Geometric Templates for Improved Tracking Performance in Monte Carlo Codes

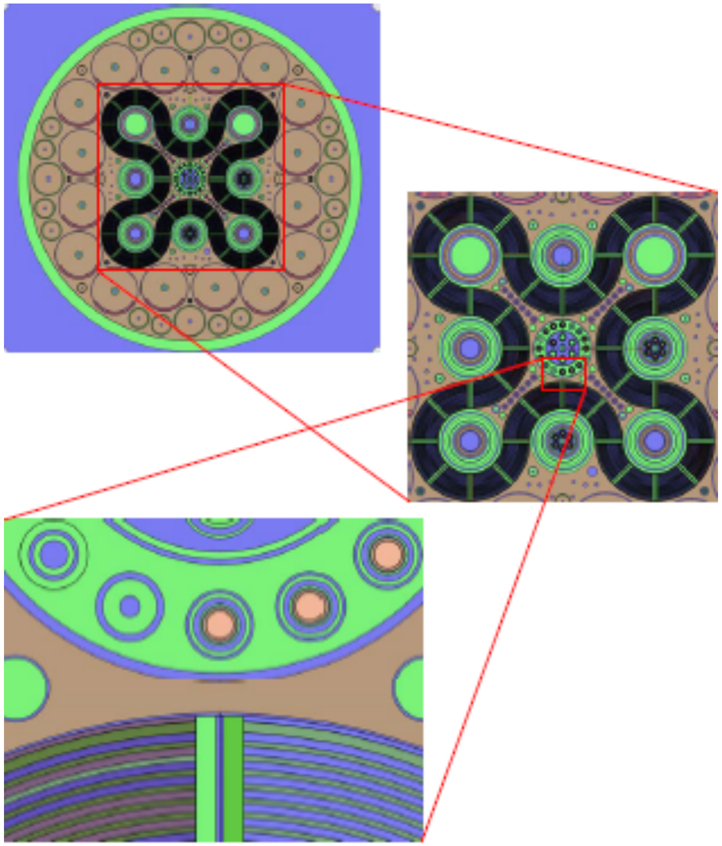
Brian R. Nease, David L. Millman,
David P. Griesheimer, and Daniel F. Gill

Bettis Laboratory, Bechtel Marine Propulsion Corp.

Joint International Conference on Supercomputing in Nuclear Applications
and Monte Carlo 2013 (SNA + MC 2013)
La Cité des Sciences et de l'Industrie, Paris, France, October 27–31, 2013

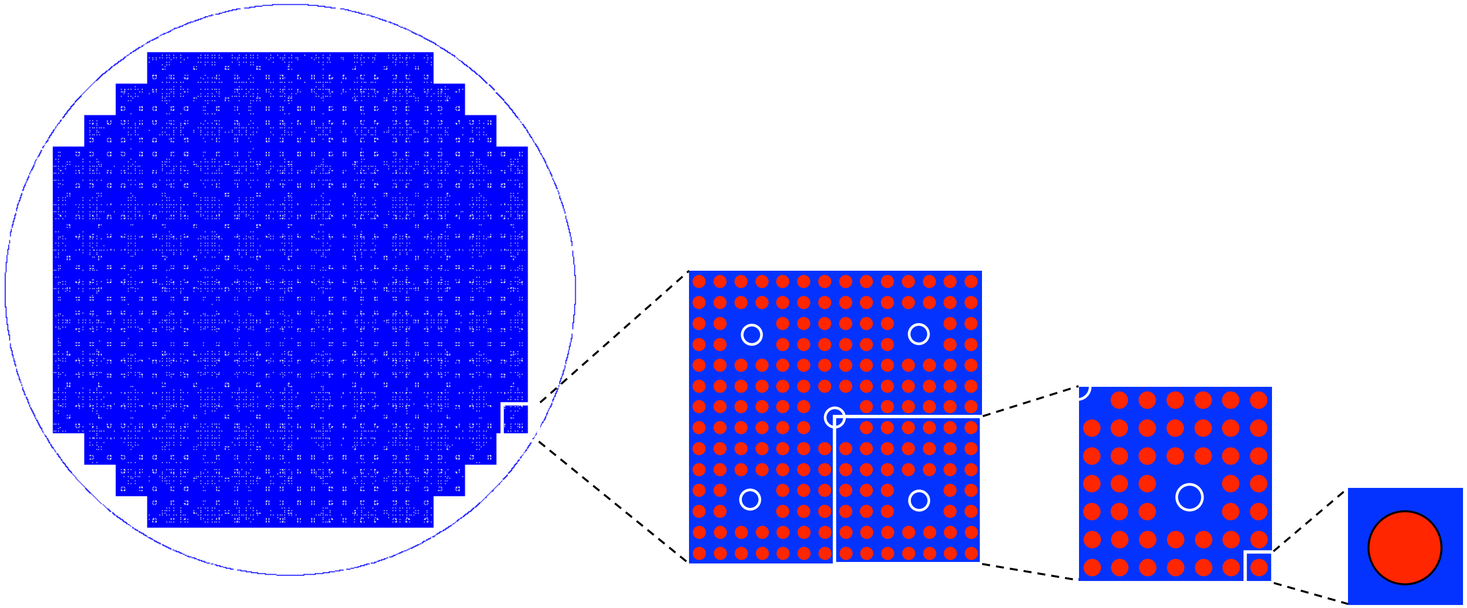
Motivation

- In Monte Carlo codes, a large fraction of simulation time is spent in **particle tracking**
- Most codes use constructive solid geometry (CSG)
  - Flexible geometric representation
  - Slow to track over
    - Cannot take advantage of known geometric characteristics, such as hierarchy or repetition



Motivation

- But, flexibility is not always needed in models!
- Most codes have extended their representation beyond CSG
  - Often combine several different systems
    - Universes, lattices, repeated structures (MCNP)
    - Grids, 2-D lattice (MC21)
  - Typically hard-coded, not extendable
  - Systems are often very disjoint and provide only minor improvements (such as simplifying user input)



Motivation

- Goal is to develop a framework that unifies the geometry systems in MC codes
  - Must be easily extendable
    - Allow developers to create new systems based on design applications
  - Must allow for systems to be individually tailored
    - CSG provides flexibility, but not speed
    - Other systems could improve tracking speed or memory usage, at the cost of flexibility

New Geometric Framework

- New framework is based on templates and overlays
  - Geometry is separated from properties for better modularity of code

- Templates
  - Contain purely geometric information
    - i.e., shapes and arrangements of objects
  - Developer creates multiple types of templates
  - Each template is limited in what it can represent
    - Less flexibility allows for better optimization
    - Each template typically based on a simple, commonly repeated geometric shape
  - Defined on a tile in a unique coordinate frame
  - Each template has unique tracking routines and data structures, but each shares a common interface

- Overlays
  - Assign properties to template and maps it to grid coordinate frame
    - i.e., material, temperature, rotation matrix, etc.

Implementation in MC21

- Highly amenable to object-oriented programming
  - All templates share a common interface
  - Each template also has its own private data/methods

Templates

- Shared data:
  - *None*
- Shared methods:
  - `handleScatter(p,d)`
  - `inside(T,p)`
  - `distance(p,c)`
  - `moveToBoundary(p,c)`
  - `crossBoundary(p,c)`

Overlays

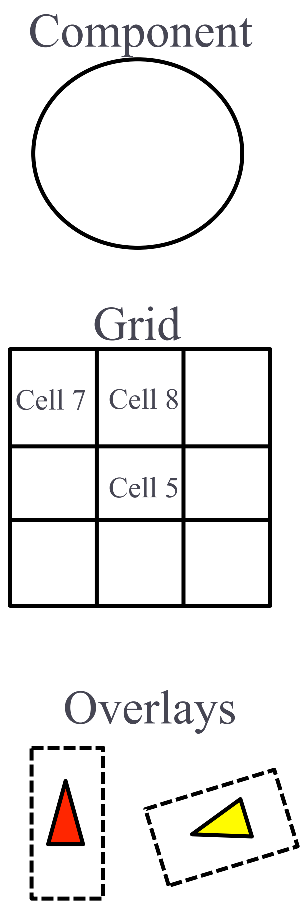
- Data:
  - `templateID`
  - `properties`
  - `translationVector`
  - `rotationMatrix`
- Methods:
  - `handleScatter(p,d)`

`p`	=	particle
`c`	=	template cell
`d`	=	distance
`T`	=	template

Implementation in MC21

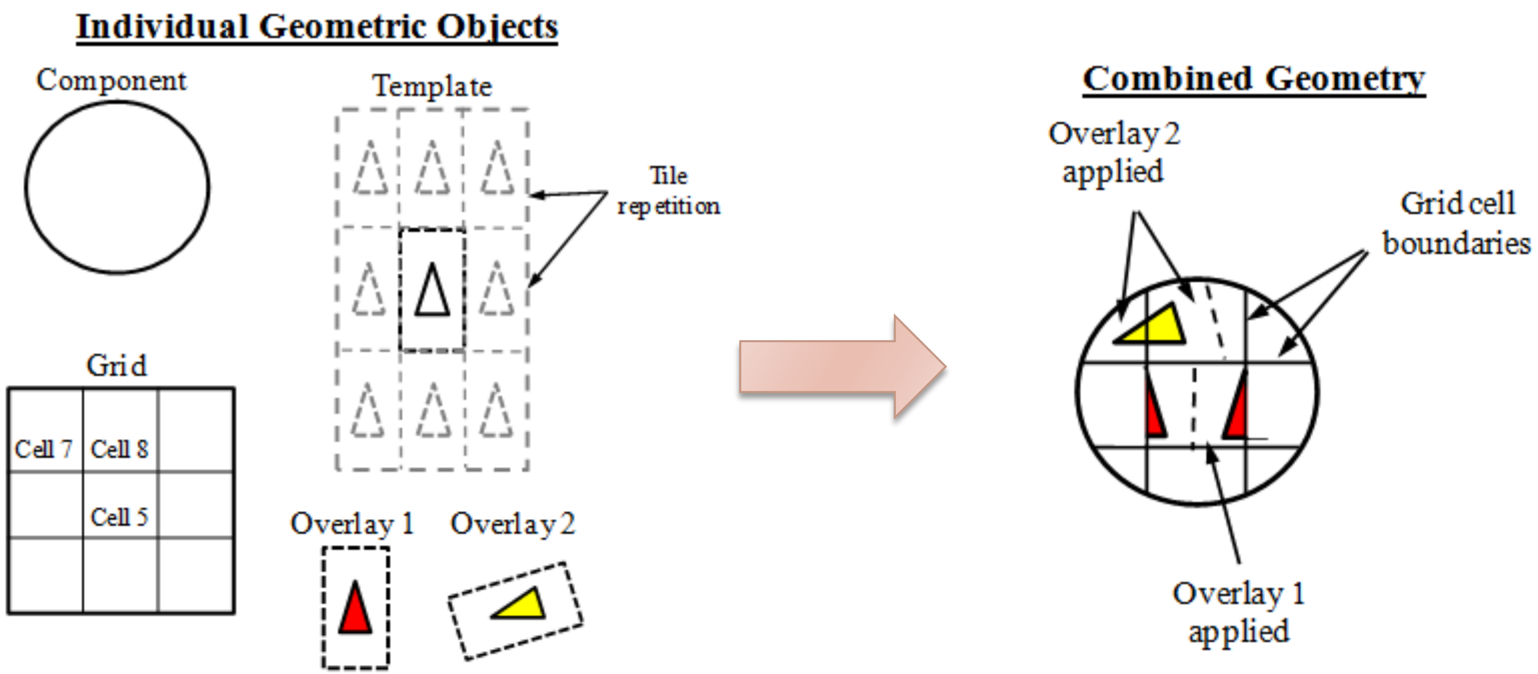
- For improved efficiency each template has its own associated cache
  - Stores information that can be reused by different operations (for instance, the information computed in the `inside` routine can often be reused by the `distance` routine)
  - Cannot be modified by other templates
  - Each cache can determine whether its information is stale

Implementation in MC21

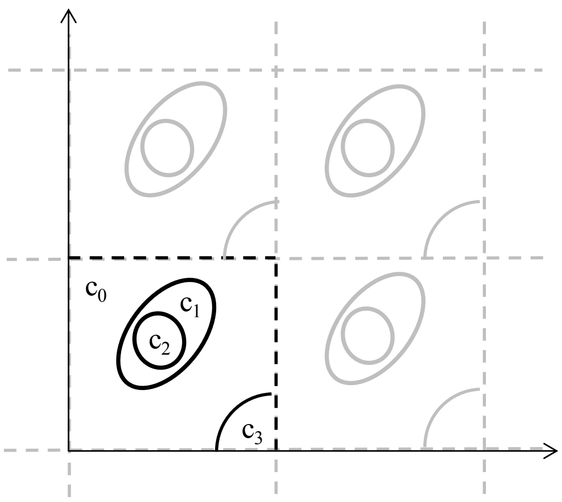


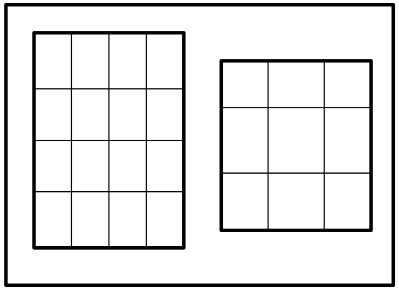
- Altogether, MC21 has three levels of representation
  - Components
    - CSG representation
    - All models must have at least one component
    - Each component contains a grid
  - Grids
    - Subdivides interior of component
    - Grid cells that extend beyond component boundary are truncated
    - Each grid cell can be assigned an overlay
  - Overlays
    - Multiple overlays can use the same template

Implementation in MC21



Implementation in MC21

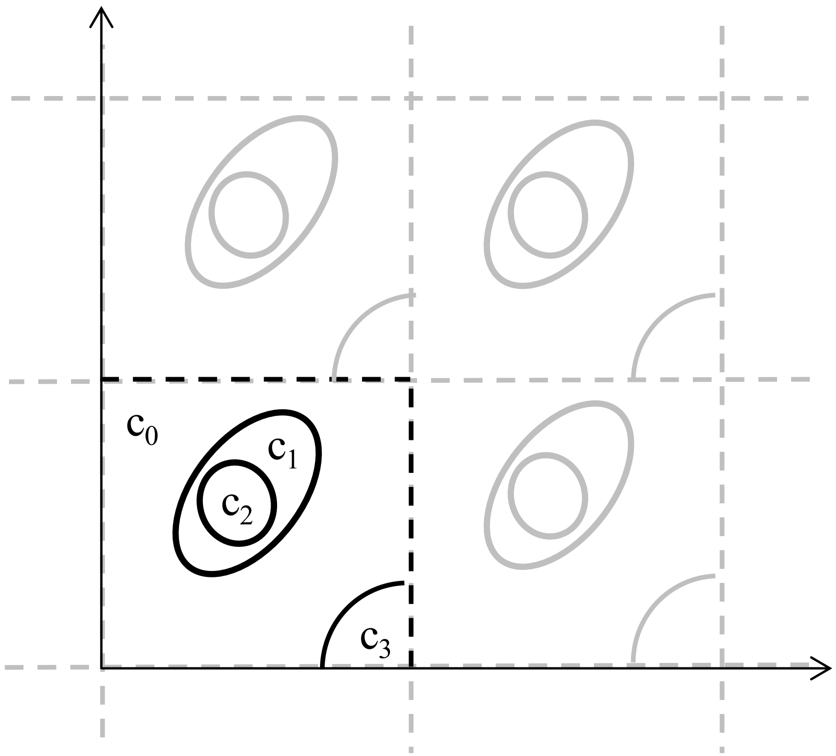




- MC21 has three template types
  1. Repeated ellipse template
     - Used to model infinitely repeating pattern of parallel, extruded elliptical cylinders
  2. Repeated ellipsoid template
     - Used to model infinitely repeating pattern of ellipsoids

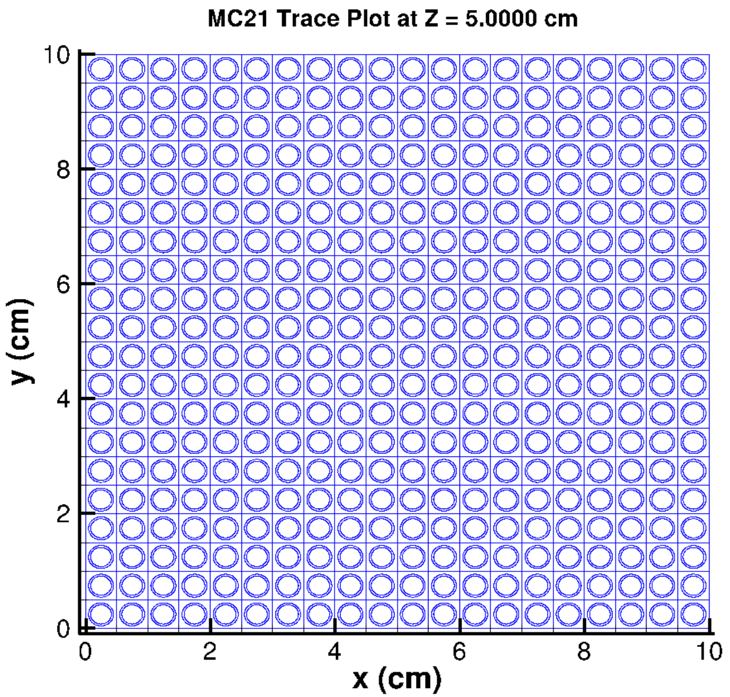
       (3-D extension of repeated ellipse template)
  3. Box template
     - Used to model a pattern of gridded boxes

Repeated Ellipse Template



- Description
  - Infinite tile extent in z
  - One or more ellipses can be defined on the tile
  - Ellipses cannot intersect within tile
  - Portions of ellipses that extend beyond tile are truncated
  - Ellipses may have arbitrary nesting, positions, rotations, and sizes
- Implementation
  - Data stored in a cell tree data structure represent
  - Ellipses represented as a general conic polynomial in matrix form
  - Tile tracking occurs via a periodic boundary condition

Repeated Ellipse Template



Case 1 - Each cylinder a component
Case 2 - Template applied to coarse grid cells
Case 3 - Template applied using repeat capability

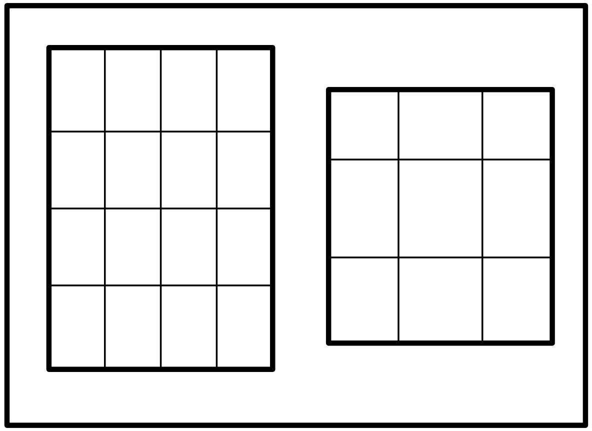
	Runtime (in seconds)		
Problem Size	Case 1 Component	Case 2 No Repeat	Case 3 Repeat
10×10	3.33	1.03	0.77
20×20	13.83	1.70	1.18
30×30	40.63	2.37	1.57
40×40	92.59	2.99	2.01

- Number of tests reduced from $O\left(k \cdot N^2\right)$ to $O\left(k\right)$
- Case 3 has least amount of event processing

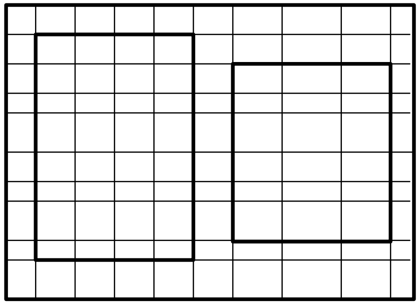
	Memory Usage (in MB)		
Problem Size	Case 1 Component	Case 2 No Repeat	Case 3 Repeat
1000×1000	3959	71	0.26

- Components are expensive to store
- Case 2 stores unique properties for each cylinder

Box Template

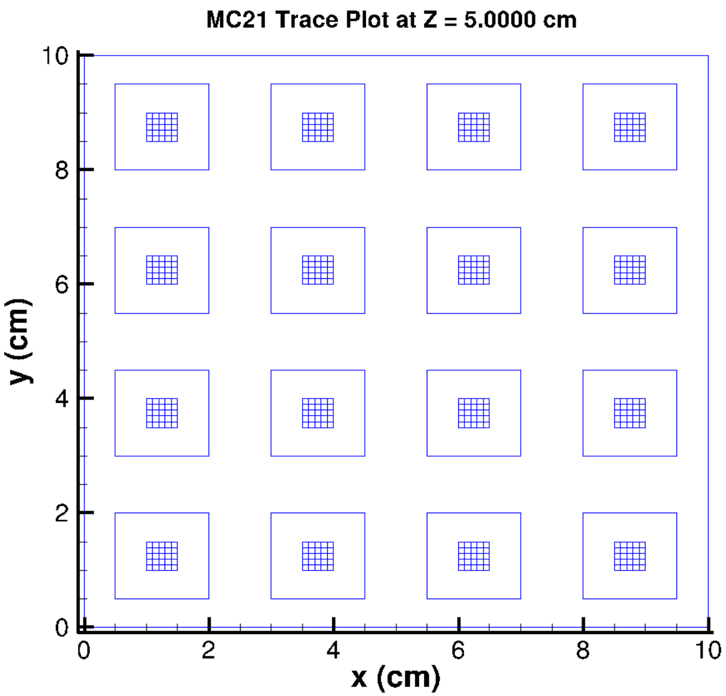


Same model with standard gridding applied
Much more memory intensive!



- Description
  - Infinite tile extent
  - Can define one or more axis-aligned boxes
  - Boxes cannot intersect but can be nested
  - Boxes can have internal Cartesian gridding
  - Minimizes necessary grid cells
- Implementation
  - Data stored in a cell tree data structure
  - Boxes represented as simple Cartesian grids

Repeated Ellipse Template



Case 1 - Each box a component w/ internal grid
Case 2 - Entire model gridded (not shown)
Case 3 - Template applied to coarse grid cells

	Runtime (in seconds)		
Problem Size	Case 1 Component	Case 2 Grid	Case 3 Template
10×10	3.92	0.65	0.55
20×20	16.38	0.82	0.65
30×30	40.04	0.94	0.72
40×40	75.32	1.04	0.77

- Number of tests reduced from $O\left(k \cdot N^2\right)$ to $O\left(k\right)$
- Case 2 is over-gridded, requiring extra event processing

	Memory Usage (in MB)		
Problem Size	Case 1 Component	Case 2 Grid	Case 3 Template
10×10	1.11	0.61	0.41
20×20	4.13	2.00	1.01
30×30	9.54	4.23	2.01
40×40	17.72	7.42	3.41

- Components are expensive to store
- Case 2 is over-gridded, requiring extra memory

Conclusion

- Framework introduced to manage multiple, individually tailored geometry systems
- Highly extendable
  - Easy to add new geometry systems
  - Each system can be optimized for specific design applications
- Final aside:
  - Template/overlay framework works naturally with delta scattering
    - Requires only `inside` routine
  - Appealing way to prototype new templates