Handling Large Numbers of Similar Items

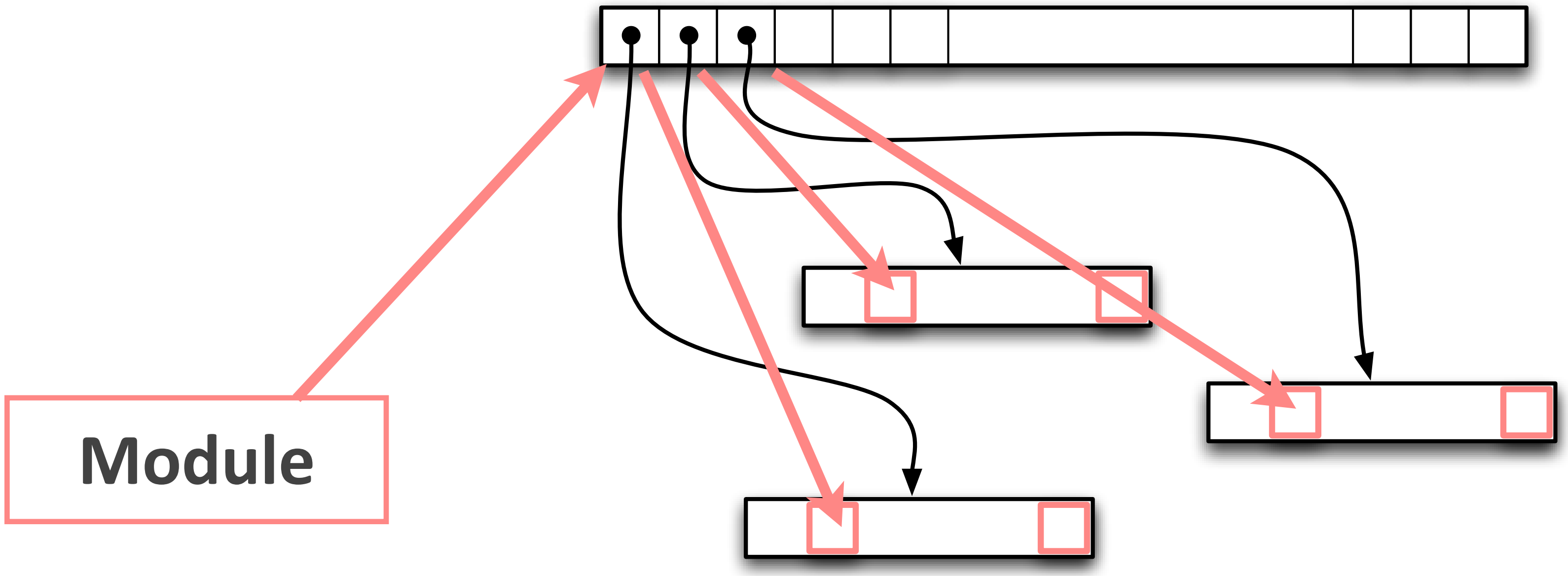
Performance

- fast access to attributes:
  not all attributes are accessed all the time
- avoid indirection

Extendibility

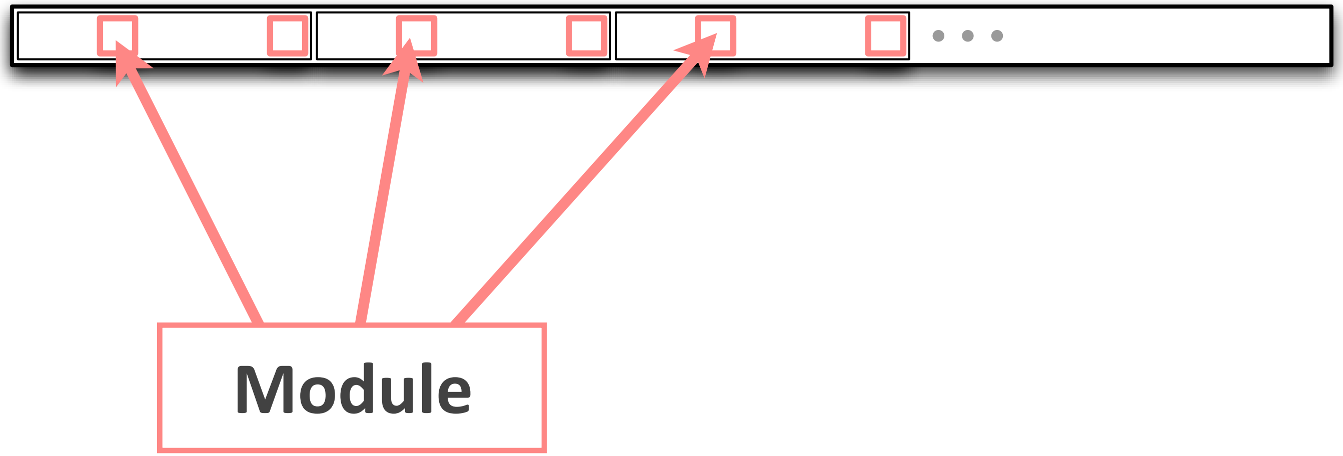
- add additional attributes later in the design
- variable numbers of attributes
- hide attribute changes behind interfaces

Conventional Approach using Classes



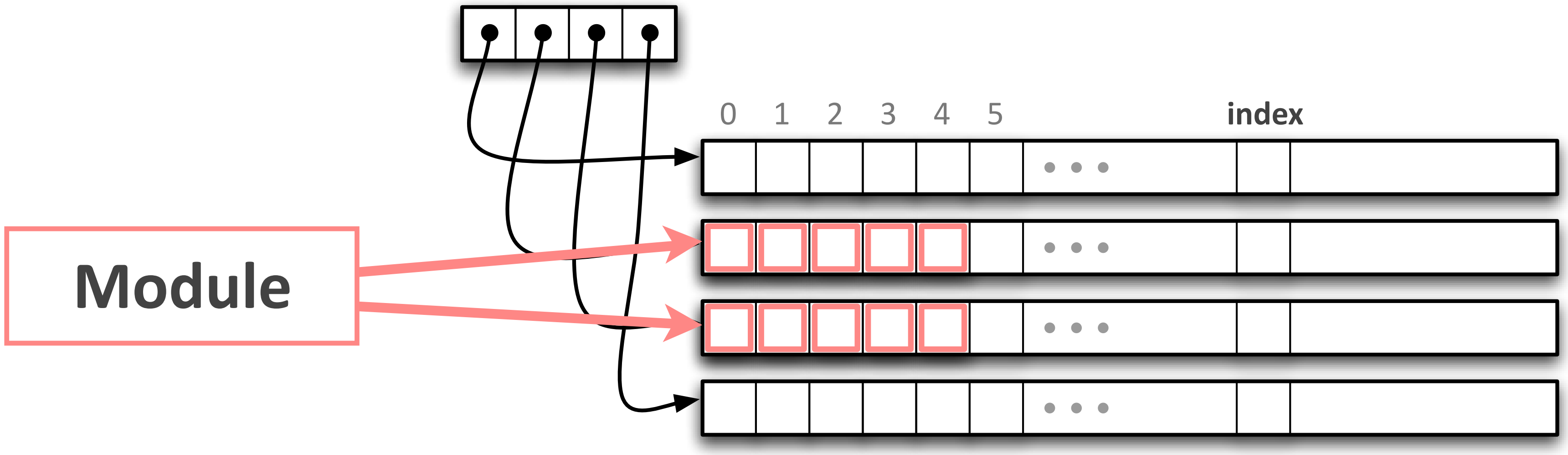
- access indirect (array and object)
- access cache inefficient, if only a few fields are accessed
- flexibility and shielding of modules via class inheritance

Conventional Approach using Structures



- access direct
- access cache inefficient, if only a few fields are accessed
- flexibility and shielding of modules via generics and interfaces

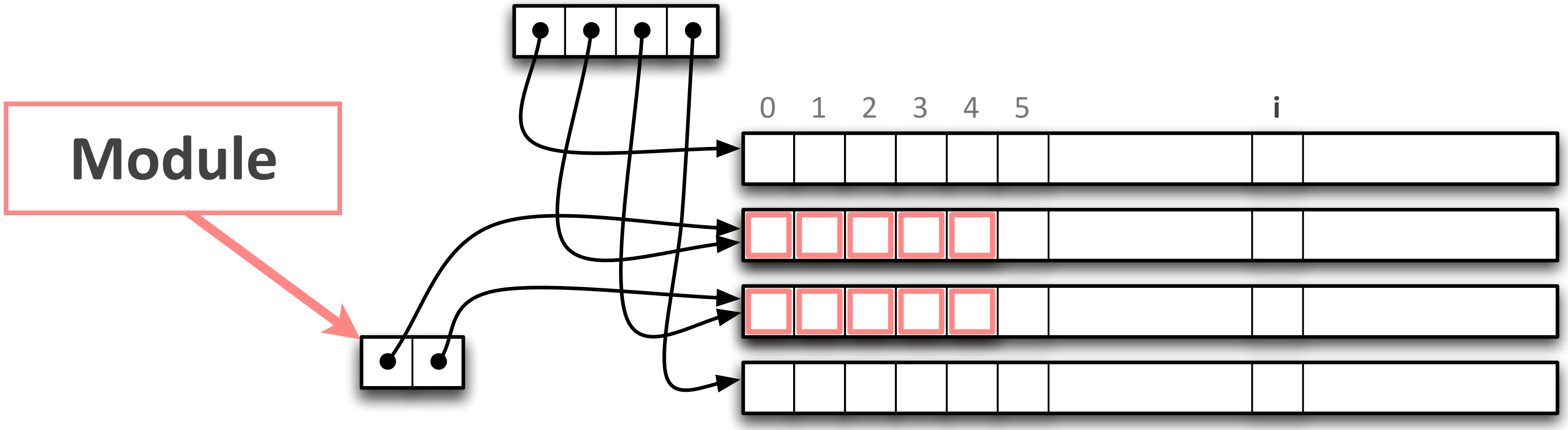
Transposed Approach



Only consider sets of objects

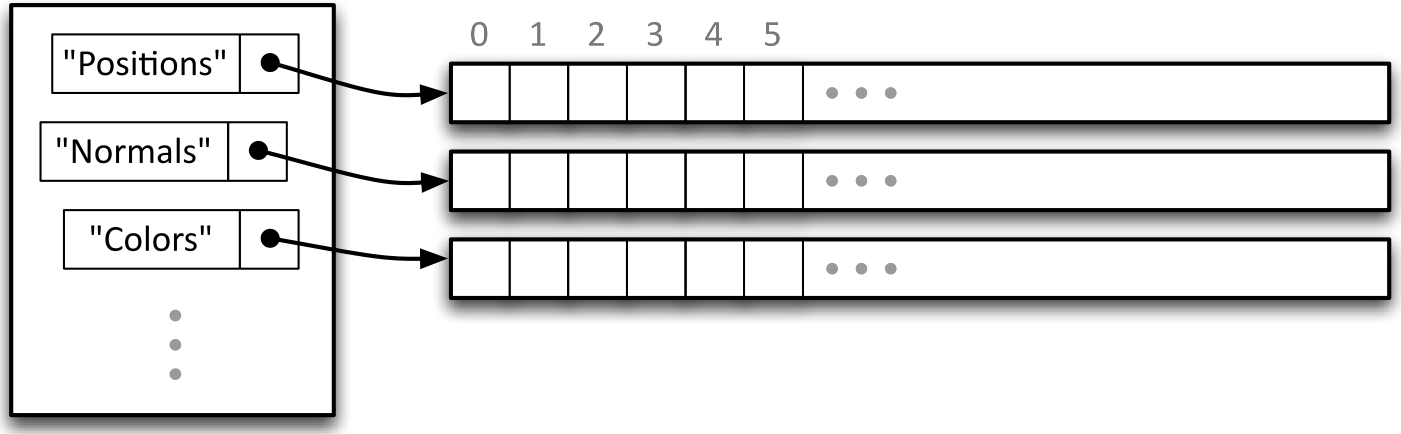
- attributes are stored in arrays of primitives (**int**, **float**, **V3f**, …)
- individual objects identified by their **index**
- access direct and cache efficient

Shielding of Modules via Facades



- the **Facade** hides changes in the attributes of the object set
- the module cannot access attributes it does not need

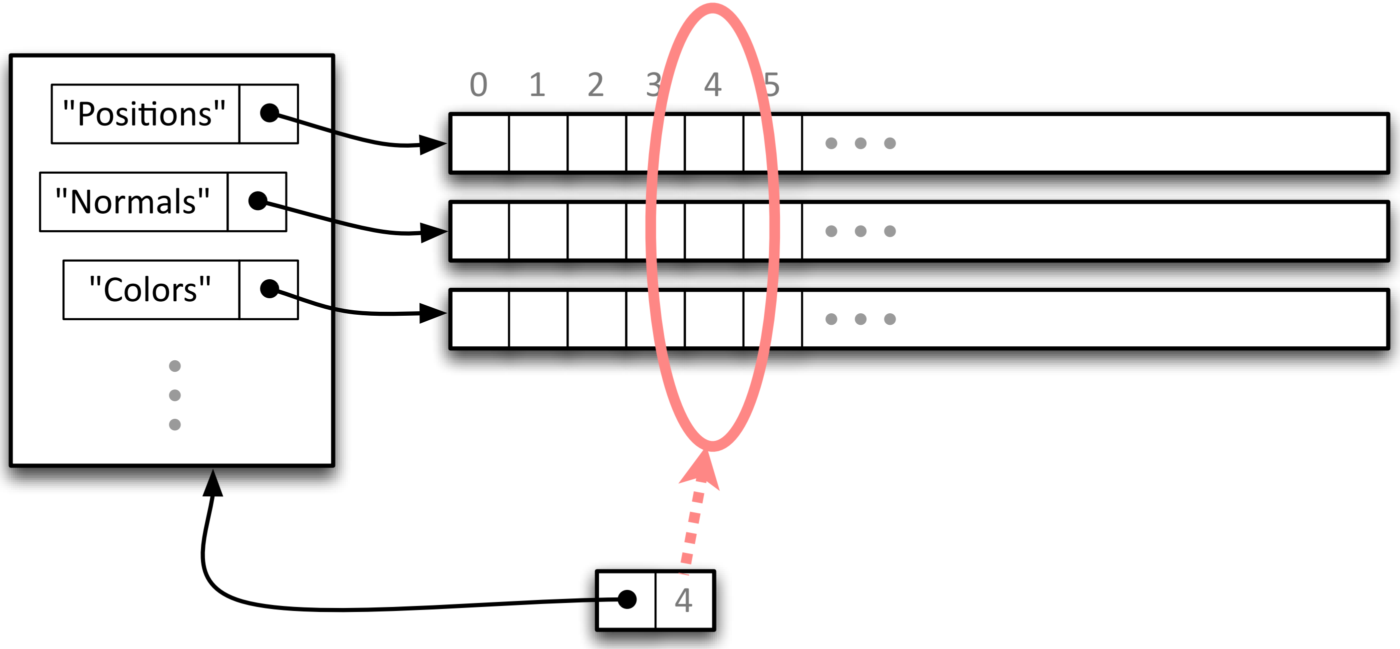
Flexibility in the Transposed Approach



Dictionary (Hash-table) of attribute-arrays (primitives)

- attribute names as keys
- flexible in the number of attributes (even at run-time)

Identifying Objects across Object Sets



Light weight object facade

- contains reference to the object set, and index of object
- all attributes of single object can be accessed via interfaces

Implications of the Transposed Approach

Performance

- gather items in sets with the same attributes
- design algorithms to take advantage of fast linear access
- avoid resizing/modifying object sets, create new ones instead

Extendibility

- simple to add or remove attributes

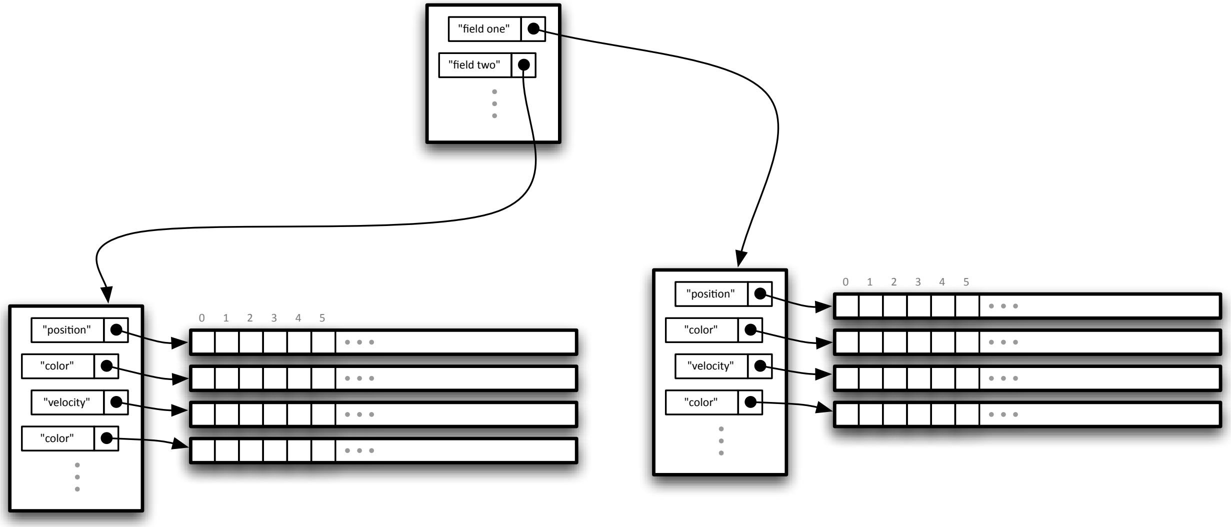
Shielding of Modules

- modules only get access to necessary fields via facades

Geometry Generation Example: Reading a VRML File

Parsing VRML file

- build hierarchical in-memory representation of VRML file
- parse intermediate nodes into Dictionaries of Dictionaries
- parse leaf nodes into Dictionaries of primitive arrays:



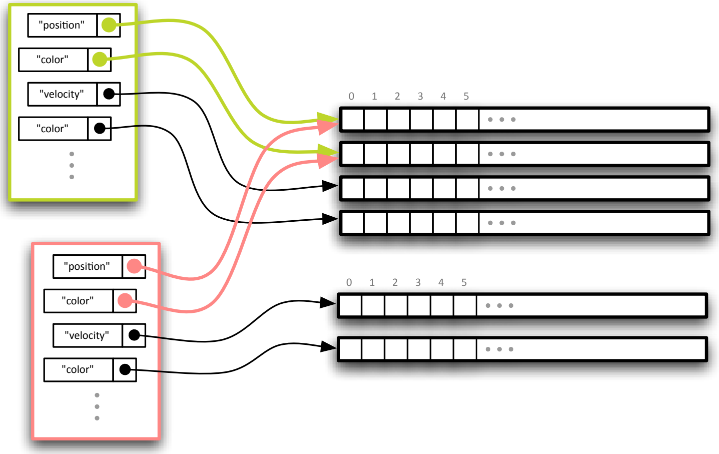
Geometry Processing

Processing modules access dictionaries of primitive arrays

- add additional attributes (primitive arrays) during processing
- create new dictionaries of primitive arrays

Avoid copying of primitive arrays

- if an attribute can be used without change, it is not copied



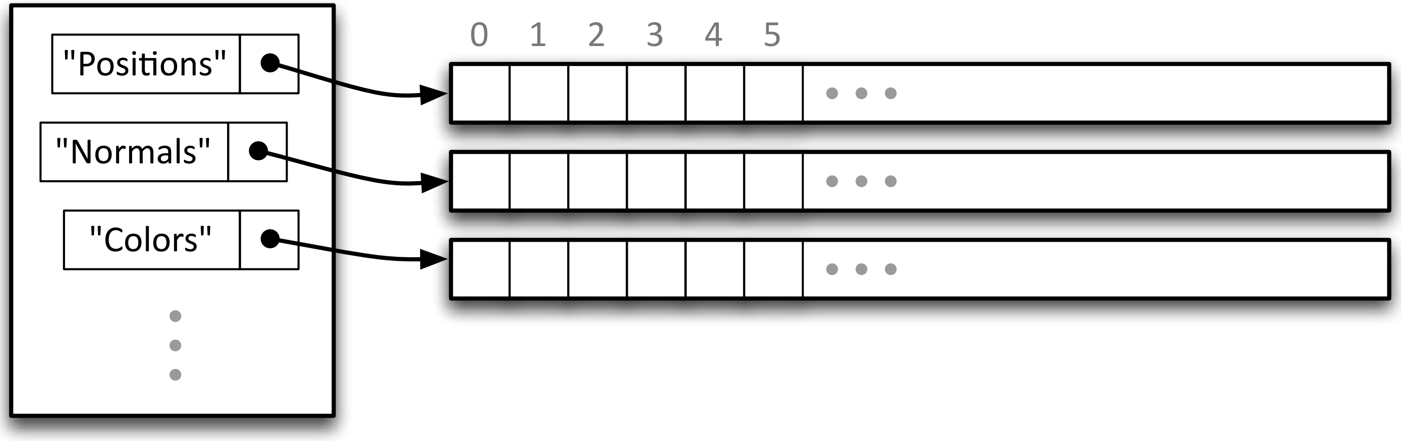
Prepare primitive arrays for fast rendering

- create arrays of primitives so that they can be directly submitted to graphics hardware

Rendering

Submit sets of items to the rendering hardware

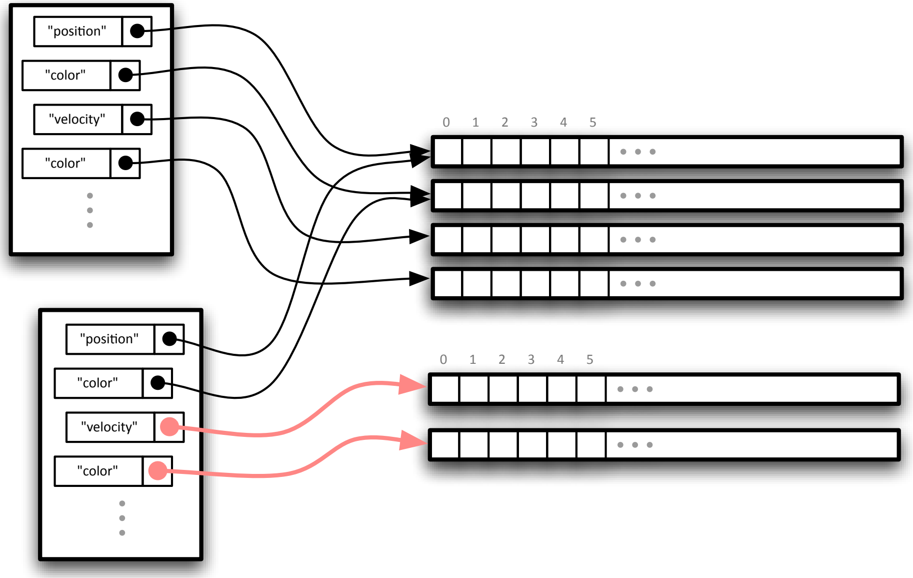
- dictionaries of primitive array have been prepared by geometry processing
- arrays of primitives are bound as Vertex Buffer Objects (VBOs)
- rendering calls are submitted to display sets of VBOs



Parallelizing Geometry Processing

- operate on primitive arrays in parallel
- do not modify existing dictionaries of primitive arrays
- newly created dictionaries reference existing primitive arrays

- **copy-on-write semantics**: create new arrays, instead of modifying existing arrays



Literature

Pitfalls of Object Oriented Programming

- http://research.scee.net/files/presentations/gcapaustralia09/Pitfalls_of_Object_Oriented_Programming_GCAP_09.pdf