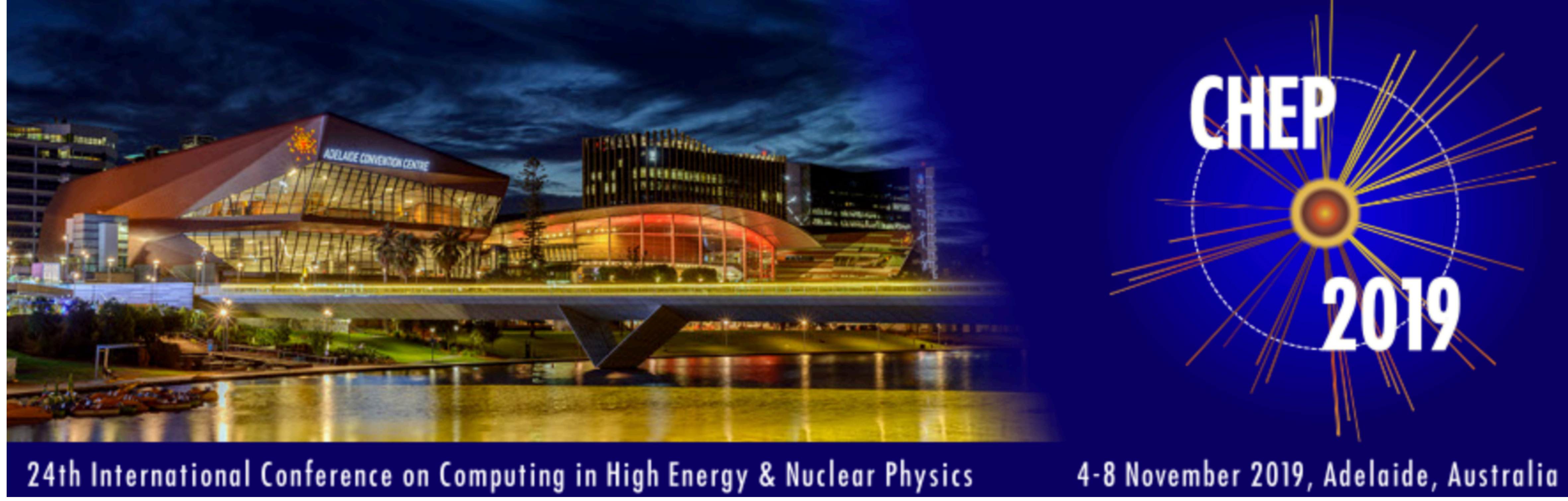
Track2 (Offline Computing); 5-November-2019



A VecGeom navigator plugin for Geant4

Sandro Wenzel (CERN - ALICE), John Apostolakis (CERN - SFT) and Gabriele Cosmo (CERN - SFT)

VecGeom

GEANT4 A SIMULATION TOOLKIT

CERN

ALICE

VecGeom and current Geant4 integration status

https://gitlab.cern.ch/VecGeom/VecGeom

AIDA 2020

https://aidasoft.web.cern.ch/USolids

- VecGeom: Incubated in Geant-V and supported by AIDA2020, targeting common evolution of Geant4/TGeo geometry functionality
  - modernize / revise / optimize geometry algorithms in a single place
  - be performance oriented and use SIMD acceleration as much as possible (multi-particle API, single-particle API)

- Two main components provided for use in simulation:
  - **elementary and composite geometry primitives** (box, tube, boolean, …)
    - distance, containment, etc.
    - bricks to build complex geometries

Integrated into G4 (10.5)
* beneficial when using complex shapes (Polycone, Tesselated, Multiunion)
* CMS reported speedup

VecGeom and current Geant4 integration status

https://gitlab.cern.ch/VecGeom/VecGeom

AIDA 2020

https://aidasoft.web.cern.ch/USolids

- VecGeom: Incubated in Geant-V and supported by AIDA2020, targeting common evolution of Geant4/TGeo geometry functionality
  - modernize / revise / optimize geometry algorithms in a single place
  - be performance oriented and use SIMD acceleration as much as possible (multi-particle API, single-particle API)

- Two main components provided for use in simulation:
  - **elementary and composite geometry primitives** (box, tube, boolean, …)
    - distance, containment, etc.
    - bricks to build complex geometries
  - **navigation in complex geometries**
    - “fast” determination of the next (straight) line intersection of a ray and the distance
    - isotropic distance (safety)
    - determination of current tracking medium

Integrated into G4 (10.5)
* beneficial when using complex shapes (Polycone, Tesselated, Multiunion)
* CMS reported speedup

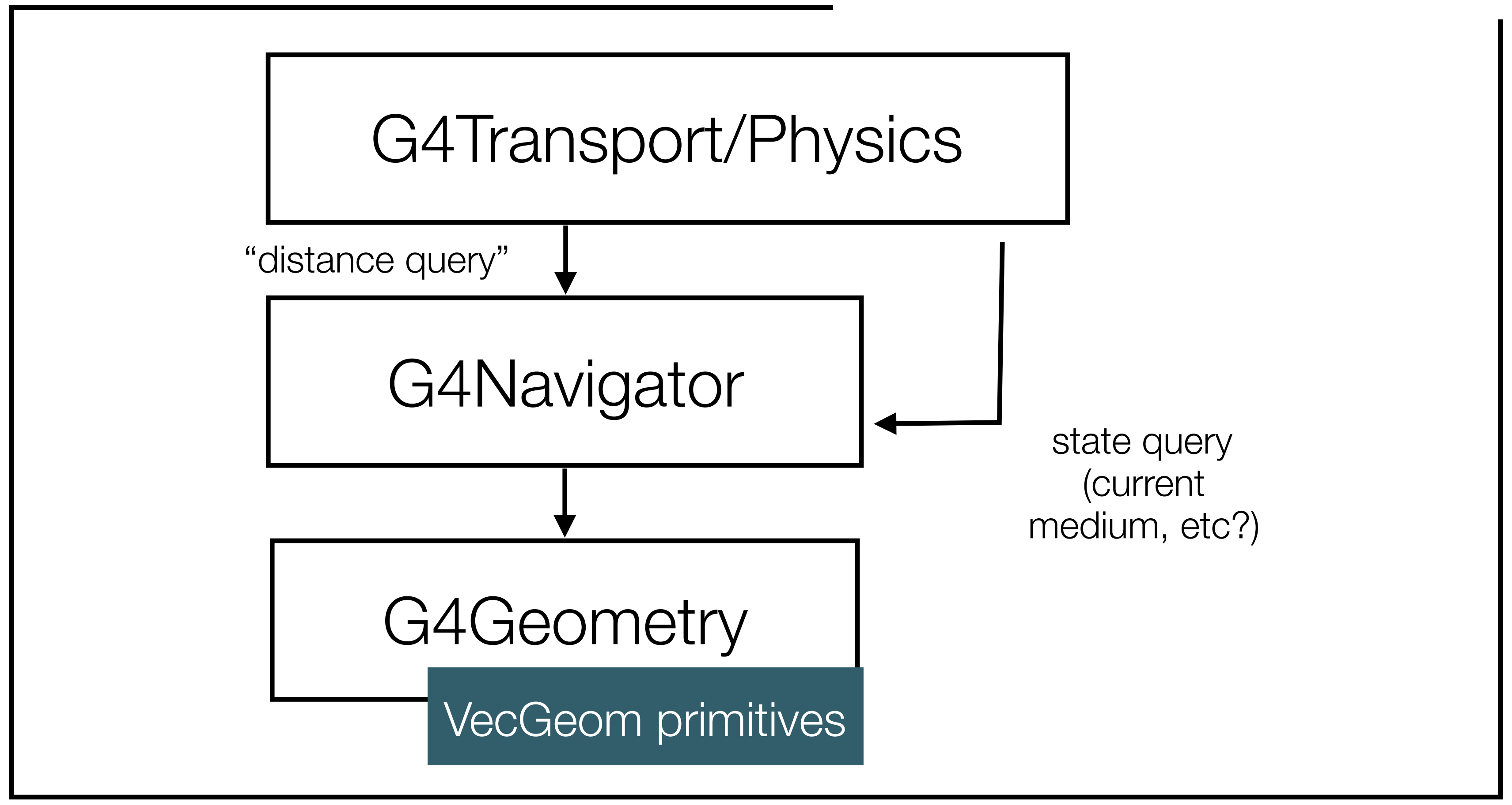
Not yet integrated
* opportunity for further speedup

This presentation discusses advances in this direction

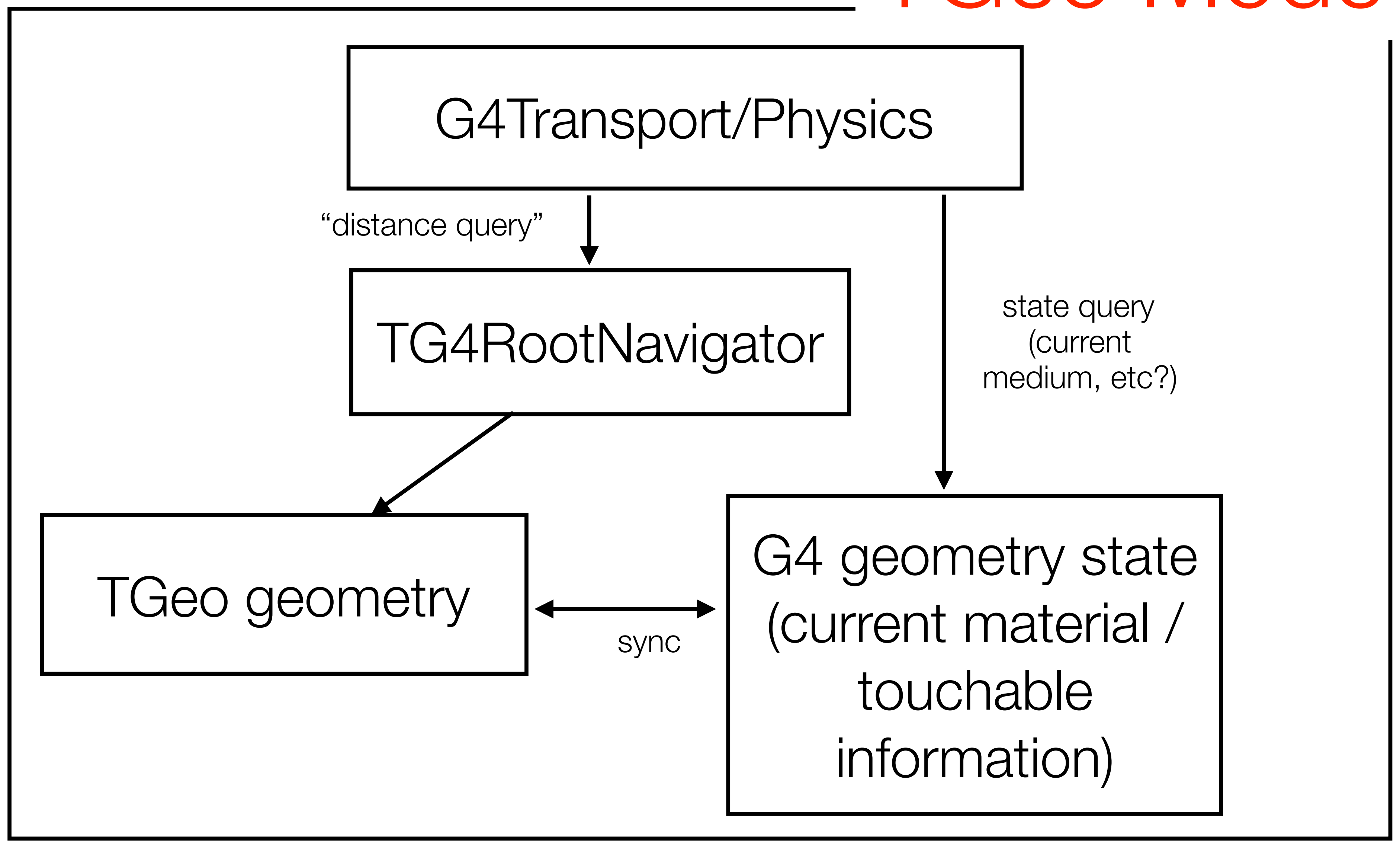
Interest of VecGeom for the VMC ecosystem

- Detector geometry description made with ROOT/TGeo classes in Virtual Monte Carlo (VMC).
- Dedicated `TG4RootNavigator` class interfaces TGeo with Geant4 to use the same TGeo geometry for simulation.
  - mainly because of special features offered by TGeo (assemblies, special solids, overlap handling, …)
- These users can currently not benefit from VecGeom developments.

Native Mode



TGeo Mode

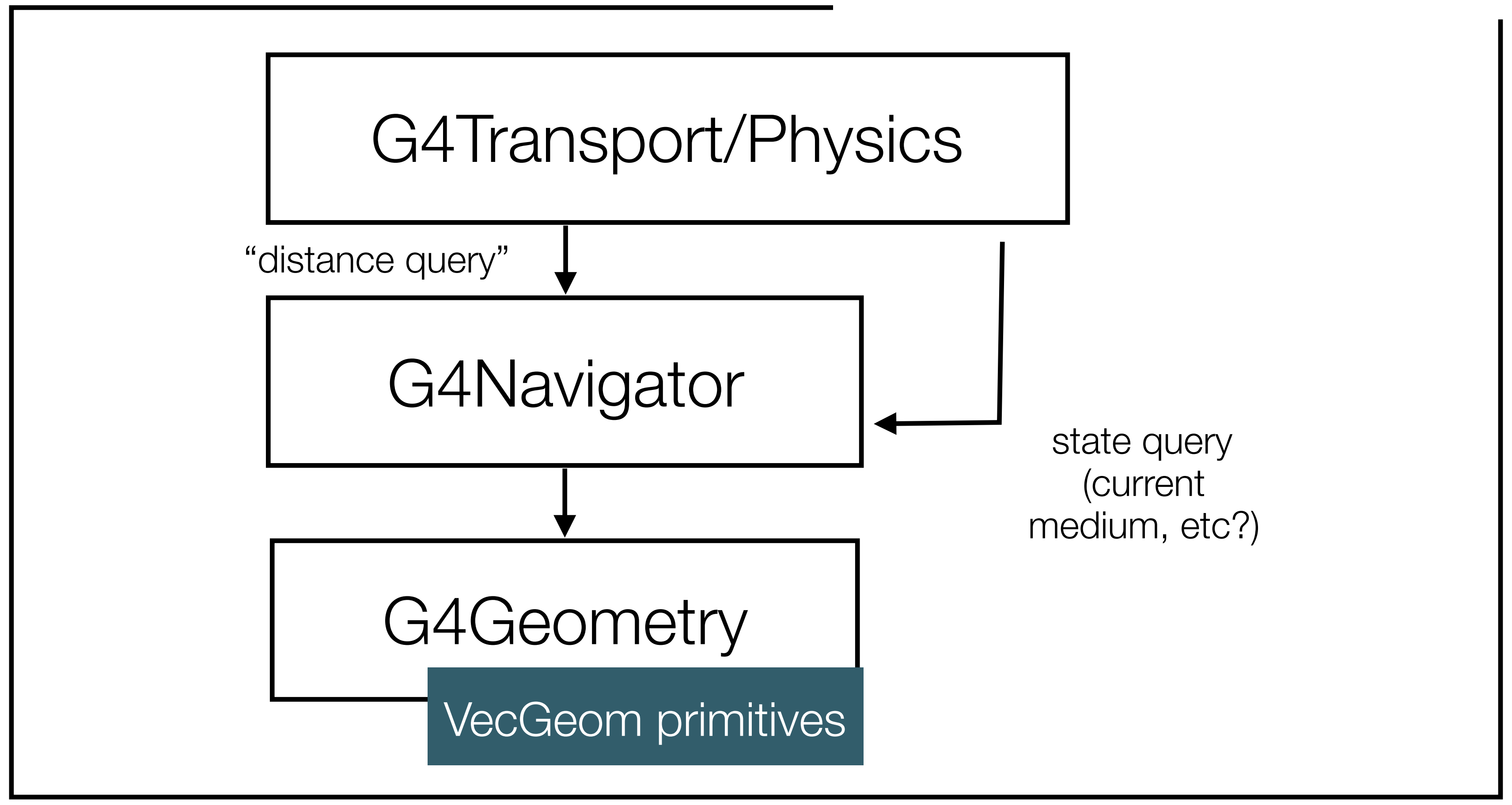


https://github.com/vmc-project/geant4_vmc

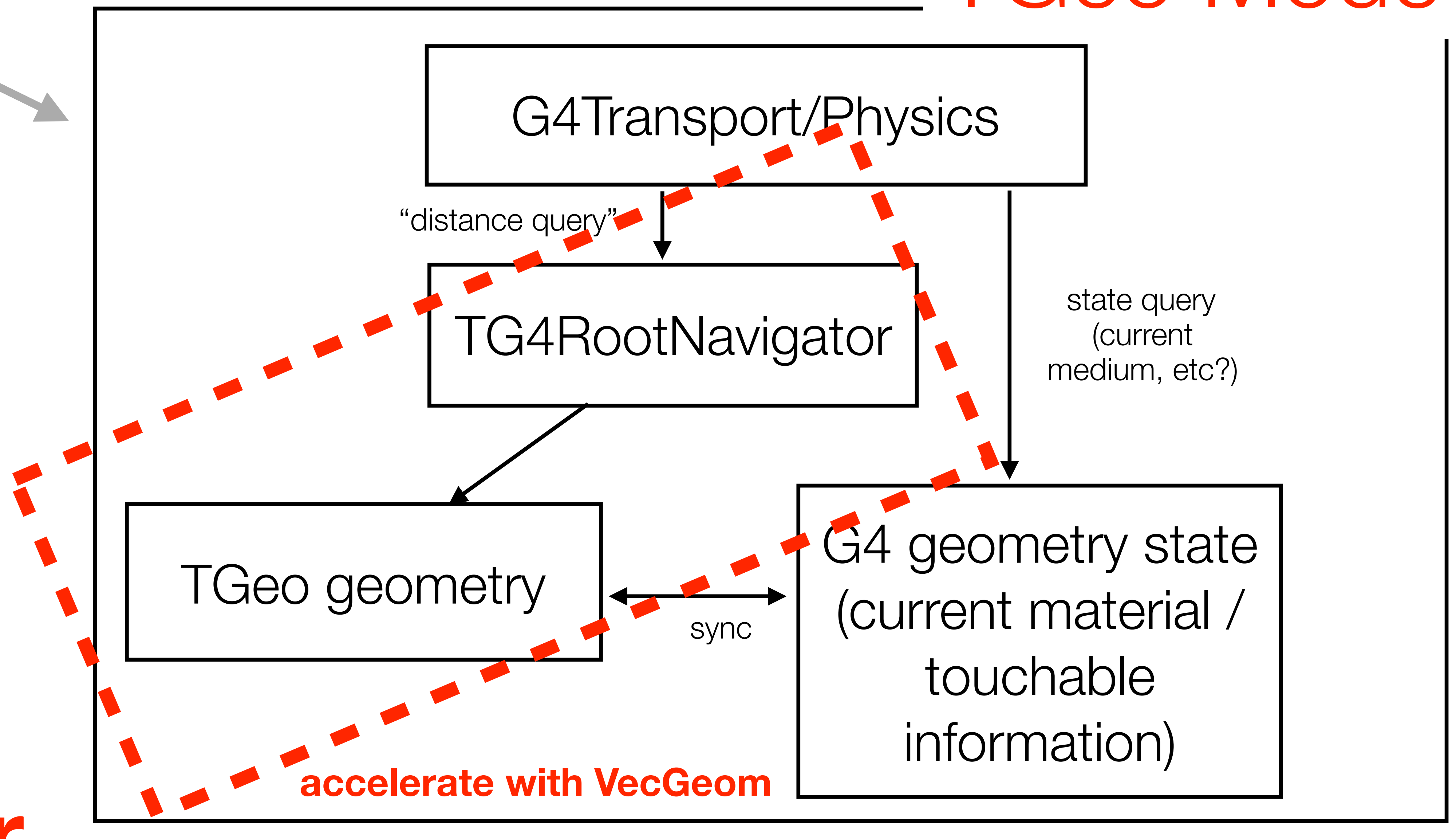
Interest of VecGeom for the VMC ecosystem

- Detector geometry description made with ROOT/TGeo classes in Virtual Monte Carlo (VMC).
- Dedicated `TG4RootNavigator` class interfaces TGeo with Geant4 to use the same TGeo geometry for simulation.
  - mainly because of special features offered by TGeo (assemblies, special solids, overlap handling, …)
- These users can currently not benefit from VecGeom developments.
- **Motivation to provide VecGeom features for these simulations!**

Native Mode



TGeo Mode

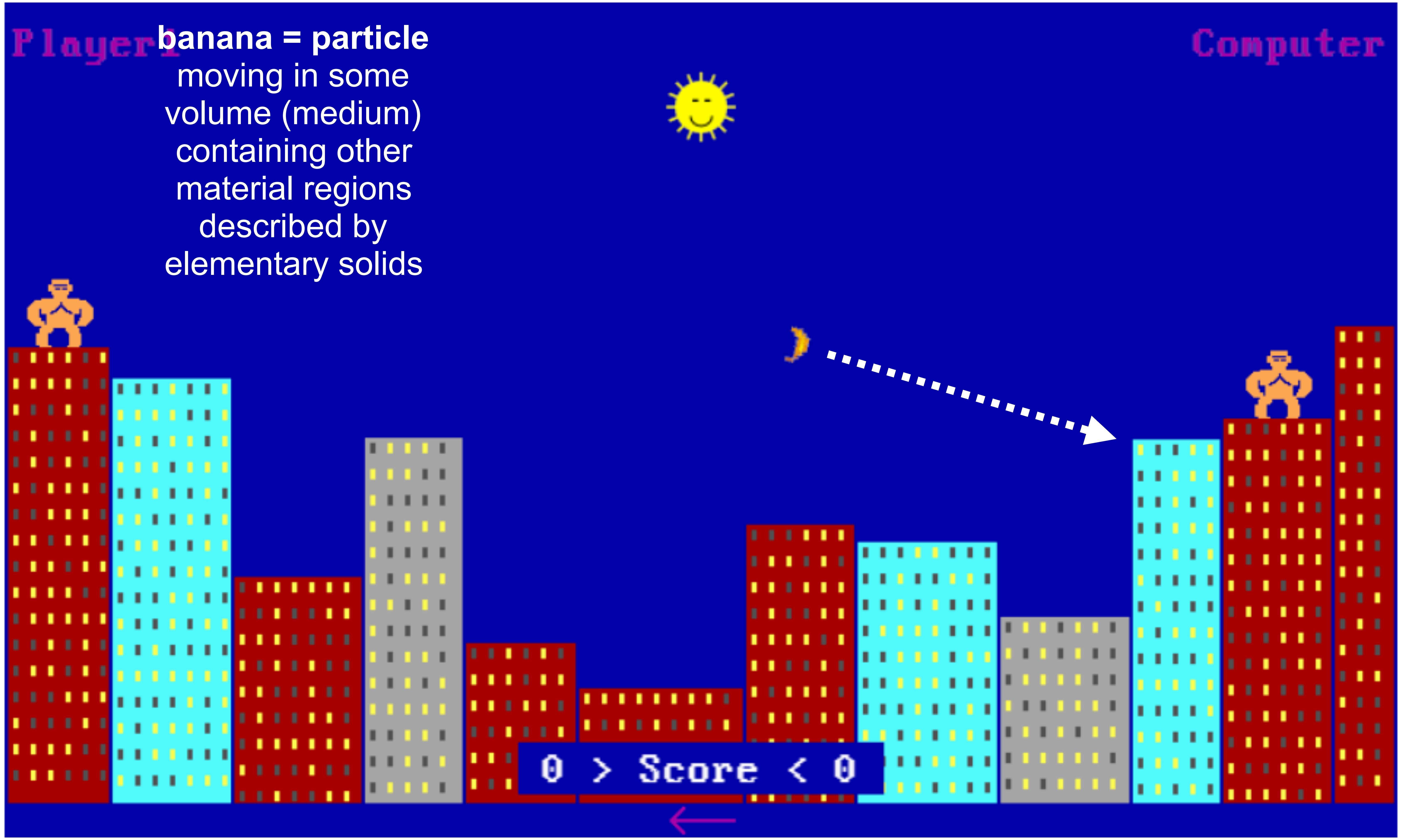


https://github.com/vmc-project/geant4_vmc

The G4Navigator: The main geometry tasks

G4Navigator

1 ComputeStep



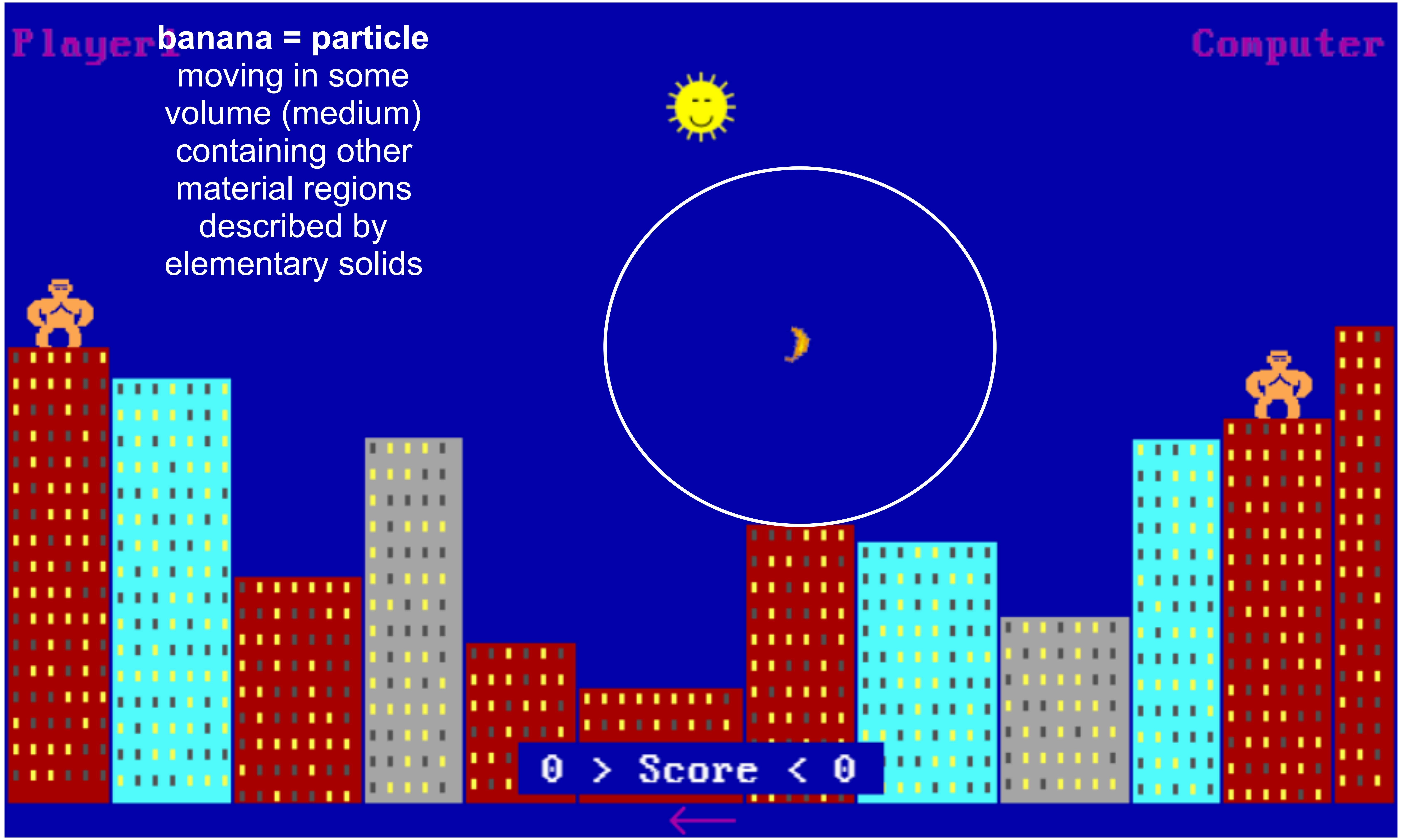
The old QBasic game “Gorillas”

Core parts of particle transport simulation resemble problems found in computer games:

1. Determination of **distance to next object** and **which object** (along current straight line trajectory)

The G4Navigator: The main geometry tasks

G4Navigator	
1	ComputeStep
2	ComputeSafety



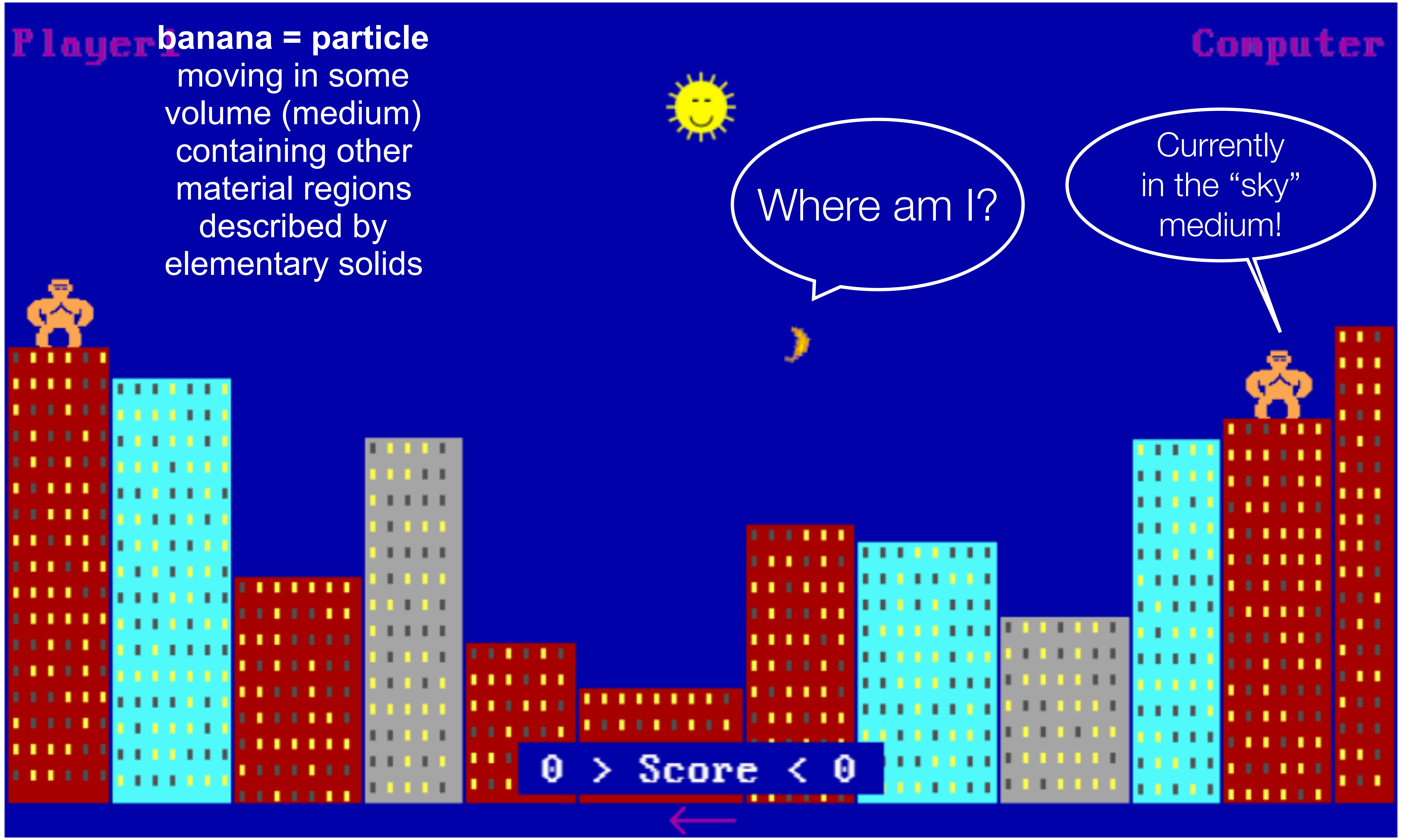
The old QBasic game "Gorillas"

Core parts of particle transport simulation resemble problems found in computer games:

1. Determination of **distance to next object** and **which object** (along current straight line trajectory)
2. Determination of **isotropic distance (safety)** to any other object

The G4Navigator: The main geometry tasks

G4Navigator	
1	ComputeStep
2	ComputeSafety
3	Locate

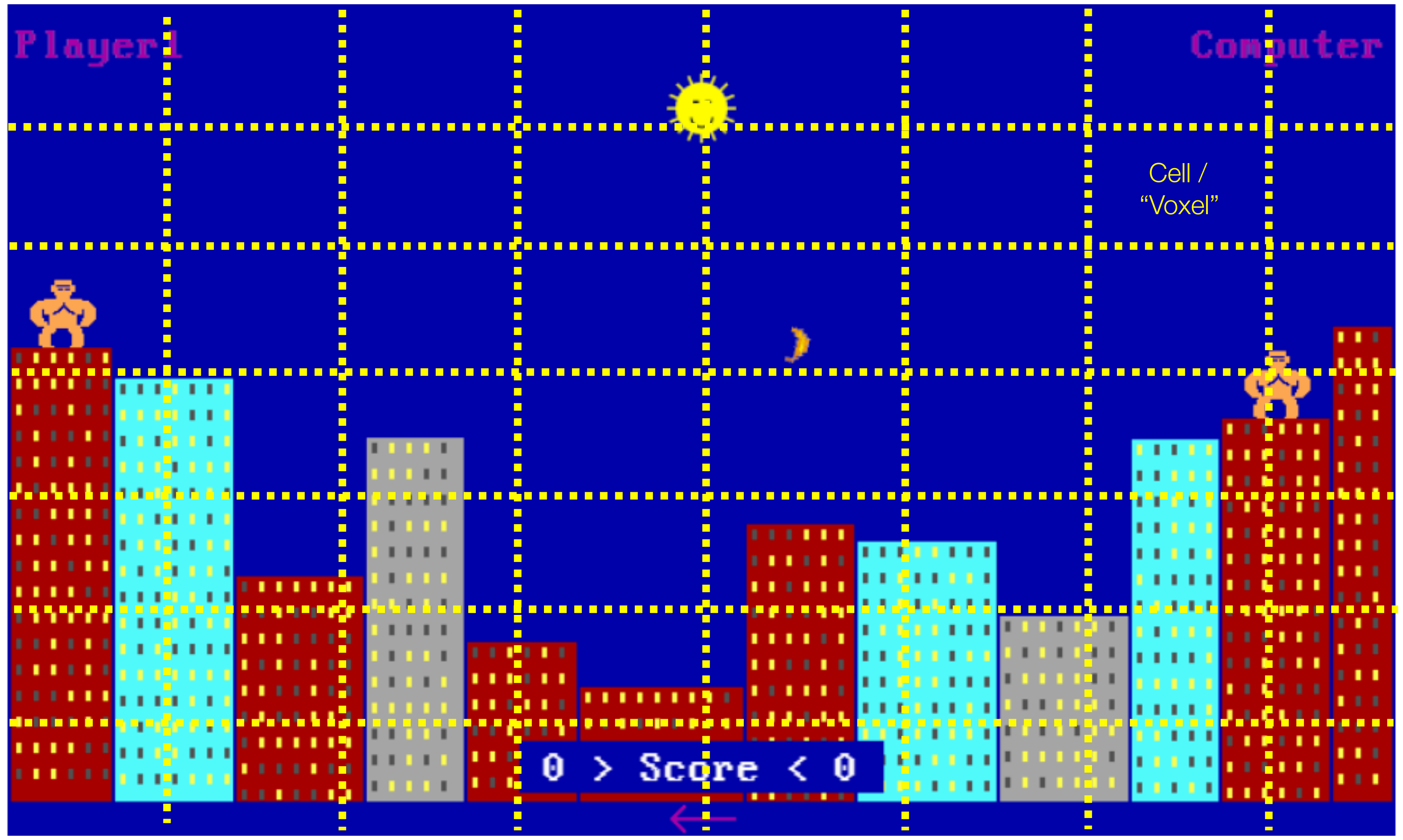


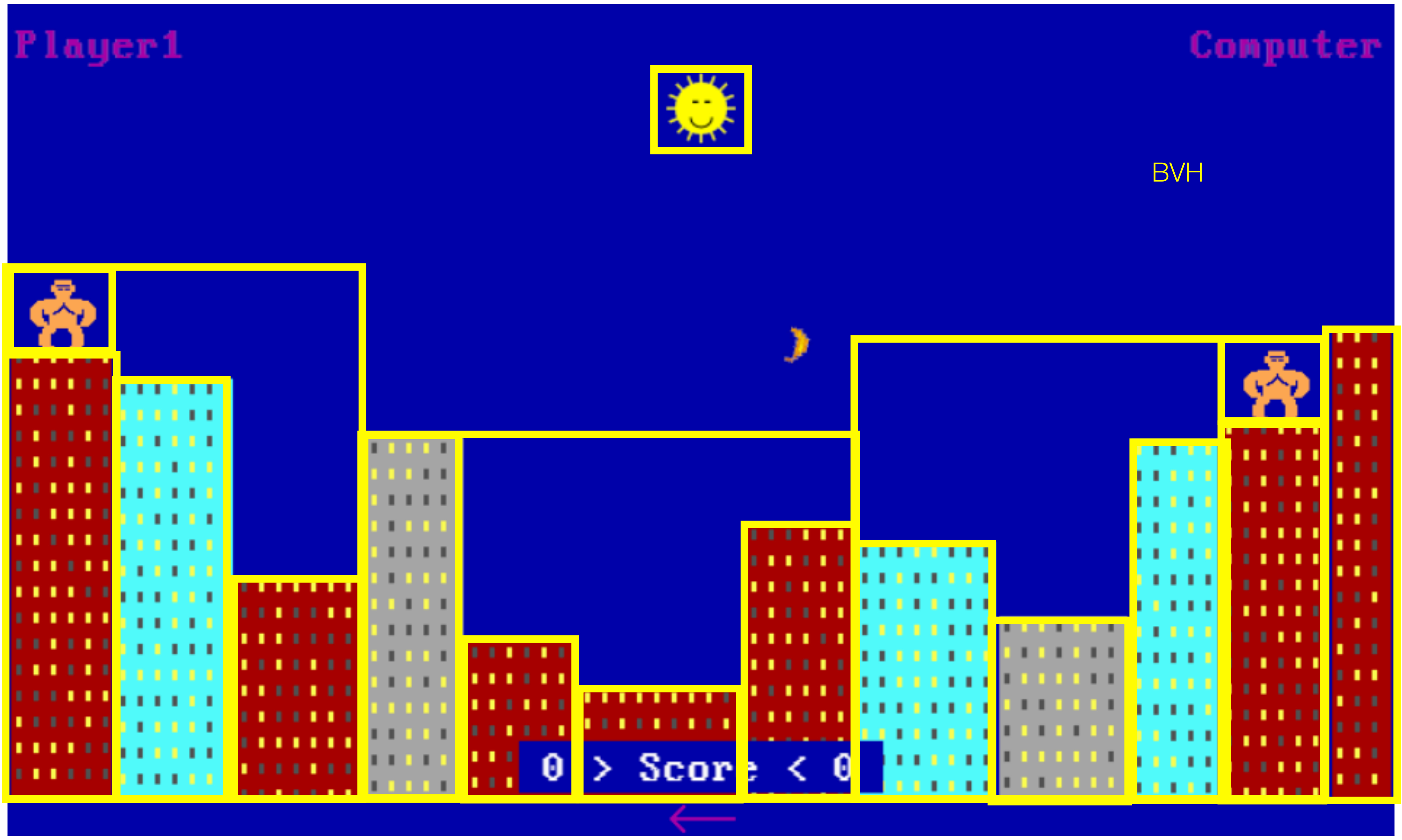
The old QBasic game “Gorillas”

Core parts of particle transport simulation resemble problems found in computer games:

1. Determination of **distance to next object** and **which object** (along current straight line trajectory)
2. Determination of **isotropic distance (safety)** to any other object
3. Determination of **medium** in which the particle currently is

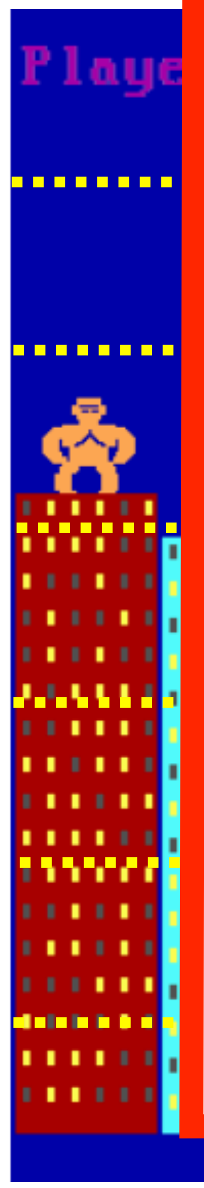
Navigator acceleration structures





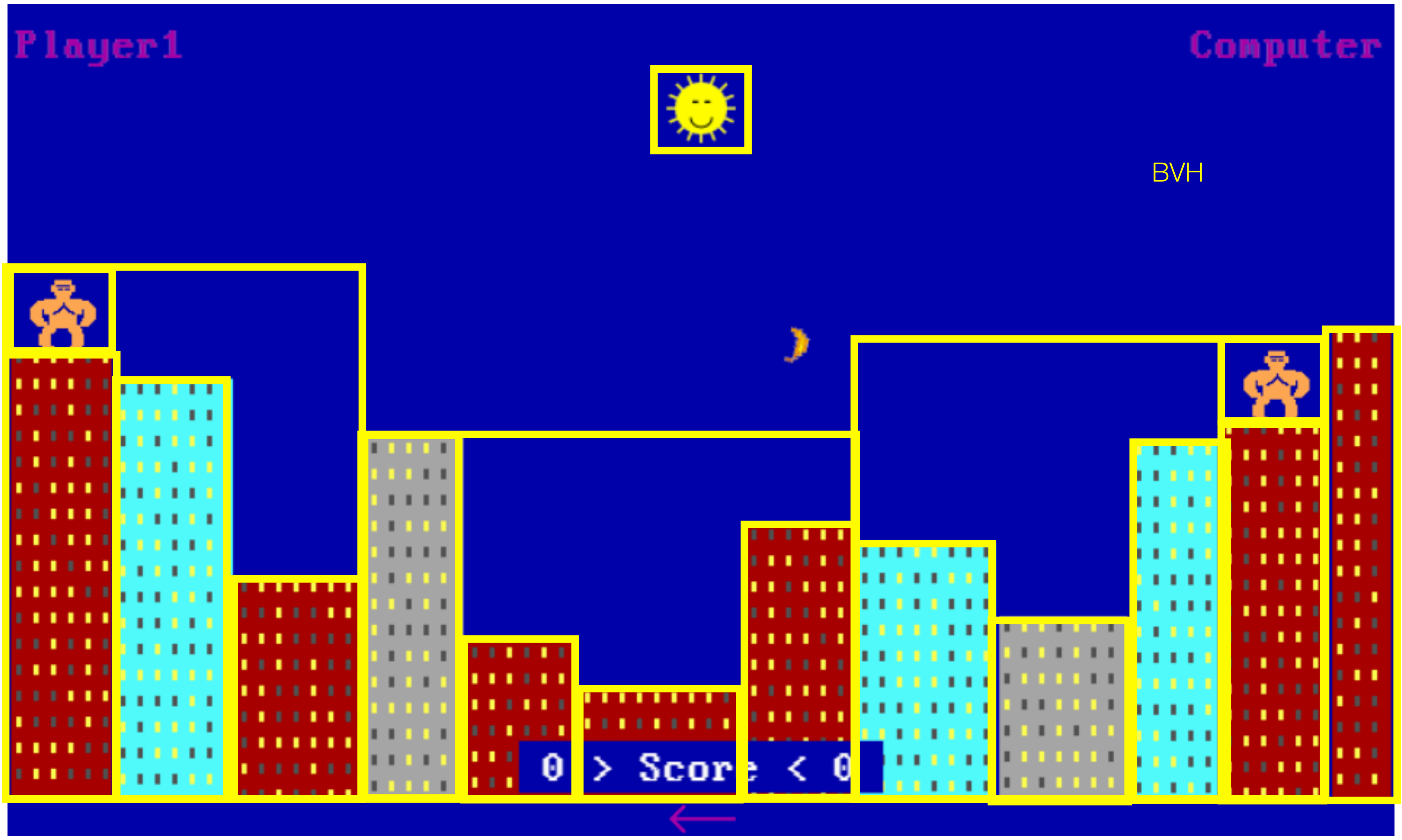
- **Acceleration structures** typically employed to **reduce complexity** in the main navigator tasks, e.g.:
  - (Sparse) cell/voxel structure (hash map) keeping a list of candidate object per cell
  - R-trees / octrees / you name it
  - Hierarchies of bounding volumes (BVHs)
- In Geant4, a fixed voxel hierarchy used in all main tasks
  - but hard coupled to navigator and rather stateful
- Revised approach in VecGeom
  - more **modular approach**
  - **wider set of implementations** which can be **mixed together** and **configured per logical volume**
    - e.g., **BVH with SIMD acceleration** for ComputeStep
    - e.g., **Any mix of voxel + BVH** for Safety and Locate
  - **can benefit from external packages** such as ray-traycing (e.g., Intel Embree)

Navigator acceleration structures



GOALS

- **Provide a demonstrator** integration into Geant4
- **Evaluate** against existing G4 navigation solutions and identify potential opportunities for large-scale experiment simulations
- **Enable VecGeom** functionality for users of **ROOT/TGeo in VMC**

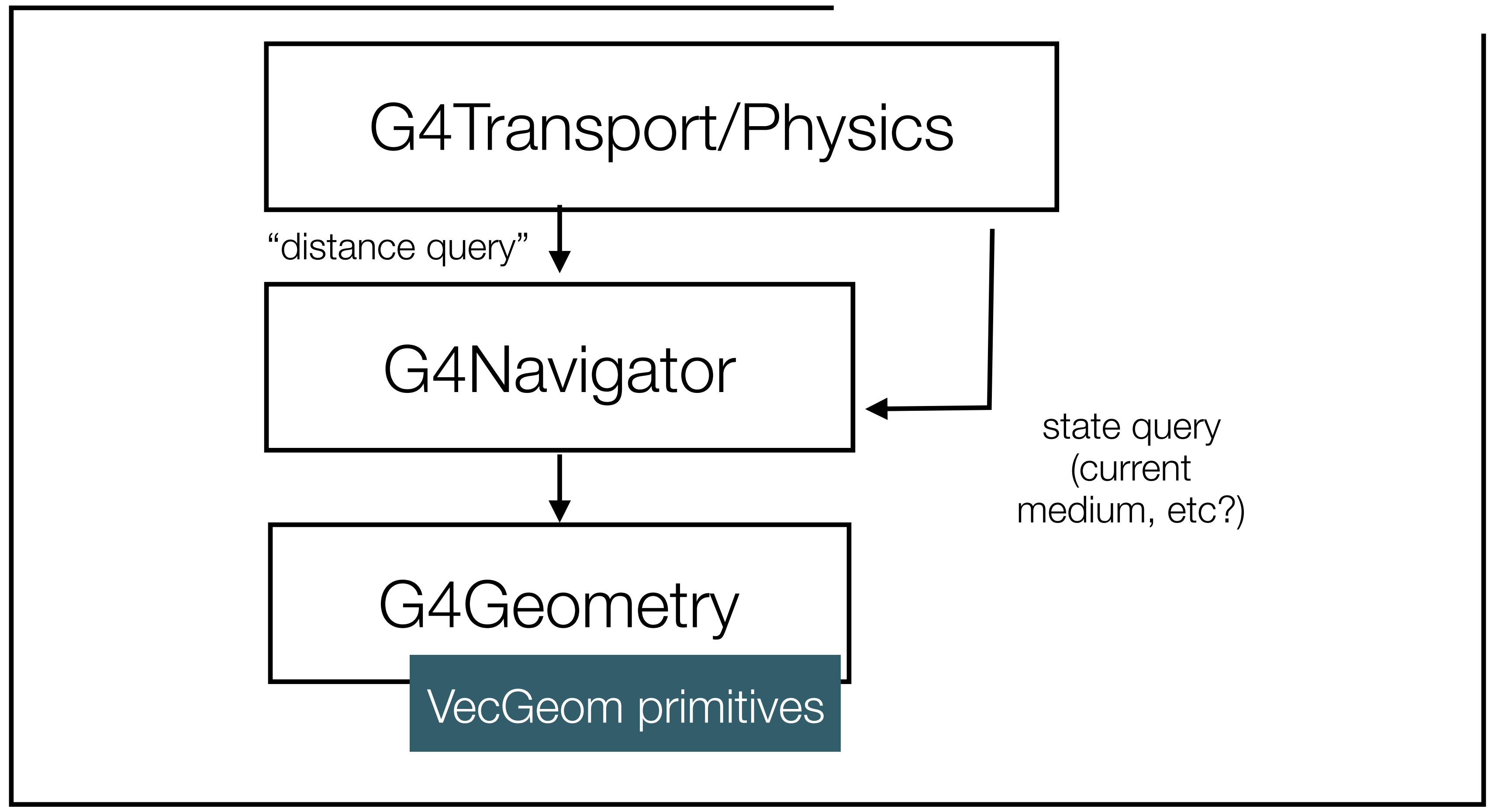


- Revised approach in VecGeom
  - more **modular approach**
  - **wider set of implementations** which can be **mixed together** and **configured per logical volume**
    - e.g., **BVH with SIMD acceleration** for ComputeStep
    - e.g., **Any mix of voxel + BVH** for Safety and Locate
  - **can benefit from external packages** such as ray-traycing (e.g., Intel Embree)

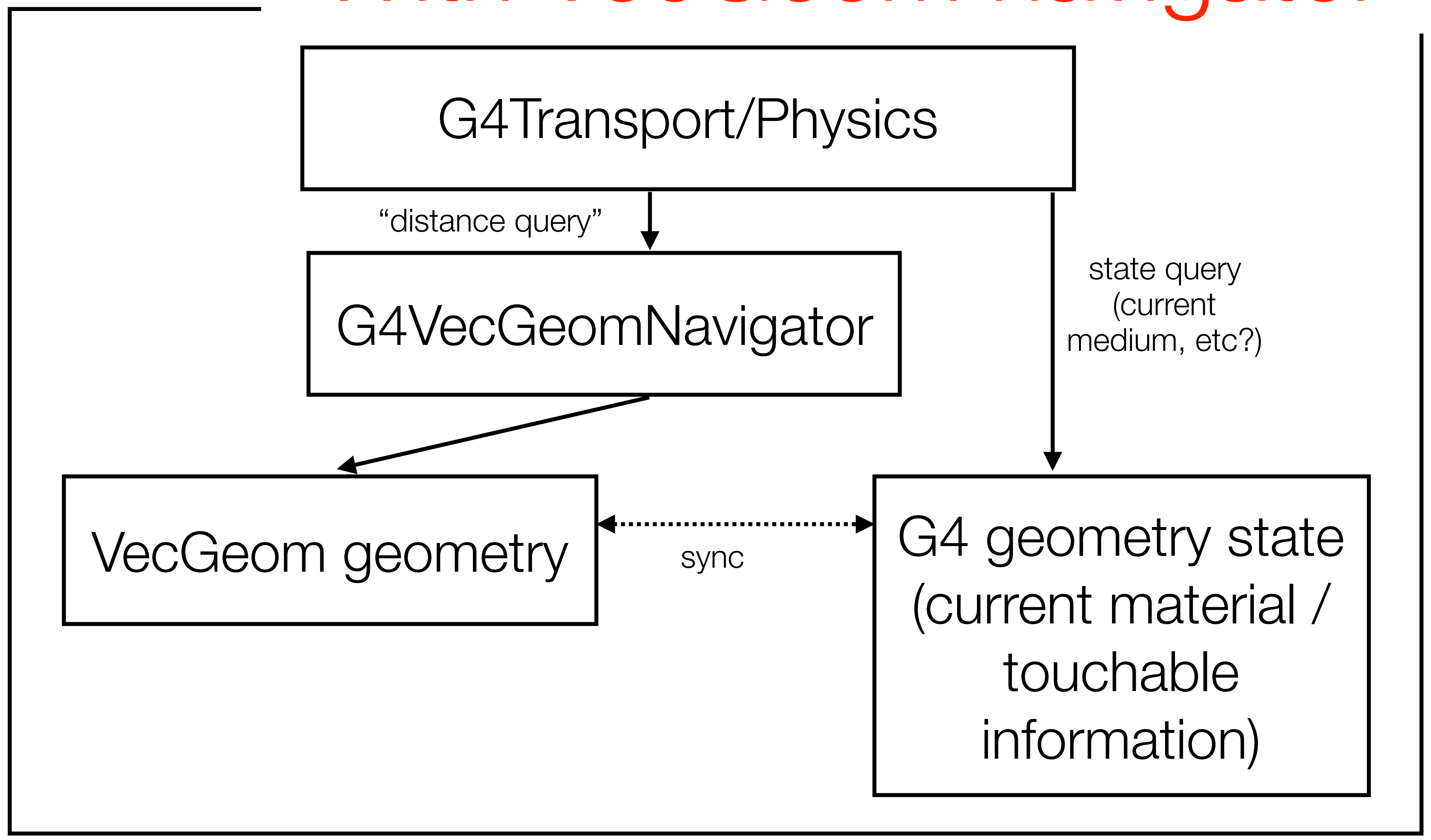
Integration approach

- Target **least user disturbance** and **least development effort**
- Provide VecGeom acceleration via a sub-class of G4Navigator, already designed to be extensible
  - follow pattern of TG4RootNavigator
- **Pay price of more memory** and **some synchronization overhead** since multiple representations of the geometry (G4 native, VecGeom) need to be kept in memory
  - memory read only and can be shared among all worker threads/processes
- Deeper integration requires substantially more work and might not be backward compatible
  - types of Geant4 and VecGeom differ

Native Mode



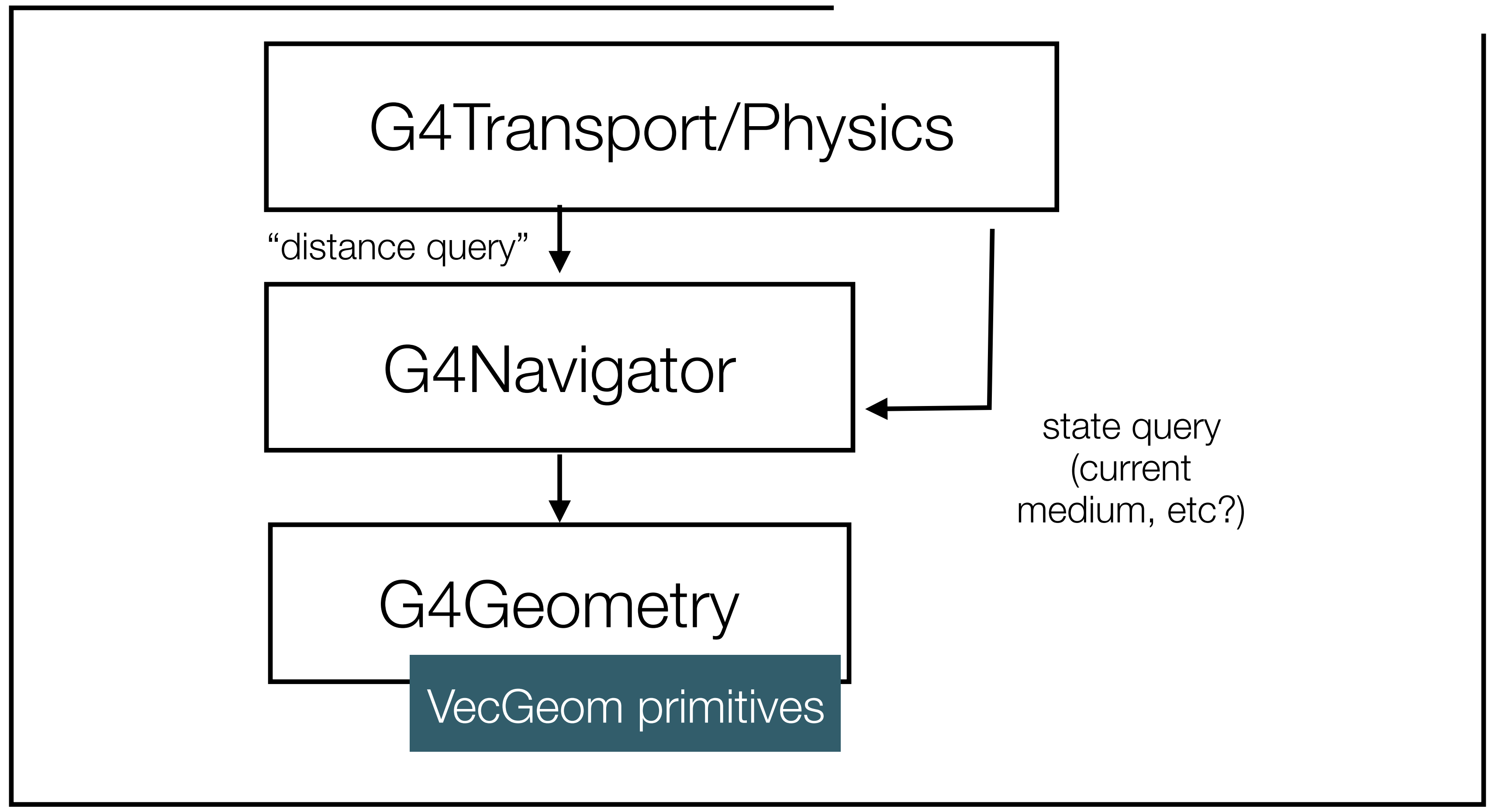
With VecGeom navigator



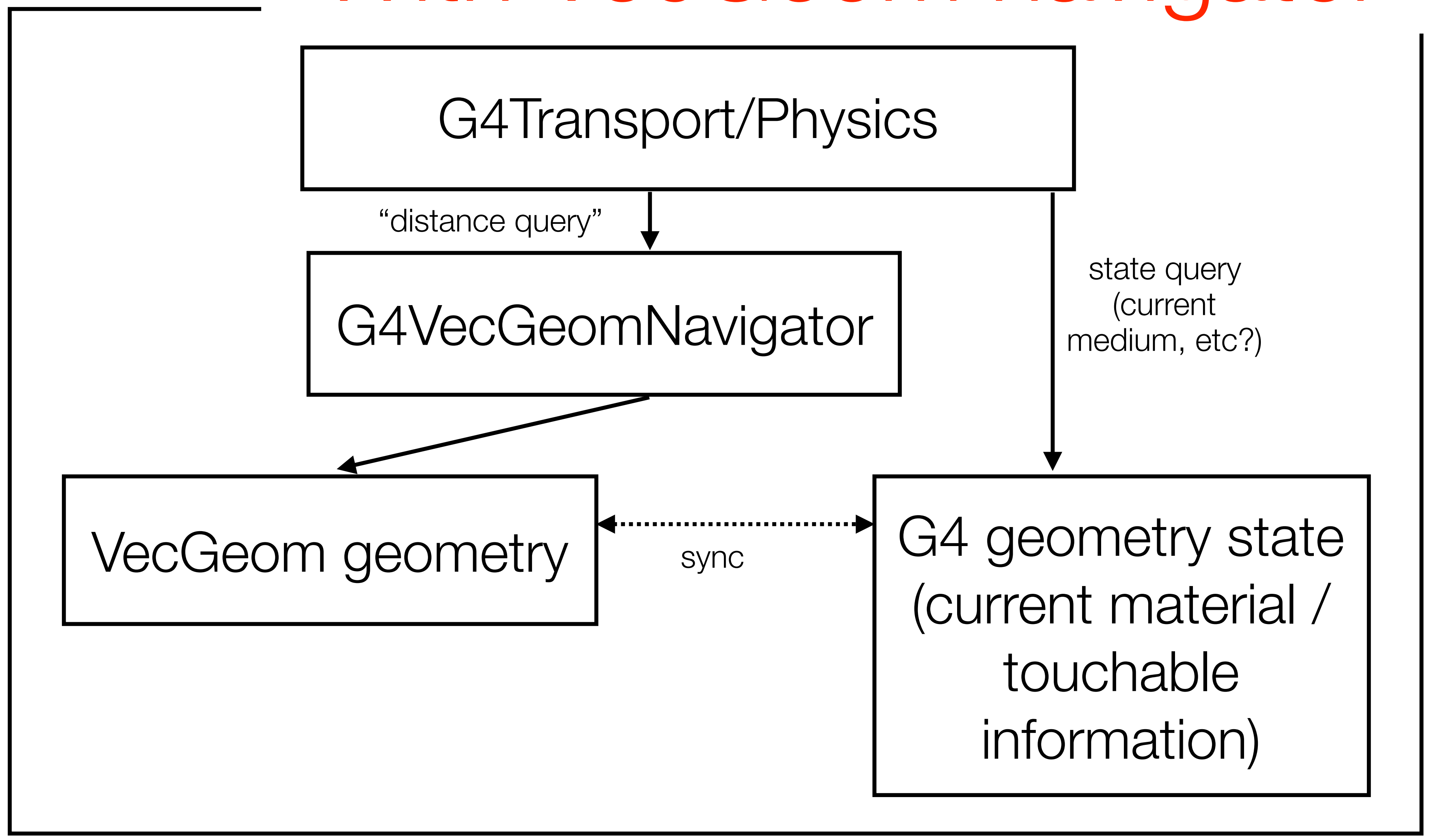
Integration approach

- Target **least user disturbance** and **least development effort**
- Provide VecGeom acceleration via a sub-class of G4Navigator, already designed to be extensible
  - follow pattern of TG4RootNavigator
- **Pay price of more memory** and **some synchronization overhead** since multiple representations of the geometry (G4 native, VecGeom) need to be kept in memory
  - memory read only and can be shared among all worker threads/processes
- Deeper integration requires substantially more work and might not be backward compatible
  - types of Geant4 and VecGeom differ

Native Mode



With VecGeom navigator



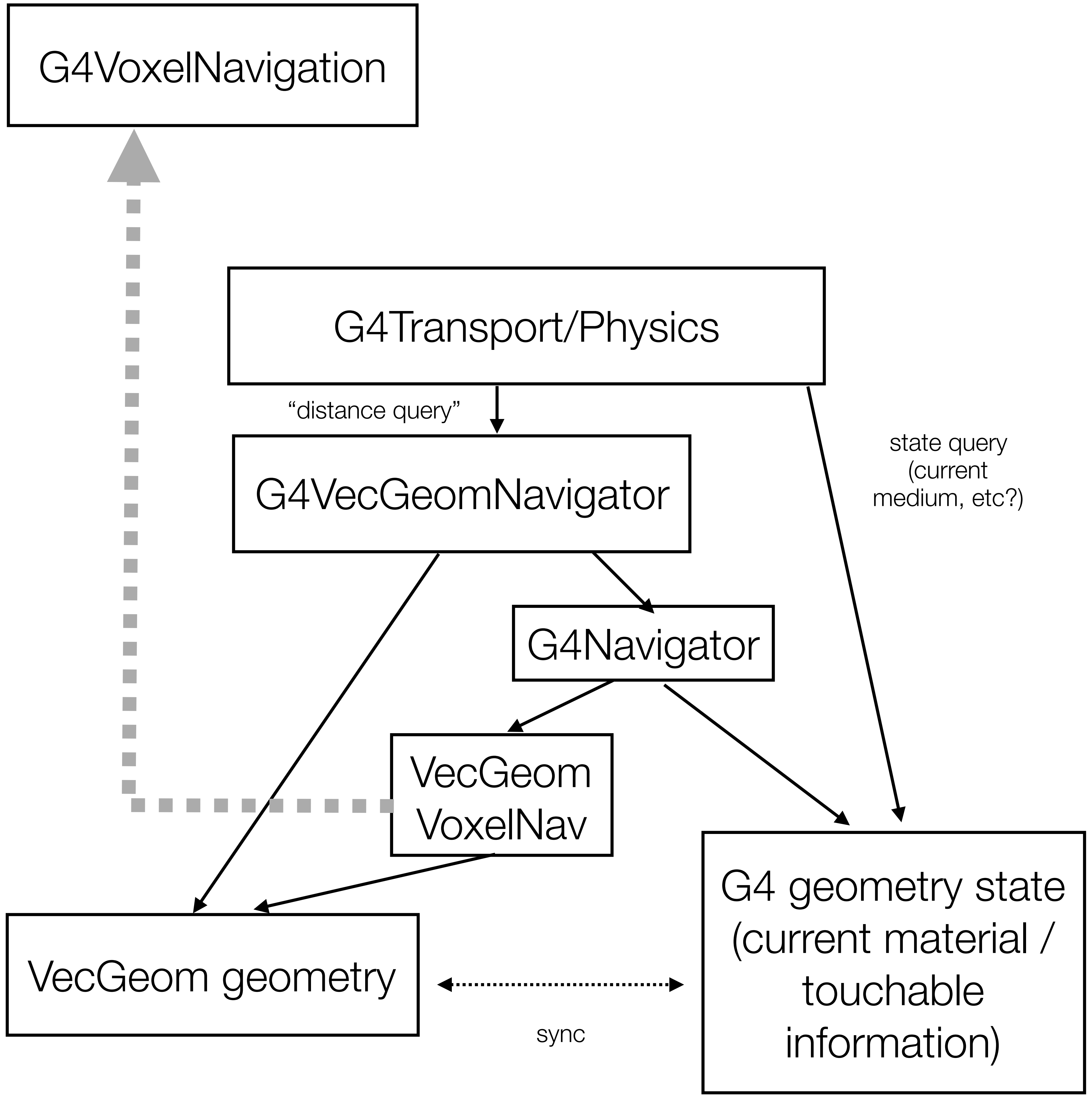

```
// make the G4VecGeomNavigator
auto nav = new G4VecGeomNavigator(g4geometry);

// hooking the navigator into G4
auto trMgr =
G4TransportationManager::GetTransportationManager();

trMgr->SetNavigatorForTracking(nav);
```

minimal change in user code

no change inside Geant4

Code development status

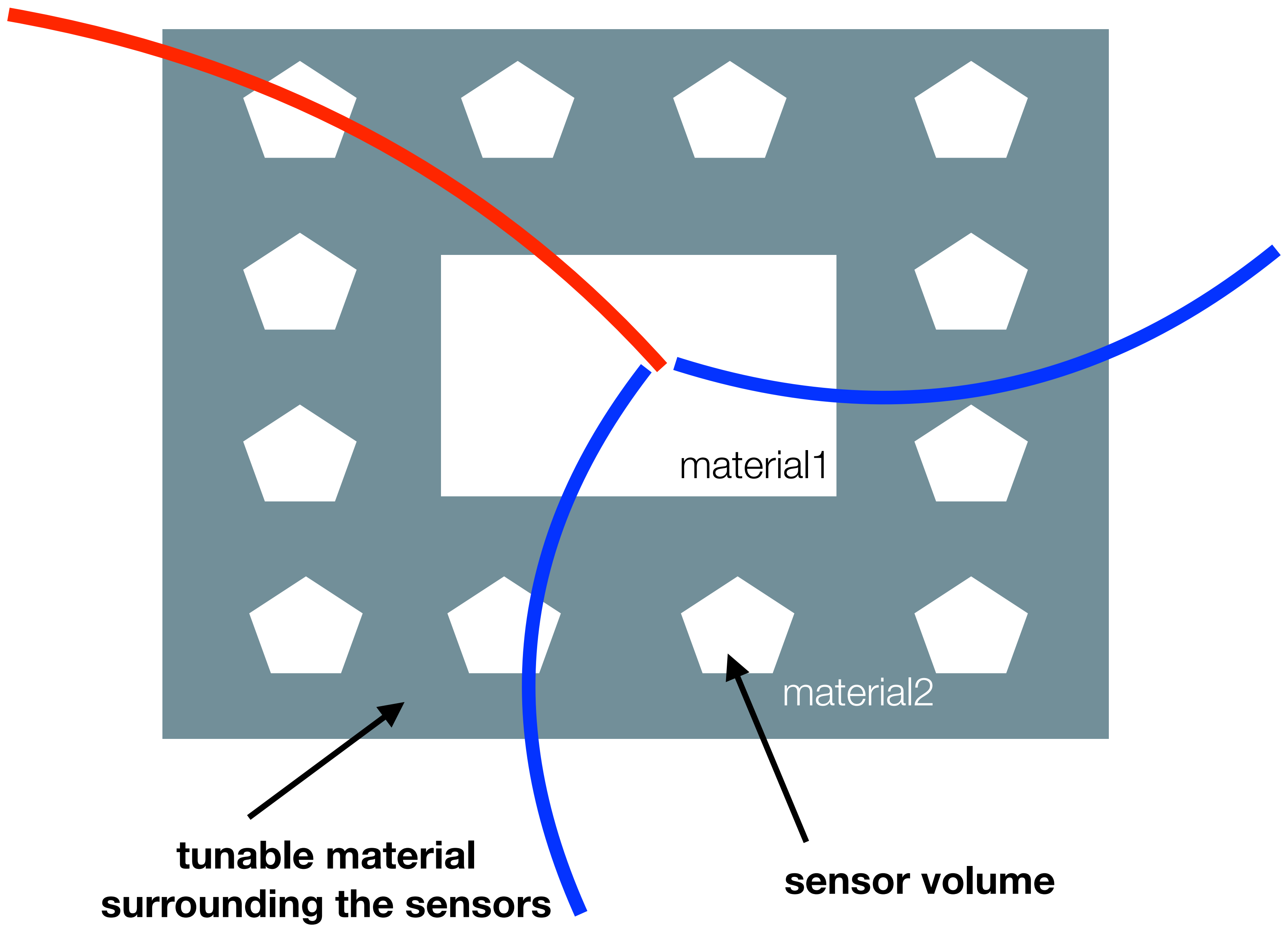
- **Multiple variants of plugin prototyped**
  - gitlab.cern.ch/VecGeom/G4VecGeomNav
- **Version 1**: **Full replacement** of all virtual functions of G4Navigator
  - complete geometry dispatch to VecGeom
- **Version 2**: **Partial replacement** - Use VecGeom only in complex logical volumes and reuse some of the G4Navigator state handling
  - partial geometry dispatch to VecGeom where it most makes sense
  - allows to use best of 2 worlds
  - can keep error and state handling of G4Navigator
  - do not need to provide all features (replicas, parameterised geometries) immediately in VecGeom navigator
  - a good “sweet spot”
- **Version 2 currently more stable** and results shown here use this version; development of version 1 continues



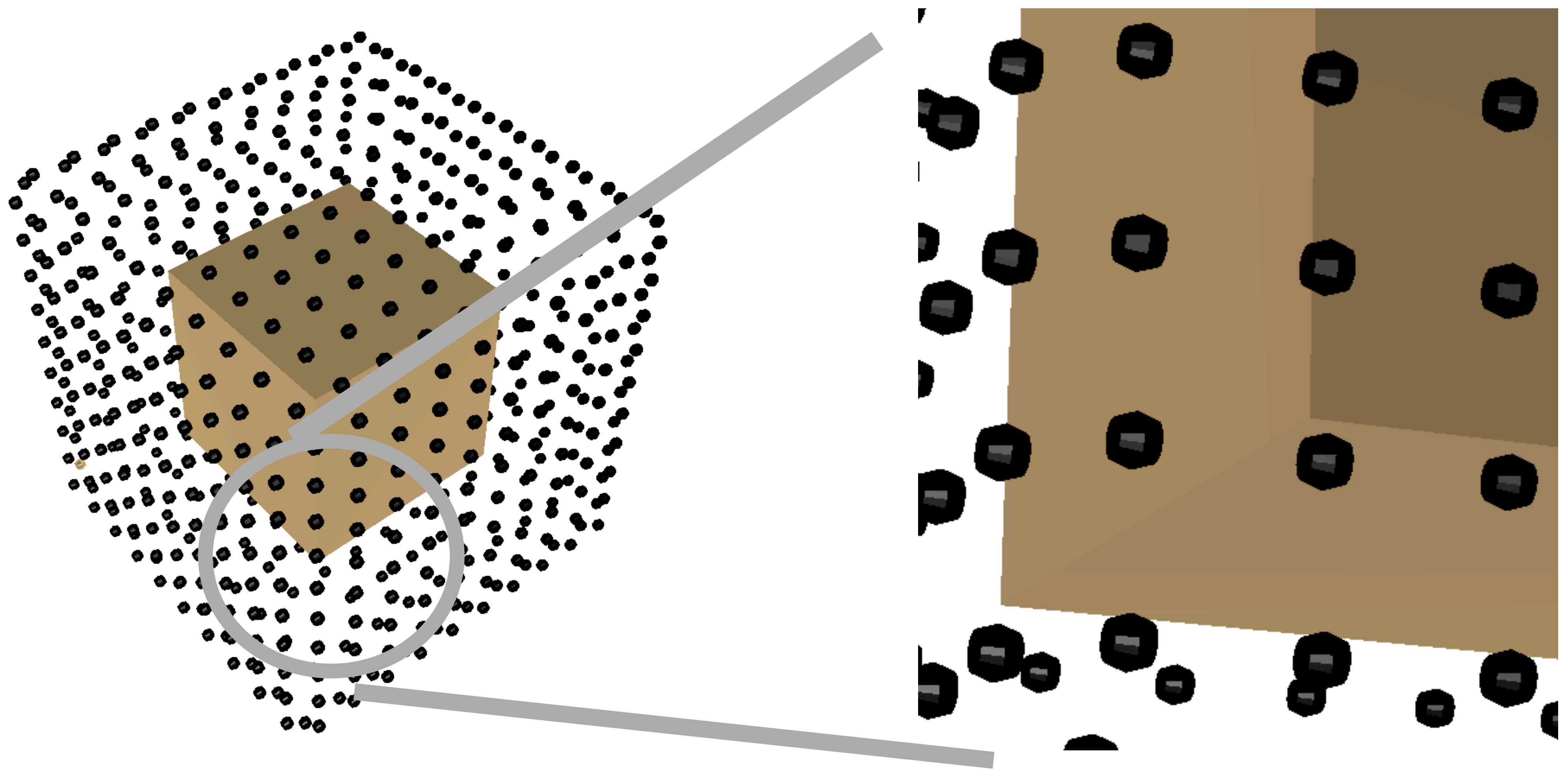
Version 2 illustration: We dispatch to VecGeom for certain G4Navigator functions (Safety) and partially for other functions (ComputeStep, Locate) via a sub-class of G4VoxelNavigation.

The test Geant4 application

- Test geometry
  - A simple geometry with few layered materials and a number of sensor volumes in material2
  - complexity tuneable via material settings, type of shape sensors and number of sensors
  - simple enough to debug and not to get trapped in complicated corner cases
  - good enough to validate and get first performance indications



3D view on example with 600 small polycone sensors :

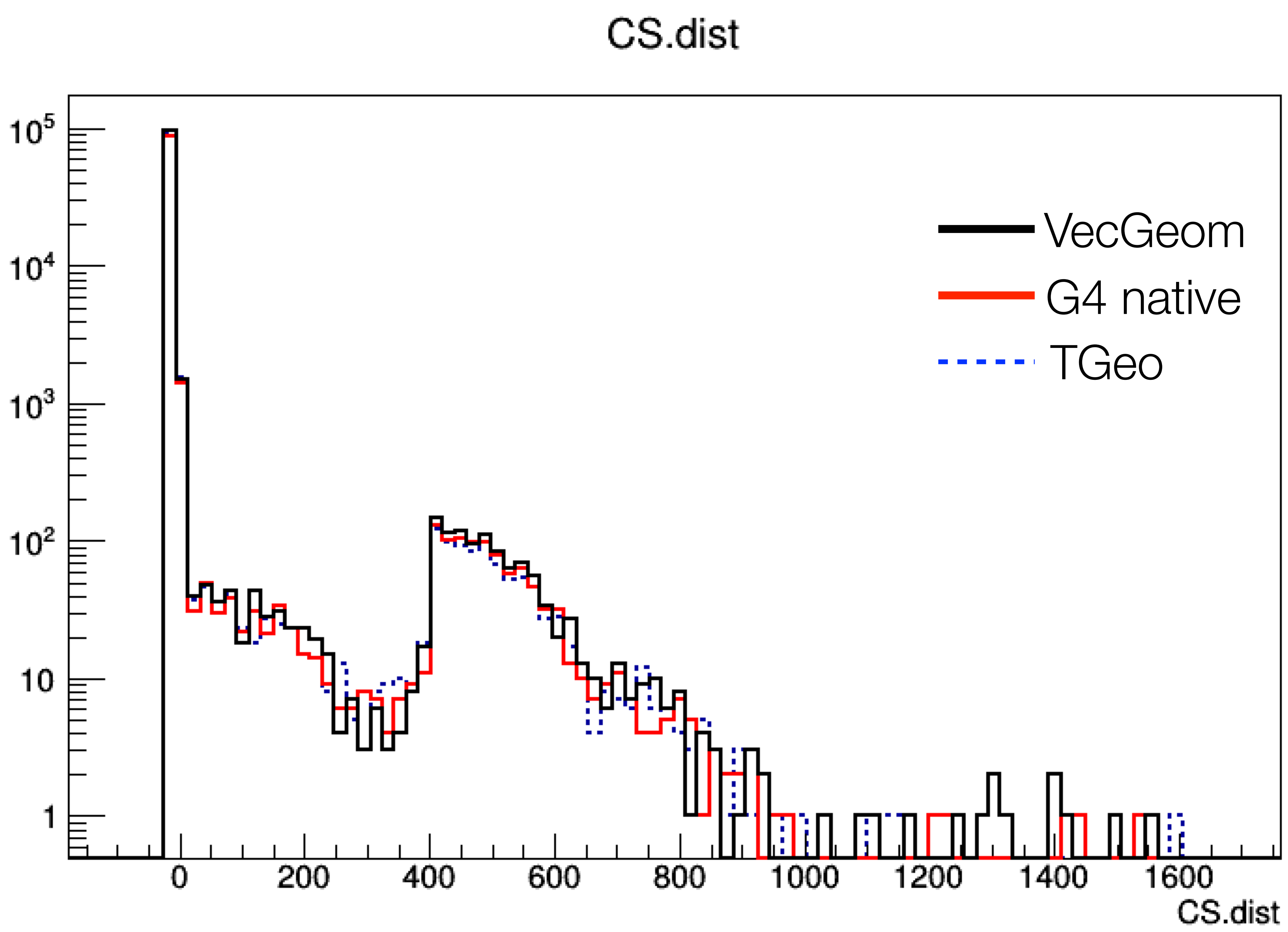


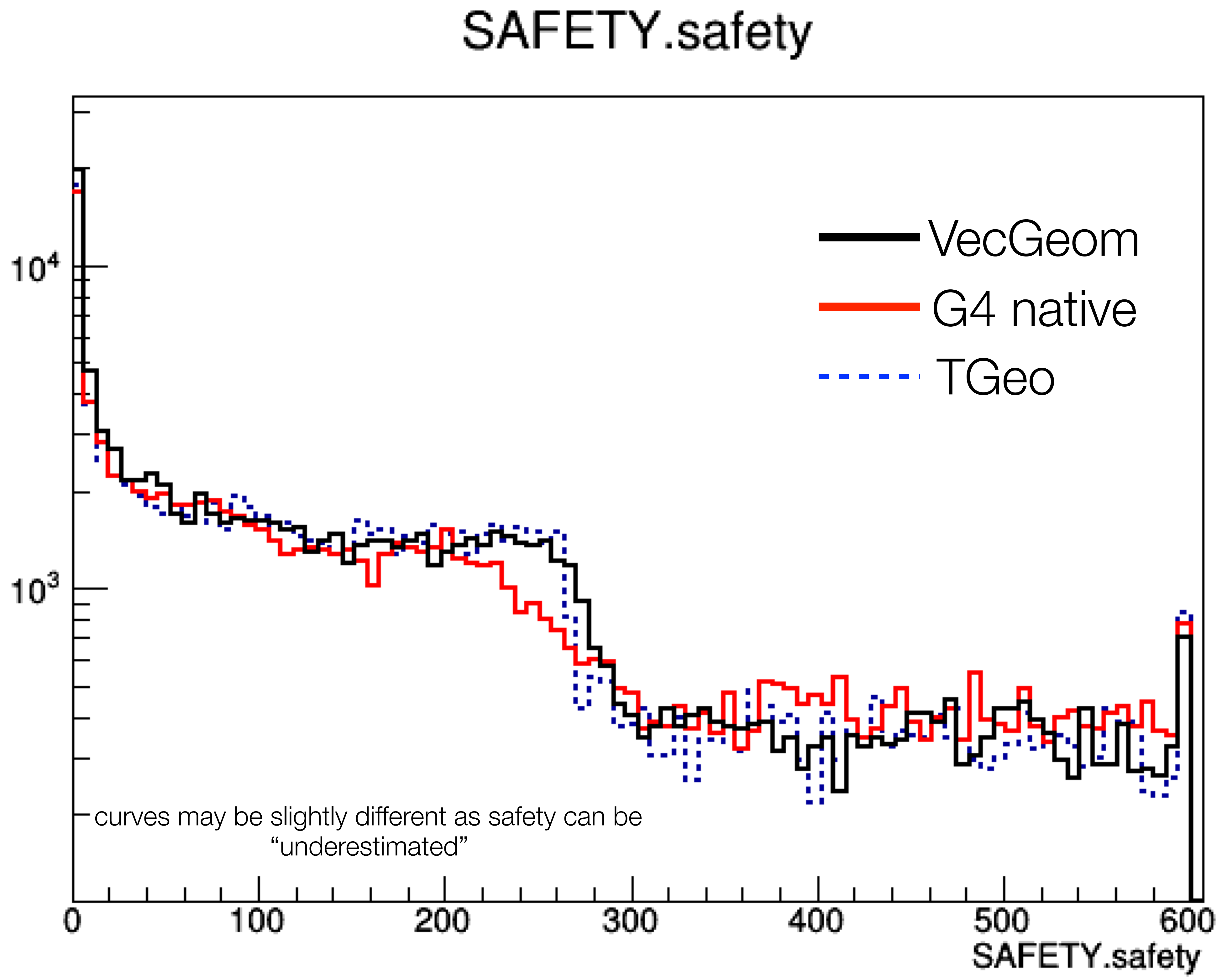
- Test Geant4 application
  - generic app taking any “gdml” geometry description for particle transport in a uniform magnetic field
    - based on “FullCMS” example in VMC repo
  - run with either [Geant4 | TGeo | VecGeom] navigation

Validation Example: Low-density limit

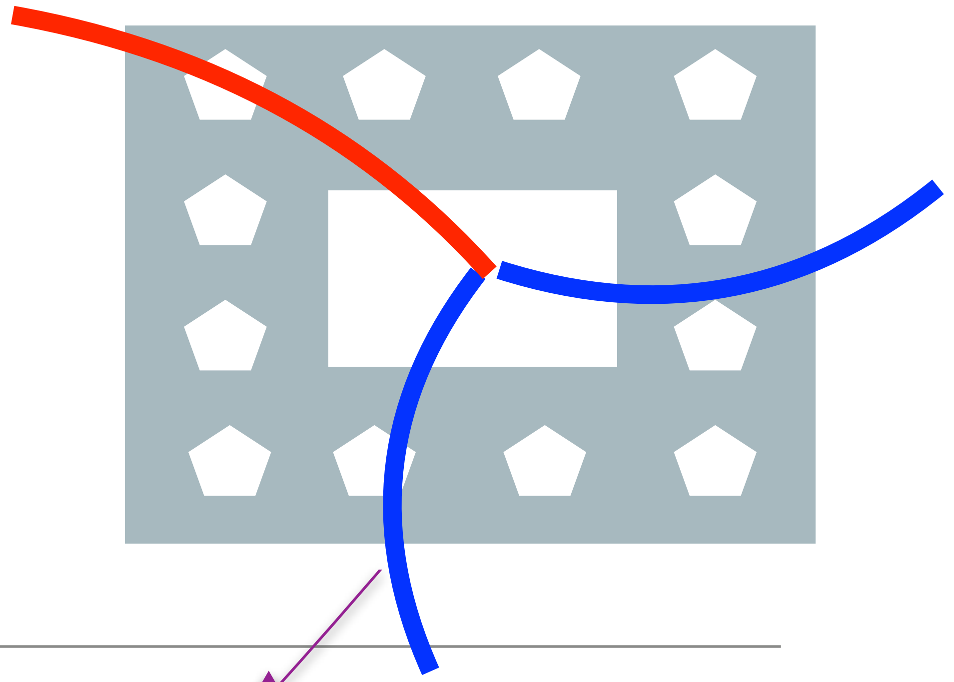
- Validate implementation by comparing new navigator to existing G4 native and TGeo-based runs: They cause a **similar number of calls to the geometry** with **consistent distributions for steps and safeties**

- Example for a geometry with 600 sensors modelled by a small “polycone” and identical initial events

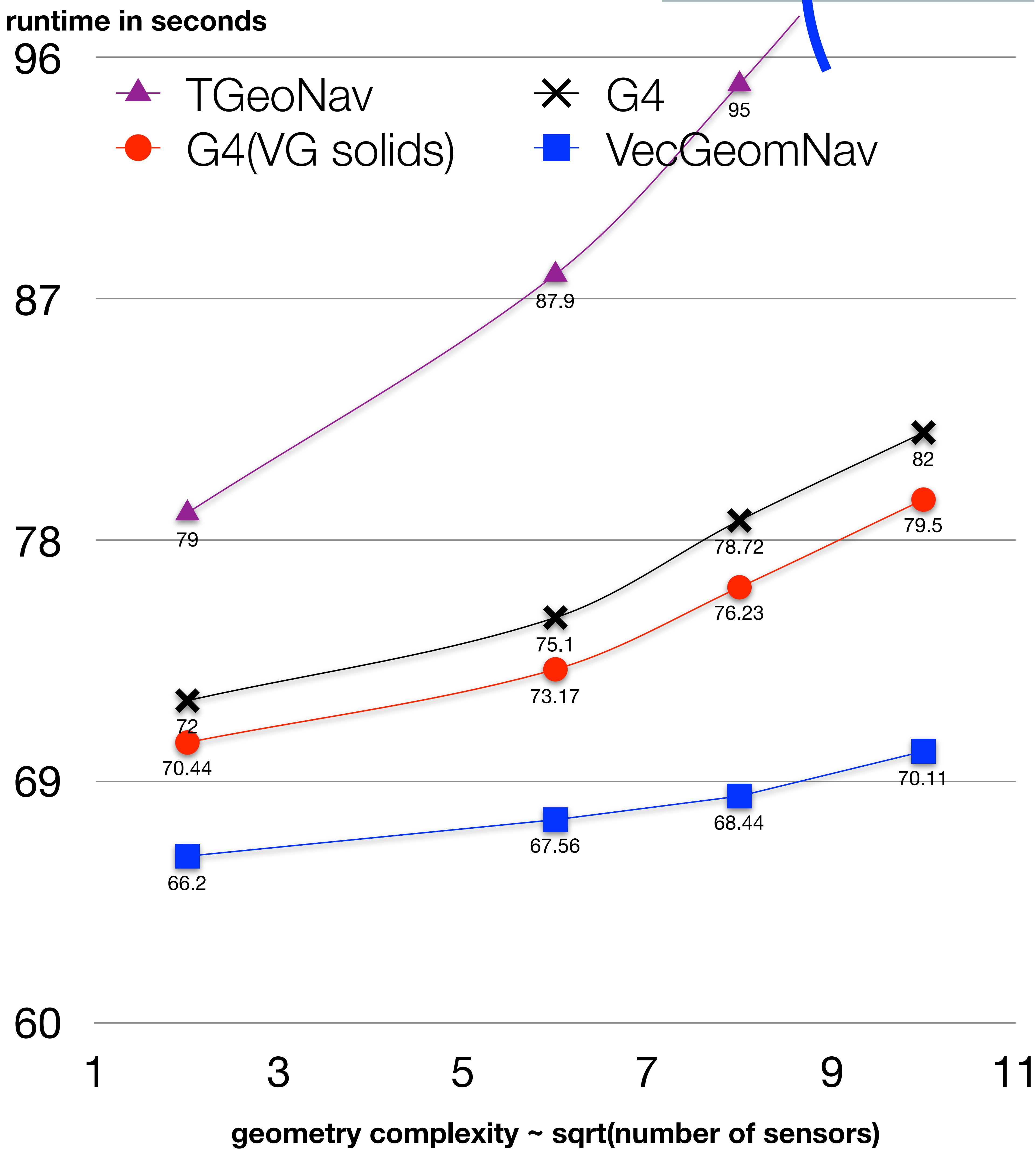




Low-density limit: View on performance

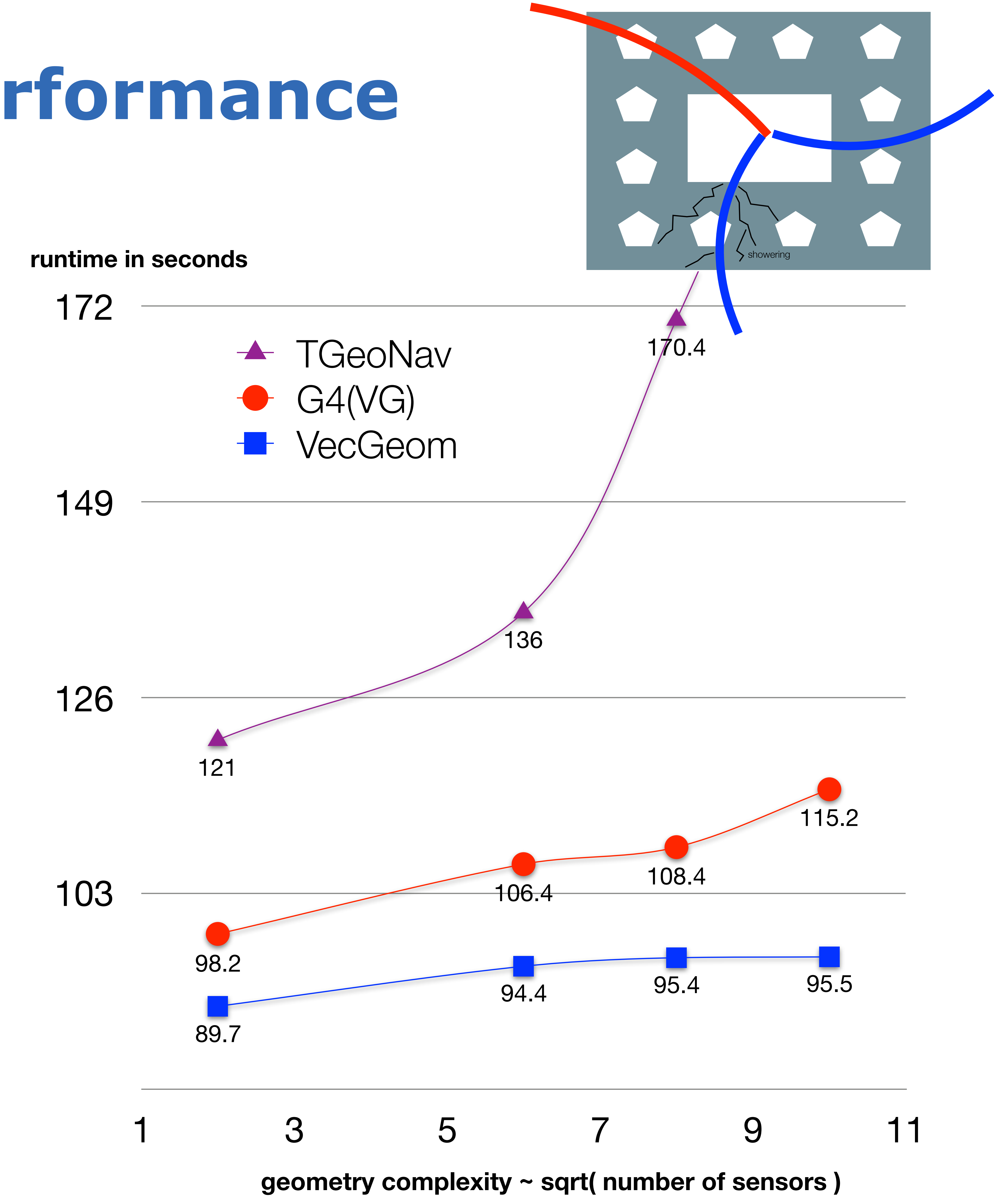


- Gas material (Kr): few physics; should be geometry-step dominated
- First performance benchmark transporting 100K primary electrons as a function of geometry complexity (number of sensors) yields encouraging numbers
- The VecGeom navigator plugin **accelerates overall simulation time**
  - here between 8 and 17% compared to plain G4
  - demonstrates additional benefit compared to G4 with only VecGeom solids integration
  - potentially slightly better scaling
- Points to potentially **substantial opportunity for TGeo based simulations** (ALICE)



Dense material limit: View on performance

- Dense material (Al): lots of physics processes and produced secondaries
- Benchmark transporting 40 primary electrons as a function of geometry complexity (number of sensors)
- Results overall similar to those in low-density material limit: **The VecGeom navigator plugin is seen to accelerate native G4 in the 10-15% range in this example**, beyond previous integration results



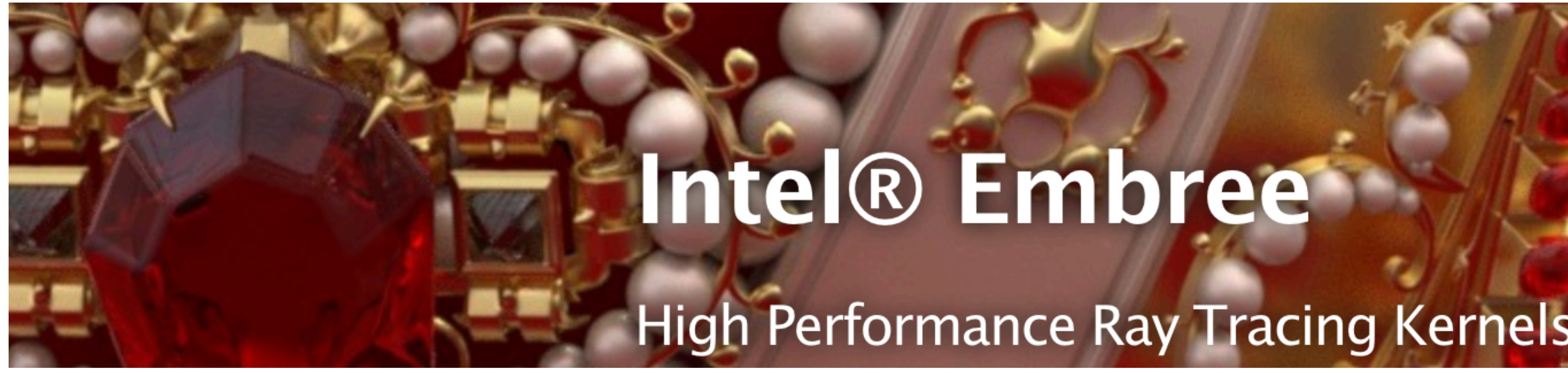
Summary

- First implementation of a Geant4 navigation plugin, using VecGeom capabilities
  - This goes substantially beyond the currently available level of integration on the solids level
  - Enables exploitation of the modular and extensible acceleration structures of VecGeom
- First evaluations indicate potential to accelerate HEP Geant4 simulations
- Results provide incentive to invest more time and effort on this topic
- Next steps:
  - Tests on full detector geometries (in particular targeting ALICE VMC)
  - Polish and make work available to any Geant4 user

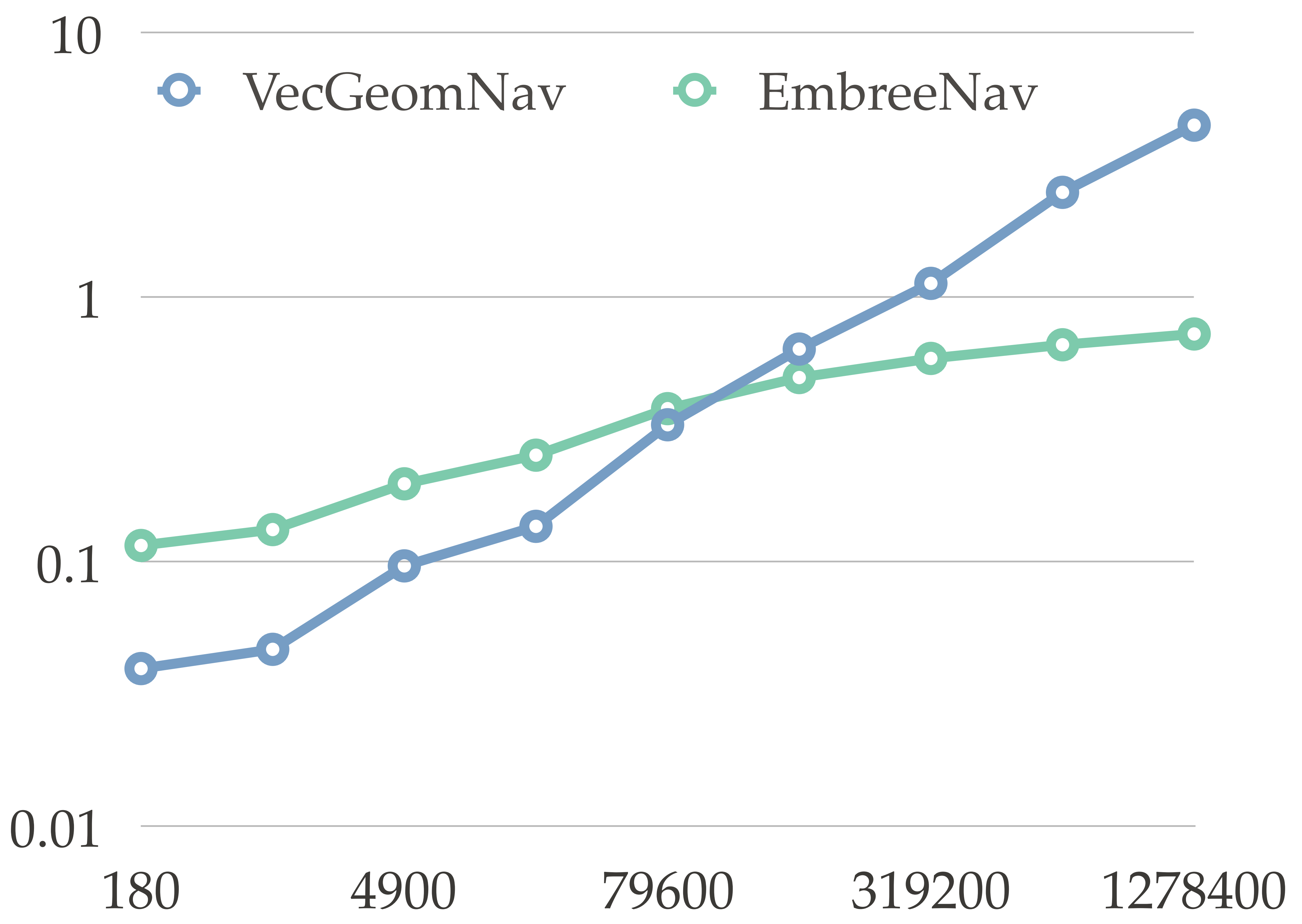
Backup material

VecGeom and External Libraries

- VecGeom is modular: New acceleration structures can be added easily or external packages interfaced
- Can benefit from external (industry developed) libraries
  - best scaling and low internal maintenance effort
- One example is Intel(R) Embree, providing SIMD accelerated ray-tracing kernels
  - useful for “distance” calculations
  - already optionally available in VecGeom
- We see that this is useful in the very complex limit (many objects, facets) but not necessarily in typical HEP geometries (much less complex compared to ray-tracing tasks).
- But certainly useful to extent VecGeom. VecGeom expects to dispatch to Embree for very large tessellated solids in the future.



https://www.embree.org/



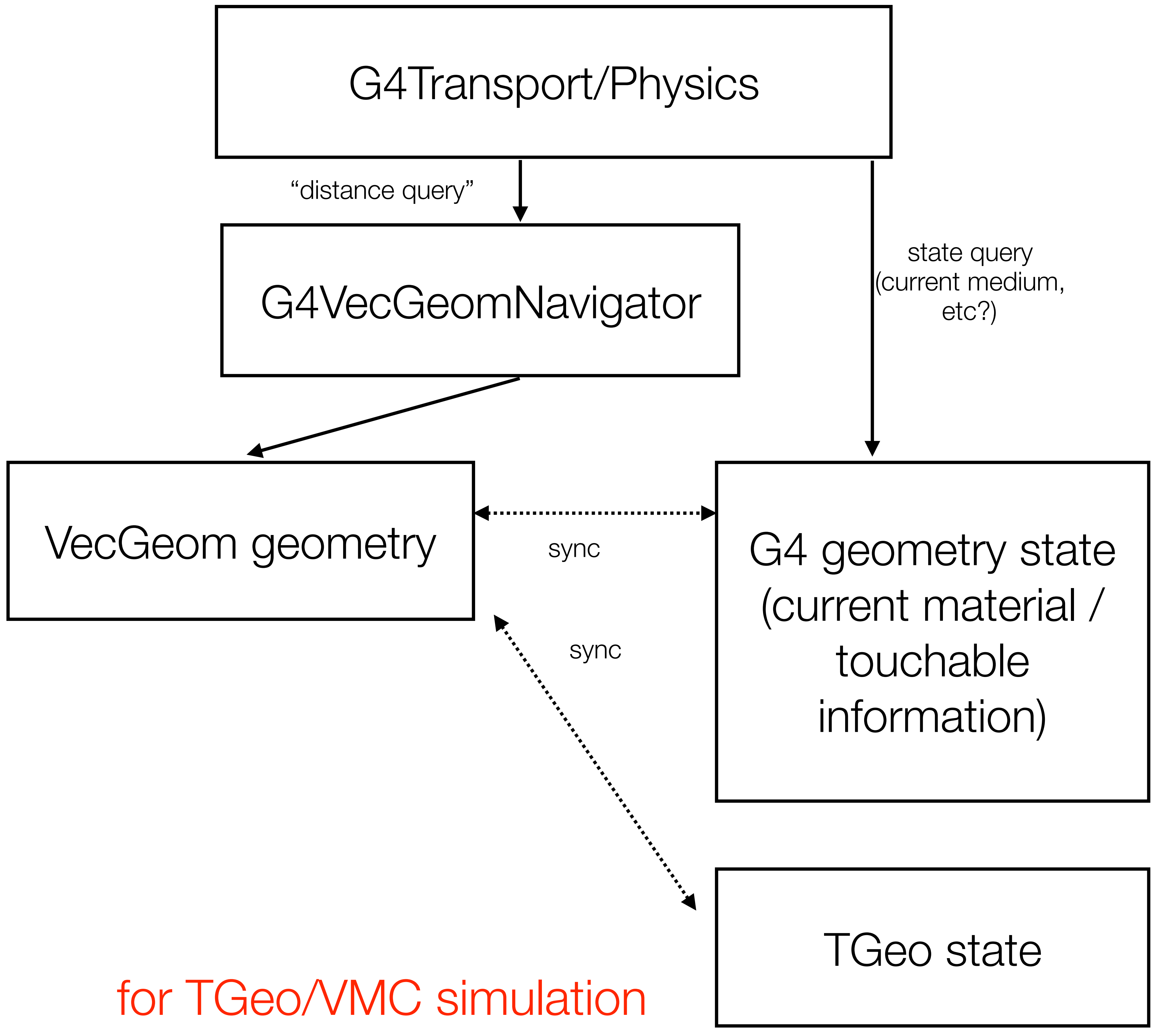
Runtime comparison of a tesselated solid using VecGeom own acceleration structure vs using Embree (as a function of the number of facets). Embree has a much better scaling in the very complex limit typical for ray-tracing applications.

Integration approach for VMC

- Integration into Virtual Monte Carlo Geant4 simulations using TGeo as modelling framework follows slightly more complicated pattern but straightforward

- Essentially, more geometries need be kept in memory and more syncing operations necessary

- Currently in development for the ALICE simulation



Generic backup information

- Benchmark architecture
  - Intel® Xeon® CPU E5-2697 v2 server
  - gcc 7.2 and normal SSE4.2 release builds; task pinning; timings are mean of 5 runs
  - Geant4 version 10.5.1; Geant4_VMC 3.6
  - VecGeom compiled with template specialized volume algorithms switched on
- Notes about VecGeom acceleration structures
  - there is some freedom / user choice for best acceleration structure which might require a pre-simulation tuning
  - some time needed to calculate VecGeom acceleration structure (currently not optimized); may benefit from GPU in future
  - acceleration structures can be distributed as “binary” condition objects and do only need to be recalculated if geometry changes

Validation approach

Correctness of the navigator plugin is studied and ensured by

- Simulation instrumentation
  - Monitor how API of G4Navigator is called from Geant4 and what is returned
  - Record positions and return values for steps, safety in a ROOT TTree
  - Compare between various implementation
- Callgrind analysis workflow
  - Run simulation in valgrind / callgrind
  - Automatic workflow to extract and compare cycles and call counts across navigator implementations