Courier V3.2.1

Foundation Changes

Time stamp and Source ID have been added to the Touch Message and the Touch Info.

See also:

Courier::TouchMsg::SetTimeStamp(), Courier::TouchMsg::SetSourceId() and Courier::TouchInfo.

Visualization Changes

- Renderjob Strategies introduced
- Create View by View ID
- Dirty Area Management

Renderjob Strategies introduced

A new class Courier::RenderJobStrategy has been introduced which is used to sort the RenderJobs. The default sort criterias are:

1. offscreen render jobs before Display render jobs.
2. 2d render jobs before 3d render jobs
3. clear render jobs before normal view render jobs
4. lower camera sequence number before higher camera sequence number For custom sort criteria simply derive from RenderJobStrategy and use the Renderer::SetRenderJobStrategy. The Courier::Renderer will use the Courier::RenderJobStrategy and utilize also the Courier::Gdu::RenderState structure.

Create View by View ID

The class Courier::ViewFactory has been enhanced by a method Courier::ViewFactory::Create which creates a View object identified by its View id. Used by Courier::ViewHandler class.

Dirty Area Management

Dirty Area Management in Widgets

The easiest way for a widget developer is to simply call the Invalidate method of the Courier::FrameworkWidget whenever the scene tree has been changed (structure and setting changes). By default this will lead to an invalidation of the associated view with a dirty area that marks the complete area as dirty.

The Courier dirty area management enables widget developers to overwrite that Invalidate method. To provide an out-of-the-box dirty area based implementation a base widget is recommended that takes the bounding rectangle (2D) resp. bounding box (3D) before the change and uses it as source for the dirty area of the current frame.

The bounding rectangle/box has first to be transformed into the view space of the camera.

Scene Dependencies

Invalidation dependencies have been extended by an optional dirty area. This enables the application developer to limit invalidation dependencies to dirty areas. Dependent on the size of the dirty area invalidation of a view in a small area no longer leads to an invalidation of the complete chain of invalidation dependencies.

The only change required on application side to limit invalidation dependencies is to provide a dirty area as small as possible when calling Invalidate. If this is already handled by a common widget base class, see next section, no additional changes are required.

Widget Changes

A common base class for 2D and 3D is already provided which overwrites the following method:

See also:

Courier::FrameworkWidget::Invalidate

If the optional dirtyArea parameter is not set, the bounding rectangle (in 2D) resp. the bounding box (in 3D) of the attached node is taken as dirty area. This base class is called BaseSampleWidget, which can be found in "<cgi_studio_root>/cgi_studio_cit/src/CitApps/CitDemoApp/Widget".

For default dirty area management all widgets need to be derived from this widget class.

In case that the bounding rectangle/box is not sufficient for correct invalidation, the widget itself has to provide a corresponding dirty area via the FrameworkWidget::Invalidate method. In this case it is also up to the widget to take care of correct invalidation of the old dirty area.

Camera View Space Transformation

In case an own dirty area is provided, the widget first has to transform its specific dirty area into the camera view space before calling Invalidate. This is done by multiplying the world transformation matrix of the attached node with the view matrix of the camera, as done in the following code snippet:

```
Candera::Matrix3x2 worldTransformation = node->GetWorldTransform();
cameraDirtyArea.Transform(worldTransformation * camera->GetViewMatrix()); // transform dirty are
```

The camera can either be added to the widget via an own property or be retrieved from the parent view in the following way:

```
Courier::ViewScene2D::CameraPtrVector& cameras = GetParentView()->ToViewScene2D()->GetCameraPtrV
```

In case that a scene contains more than one camera the transformation of the dirty area has to be done for each camera.

See also:

BaseSampleWidget::Invalidate implementation as a reference.

Revision #3
Created 2023-03-02 05:49:33 UTC by Kanai Tomoaki
Updated 2023-06-19 03:06:48 UTC by Tsuyoshi.Kato