G4 Geometry Meeting

Reference Counted Touchables

Design

Radovan Chytracek

CERN IT/API Geant4

Motivation

- Use case
  - Transportation creates touchables as a track propagates across volumes
  - User actions get pointers to the touchables allowing to access their data
- Problem
  - Transportation uses two touchables in the flip-flop way for pre-step and post-step points.
  - Pointers passed to user actions become invalid after each step so the touchables must be copied
  - Using touchable pointers implies explicit memory management in a user code and is error prone

Requirements

- Touchables must become "persistent"
  - remain valid after each step (during one run)
- Simpler (automatic) memory management
  - no user action is required to create or delete them
- The existing interface should be preserved if possible

G4 Geometry

Affected classes and categories

- G4VTouchable and its derivates
  - G4GRSSolid, G4GRSVolume, G4TouchableHistory
- G4Navigator
- G4TransportationManager
- G4SteppingManager, G4Track, G4Step
- User actions
- Parametrization and ReadOut Geometry

- May be others...

Possibilities

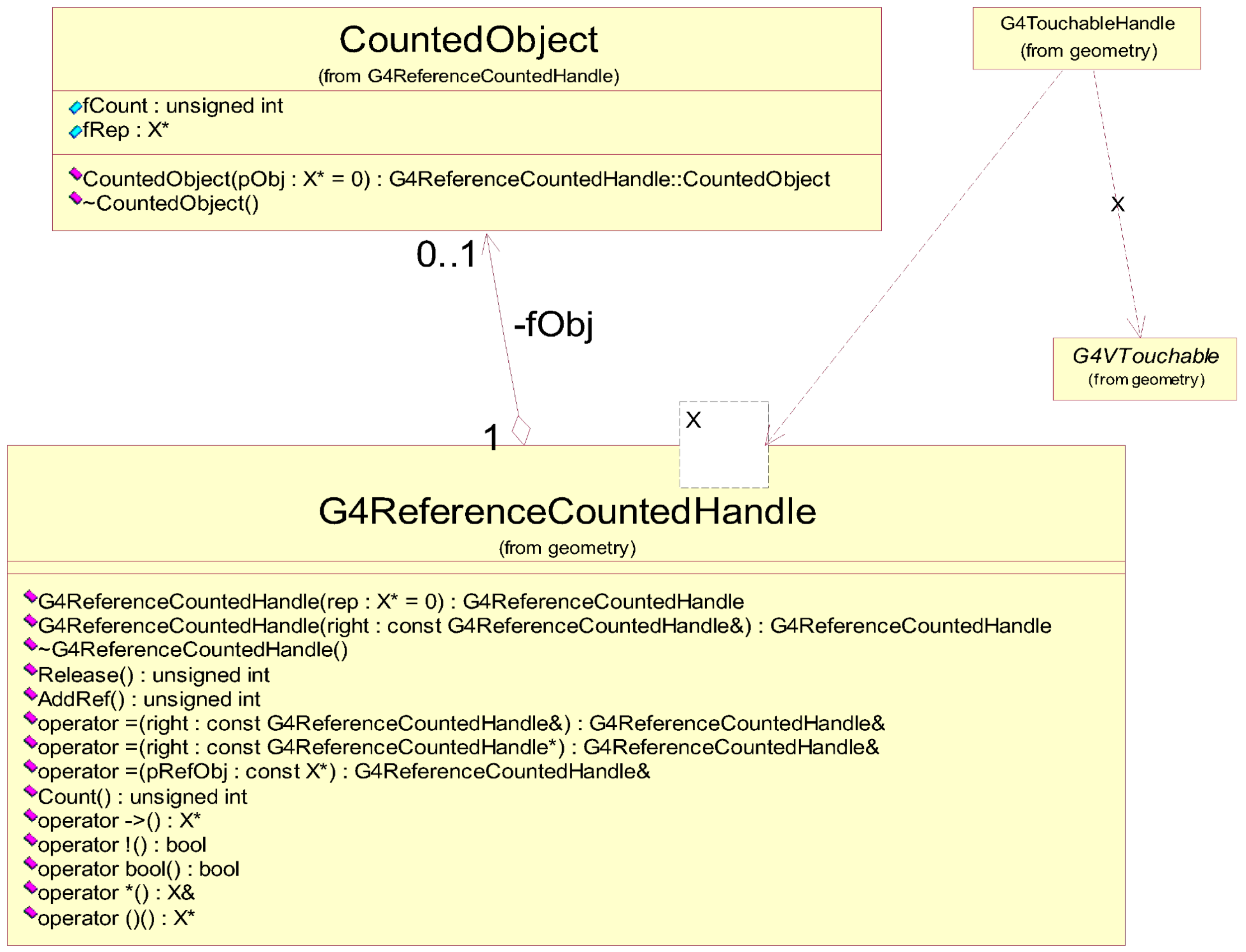
- Handle-Body design pattern
  - G4NavigationLevel already implemented this way
- Smart-pointer idiom
- In both cases the use of pointers must go away
  - Passing by pointer to an object provides no way to trigger proper reference counting unless the users are forced to follow the rule of adding and removing object reference counts by hand
    - Microsoft (COM), Gaudi (Interfaces and Services)
- Both ways imply the clients keep or manipulate the counted objects only using their handles

Prototype implementation

- **The hybrid of auto_ptr (STL) with reference counting ability**
- **A templated class G4ReferenceCountedHandle is introduced**
  - **Essentially a wrapping class emulating dumb pointers**
    - **basically any object can be made reference countable**
  - **Automagically deletes the object when object’s count is zero**
  - **Implements copy constructor, assignment, dereferencing**
  - **Supports conversions to the type of the counted object**
  - **Can’t be created by new or deleted by calling delete**
  - **Must be passed always by reference(&) or copy**
  - **Multiple handles share one instance of counted object**
  - **Does not support exclusive ownership**
    - **an object behind the pointer can be modified**

Design - Class Diagram

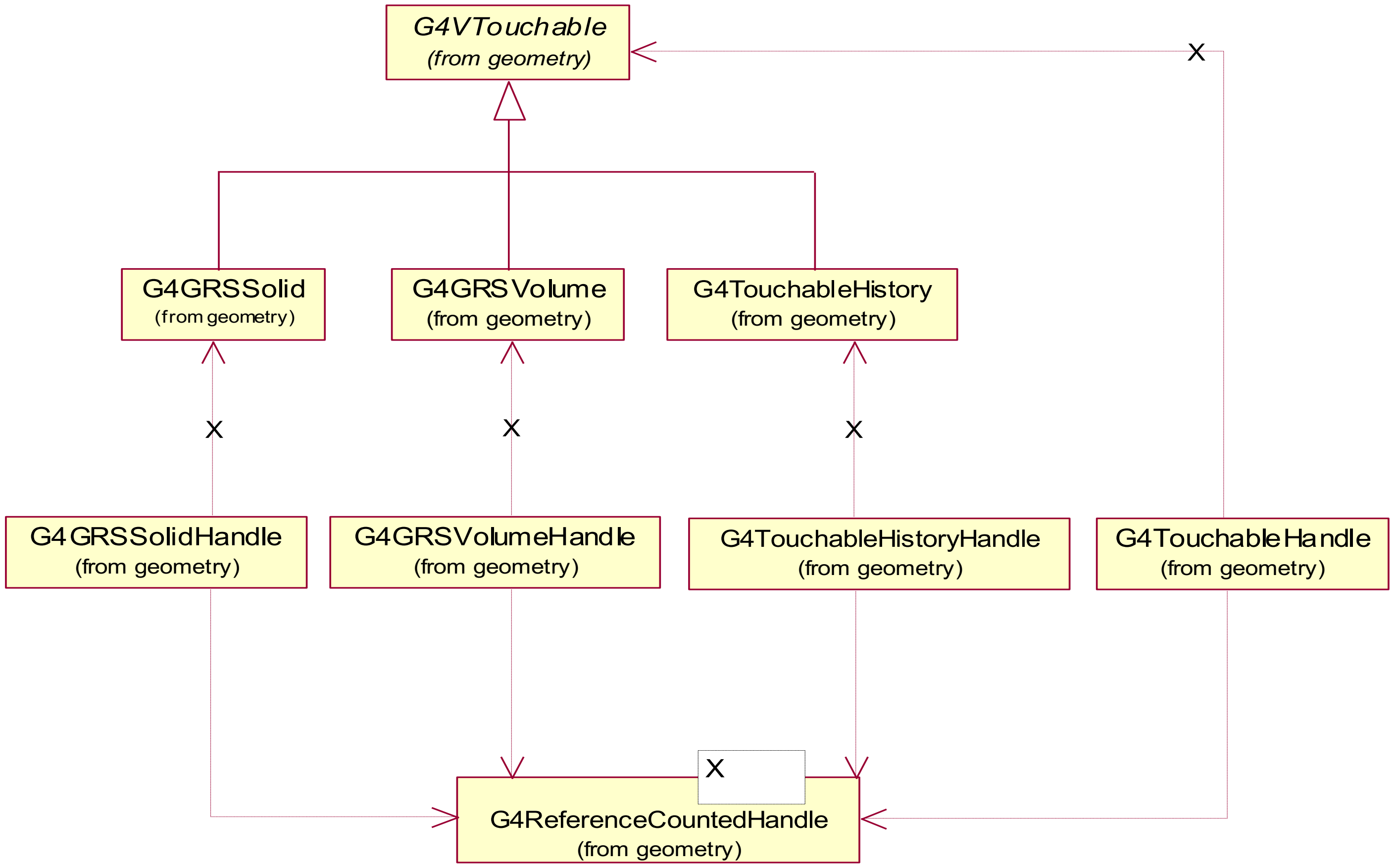
Reference Counted Handle



CountedObject
(from G4ReferenceCountedHandle)
fCount : unsigned int
fRep : X*
CountedObject(pObj : X* = 0) : G4ReferenceCountedHandle::CountedObject
~CountedObject()
G4TouchableHandle
(from geometry)
X
G4VTouchable
(from geometry)
0..1
-fObj
1
X
G4ReferenceCountedHandle
(from geometry)
G4ReferenceCountedHandle(rep : X* = 0) : G4ReferenceCountedHandle
G4ReferenceCountedHandle(right : const G4ReferenceCountedHandle&) : G4ReferenceCountedHandle
~G4ReferenceCountedHandle()
Release() : unsigned int
AddRef() : unsigned int
operator =(right : const G4ReferenceCountedHandle&) : G4ReferenceCountedHandle&
operator =(right : const G4ReferenceCountedHandle*) : G4ReferenceCountedHandle&
operator =(pRefObj : const X*) : G4ReferenceCountedHandle&
Count() : unsigned int
operator ->() : X*
operator !() : bool
operator bool() : bool
operator *() : X&
operator ()() : X*

Design - Class Diagram

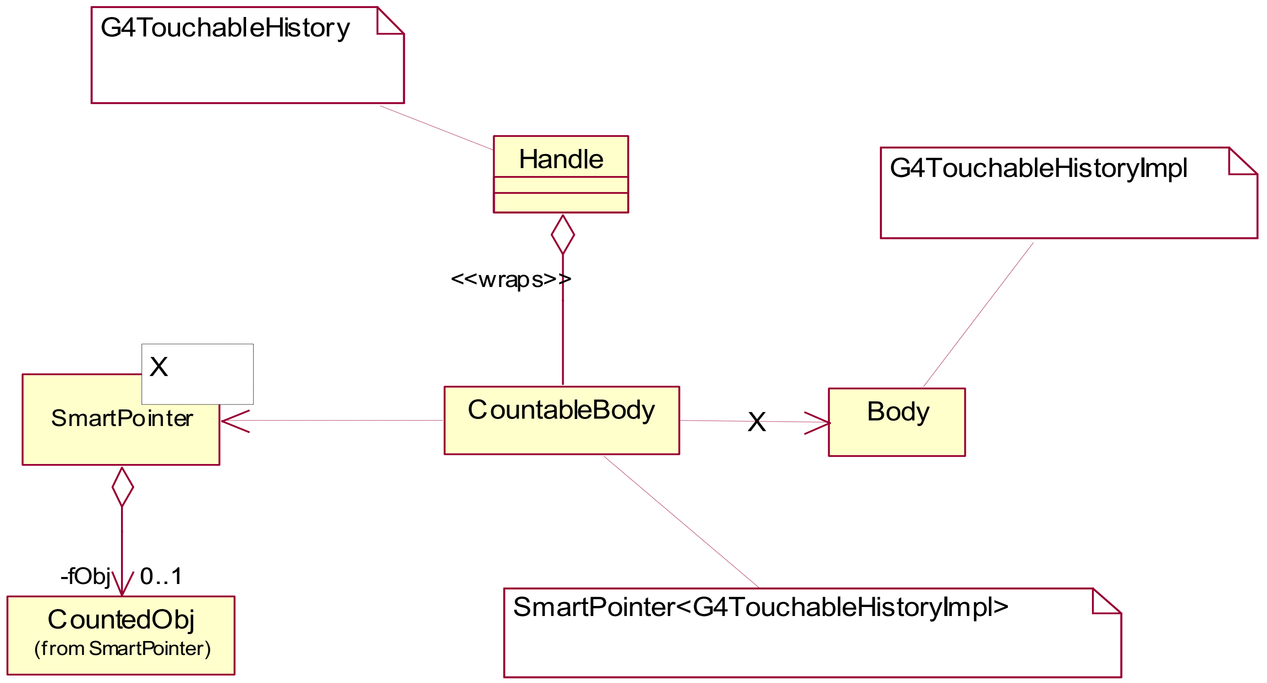
Reference Counted Touchables



Possible extensions

- **Hide the operators new() and delete()**
  - **an attempt to dynamically create or delete is detected at compile time**
- **Implementation using G4Allocator and related classes, if needed, to make it faster**
- **Alternative implementation of client classes**
  - **smart-pointers not used directly as a reference counting handle**
  - **client wrapper classes(handle) hold the smart-pointer to their implementation classes(body)**

Alternative Implementation



```
class Handle {
private:
SmartPointer<Body> fCountableBody;
};
```