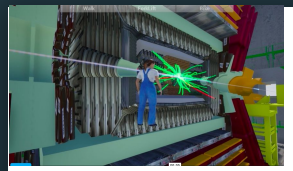
vDUNE Status

Lucas Sorenson, David DeMuth, March 30, 2021

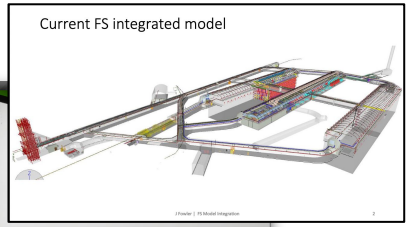
Link to Slides

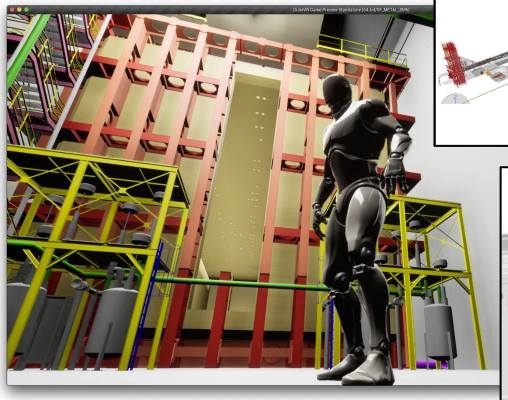


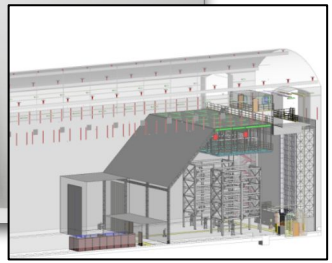
Previous Progress

Goal: Develop a 3D game environment conducive for use by high school teachers and their students to better identify with the inner workings of the DUNE particle detector as a complement to a supernova focused neutrino masterclass.

- A 3D model of the DUNE underground lab at Lead, SD was developed using NavisWorks models created by J. Fowler and hosted at www.edms.cern.ch
- The far detector complex uses Epic Games UnReal Engine to render environments.
- An avatar can navigate and interact with selected model elements and can walk into the open cryostat to learn explore detector components.

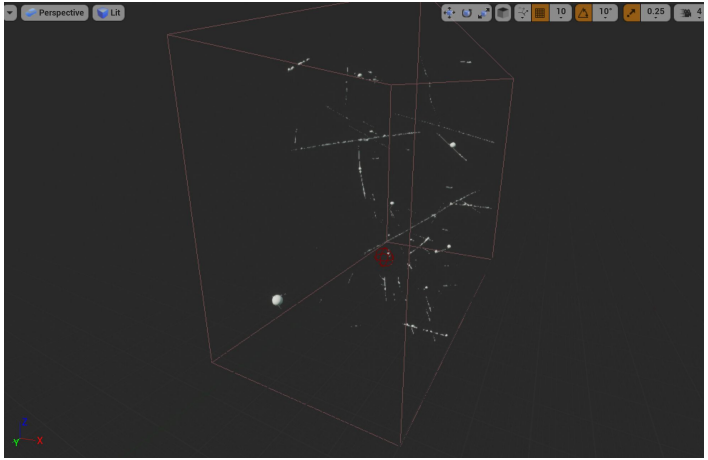


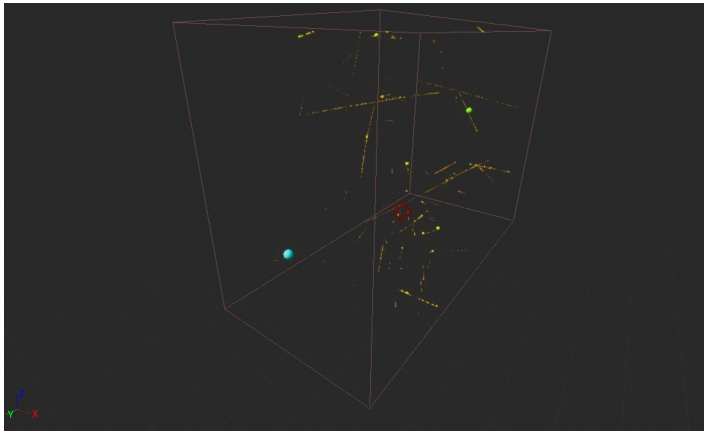




Overview of Progress

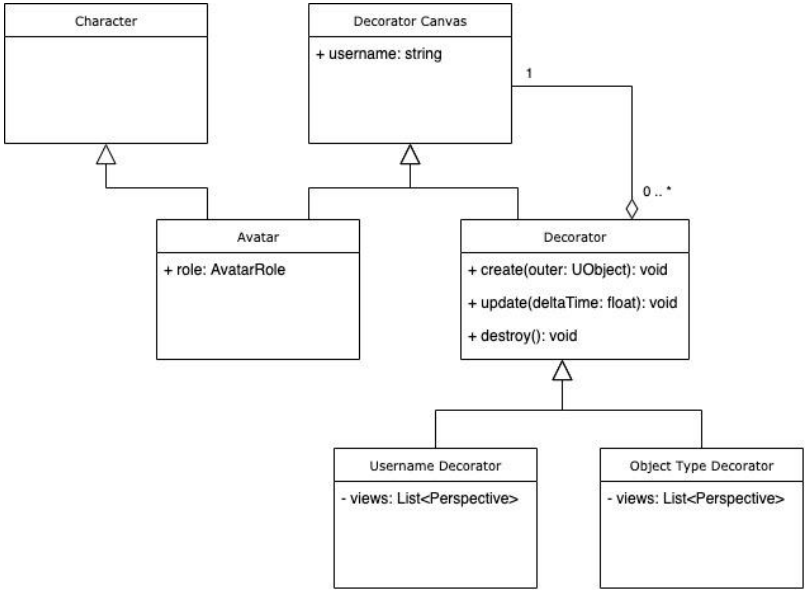
- ProtoDUNE 1 data has been converted to JSON data structures and displayed in a model detector
- QT generated metadata for each event added to json data structure, and parsed for event display information.
- Event display boundary box
    - When processed in the QT ETL process, metadata is produced for events including information about the event boundary.
    - This data can be used to position cameras and scale the event to display it in our virtual environments.



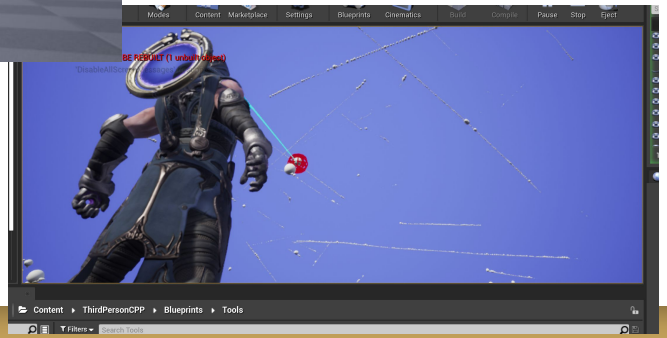


Tools & Decorators

- Tools can be used to interact with both the environment and with neutrino events.
  - With a laser pointer the user can extract more information about a subject.
  - The user can measure the distance between two points.
  - Items can be picked up and stored in a user's inventory.
- Decorators are used to provide variations on how information is displayed to the user. Thus far we have focused on two main implementations of this:
  - Displaying information about a player.
  - Displaying information about a neutrino point.



Neutrino Event Lifecycle



1. When an event actor is loaded, Neutrino Event Data is extracted from a JSON file.
2. Data pertaining to each neutrino point is stored on an object in memory.
3. A 3D mesh is generated for each point based on a simple sphere model.
4. RGB values for each point are derived from its charge.
5. The resulting color is applied to a procedurally generated material, which is then applied to the mesh for the corresponding neutrino point.

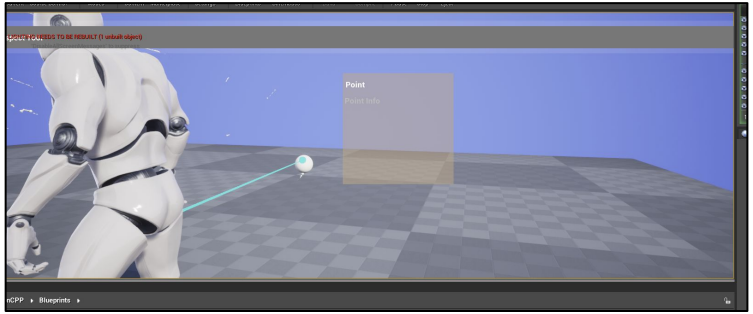
Material Code

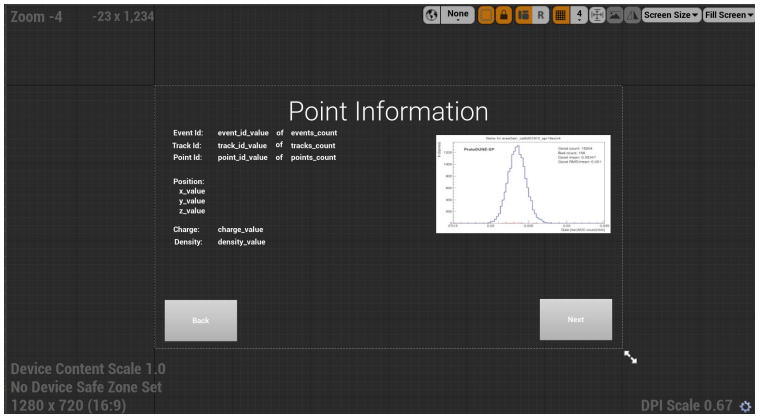
```
48  void ANeutrinoPoint::set_color_by_charge(double charge)
49  {
50      UE_LOG(LogClass, Error, TEXT( x: "setting color for charge: %f"), charge);
51      // if the charge is less than zero, the parameter is invalid.
52      if (charge < 0) return;
53      // if the mesh is null, no color can be assigned.
54      if (!point_mesh_) return;
55
56      // normalize charge
57      float spectrum_position;
58      spectrum_position = charge / max_charge_;
59      UE_LOG(LogClass, Error, TEXT( x: "Charge: %f Max Charge: %f"), charge, max_charge_);
60
61      auto category = get_category(spectrum_position);
62
63      // Generate a set of rgb values.
64      auto rgb = get_rgb(category);
65
66      // Try to create a material.  Skip application if the material construction fails.
67      UMaterialInstanceDynamic *material = UMaterialInstanceDynamic::Create(material_interface_, this);
68      if (!material) return;
69
70      // Apply the generated color to the material.
71      material->SetVectorParameterValue("BaseColor", FLinearColor(rgb[0], rgb[1], rgb[2], 1.0f));
72      point_mesh_->SetMaterial(0, material);
73  }
```

Interactive Particle Track Elements

Goal: From an event, allow user to select a point on any track to review metadata.

- All points on the track become touchable by a pointable laser tool.
- Pop up window to display point data:
  - Position (x,y,z)
  - Charge, and charge density
  - Associated Track id
  - Sum of charge on track
- Popup window can include a graphic such as a histograms.
- Popup window includes links that can be selected by the avatar that open an in game browser, allowing for more fluid content hosting.





Future and In-progress Use Cases

1. Networking (Connecting to vDune Instances and multiple user interaction)
    a. Assigning user roles and allowing teacher to gather students into a specific location.
    b. Defining user abilities and actions specific to their user role.
2. Display information about characters or objects.
3. Introducing new avatar actions including dropping a ball.
4. Giving the user the ability to get a closer view of individual neutrino tracks and points.
5. Loading neutrino events asynchronously.