - start with static partition that matches projector assignment
  - based on profiled workload, move work around to balance, in units that match hardware rendering capabilities
    - » task stealing or task pushing
  - previous frame partition can be used as starting point
- Treat as general partitioning problem; constraints may refine
  - repartition from scratch, or use previous frame as starting point
- Focus on latter approach for now, ignoring system constraints

The General Partitioning Problem

- Goal: contiguous partitions that are load balanced
- General class of problems: Mesh partitioning
  - Partition the elements of an irregular mesh such that load is balanced and communication among partitions minimized
- Dual of mesh partitioning: graph partitioning
  - e.g. nodes of graph are elements that have computation costs, edges denote connectivity and have comm. costs when cut
  - goal: partition to balance and reduce computation and comm. costs
- Problem: NP-complete, so use heuristics
  - want them to be cheap and effective; exploit structure of problem
- In polygon rendering:
  - polygons are elements
  - comm. represented by adjacency, to ensure contiguous partitions

Approaches to Partitioning Irregular Meshes

Some also apply to many other irregular computations

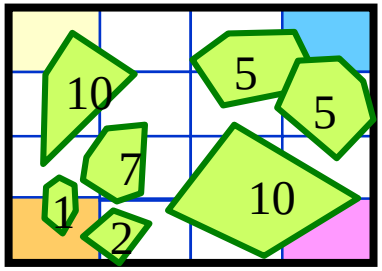
- Merge
  - Start with many pieces, then merge
- Partition
  - Global partitioning methods
  - Multi-level methods
- Optimization
  - Dynamic adjustment
    - » start with some partition, then steal or donate dynamically
  - Local refinement methods
    - » start with a guess, and adjust based on localized criteria
- Hybrids

Merge Methods

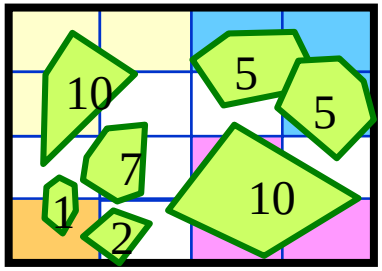
- Random Assignment
- Scattered Assignment
- The Greedy Algorithm
  - " grow" partitions from starting points
  - starting points must be well chosen

Merging of Regular Grid Tiles

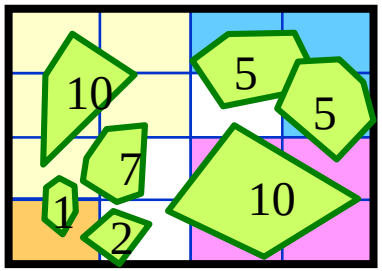
- Starting from four corners
- Try to merge the tile which may make the maximum partition weight grow as less as possible



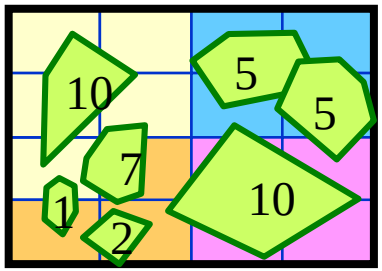
Max = 10



Max = 10



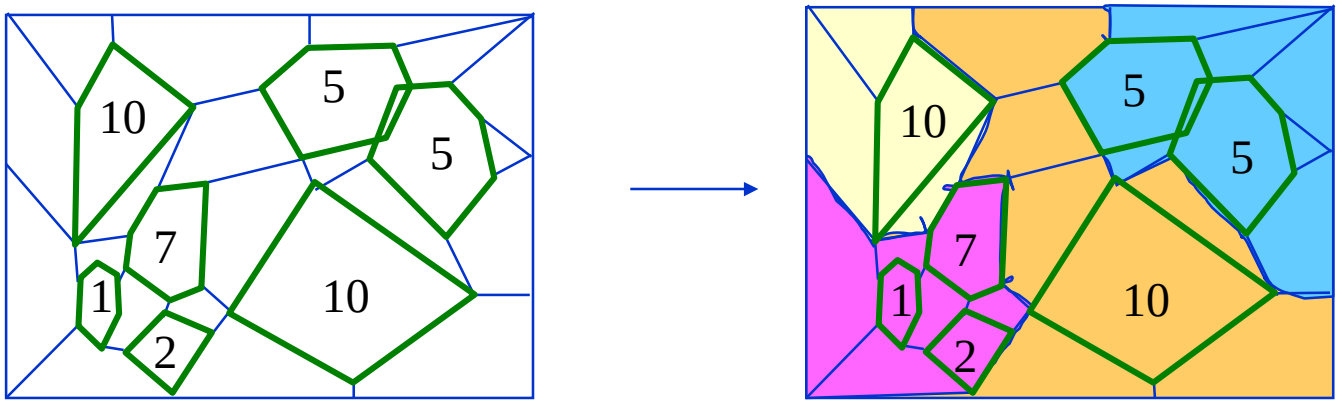
Max = 18



Max = 20

Merging of Irregular Tiles

- Can use irregular initial tiles also. For example, create initial tiles according to primitive geometry.



Max = 10

Partition Methods

- Direct P-way
- Recursive
  - Geometry based
    - » partition mesh/domain recursively
  - Graph based
    - » partition graph representation recursively

Direct P-way Partition Methods

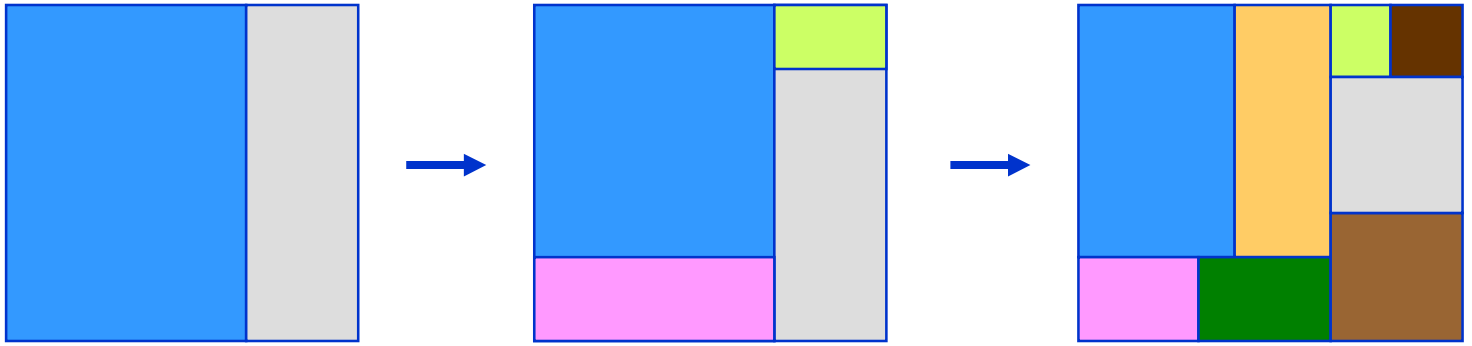
- Random or Scattered Assignment
- Linear, with Bandwidth Reduction
  - order nodes for contiguity, then partition linearly
  - e.g. Morton Ordering, Peano/Hilbert ordering
- Tree partitioning
  - represent spatial contiguity hierarchically using a tree
  - inorder traversal of tree yields an ordering
  - partition tree " linearly"
  - achieves above effect

Recursive Partition Methods

- Geometry-based
  - Coordinate Partitioning
    - » along X, Y, Z axes
  - Inertial Partitioning
    - » choose axes intelligently according to measures of inertia
- Graph based
  - Layered Partitioning
    - » recursive using greedy-like approach on graph
  - Spectral Partitioning
    - » find matrix that represents structure of graph (Laplacian matrix)
    - » find first nontrivial eigenvector of this matrix (Fiedler vector)
    - » use this as separator field for partitioning (e.g. bisection)
    - » very good results, but quite expensive to compute

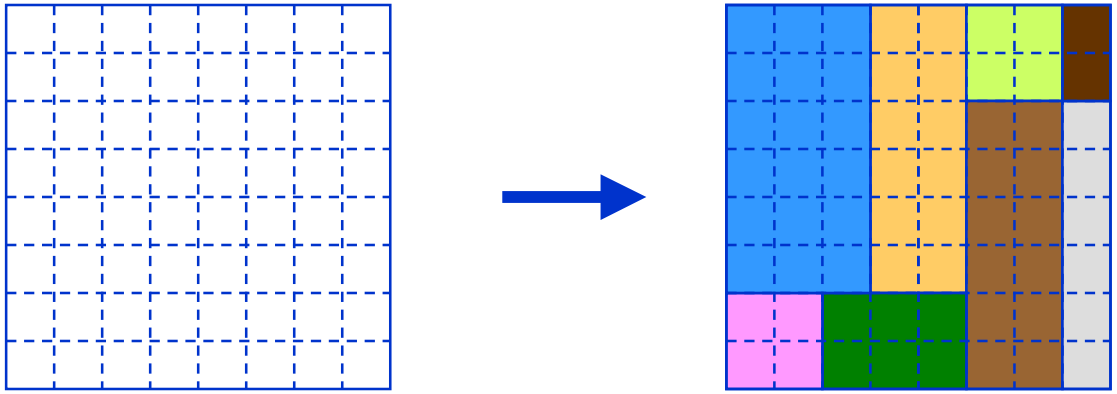
Recursive Partition

- Whelan’ s median-cut method
  - each primitive is represented by its centroid
  - using the number of primitives falling in each region as load estimation
  - recursively divide the longer dimension of the screen using the median-cut until the number of tiles equals the number of processors.



Mueller's mesh-based hierarchical decomposition method

- Rendering primitive' s bounding box to a fine mesh, add 1/A to the cell it overlaps (A is the total number of cell it overlaps)
- Sum the cells weight into a summed area table
- Recursively divide the screen using binary search



Optimization Methods

- Develop a cost function (sum of comp and comm costs)
- Minimize the function, subject to constraints
- Difficult search problem: many local minima
  - need a good starting guess
- Refinement based on Global Criteria
  - Simulated Annealing
  - Chained Local Optimization
  - Genetic Algorithms
- Refinement based on Local Criteria
  - Kernighan-Lin
  - Jostle

Local Refinement Methods

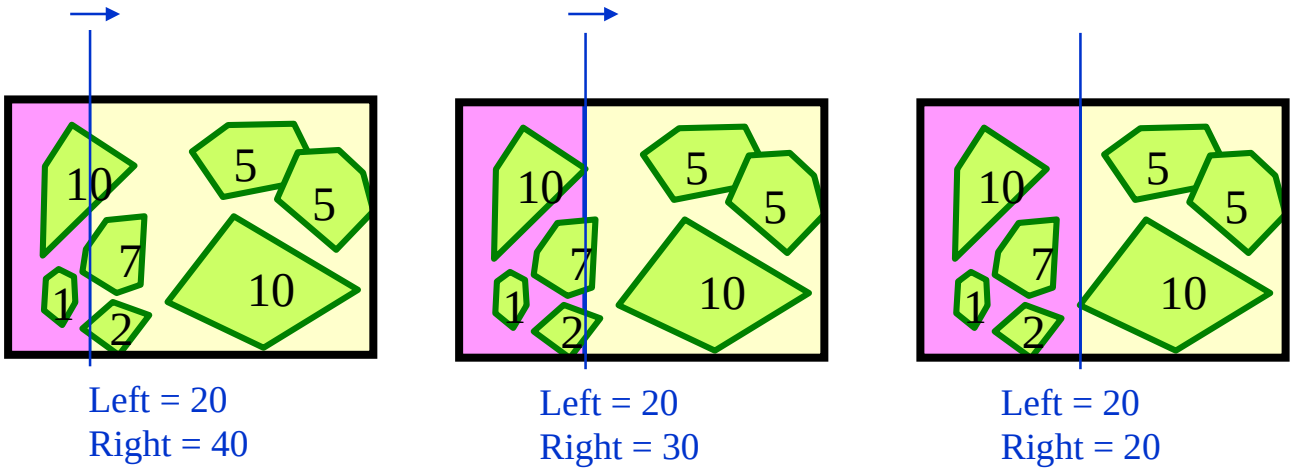
- Kernighan-Lin
  - swap elements with neighbors to improve matters
  - try all pairs to see which gives best gain in a sweep
  - iterate over sweeps until convergence
- Jostle
  - similar, but swap in chunks and preferentially swap elements at boundaries
  - can be implemented in parallel

Multilevel and Hybrid Methods

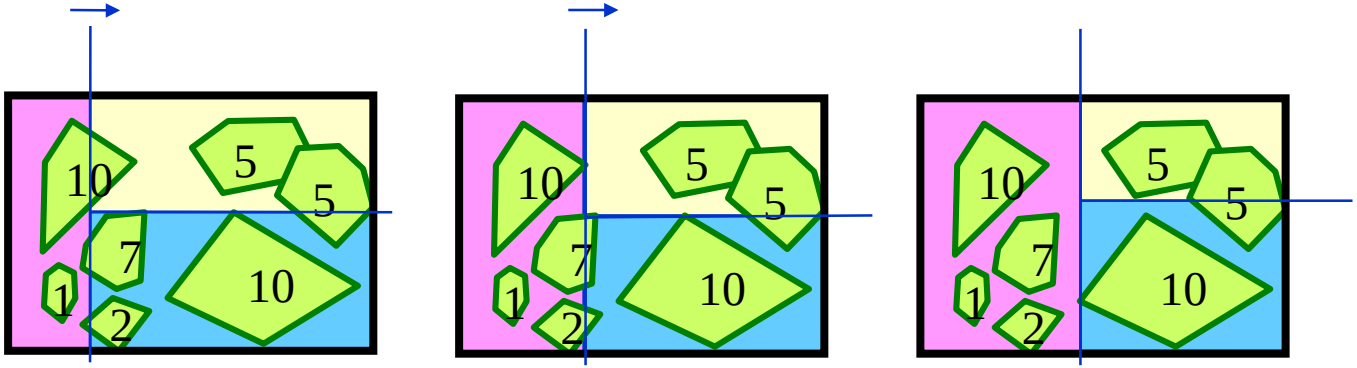
- Multilevel methods
  - Construct coarse graph/mesh as approximation
  - Partition coarse mesh
  - Project to fine mesh
  - Refine
  - Can do hierarchically
- Hybrid methods
  - e.g. combine multilevel with local refinement at each level
  - e.g. spectral may be better than inertial, but inertial plus KL may be better and faster than pure spectral

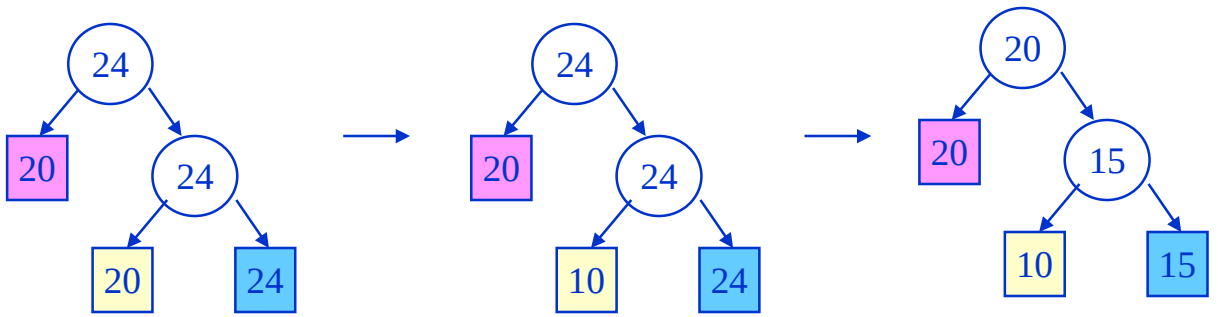
Our Approach

- 1D case: Partition the screen into vertical strips
  - Define the cost function as the number of primitives overlap each tile.
  - start from any tile assignment, moving the cut so that the tiles on both side of it have costs as balanced as possible, repeat until cannot move any cut.



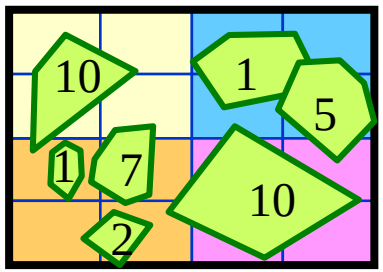
Our approach: 2D case

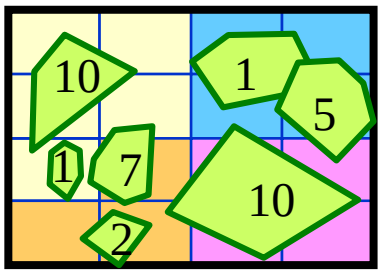




Tile swapping

Starting from a static assignment, and swap cells on the boundary





17	16
20	15

18	16
19	15

Applying Tree Partitioning to Parallel Rendering

- Divide image plane into small cells
- For each bounding box, increment cost of corr. Cells
- Build cost tree with these cells as leaves
- Each tree cell holds:
  - total pixel cost for that cell
  - total polygon cost for all polygons fully contained in cell
  - list of polygons (with costs) that are partly contained in cell
- Partition using costzones
  - but traverse partial polygons list to see if already in partition
- For display wall:

Static Plus Refinement Approach

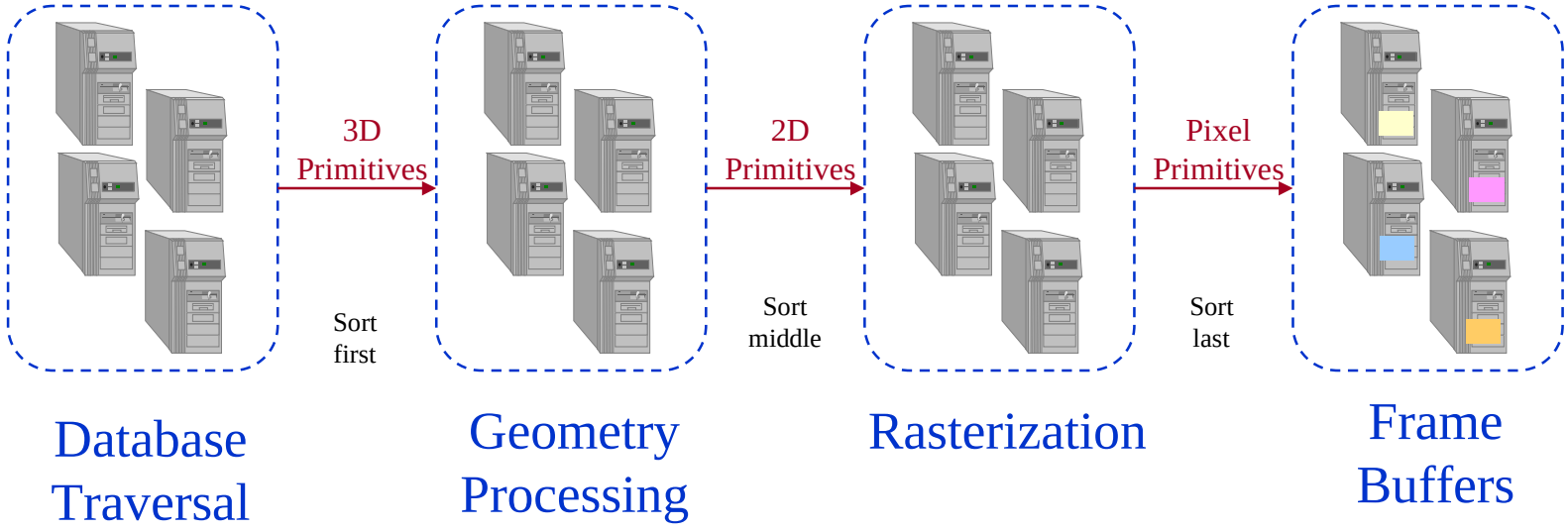
- Divide into regions that match projectors
  - a node is responsible for all tiles in its region
- Use KL or Jostle refinement to rebalance at boundaries
  - use a tile or basic cell as unit of refinement
  - tile can match hardware rendering unit
- Polygon cost of a tile
  - keep track of polygons that cross different faces of tile
  - if they cross an " internal" face for current partition, no need to subtract this cost from this partition when tile is moved out of this partition
  - if they cross an " external" face, no need to add this cost to the new partition when tile is moved to it
- Use current partition as initial partition for next frame

Taxonomy of Partition Algorithms

- Partition
  - What types of splits?
  - How choose where to split?
- Merging
  - How determine initial tiles?
  - How choose tiles to merge?
- Optimization
  - What is the state space?
  - What are the operators?
  - What is the objective function?

Can partition …
- Prior to rendering
- While rendering

Previous Approaches

- Parallel rendering classifications (Molnar94):
  - Sort-last (object load-balance, sort each pixel)
  - Sort-middle (sort between geometry and rasterization)
  - Sort-first (sort before geometry processing)

Usually tightly-coupled processors