Tools for the unified QA: json → histogram canvas generator

Ilnur Gabdrakhmanov

JINR, VBLHEP

Dubna November 23, 2020

Main objectives:

- Make addition of histogram simple and flexible
- It should not require code rebuild
- If possible make the same for filling logic

Implementation:

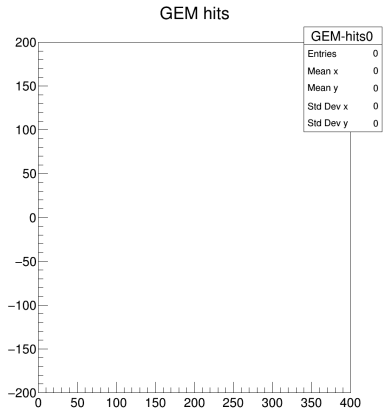
- User writes histograms configuration in the json file
- BmnPadGenerator converts json structure to the recursively nested BmnPadBranch objects
- BmnPadBranch tree can be filled or drawn afterwards

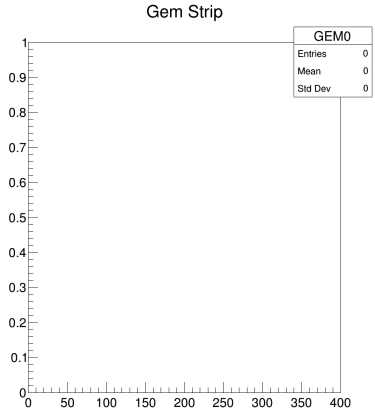
Simple configuration

JSON scheme:

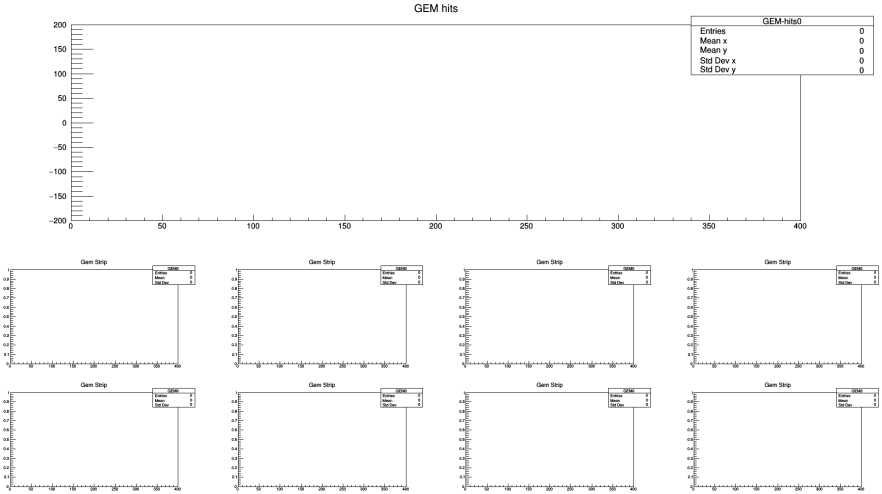
```
{
  "Name": "GEMS",
  "Title": "GEM Canvas",
  "DivX": "2",
  "DivY": "1",
  "Pads": [
    {
      "Class": "TH2I",
      "Name": "GEM-hits0",
      "Title": "GEM hits",
      "Options": "colz",
      "Dimensions": [
        200,
        0,
        400,
        400,
        -200,
        200
      ]
    },
    {
      "Class": "TH1F",
      "Name": "GEM0",
      "Title": "Gem Strip",
      "Dimensions": [
        200,
        0,
        400
      ]
    }
  ]
}
```

Canvas structure:

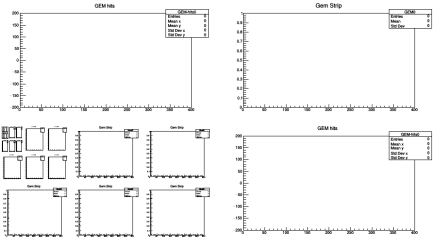


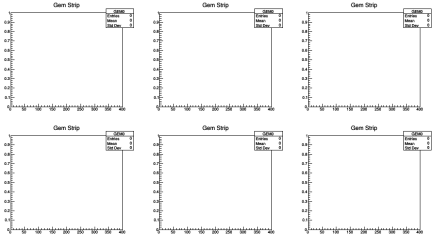


More complex configuration



More complex configuration





Code example

```
BmnPadGenerator *g = new BmnPadGenerator();
g->LoadPTFrom(FileName);
BmnPadBranch * br = g->GetPadBranch();
TCanvas* can = new TCanvas("canHits", "", 1920, 1080);
g->PadTree2Canvas(br, can);
BmnHist::DrawPadTree(br);
```

Next steps

- Implement filling procedures
- [optional] Add support for style options (or use one style for all)
- Try to implement configuration scheme for histogram filling logic