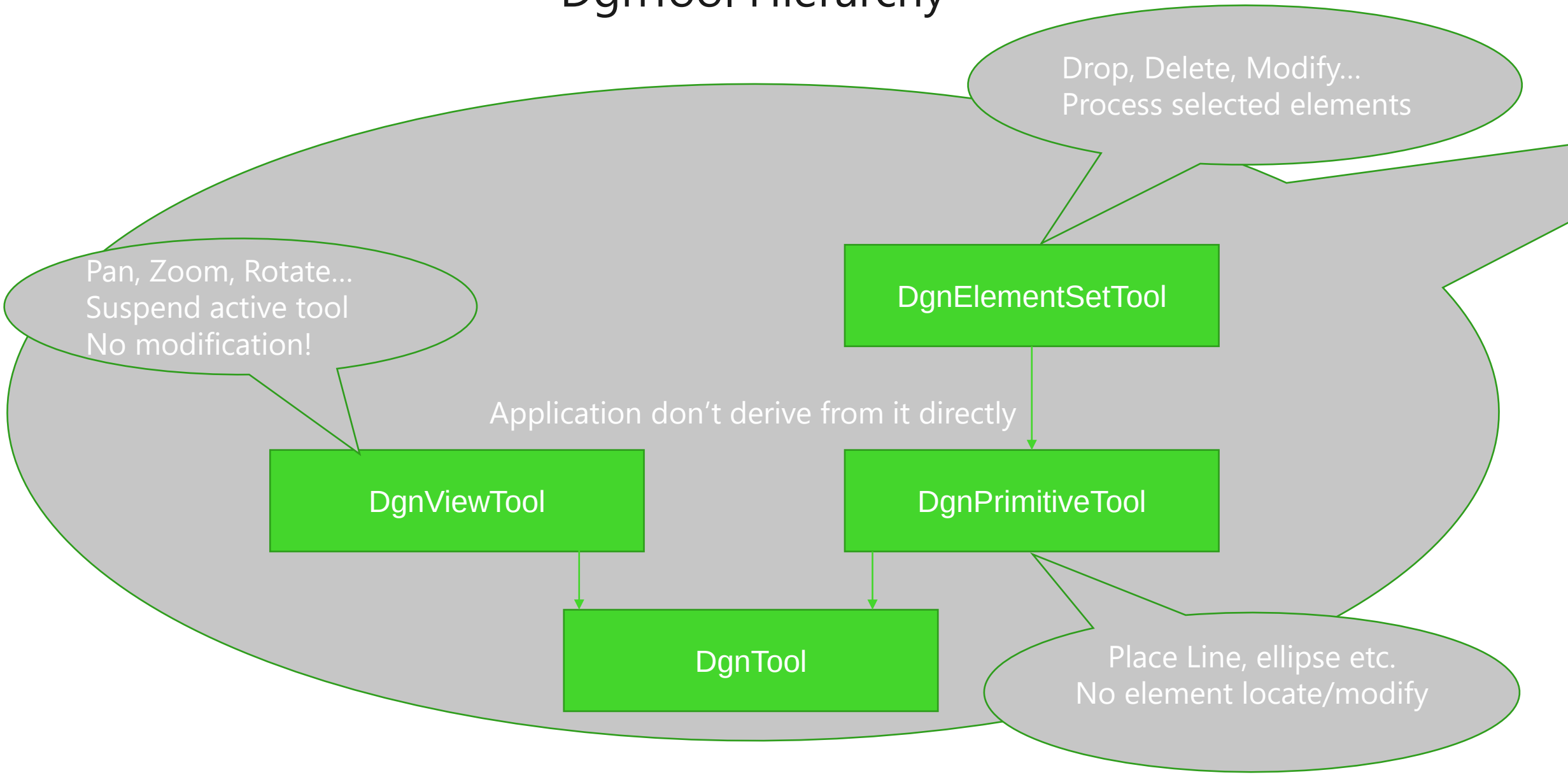
Implementing MicroStation Tools

Bentley®

DgnTool Hierarchy



Bentley

DgnTool - Application don’t derive from it directly

- DgnTool base class responsibilities
  - Tool name
  - Prompts
  - PopulateToolSettings
- Starting a tool

```
tool = MyTool(0, 0)
Tool.InstallTool();
```

  - MyTool is now the active primitive tool
  - The arguments to the constructor are not used in Python
  - Automatically freed when tool exits
- Obtaining the active tool

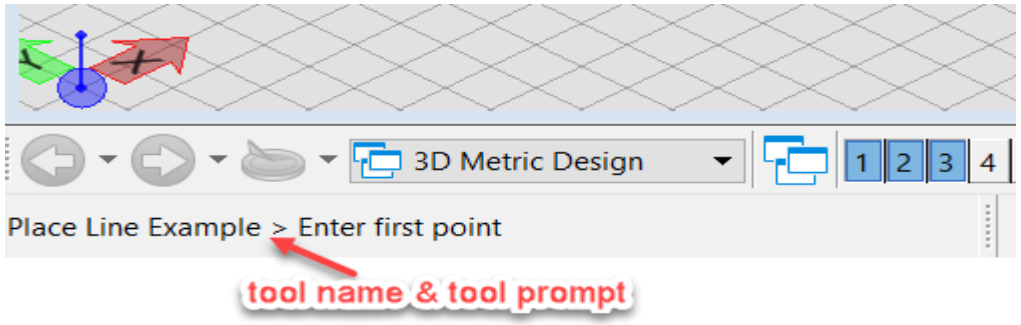
```
tool = DgnTool.GetActivePrimitiveTool()
```


DgnPrimitiveTool – construct and install the tool

- The DgnPrimitiveTool class can be used to implement a primitive command. Placement tools that don't need to locate or modify elements are good candidates for a DgnPrimitiveTool.

Generally you can define a method InstallNewInstance to be called to construct and install the tool

- *toolName, string* Id for the tool name is not used in Python implement _GetToolName instead
- toolPrompt, stringId for the initial prompt is not used in Python. Use the NotificationManager to output prompts




```
class ExampleCreateLineTool(DgnPrimitiveTool):

    def __init__ (self,  toolName, toolPrompt) : DgnPrimitiveTool.__init__ (self. toolName,
toolPrompt)
        m_self = self #Keep self reference

#The tool name will be this string in the status bar,
    def _GetToolName (WStringR name) :
        s = Wstring ("tool name")
        return s

    def InstallNewInstance (toolId, toolPrompt) :
        tool = ExampleCreateLineTool (toolId, toolPrompt)
        tool.InstallTool ()
```


Bentley

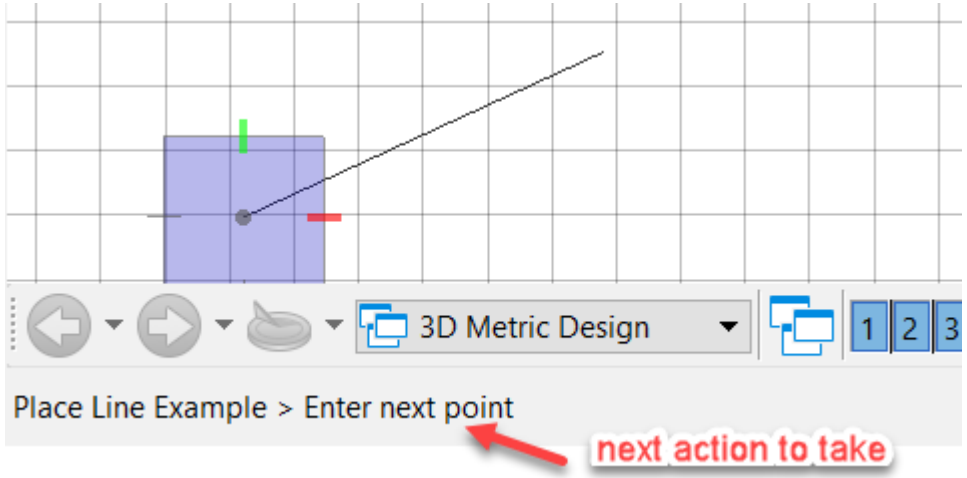
DgnPrimitiveTool – _OnDataButton and prompt user next action

The override _OnDataButton respond to mouse left button clicking

def _OnDataButton (self, ev) :

- DgnButtonEventCR ev # the button event which contains the button position/point
- In method implementation, you can use the following to prompt what action you expect user to make in next step.

NotificationManager.OutputPrompt (msgStr)




```
class ExampleCreateLineTool(DgnPrimitiveTool):

 def _OnDataButton (self, ev) :
   if len(self.m_points) == 0:
      self._BeginDynamics() # Start dynamics on first point

   self.m_points.append (ev.GetPoint ())
   self.SetupAndPromptForNextAction ()

   if len(self.m_points) < 2:
      return False

   eeh = EditElementHandle  ()
   if self.CreateElement (eeh, self.m_points):
      eeh.AddToModel ()

   self.m_points.clear ()
   self.m_points.append (ev.GetPoint ())

   return self._CheckSingleShot ()

def SetupAndPromptForNextAction ()
   msgStr = ''
   if (len(self.m_points)==1):
         msgStr = 'Pick next point:'
    else:
         msgStr = 'Pick next point or reset:'
    NotificationManager::OutputPrompt (msgStr)
```


Bentley

DgnPrimitiveTool – _OnDynamicFrame

The override _OnDynamicFrame respond to mouse move event

Def _OnDynamicFrame (self, ev) :

- DgnButtonEventCR ev # the button event which contains the button position/point

In implementation, use RedrawElems API to dynamically/temporarily draw the preview of the result of next button clicking at the same position.

```
class ExampleCreateLineTool(DgnPrimitiveTool):
    def _OnDynamicFrame (self, ev)
        tmpPts = DPoint3dArray (self.m_points)
        eeh = EditElementHandle  ()
        tmpPts.append (ev.GetPoint ())

        if not self.CreateElement (eeh, tmpPts):
            return

        RedrawElems redrawElems;
        redrawElems.SetDynamicsViews (IViewManager::GetActiveViewSet (),
ev.GetViewport ())
        redrawElems.SetDrawMode (DRAW_MODE_TempDraw)
        redrawElems.SetDrawPurpose (DrawPurpose.Dynamics)
        redrawElems.DoRedraw (eeh)
```


DgnPrimitiveTool – _OnResetButton & _OnRestartTool

The override _OnResetButton respond to mouse right button clicking

def _OnResetButton (self, ev) :

- DgnButtonEventCR ev # the button event which contains the button position/point
- Generally it only call _OnRestartTool to terminate current tool session and start a new tool session.

def _OnRestartTool ():

- Generally it just construct and install a new tool of the same type.

```
class ExampleCreateLineTool(DgnPrimitiveTool):
    def _OnRestartTool (self):
        InstallNewInstance (0, 0)

    def _OnResetButton (self, ev):
        _OnRestartTool ();
        return True
```


Bentley

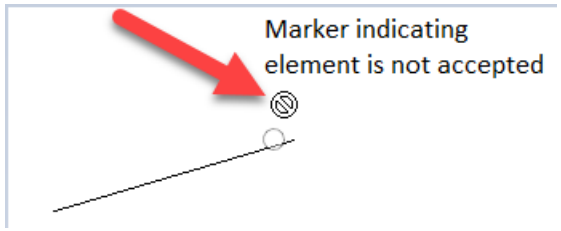
DgnElementSetTool

- Operates on ElementAgenda
  - Populated from fence, selection set, or user locate
- Key member functions
  - _NeedAcceptPoint()
    - Require data point before processing agenda
  - __WantAdditionalLocate ()
    - Require data point before processing agenda
  - _SetupAndPromptForNextAction()
    - Output context-sensitive prompt
  - _OnPostLocate()
    - Filter elements, provide reason for rejection
  - _OnElementModify()
    - Called for each element in the agenda
  - _OnModifyComplete()
    - Called after all elements modified

DgnElementSetTool – filter element that is accepted by the tool

Determine if the element that the mouse cursor hit is the kind of element the tool expect and accept.

def _OnPostLocate (self, path, cantAcceptReason);




```
class ModifyTool (DgnElementSetTool):

    def _OnPostLocate (self, path, cantAcceptReason) :
        if  not DgnElementSetTool._OnPostLocate (self, path, cantAcceptReason) :
            return False
        eh = ElementHandle(path.GetHeadElem(), path.GetRoot())
        curve = ICurvePathQuery.ElementToCurveVector(eh)

        # Accept elements that are open paths with one or more linear segments (ex. line or linestring).
        if curve==None or (not curve.IsOpenPath()):
            return False
```


Bentley

DgnElementSetTool – Dynamical preview by implementing _OnDynamicFrame yourself

```
class ModifyTool (DgnElementSetTool):

def _OnDynamicFrame (self, ev) :
        tmpPts = DPoint3dArray(self.m_points)
        eeh = EditElementHandle()

        tmpPts.append(ev.GetPoint()) # Use current button location as end point.

        if not self.CreateElement(eeh, tmpPts):
            return

        redrawElems = RedrawElems()
        redrawElems.SetDynamicsViews(IViewManager.GetActiveViewSet(), ev.GetViewport()) # Display in all
views, draws to cursor view first...
        redrawElems.SetDrawMode(DRAW_MODE_TempDraw)
        redrawElems.SetDrawPurpose(DrawPurpose.Dynamics)

        redrawElems.DoRedraw(eeh)
```


DgnElementSetTool –Select elements and process them in _OnDataButton

To select element and operate on the selected element.

def _OnDataButton(self, ev):

Call it in _OnDataButton to locate/select element to operate on
def _DoLocate (self, ev, newSearch, complexComponent):

```
class ModifyTool (DgnElementSetTool):

    def _OnDataButton(self, ev):
        if m_textBlock.IsNull () Or m_textBlock.IsEmpty ():
            #Extract the text block from the selected text element.
            path = _DoLocate (ev, true, ComponentMode::Innermost)
            if path == None :
                return False
            m_textBlock = ValidateSelection (path)
            _SetupAndPromptForNextAction ()
            _BeginDynamics ()
        else
            textElem = EditElementHandle ()
            CreateTextElement (textElem, ev)
            textElem.AddToModel ();
            IncrementText ();
        return True
```


Bentley

DgnElementSetTool – Select elements using default implementation

Don’t’ override _OnDataButton to locate element yourself, use the following to get element agenda the default implementation locates and collect for you

GetElementAgenda ()

Determine whether the selection set before tool starting is to be used by the tool.
def _AllowSelection ():

```
enum UsesSelection
    {
    USES_SS_Check,          #Active Selection Set is allowed as ElemSource
    USES_SS_Required,       #Active Selection Set is required as ElemSource
    USES_SS_None,           # Active Selection Set is not supported as ElemSource
    }
```


Bentley

DgnElementSetTool – Dynamical preview using default _OnDynamicFrame

DgnElementSetTool implement it for us which is sufficient in most cases.

What we may do is to override the following three

1. def _WantDynamics ():
   - return true if want dynamical preview
2. def _SetupForModify (self, ev, isDynamics):
   - ev, button event of a mouse move or mouse button clicking
   - isDynamics, whether this is called for dynamical preview or for finally processing selected element agenda.
   - Purpose: setup information based on ev and isDynamics which is to be used in _OnElementModify
3. def _OnElementModify (self, eeh) :
   - Purpose: use the information, may be a member variable, stored by _SetupForModify to modify some element in the selected element agenda. The modified result will be used to draw dynamically in the view. This is called for each element in the selected element agenda.

DgnElementSetTool –process selected elements in following overrides but not in _OnDataButton

DgnElementSetTool implement _OnDataButton for us which is sufficient in most cases.

What we may do is to override the following four

1. def _SetupForModify (self, ev, isDynamics)
   - This time isDynamics is false, means that be called to process element finally but not for dynamic preview
2. def _OnElementModify (self, eeh) :
   - It is called by DgnElementSetTool._ProcessAgenda.

You may also override the following to process all elements together but not one by one if it is necessary.

def _ProcessAgenda (self, ev)

```
class  ExampleModifyElementTool (DgnElementSetTool):

def  _SetupForModify (self, ev, isDynamics) :
    m_isDynamics = isDynamics
    m_ev = ev

    return True

def  _OnElementModify (self, eeh):
    locatePoint = Dpoint3d ()
    curve = ICurvePathQuery.ElementToCurveVector (eeh)
    CurveLocationDetail location;

    _GetAnchorPoint (self, locatePoint)

    # modify curve using locatePoint and m_ev to modify the curve

    # Give the element's handler a chance to update itself from the modified curve vector.
    pathEdit = eeh.GetHandler ()
    pathEdit.SetCurveVector (eeh, curve)

    # Handler didn't choose to update itself, create a new element to represent the modified
curve vector.
    DraftingElementSchema.ToElement (eeh, curve, eeh, eeh.GetModelRef ().Is3d (),
eeh.GetModelRef ())
```


Bentley

DgnElementSetTool – complete the current tool session

def _OnModifyComplete (self, ev)

- Take some final actions before exiting the current tool session. Typically, do not need to override it, just use the behavior of DgnElementSetTool itself which will re-start a new tool of the same type.

DgnElementSetTool – the usage of _WantAdditionalLocate and _NeedAcceptPoint

```
class  TwoElementTool(DgnElementSetTool):

    def _WantDynamics (self):
        return False

#if don't want extra accept point, don't override at all.  Because override and return false would cause problems.
    def _NeedAcceptPoint (self):
        return true
    def  _WantAdditionalLocate (self, ev) :
         return len (GetElementAgenda ()) < 2; # need to select another element until two element are selected
```


Bentley

DgnViewTool

- DgnViewTool can be used to implement a viewing command. A viewing command will suspend the currently active primitive command until it exits. Viewing commands should not change the file or anything else that might affect the active primitive command.

```
class       ExampleViewTool (DgnViewTool):

   def __init__  (self, toolName, toolPrompt) : DgnViewTool.__init__ (toolName,
toolPrompt)
      self.m_self = self

   def _OnDataButton (self, ev):
      vp = ev.GetViewport ()

      if vp == None
         return False  #Sub-classes may ascribe special meaning to this status, it's not
checked by DgnViewTool.

      centerPt = ev.GetPoint () #The DgnButtonEvent point is always ACTIVE coords.

      vp.ActiveToRoot(centerPt, centerPt) # Need point in ROOT coords for new zoom
center (ACTIVE != ROOT when a reference is activated).
      vp.Zoom(centerPt, 0.5, True) # Change the frustum for this viewport by tje
supplied scale factor, does not update the view.
      vp.SynchWithViewInfo(True, True) # Add entry to view undo stack so that view
previous can nr used to get back to the current view.

      info = IndexedViewSet.FullUpdateInfo()

      info.SetStartEndMsg(True) # View tools should output progress/display complete
messages...
      IViewManager.GetActiveViewSet().UpdateView(vp, DRAW_MODE_Normal,
DrawPurpose.Update, info) # Update the display by doing a full update.

      return True
```


Bentley

DgnTool Examples

- DgnViewTool
  - ExampleViewTool : ExampleViewTool.py
- DgnPrimitiveTool :
  - LineCreator : LineCreator.py
- DgnElementSetTool
  - ElementModifier : ElementModifier.py

Connect with Bentley

Connect with us

www.bentley.com

developer.bentley.com

communities.bentley.com/products/programming

Social Media

Medium.com

MicroStation

