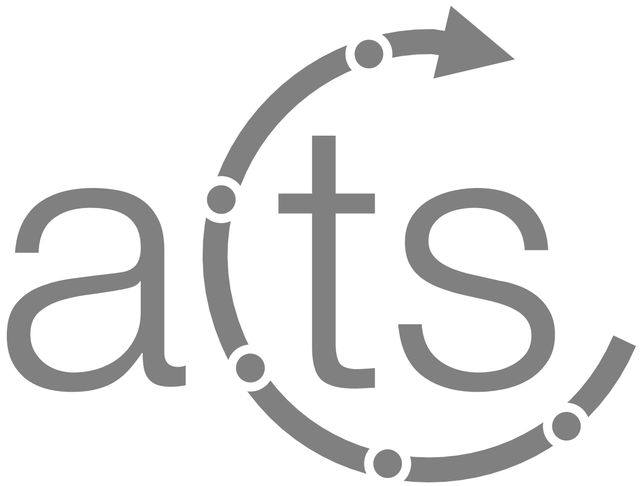


Geometry Developments

supported by

EP-R&D
Programme on Technologies for Future Experiments

iris hep

Bundesministerium für Bildung und Forschung

cooperations [1]

HSF
HEP Software Foundation

intel

CERN openlab

A. Salzburger (CERN) for the ACTS project

@SaltyBurger

CERN

Geometry - Surfaces

Surface class is the base component of all geometry objects

- Layers (may be dropped, see tomorrow) have approach surfaces
- Volumes are composed as set of boundary surfaces

This is a key element of the conceptual design:

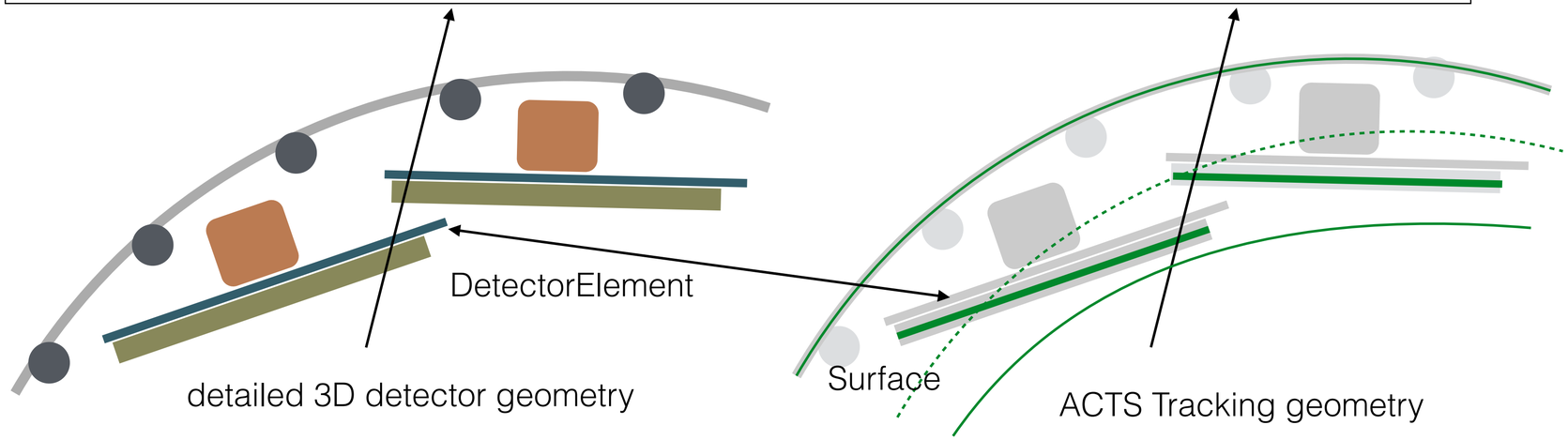
- makes all geometry objects compatible with the central Propagator module
- Bound surfaces act as measurement reference surfaces but also as navigation hooks and boundary surfaces (portals)

Geometry - DetElementBase

Binding the ACTS geometry to an existing Geometry model is done via declaration of a `Acts::DetectorElementBase` object

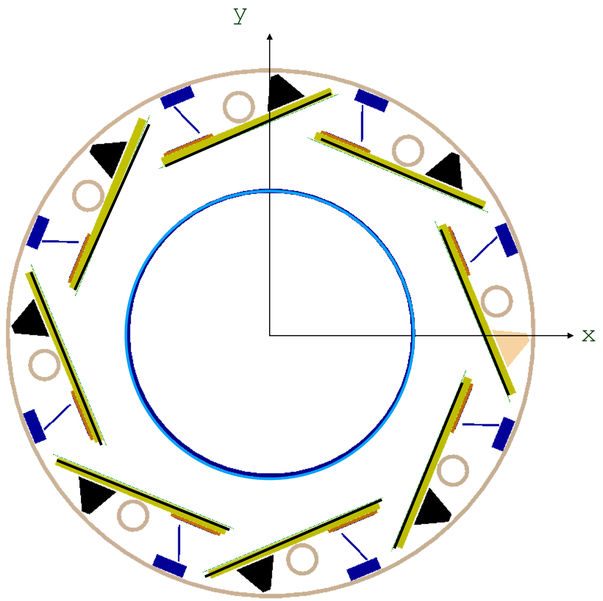
```
namespace Acts {
  /// doxygen documentation
  class DetectorElementBase {
   /// the according represented surface
    virtual const Surface& associatedSurface() const = 0;
  };
}
```

```
class MyDetectorElement {
  /// @copydoc DetectorElementBase::asscociatedSurface
   const PlaneSurface& associatedSurface() const;
 };
```



Geometry - Geometry building

(Current) geometry building follows a bottom up approach



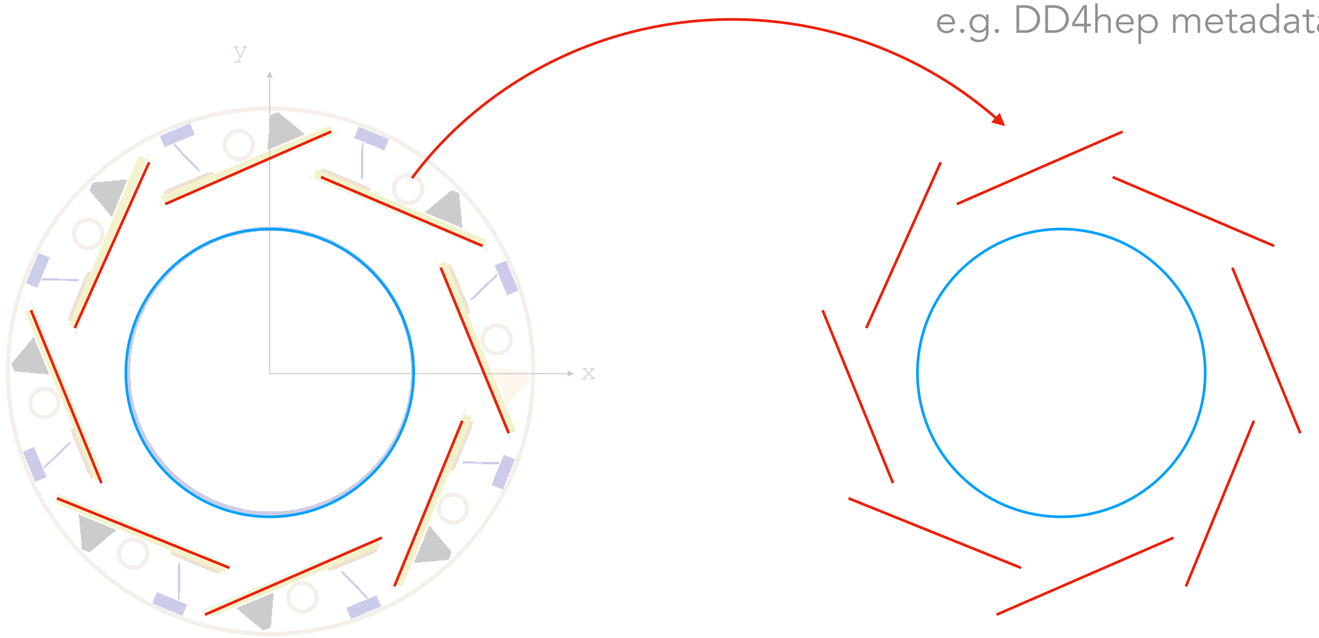
3D geometry model

(Plugin, e.g. GeoModel, DD4Hep, TGeo)

Geometry - Geometry building

(Current) geometry building follows a bottom up approach

collected through Plugin:
e.g. DD4hep metadata, TGeo naming, json files



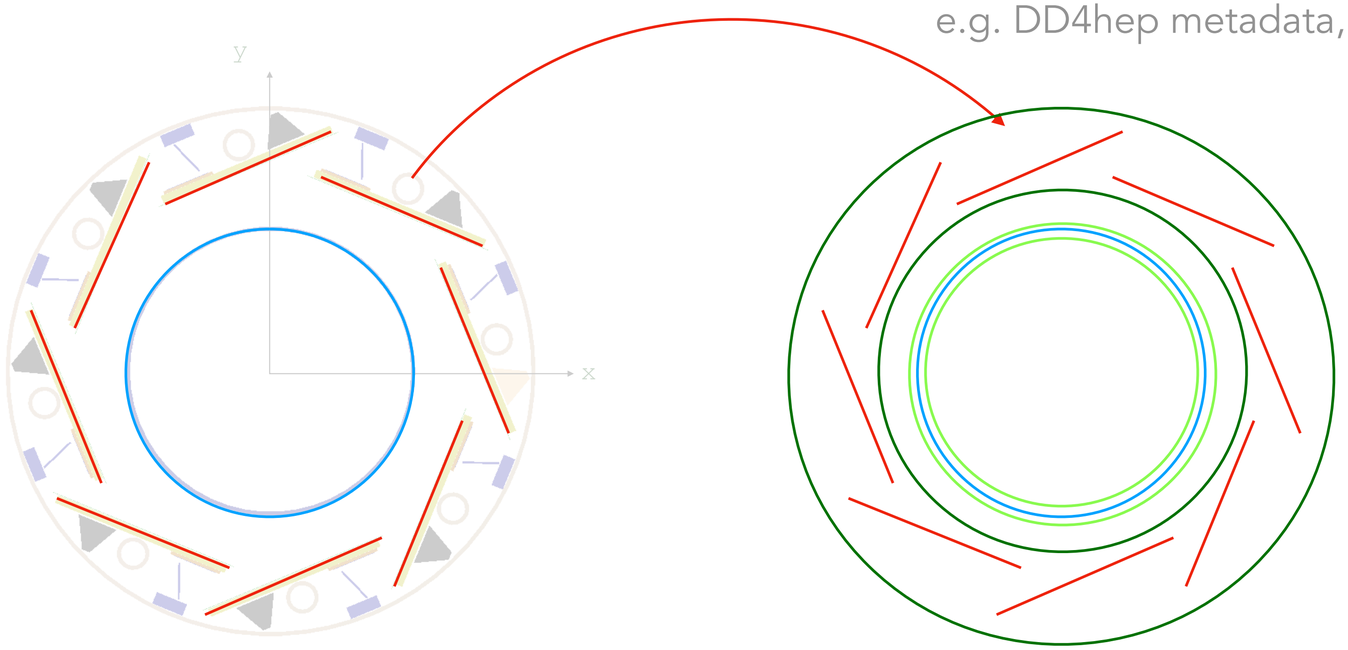
3D geometry model + `PluginDetectorElement` + `PassiveLayer`

(Plugin, e.g. GeoModel, DD4Hep, TGeo)

Geometry - Geometry building

(Current) geometry building follows a bottom up approach

collected through Plugin:
e.g. DD4hep metadata, TGeo naming, json files



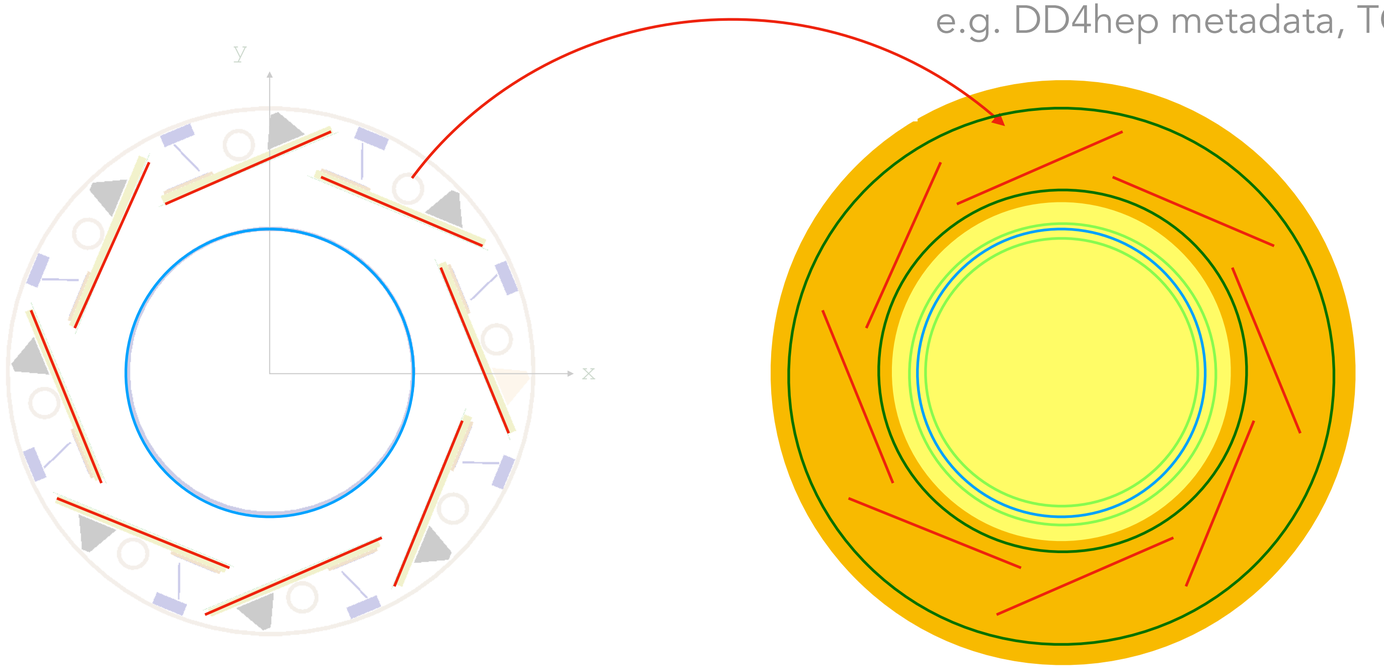
3D geometry model + `PluginDetectorElement` + `PassiveLayer` + `LayerBuilders`

(Plugin, e.g. GeoModel, DD4Hep, TGeo)

Geometry - Geometry building

(Current) geometry building follows a bottom up approach

collected through Plugin:
e.g. DD4hep metadata, TGeo naming, json files

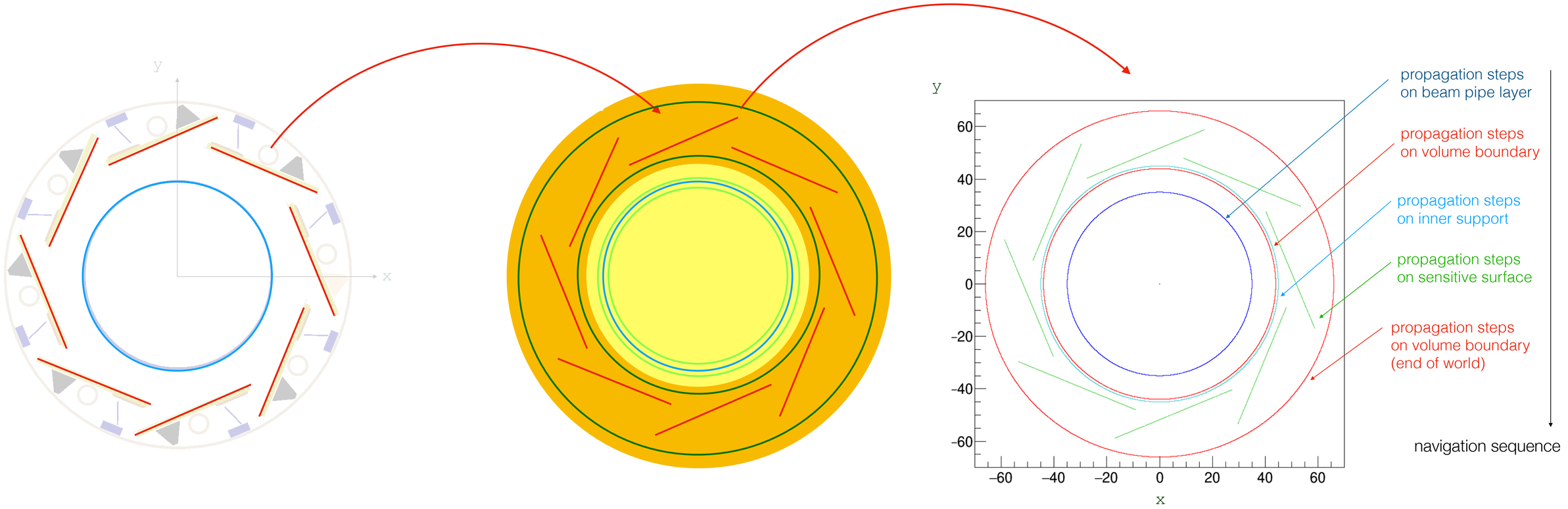


3D geometry model + `PluginDetectorElement` + `PassiveLayer` + `LayerBuilders` + `VolumeBuilders`

(Plugin, e.g. GeoModel, DD4Hep, TGeo)

Geometry - Geometry building

(Current) geometry building follows a bottom up approach



Geometry - Navigation consequence

Navigator in ACTS (only for layer setup) is relatively complex

- It has to resolve a three-body system + edge cases

Test implementation without layers

- **saves 3/4** of the code lines
- With same functionality:
  - Sensitive, passive surfaces
  - Material integration possible

`Acts::Experimental::Navigator`

LOC: 170

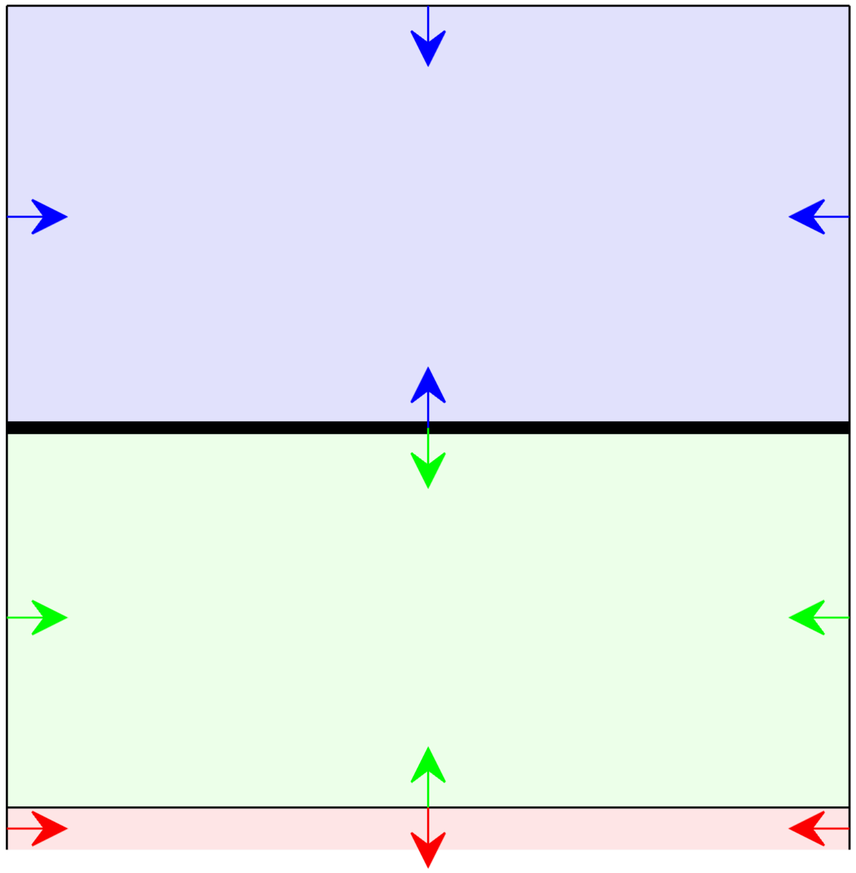
`Acts::Navigator`

LOC: 768

Geometry - Navigation proposal

Instead of one navigator having to deal with different setups

- delegate to volumes (literally using the `Delegate` class)



visualisation of prototype using actsvg

```
/// It has a Surface representation for navigation and propagat
/// and guides from one volume to the next.
///
/// The surface can carry material to allow mapping onto
/// portal positions if required.
///
class Portal : public std::enable_shared_from_this<Portal> {

 protected:
  /// Constructor from surface w/o portal links
  ///
  /// @param surface is the representing surface
  Portal(std::shared_ptr<Surface> surface);

 private:
  /// The surface representation of this portal
  std::shared_ptr<Surface> m_surface;

  /// The portal targets along/opposite the normal vector
  std::array<ManagedDetectorVolumeLink, 2> m_volumeLinks;

};
```

Geometry - VolumeLink + ManagedVolumeLink

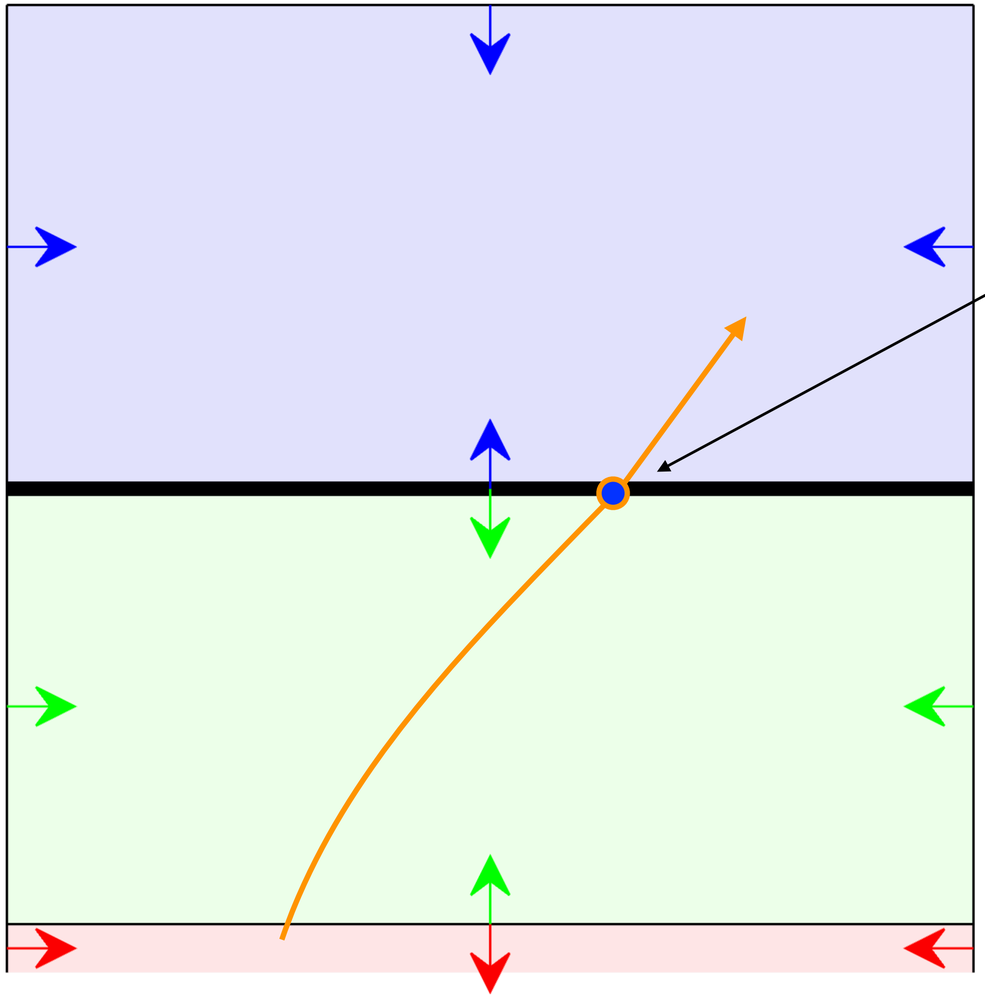
- Delegate to update a `Experimental::NavigationState` struct (that is used by the `Experimental::Navigator`)

```
/// Base class for all link implementations that need class structure
class INavigationDelegate {
 public:
  virtual ~INavigationDelegate() {}
};

/// Memory managed delegate to guarantee the lifetime
/// of eventual unterlying delegate memory and the
/// delegate function
///
template <typename deletage_type>
struct ManagedDelegate {
 public:
  deletage_type delegate;
  std::shared_ptr<INavigationDelegate> implementation = nullptr;
};
```

```
/// Memory  managed navigation state updator
using ManagedNavigationStateUpdator =
    std::tuple<NavigationStateUpdator, NavigationStateUpdatorStore>;

/// Declare a Detctor Volume Switching delegate
///
/// @param gctx is the current geometry context
/// @param position is the position at the query
/// @param direction is the direction at the query
///
/// @return the new DetectorVolume into which one changes at this switch
using DetectorVolumeLink = Delegate<const DetectorVolume*(
    const GeometryContext& gctx, const Vector3& position,
    const Vector3& direction)>;

/// Memory managed detector volume link
using ManagedDetectorVolumeLink = ManagedDelegate<DetectorVolumeLink>;
```

Geometry - Navigation proposal

Portal leads to new volume, volume gives access to local navigation

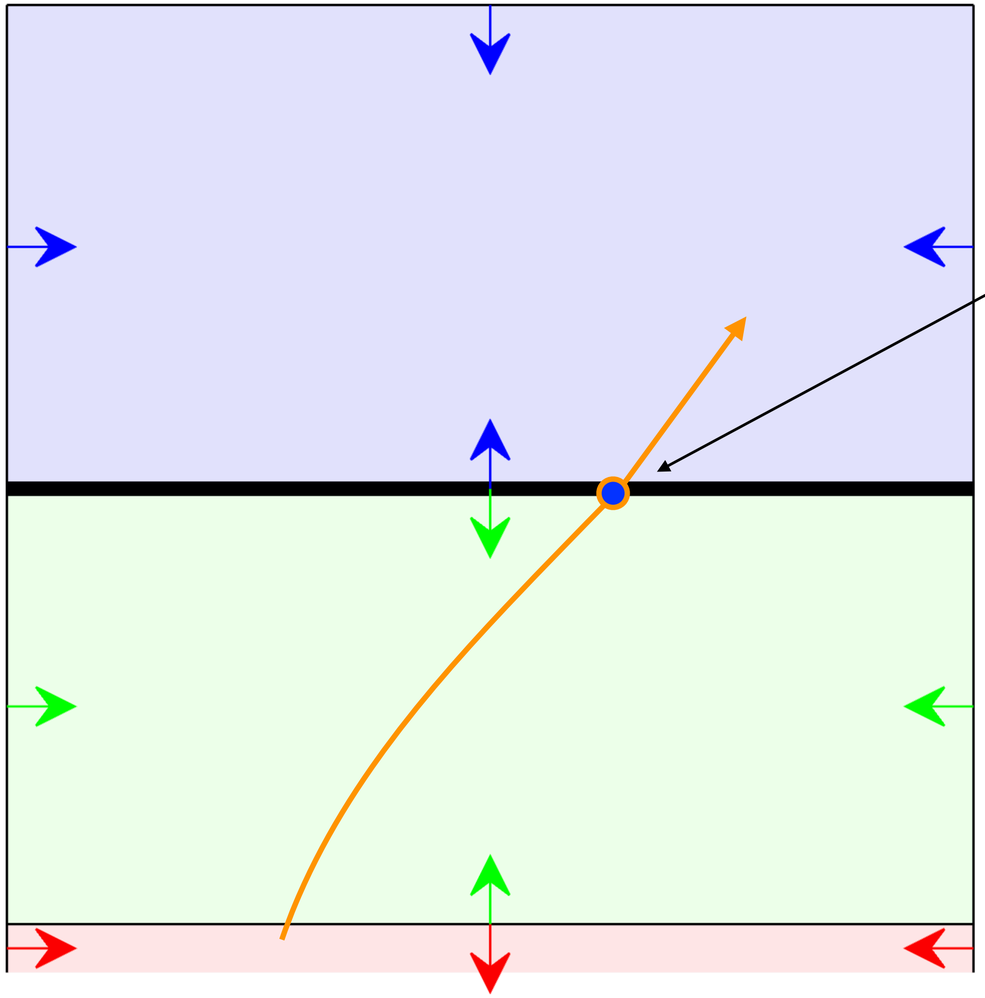


This can be implemented using straight line, frustum, blind search, whatever …

```
/// Declare a navigation state updator
///
/// This delegate dispatches the local navigation action
/// to a dedicated struct or function that is optimised for
/// the given environment.
///
/// @param nState is the navigation state to be updated
/// @param volume is the volume for which this should be called
/// @param gctx is the current geometry context
/// @param position is the position at the query
/// @param direction is the direction at the query
/// @param absMomentum is the absolute momentum at query
/// @param charge is the charge to be used for the intersection
///
using NavigationStateUpdator = Delegate<void(
    NavigationState& nState, const DetectorVolume& volume,
    const GeometryContext& gctx, const Vector3& position,
    const Vector3& direction, ActsScalar absMomentum, ActsScalar charge)>;

/// Memory  managed navigation state updator
using ManagedNavigationStateUpdator = ManagedDelegate<NavigationStateUpdator>;
```

Geometry - Navigation proposal

Portal leads to new volume, volume gives access to local navigation

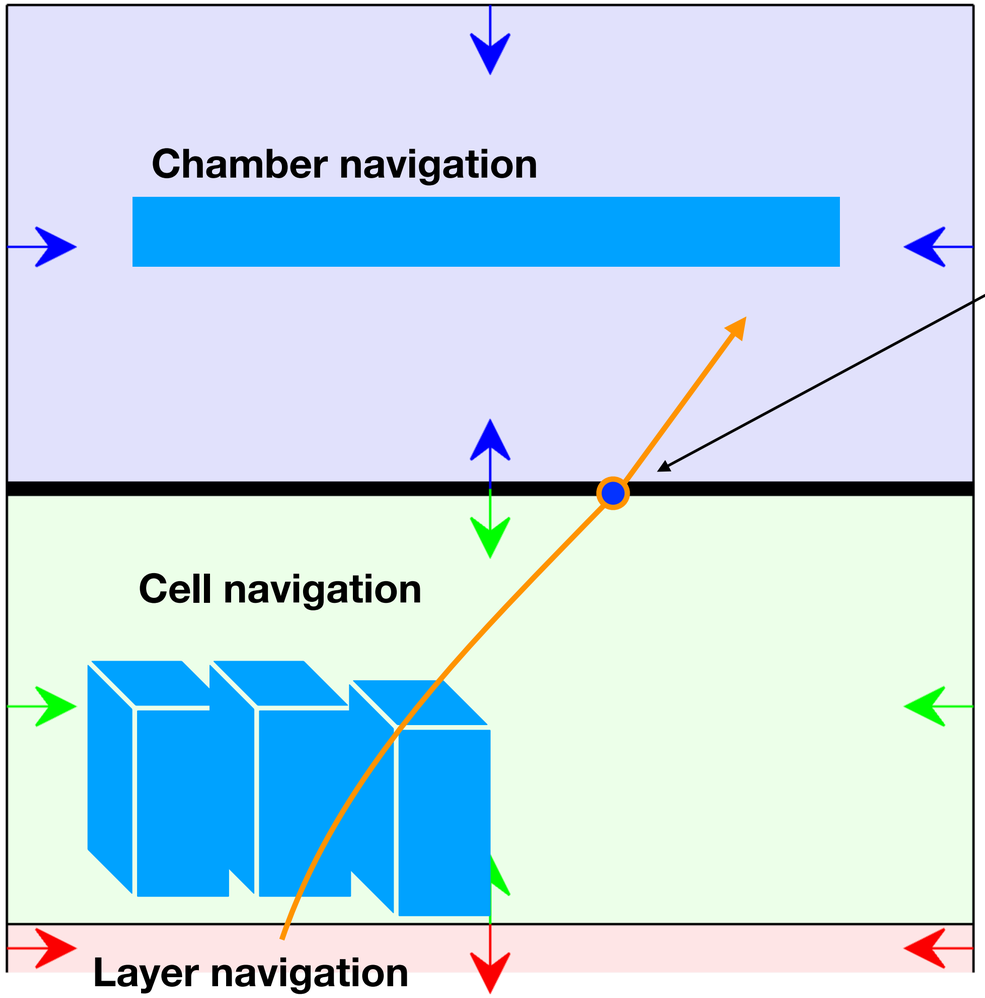


This can be implemented using straight line, frustum, blind search, whatever …

```
/// Declare a navigation state updator
///
/// This delegate dispatches the local navigation action
/// to a dedicated struct or function that is optimised for
/// the given environment.
///
/// @param nState is the navigation state to be updated
/// @param volume is the volume for which this should be called
/// @param gctx is the current geometry context
/// @param position is the position at the query
/// @param direction is the direction at the query
/// @param absMomentum is the absolute momentum at query
/// @param charge is the charge to be used for the intersection
///
using NavigationStateUpdator = Delegate<void(
    NavigationState& nState, const DetectorVolume& volume,
    const GeometryContext& gctx, const Vector3& position,
    const Vector3& direction, ActsScalar absMomentum, ActsScalar charge)>;

/// Memory  managed navigation state updator
using ManagedNavigationStateUpdator = ManagedDelegate<NavigationStateUpdator>;
```

Geometry - Navigation proposal

Portal leads to new volume, volume gives access to local navigation

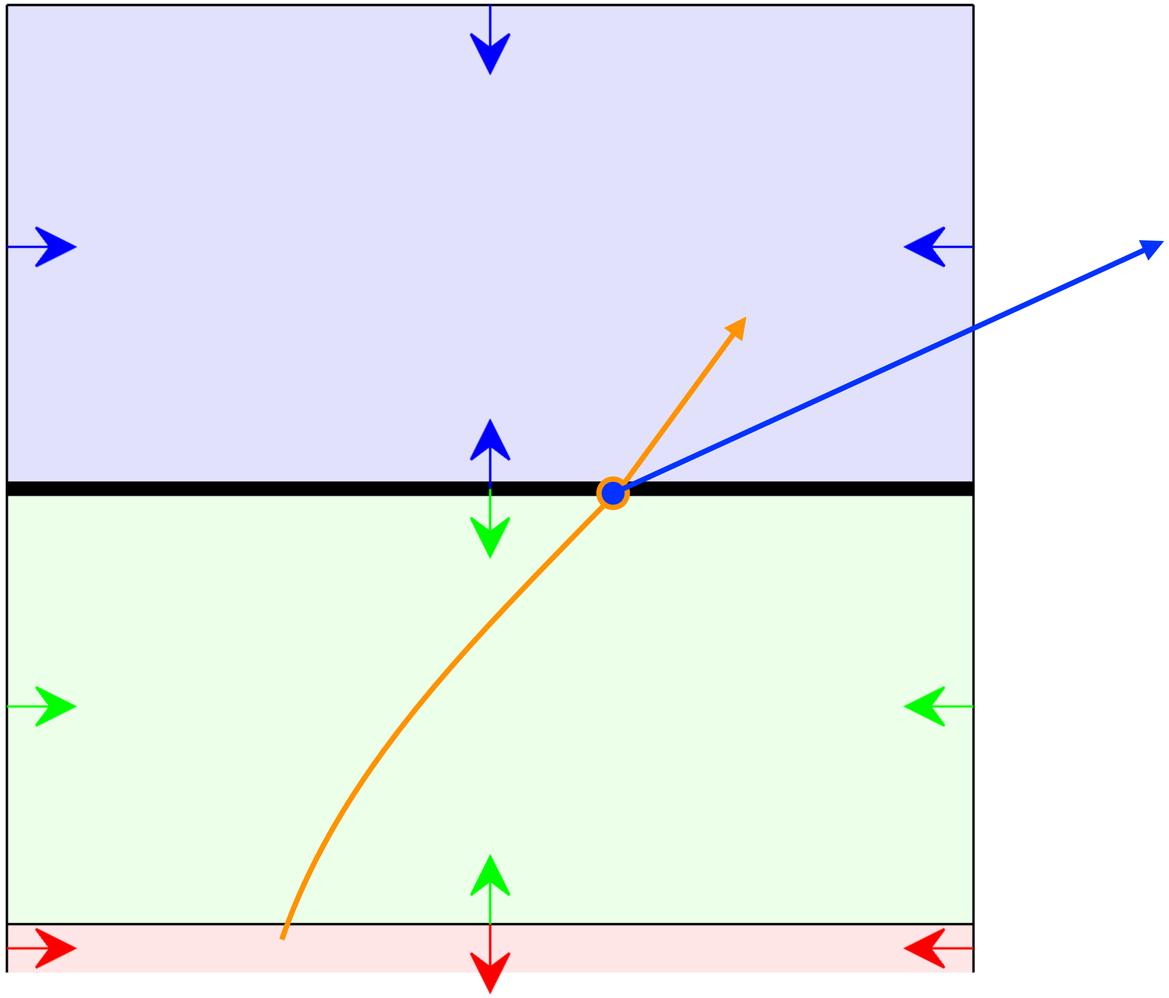


This can be implemented using straight line, frustum, blind search, whatever …

```
/// Declare a navigation state updator
///
/// This delegate dispatches the local navigation action
/// to a dedicated struct or function that is optimised for
/// the given environment.
///
/// @param nState is the navigation state to be updated
/// @param volume is the volume for which this should be called
/// @param gctx is the current geometry context
/// @param position is the position at the query
/// @param direction is the direction at the query
/// @param absMomentum is the absolute momentum at query
/// @param charge is the charge to be used for the intersection
///
using NavigationStateUpdator = Delegate<void(
    NavigationState& nState, const DetectorVolume& volume,
    const GeometryContext& gctx, const Vector3& position,
    const Vector3& direction, ActsScalar absMomentum, ActsScalar charge)>;

/// Memory  managed navigation state updator
using ManagedNavigationStateUpdator = ManagedDelegate<NavigationStateUpdator>;
```

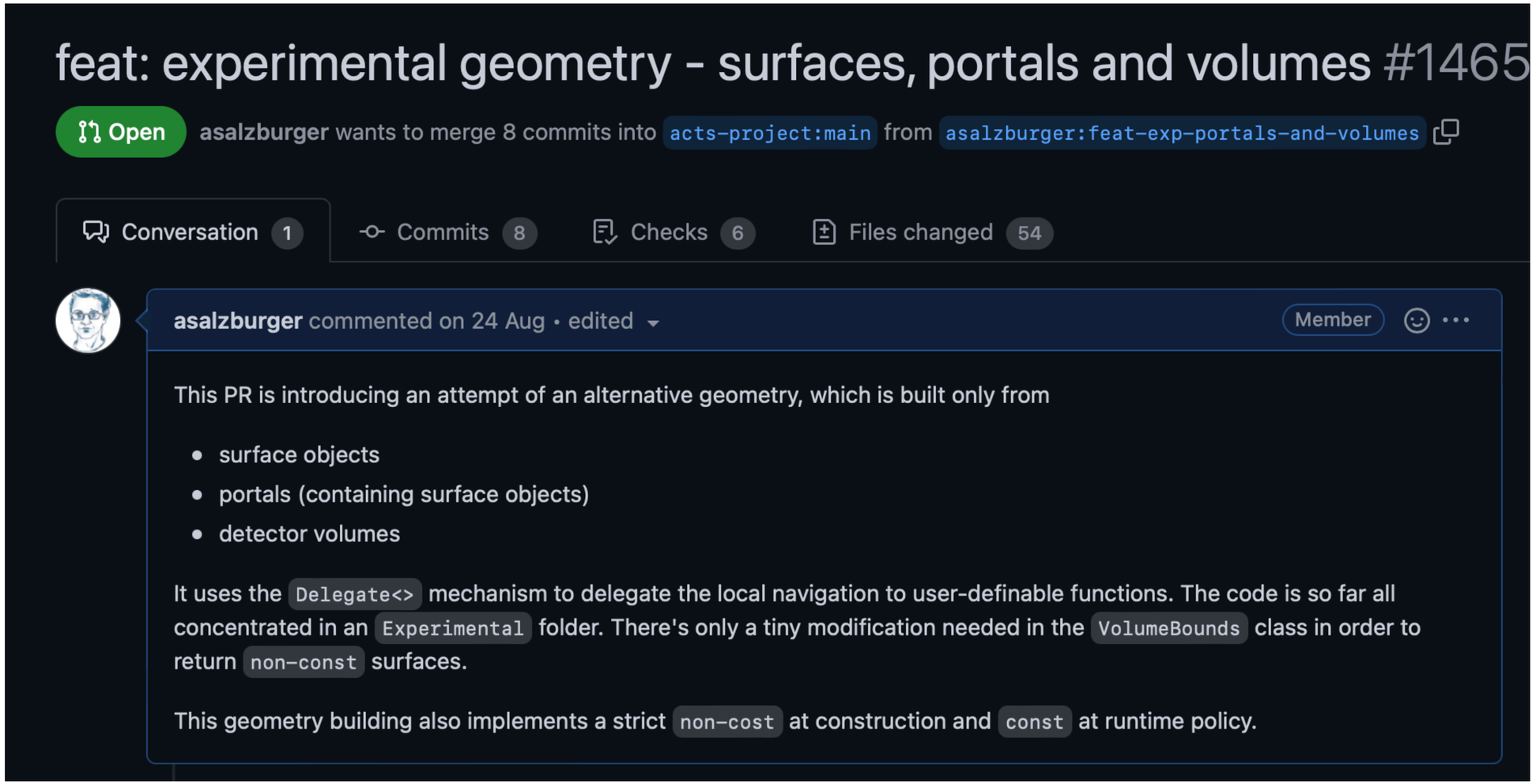
Geometry - Navigation proposal

New volume updates navigation state



```
/// @brief A navigation state holding the current information
/// about volume, surfaces, and portals
struct NavigationState {
  /// @brief  A surface candidate and its intersection
  struct SurfaceCandidate {
    /// A candidate intersection, in Surface view
    ObjectIntersection<Surface> objectIntersection;
    /// A candidate is either a detector Surface
    const Surface* surface = nullptr;
    /// Or a portal
    const Portal* portal = nullptr;
    /// The boundary check used for these
    BoundaryCheck bCheck = true;
  };

  using SurfaceCandidates =
      boost::container::small_vector<SurfaceCandidate, 16u>;

  /// The current volume in processing
  const DetectorVolume* currentVolume = nullptr;

  /// The current surface, i.e the track is on surface
  const Surface* currentSurface = nullptr;

  /// That are the candidate surfaces to process
  SurfaceCandidates surfaceCandidates = {};
  SurfaceCandidates::iterator surfaceCandidate = surfaceCandidates.end();

  /// Boundary directives for surfaces
  BoundaryCheck surfaceBoundaryCheck = true;

  /// An overstep tolerance
  ActsScalar overstepTolerance = -0.1;

  /// Auxilliary attached information
  std::any auxilliary;
};
```

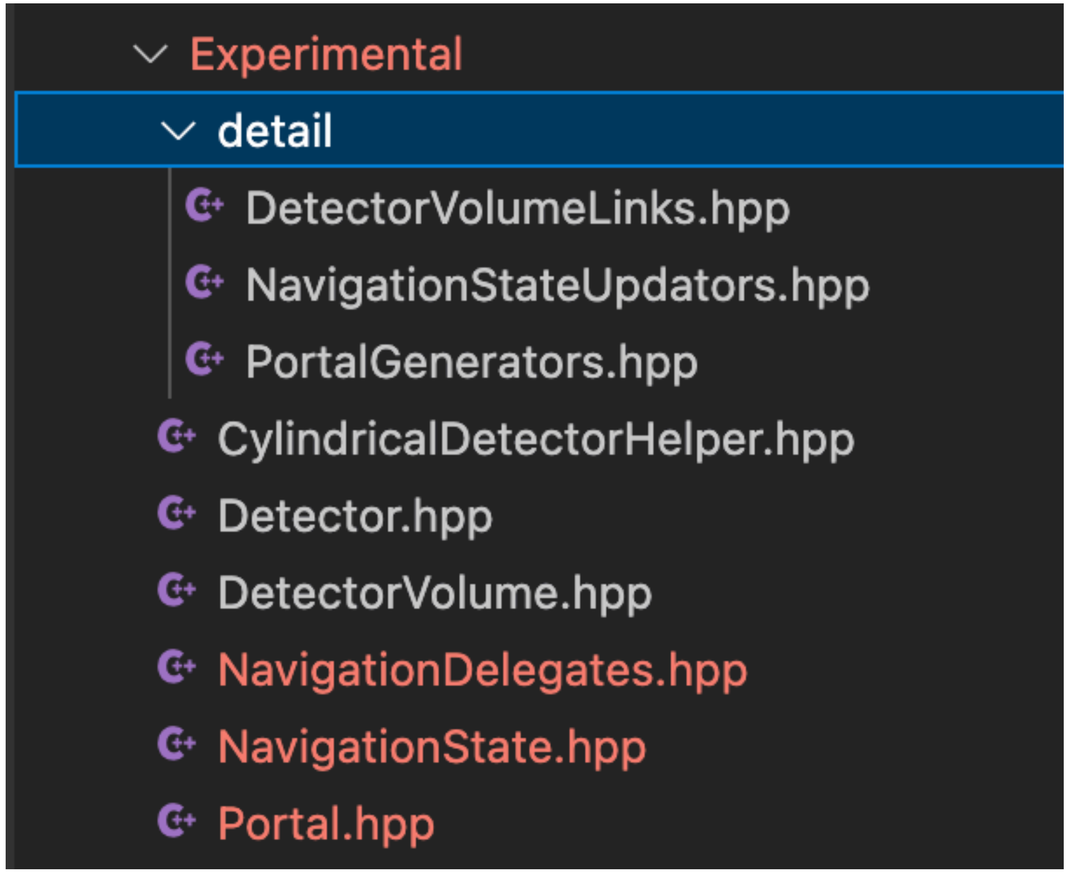
Geometry - Current status (1)



https://github.com/acts-project/acts/pull/1465

Geometry - Current status (2)

All new code in `Experimental` folder and namespace, can be deployed in parallel

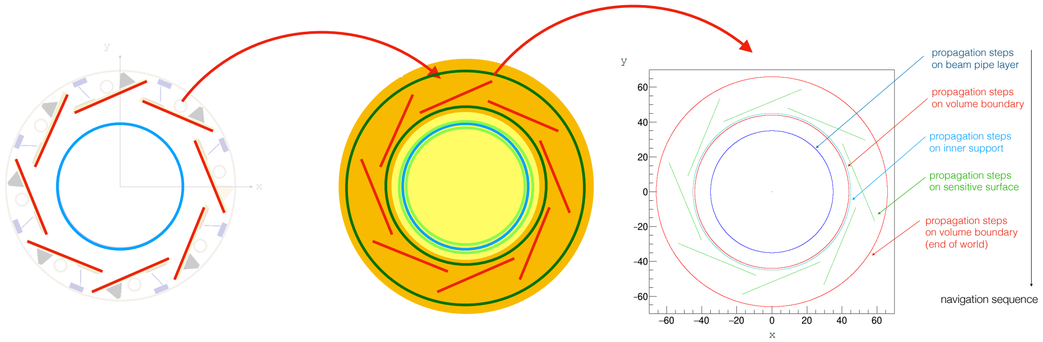


Almost complete setup:
Way less code/classes than in Geometry.

https://github.com/acts-project/acts/pull/1465

Geometry - Other problems/Issues

(Current) geometry building follows a bottom up approach



This setup can be quite brittle:

- potential overlap of volumes (often caused by actual extent + clearance envelopes)

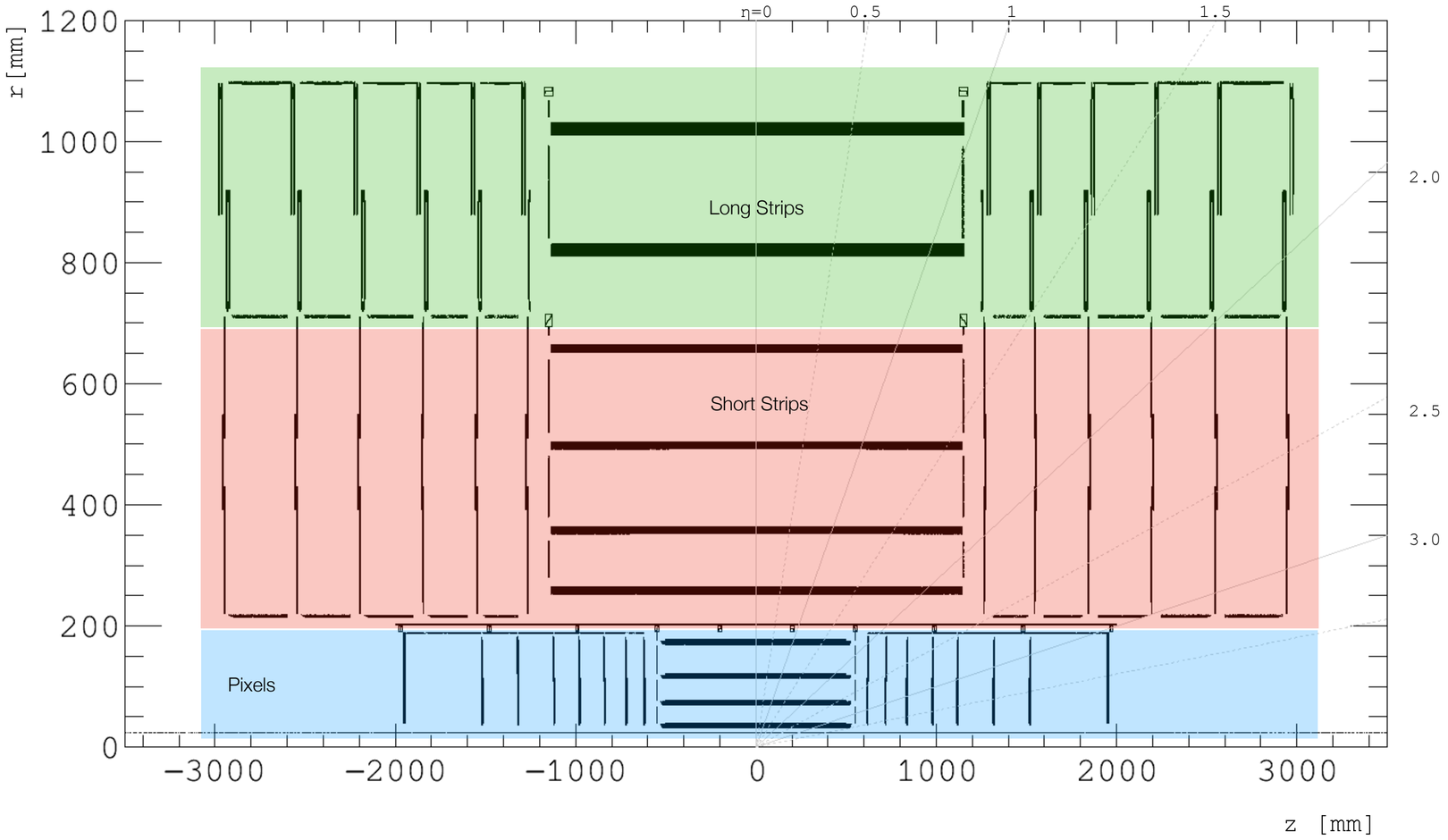
Often 1000s of surfaces are parsed to construct a fixed global skeleton of volumes

- Only few volumes are actually needed

Shall we **reverse** the logic?

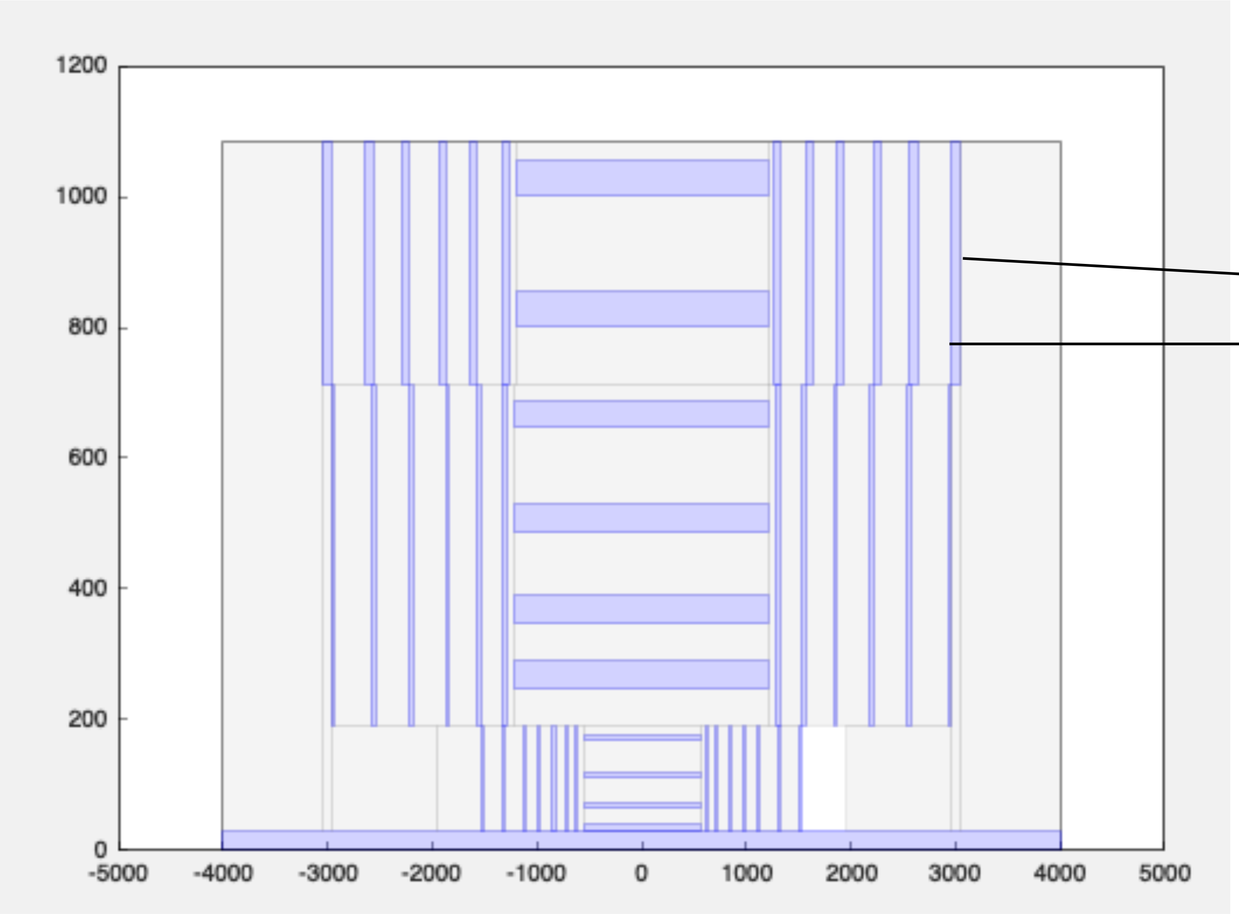
Geometry - Reverse building (1)

<u>Current approach</u>: sensitive modules parsed & ordered to estimate layer dimensions and subsequently volume dimensions:



Geometry - Reverse building (2)

We could give a very light-weight way to build a volume frame first and then “fill” it with local navigation delegates.



“Fillers” that create a substructure delegate.