```
//
// ********************************************************************
// * License and Disclaimer                                           *
// *                                                                  *
// * The  Geant4 software  is  copyright of the Copyright Holders  of *
// * the Geant4 Collaboration.  It is provided  under  the terms  and *
// * conditions of the Geant4 Software License,  included in the file *
// * LICENSE and available at  http://cern.ch/geant4/license .  These *
// * include a list of copyright holders.                             *
// *                                                                  *
// * Neither the authors of this software system, nor their employing *
// * institutes,nor the agencies providing financial support for this *
// * work  make  any representation or  warranty, express or implied, *
// * regarding  this  software system or assume any liability for its *
// * use.  Please see the license in the file  LICENSE  and URL above *
// * for the full disclaimer and the limitation of liability.         *
// *                                                                  *
// * This  code  implementation is the result of  the  scientific and *
// * technical work of the GEANT4 collaboration.                      *
// * By using,  copying,  modifying or  distributing the software (or *
// * any work based  on the software)  you  agree  to acknowledge its *
// * use  in  resulting  scientific  publications,  and indicate your *
// * acceptance of all terms of the Geant4 Software license.          *
// ********************************************************************
//
/// \file electromagnetic/TestEm3/src/HEPDSWPrimaryGeneratorMessenger.cc
/// \brief Implementation of the HEPDSWPrimaryGeneratorMessenger class
//
// $Id$
//
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

#include "HEPDSWPrimaryGeneratorMessenger.hh"

#include "HEPDSWPrimaryGeneratorAction.hh"
#include "G4UIdirectory.hh"
#include "G4UIparameter.hh"
#include "G4UIcommand.hh"
#include "G4UIcmdWithoutParameter.hh"
#include "G4UIcmdWithADoubleAndUnit.hh"
#include "G4UIcmdWithAString.hh"

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

HEPDSWPrimaryGeneratorMessenger::HEPDSWPrimaryGeneratorMessenger(HEPDSWPrimaryGe
neratorAction* Gun):fAction(Gun)
{
  fGunDir = new G4UIdirectory("/hepd/gun/");
  fGunDir->SetGuidance("gun control");

  fDefaultCmd = new G4UIcmdWithoutParameter("/hepd/gun/setDefault",this);
  fDefaultCmd->SetGuidance("set/reset kinematic defined in PrimaryGenerator");
  fDefaultCmd->AvailableForStates(G4State_PreInit,G4State_Idle);

  fRndmCmd = new G4UIcmdWithoutParameter("/hepd/gun/random",this);
  fRndmCmd->SetGuidance("random position of initial point");
  fRndmCmd->AvailableForStates(G4State_PreInit,G4State_Idle);

  fPntngCmd = new G4UIcmdWithoutParameter("/hepd/gun/toCenter",this);
  fPntngCmd->SetGuidance("direction of particle always pointing to center");
  fPntngCmd->AvailableForStates(G4State_PreInit,G4State_Idle);

  fEnrgCmd = new G4UIcmdWithADoubleAndUnit("/hepd/gun/energy",this);
  fEnrgCmd->SetGuidance("Set the energy of the particle");
  fEnrgCmd->SetParameterName("Energy",false);
  fEnrgCmd->SetUnitCategory("Energy");
  fEnrgCmd->AvailableForStates(G4State_Idle);
```

```
  fPartCmd = new G4UIcmdWithAString("/hepd/gun/particle",this);
  fPartCmd->SetGuidance("Set the particle type");
  fPartCmd->SetParameterName("Particle Type",false);
  fPartCmd->AvailableForStates(G4State_Idle);

  G4UIparameter* param;

  fDummyCmd = new G4UIcommand("/hepd/gun/dummy",this);
  fDummyCmd->AvailableForStates(G4State_Idle);
  param = new G4UIparameter("X",'d',false);
  param->SetGuidance("X position");
  fDummyCmd->SetParameter(param);
  param = new G4UIparameter("Y",'d',false);
  param->SetGuidance("Y position");
  fDummyCmd->SetParameter(param);
  param = new G4UIparameter("unit",'s',false);
  param->SetGuidance("length unit");
  fDummyCmd->SetParameter(param);
  param = new G4UIparameter("Theta",'d',false);
  param->SetGuidance("Theta angle");
  fDummyCmd->SetParameter(param);
  param = new G4UIparameter("unit",'s',false);
  param->SetGuidance("angle unit");
  fDummyCmd->SetParameter(param);


  fPowerLawCmd = new G4UIcommand("/hepd/gun/powerlaw",this);
  fPowerLawCmd->SetGuidance("Set the power law with Emin Emax [unit]  Gamma");
  fPowerLawCmd->AvailableForStates(G4State_Idle);
  param = new G4UIparameter("Emin",'d',false);
  param->SetGuidance("E min");
  fPowerLawCmd->SetParameter(param);
  param = new G4UIparameter("Emax",'d',false);
  param->SetGuidance("E max");
  fPowerLawCmd->SetParameter(param);
  param = new G4UIparameter("unit",'s',false);
  param->SetGuidance("E unit");
  fPowerLawCmd->SetParameter(param);
  param = new G4UIparameter("gamma",'d',false);
  param->SetGuidance("gamma");
  fPowerLawCmd->SetParameter(param);

}

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

HEPDSWPrimaryGeneratorMessenger::~HEPDSWPrimaryGeneratorMessenger()
{
  delete fDefaultCmd;
  delete fRndmCmd;
  delete fEnrgCmd;
  delete fPntngCmd;
  delete fDummyCmd;
  delete fPowerLawCmd;
  delete fGunDir;
}

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

void HEPDSWPrimaryGeneratorMessenger::SetNewValue(G4UIcommand* command,G4String
newValue)
{
  if( command == fDefaultCmd )
    { fAction->SetDefaultKinematic();}
  if( command == fRndmCmd )
    { fAction->SetRandomPosition();}
  if( command == fPntngCmd )
    { fAction->SetDirectionToCenter();}
  if( command == fEnrgCmd )
```

```
    { fAction->SetEnergy(fEnrgCmd->GetNewDoubleValue(newValue));}
  if( command == fPartCmd )
    { fAction->SetParticle(newValue);}
  if( command == fDummyCmd )
    {
      G4double Xpos,Ypos,theta;
      G4String unit_l,unit_a;
      std::istringstream is(newValue);
      is >> Xpos >> Ypos >> unit_l >> theta >> unit_a;
      Xpos*= G4UIcommand::ValueOf(unit_l);
      Ypos*= G4UIcommand::ValueOf(unit_l);
      theta*= G4UIcommand::ValueOf(unit_a);
      fAction->SetDummy(Xpos,Ypos,theta);
    }
  if( command == fPowerLawCmd )
    {
      G4double Emin,Emax,gamma;
      G4String unit;
      std::istringstream is(newValue);
      is >> Emin >> Emax >> unit >> gamma;
      Emin*= G4UIcommand::ValueOf(unit);
      Emax*= G4UIcommand::ValueOf(unit);
      fAction->SetPowerLaw(Emin,Emax,gamma);
    }
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
```