```
#ifndef Vertex_h
#define Vertex_h 1
////////////////////////////////////////////////////////////////////////////////
//
#include "globals.hh"
#include <vector>

#include "G4VHit.hh"
#include "G4THitsCollection.hh"
#include "G4Allocator.hh"
#include "G4ThreeVector.hh"
#include "Track.hh"

#include "G4HCofThisEvent.hh"

////////////////////////////////////////////////////////////////////////////////
//
class Vertex : public G4VHit
{
public:
  Vertex ();
  Vertex (G4bool aQuaiselastic, G4bool aInelastic,
          G4String aVolumeName,G4ThreeVector aPosition);

  ~Vertex ();
  Vertex (const Vertex&);
  const Vertex& operator= (const Vertex&);
  int operator== (const Vertex&) const;

  inline void* operator new(size_t);
  inline void  operator delete(void*);

  inline G4String       GetVolumeName(){return theVolumeName;}
  inline G4ThreeVector  GetPosition(){return thePosition;}
  inline G4bool         IsQuasielastic(){return theQuasielastic;}
  inline G4bool         IsInelastic(){return theInelastic;}


  inline void  SetIsQuasielastic(G4bool aQuasielastic){theQuasielastic=aQuasiela
stic;}
  inline void  SetIsInelastic(G4bool aInelastic){theInelastic=aInelastic;}
  inline void  SetPosition(G4ThreeVector aPos){thePosition=aPos;}
  inline void  SetVolumeName(G4String aVolName){theVolumeName=aVolName;}


  void Draw () {};
  void Print () {};
  void clear () {};
  void DrawAll () {};
  void PrintAll () {};

private:
  G4String           theVolumeName;
  G4ThreeVector      thePosition;
  G4bool             theInelastic;
  G4bool             theQuasielastic;
};

typedef G4THitsCollection<Vertex> VertexsCollection;

extern G4Allocator<Vertex> VertexAllocator;

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

inline void* Vertex::operator new(size_t)
{
  void *aHit;
  aHit = (void *) VertexAllocator.MallocSingle();
  return aHit;
```

```
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

inline void Vertex::operator delete(void *aHit)
{
  VertexAllocator.FreeSingle((Vertex*) aHit);
}


////////////////////////////////////////////////////////////////////////////////
#endif
```