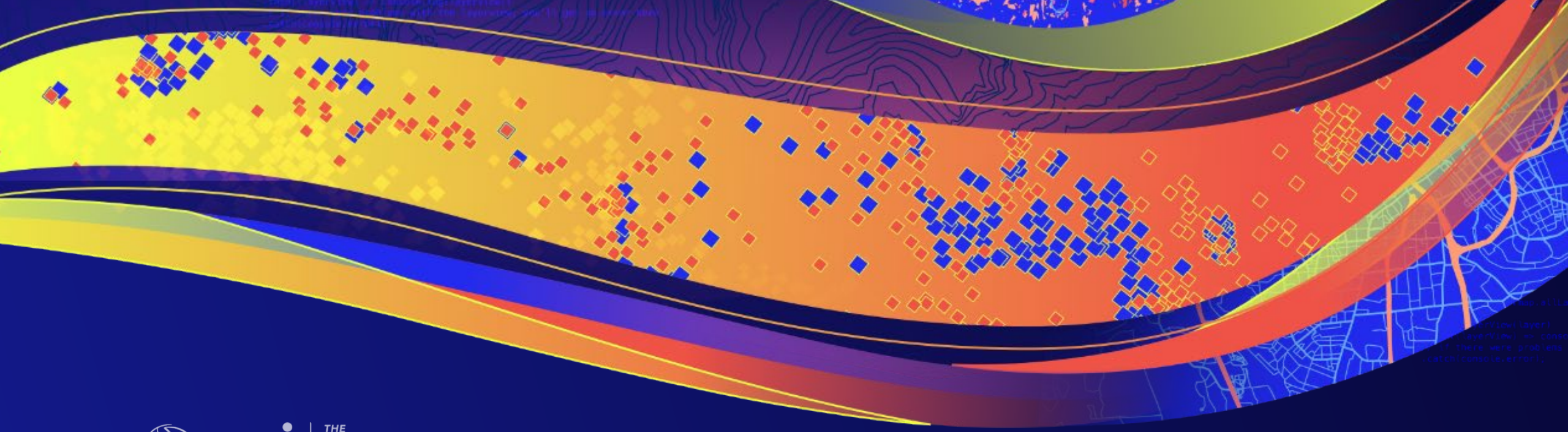


esri | THE SCIENCE OF WHERE

ESRI DEVELOPER SUMMIT 2023

ArcGIS Pro SDK for .NET: Intermediate Map Visualization Using Time API and Tray Item Template

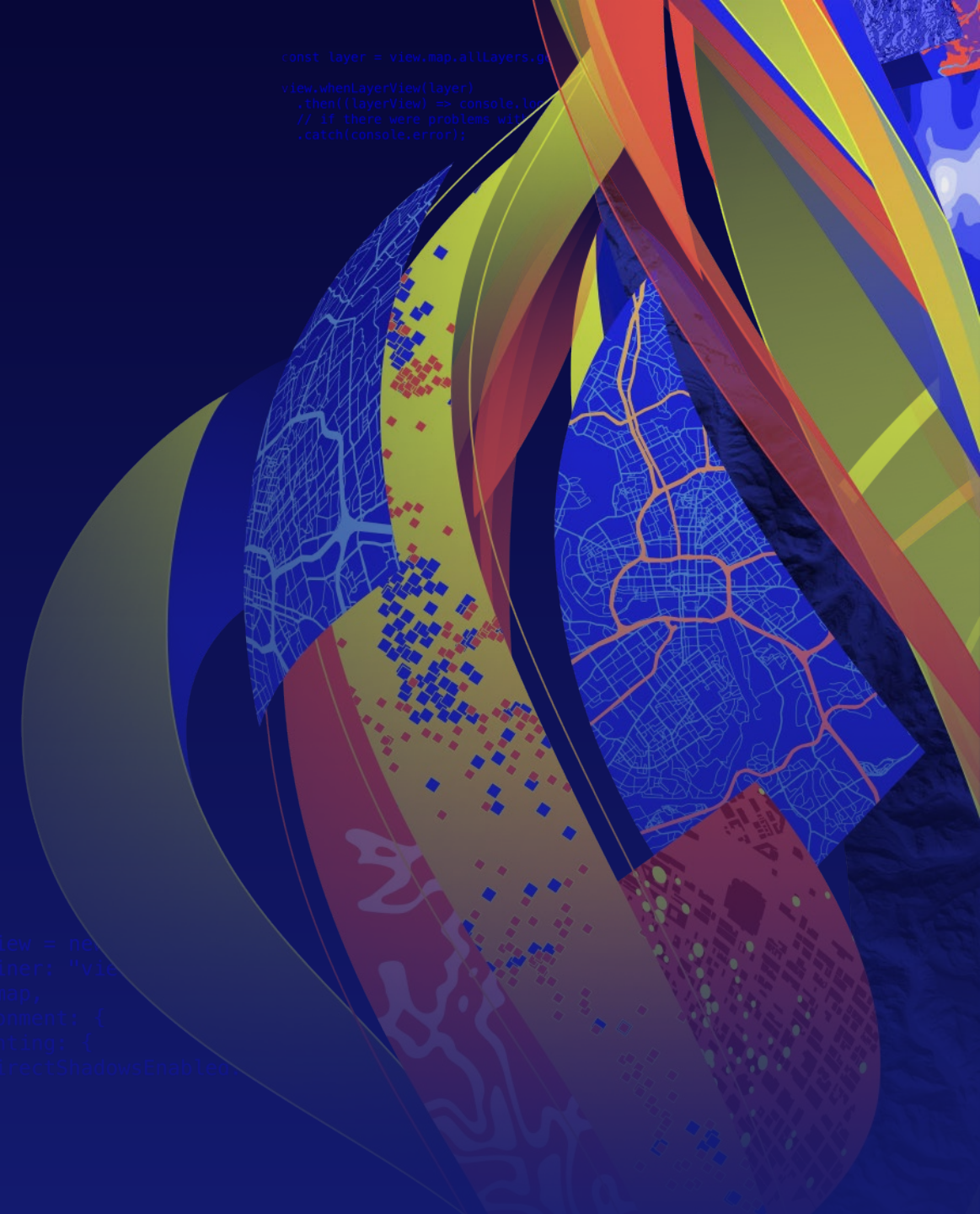
Uma Harano, Wolf Kaiser

Session Overview

- Time Filter
  - Set time properties using attribute fields
  - Set time properties using a fixed time extent
- Tray buttons
  - Map Tray button
  - Layout Tray button
- New symbology options

Time Filter

Uma Harano



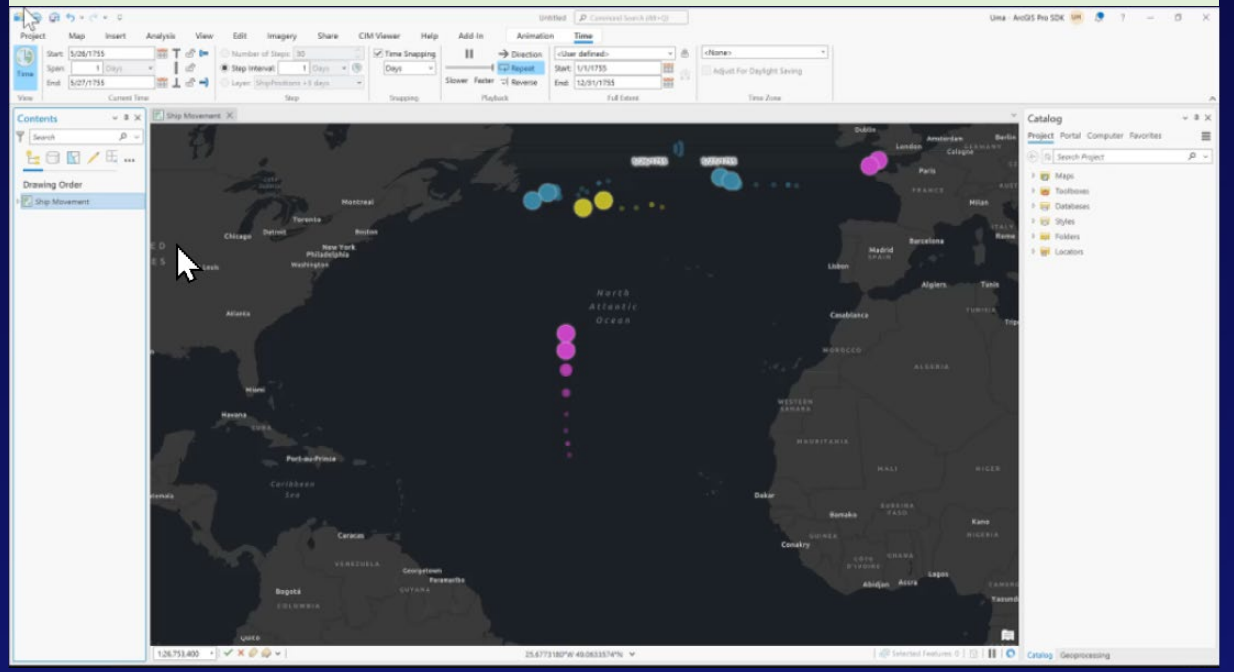
Time Filter in ArcGIS Pro

- A layer can store time values for each feature in date field(s).
  - Time aware data

- These layers can be configured to work with time by setting their temporal properties that indicate where the data exists in time.

- A Map's temporal extent is applied as a filter to all the time-aware content in the map.

Ship location data: Time enabled



Layer's Temporal Properties

- Layer's temporal properties can be configured in two ways:
  1. Use attribute fields in the data
     - Attribute fields that represent the time values for a feature is required
     - Example: Date and time of earthquakes
  2. Set layer to a fixed time extent
     - Attribute fields with dates are **not** required.
     - Example: Aerial image with an effective lifespan of 3 months.
- Pro API supports both these configuration options.

1. Use attribute fields in the data

- Layer time can be stored in a single time field or in two time fields.
  - Single time field: Each feature exists at an instant in time.
    - Ex. Earthquake
  - Two time fields: Each feature exists for a certain duration in time.
    - Ex. Start and end of a fire.

1. Use attribute fields in the data (contd)

- **"TimeParameters"** class is used to describe the time filter assigned to a layer.
  - Single field: Set the **"StartTimeFieldName"**
  - Two fields: Set the **"StartTimeFieldName"** and **"EndTimeFieldName"**

```
var tParams = new TimeParameters();
//single field - "point" in time.
tParams.StartTimeFieldName = "ConstructionDate";
//Two fields - specifies duration
tParams.StartTimeFieldName = "ConstructionDate";
tParams.EndTimeFieldName = "EndConstructionDate";
```

1. Use attribute fields in the data (contd)

- **Time extent** for the entire layer can also be specified
  - Use **"TimeRange"** property of the TimeParameters class.
    - If not specified, the layer exists within the time range of all the features.
- **"Start"** and **"End"** properties of the **"TimeRange"** class are used to specify the time extent of a layer.

```
//configure fields…
tParams.StartTimeFieldName = …
//optional – apply time extent for the layer
 tParams.TimeRange = new TimeRange();
 tParams.TimeRange.Start = new DateTime(2021, 11, 26);
tParams.TimeRange.End = new DateTime(2021, 11, 27);
```

1. Use attribute fields in the data (contd)

Apply time filter to the layer

- Apply the configured TimeParameters class to the layer using the **"SetTime"** method.
  - Accepts the configured TimeParameters class as a parameter.
  - Test the validity of the configured TimeParameters using the **"CanSetTime"** method.

```
//Testing the validity of the time filter
if (layer.CanSetTime(tParams))
{
     layer.SetTime(tParams); //apply the filter
}
```

1. Use attribute fields in the data (contd)

Configures layer time using attribute values for start and end time of a feature

```
var tParams = new TimeParameters();
tParams.StartTimeFieldName = "ConstructionBeginDate";
tParams.EndTimeFieldName = "ConstructionCompleteDate"; //optional

//optional to specify time TimeRange
tParams.TimeRange = new TimeRange();
tParams.TimeRange.Start = new Date(…);
tParams.TimeRange.End = new Date(…);

//Testing the validity of the time filter
if (layer.CanSetTime(tParams)) {
      layer.SetTime(tParams); //apply the filter
}
```

2. Set layer to a fixed time extent

- The entire dataset is configured to display during a fixed time extent.
  - Configure start and end time for a layer only.
  - Layer does not have time fields or if it does, we are not going to use them.

Mudslide: Before and After aerial Image

2. Set layer to a fixed time extent

- **Use the Time extent same as before.**
  - Use **"TimeRange" and "Start"** and **"End"** dates
    - Specifies the fixed extent of the layer

```
var tParams = new TimeParameters();
//We skip assigning the fields as we don't have any or are not using them
//Apply time extent for the layer
tParams.TimeRange = new TimeRange();
tParams.TimeRange.Start = new Date(…);
tParams.TimeRange.End = new Date(…);

//Test the validity of the time filter
if (layer.CanSetTime(tParams)) {
     //apply the filter
     layer.SetTime(tParams);
}
```

Retrieve Temporal Information from layer

- To check if a layer is time aware, use the **"IsTimeSupported"** method.

```
if (layer.IsTimeSupported()) {
  //Apply time filter
}
```

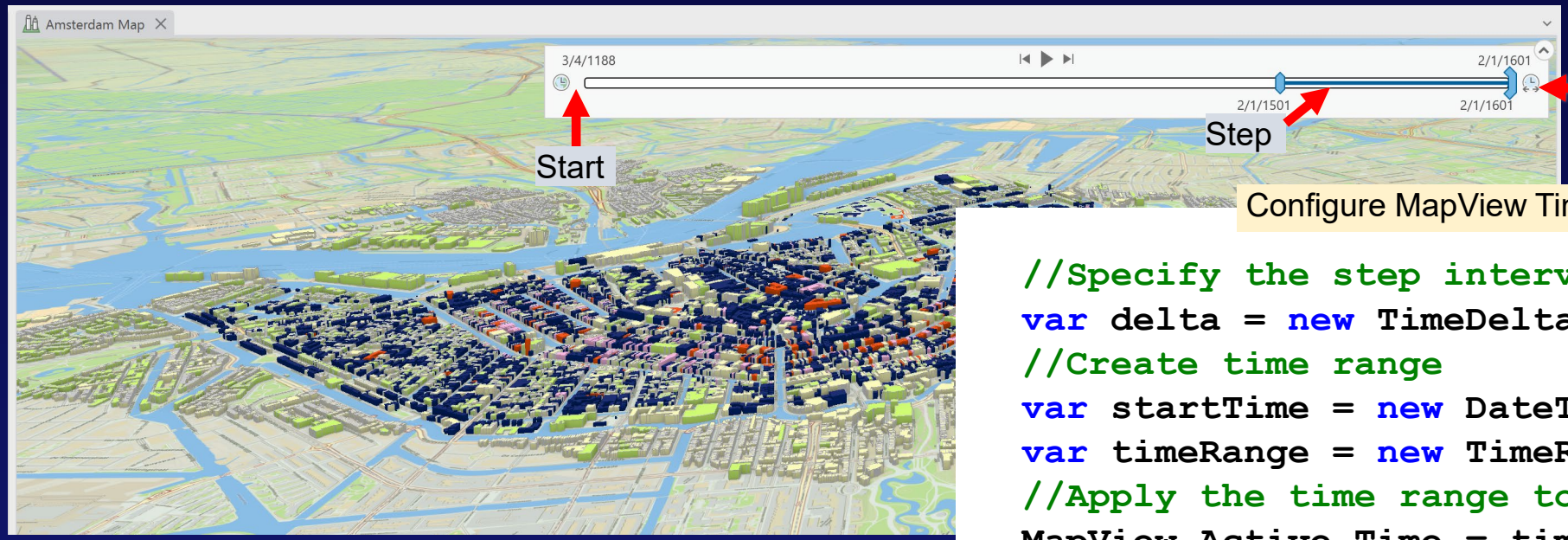
- To get the time extent of the data for the specified FieldName, use the **"GetDataTimeExtent"** method on a layer.

```
var startDate = featureLayer.GetDataTimeExtent("ConstructionDate").StartTime;
var endDate = featureLayer.GetDataTimeExtent("ConstructionDate").EndTime;
```

Configure time slider in the MapView

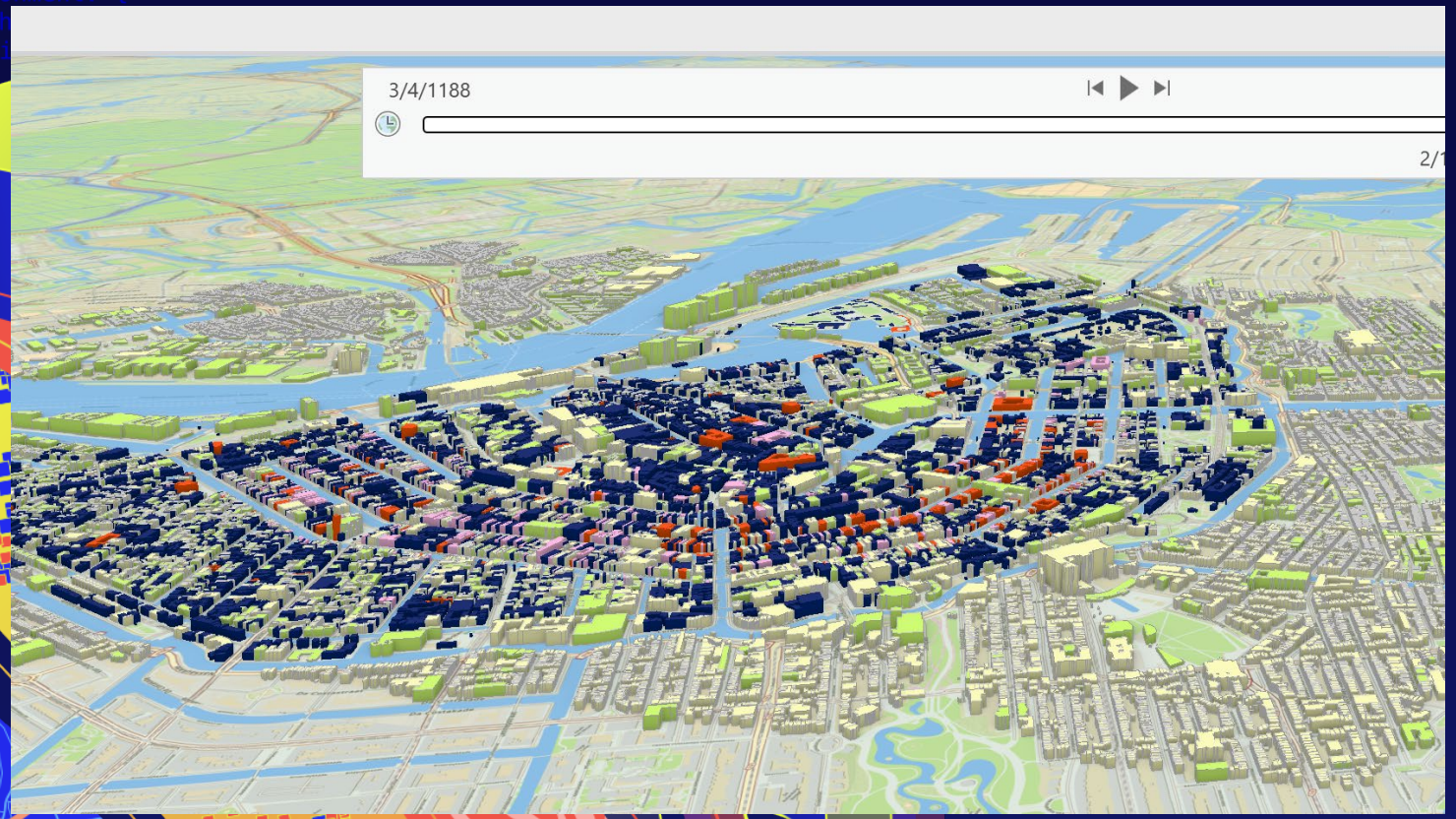
- A map view with time-aware layer(s) has an interactive time slider control.
  - allows you to explore the data.
- To configure the time slider:
  - Set the MapView's **`"Time"`** property with the **`"TimeRange"`**.
  - TimeRange is configured with the MapView's temporal extent and the Step interval.

Configure time slider in the MapView



Configure MapView Time slider

```
//Specify the step interval for visualization
var delta = new TimeDelta(100, TimeUnit.Years);
//Create time range
var startTime = new DateTime(1601, 01, 01);
var timeRange = new TimeRange(startTime, delta);
//Apply the time range to the map view
MapView.Active.Time = timeRange;
```

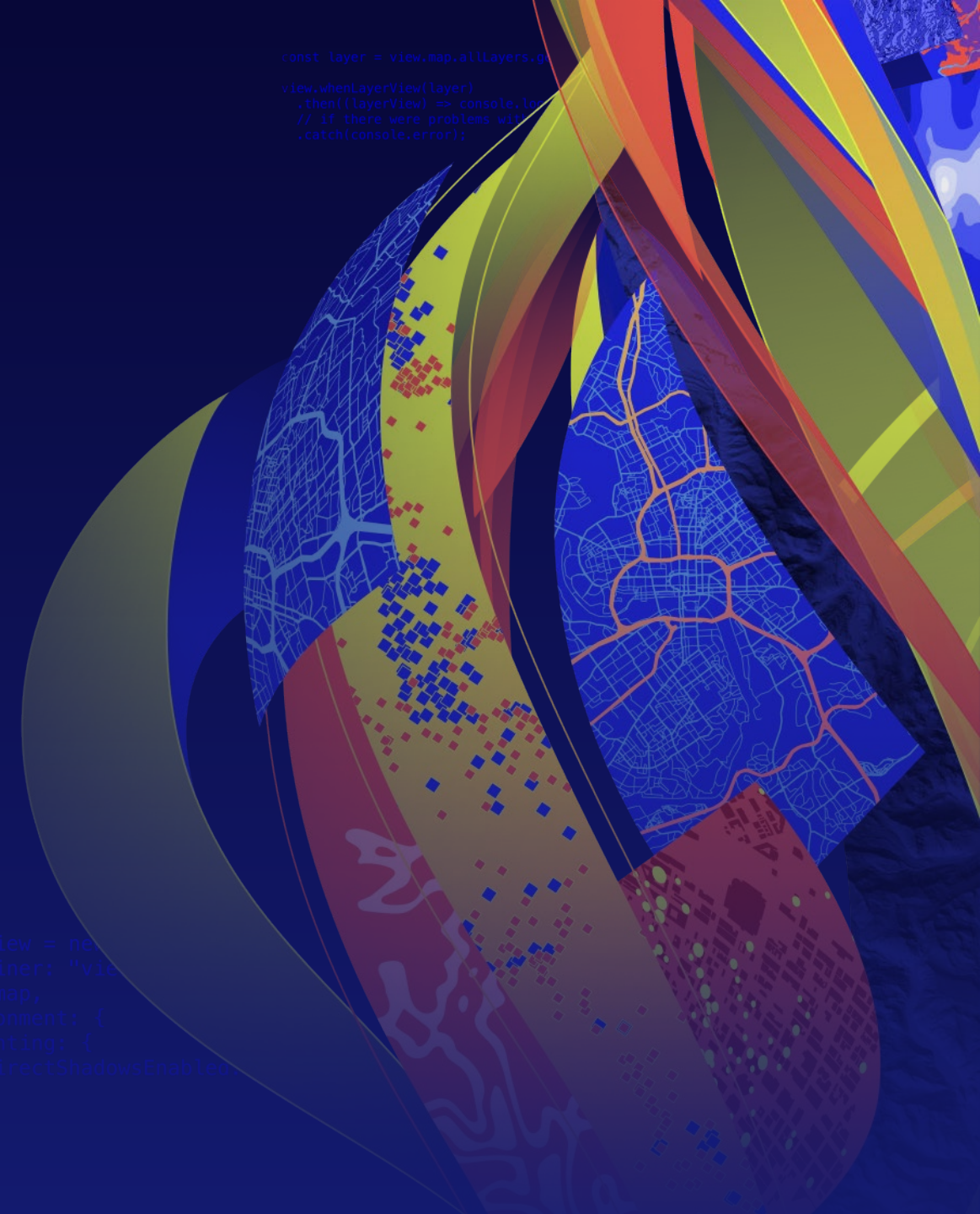


Time Filter demo

- Uma Harano

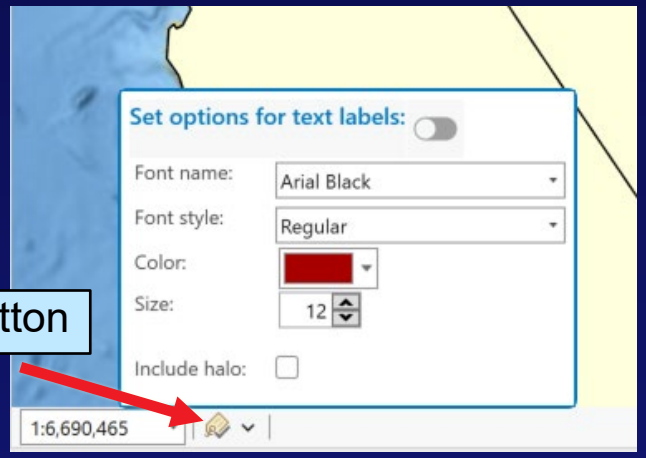
Tray buttons

Uma Harano

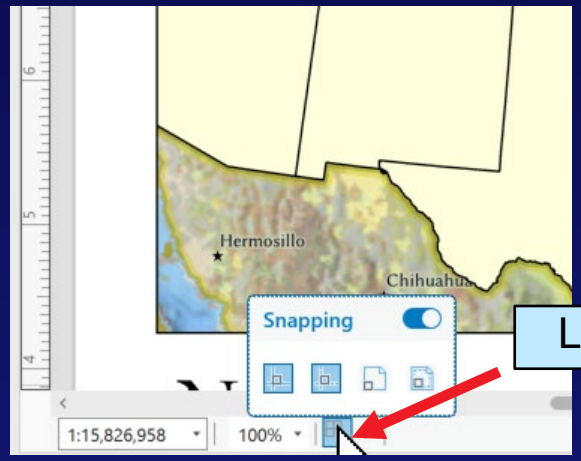


TrayButton class

- Represents a button that can be added to the ArcGIS Pro Tray.
- Base class used to create buttons that appear in the tray area of every MapView and LayoutView.
  - MapTrayButton and LayoutTrayButton derived classes.



Map Tray Button



Layout Tray Button

- Visual Studio item templates are available to create tray buttons.
  - ArcGIS Pro Map Tray Button
  - ArcGIS Pro Layout Tray Button

MapTrayButton

- A MapTrayButton is registered in the **"esri_mapping_MapTrayButtons"** category in the config.daml file.

```
<updateCategory refID="esri_mapping_MapTrayButtons">
      <insertComponent …</insertComponent>
</updateCategory>
```

- Map tray button inherits from the **"MapTrayButton"** base class.

```
internal class MapTrayLabelingButton : MapTrayButton{ }
```

LayoutTrayButton

- A LayoutTrayButton is registered in the **"esri_layout_LayoutTrayButton"** category in the config.daml file.

```
<updateCategory refID="esri_layout_LayoutTrayButtons">
      <insertComponent …</insertComponent>
</updateCategory>
```

- Layout tray button inherits from the **"LayoutTrayButton"** base class.

```
internal class LayoutTrayGuidesButton : LayoutTrayButton{ }
```

Both these base classes have the same properties and methods that can be used to configure their behavior

Tray Button types

- TrayButtons have a **"ButtonType"** property that can be set to one of the following 3 types:

1. Button
2. ToggleButton
3. PopupToggleButton

```
// internal class LayoutTrayGuidesButton : LayoutTrayButton
internal class MapTrayLabelingButton : MapTrayButton{

 protected override void Initialize(){
     ButtonType = TrayButtonType.Button;
     //ButtonType = TrayButtonType.ToggleButton;
     //ButtonType = TrayButtonType.PopupToggleButton;
 }
```

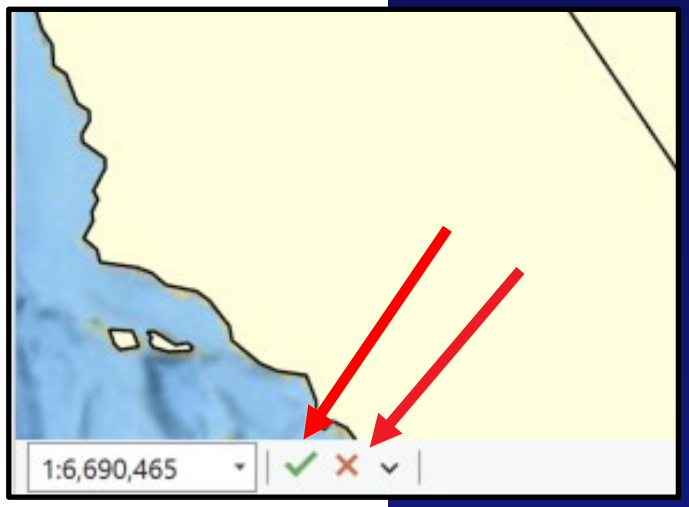
- Set ButtonType property in the **"Initialize"** override of the Tray button class.
- Each tray button type has a different configurable behavior.

ButtonType = Button

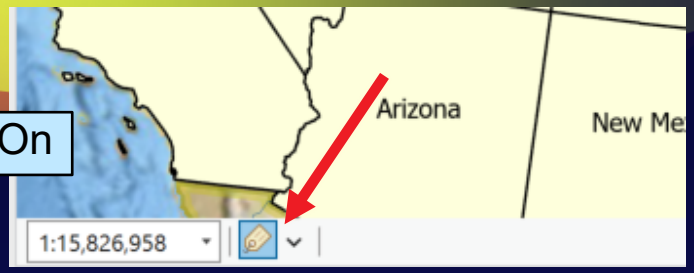
- A regular button that executes a Command when clicked.
  - Set the **"ClickCommand"** property to invoke Command when button is clicked

```
//internal class LayoutTrayGuidesButton : LayoutTrayButton
internal class MapTrayLabelingButton : MapTrayButton{

protected override void Initialize(){
      // set the button type
      ButtonType = TrayButtonType.Button;
      ClickCommand = new RelayCommand(DoClick);
  }
private void DoClick(){
// do something when the tray button is clicked
…
}
```



ButtonType = ToggleButton



- A button that has an “On” and “Off” state.
  - Toggle on/off layer labels.
- **“IsChecked”** property stores the Checked state.
  - IsChecked is set to true if button is checked, false otherwise
- Override the **“OnButtonChecked”** method to configure behavior when the button state is changed.

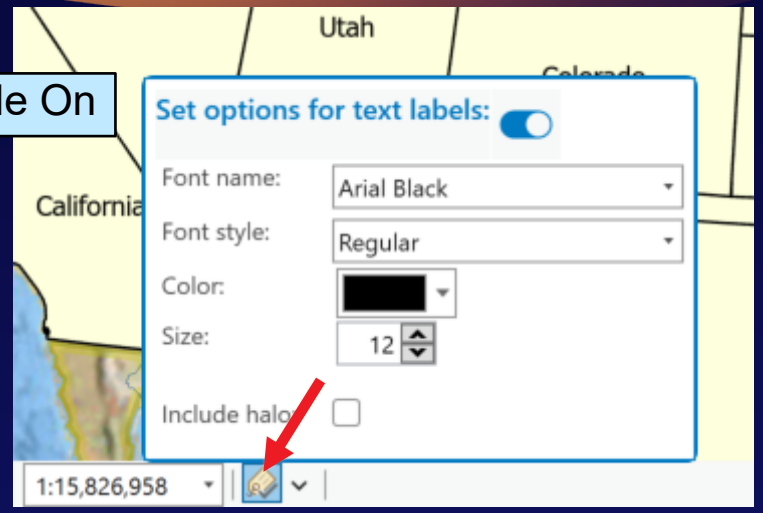
```
//internal class LayoutTrayGuidesButton : LayoutTrayButton
internal class MapTrayLabelingToggleButton : MapTrayButton{

protected override void Initialize(){
      // set the button type
      ButtonType = TrayButtonType.ToggleButton;
}
 protected override void OnButtonChecked() {
      LabelLayers(IsChecked); // do something with the checked state
 }
```

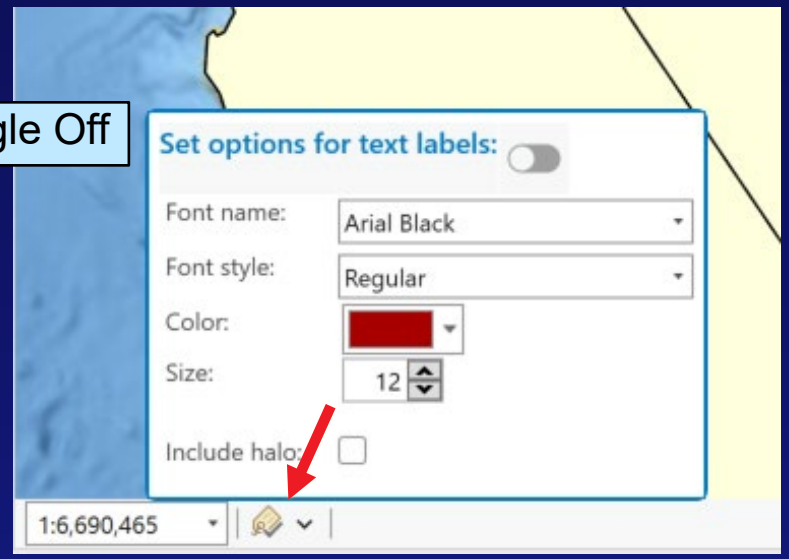
ButtonType = PopupToggleButton

- A ToggleButton that displays a Pop-up when the mouse is hovered over the button.
  - Override the "`ConstructPopupContent`" method to provide your own UI to be displayed when the button is hovered over.
  - Examples:
    - Toggle on/off layer labels.
    - Site controls on UI that configure the label font properties.

Toggle On



Toggle Off



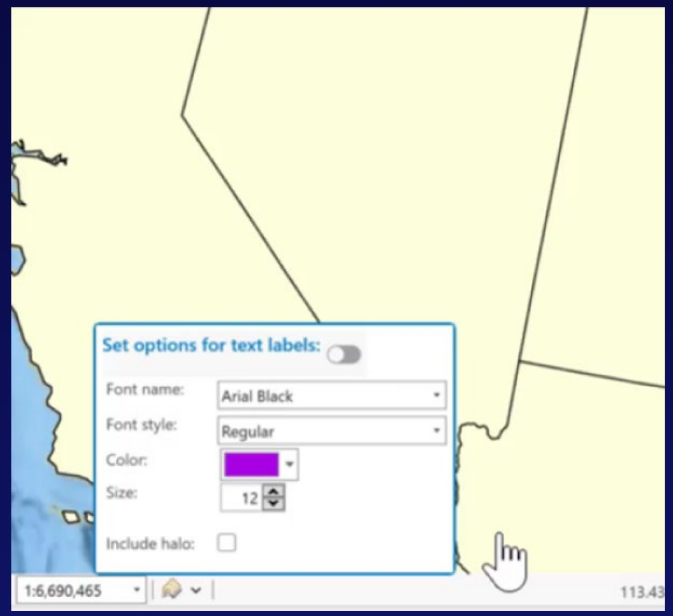
ButtonType = PopupToggleButton

```
//internal class LayoutTrayGuidesButton : LayoutTrayButton
internal class MapTrayLabelingTogglePopUpButton : MapTrayButton{

 protected override void Initialize(){
     // set the button type
     ButtonType = TrayButtonType.PopupToggleButton;
 }
 // Construct the popup view and return it.
 protected override ContentControl ConstructPopupContent() // set up the tray VM
    //_popupVM = new LayoutTrayButton1
    _popupVM = new MapTrayButton1PopupViewModel();
    // return the UI with the datacontext set
    return new MapTrayButton1PopupView() { DataContext = _popupVM };
    //return LayoutTrayButton1PopupView() {..
 }
```

Controlling AutoClose Behavior

- Default behavior of the PopupToggleButton is to auto-close when the mouse travels outside the border of the window.



- If the popup hosts other controls such as a combobox, DateTimePicker control or a ColorPicker control that launch UI not completely confined by the popup window, there can be situations where the popup closes too early.

- **`TrayButton.CanAutoClose`** - a flag on the button to control its auto-close behavior.

Controlling AutoClose Behavior

ColorPicker's PopupOpened event

TrayButton.CanAutoClose = false

Tray UI remains open.

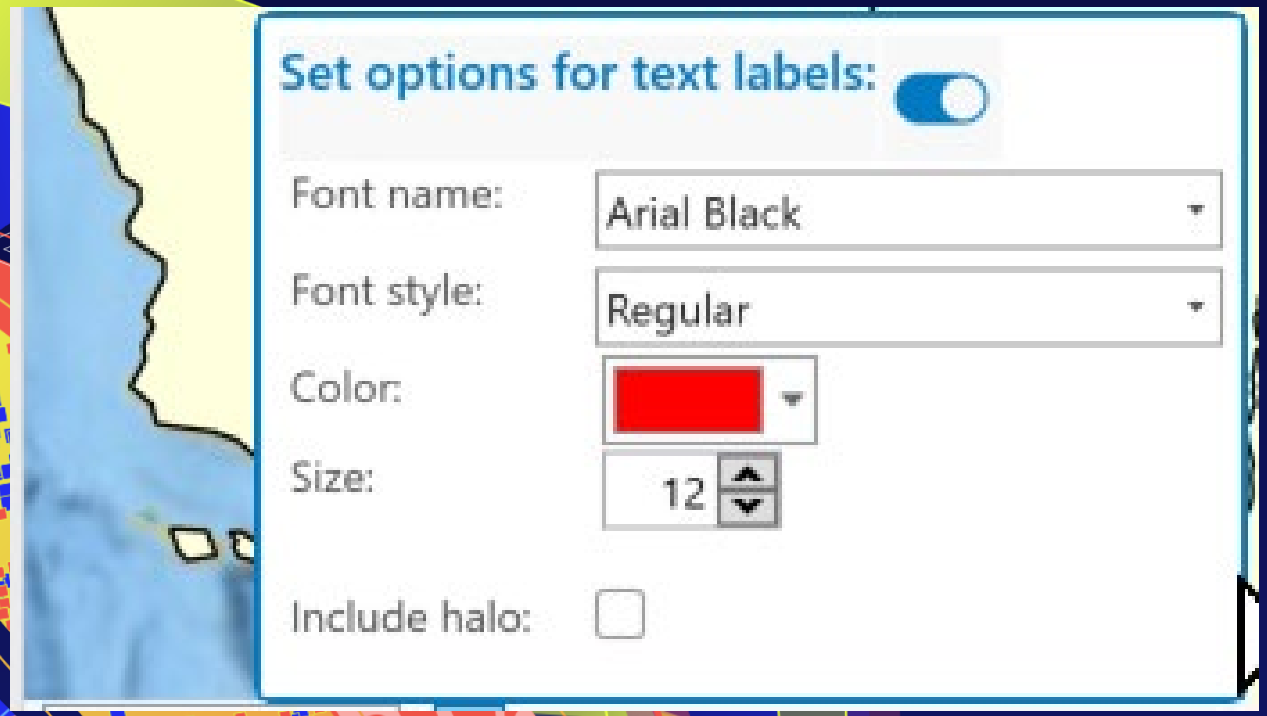
Perform other Tray UI Modification

Color Picker Popup closes.

ColorPicker's PopupClosed event

TrayButton.CanAutoClose = true

Restores Tray default Auto Close Behavior



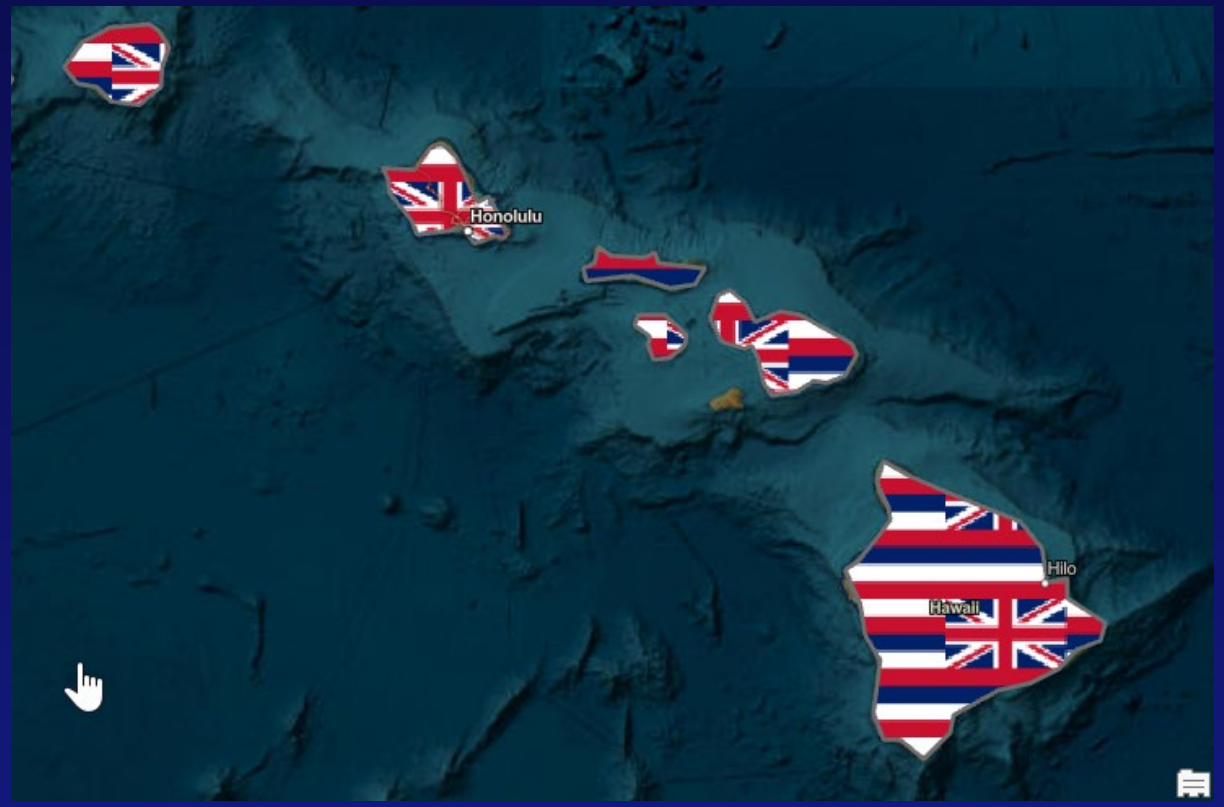
Tray Buttons Demo

- Uma Harano

New Symbology Options

Picture Fill Symbology

- Picture Fill Symbology renders images in polygons



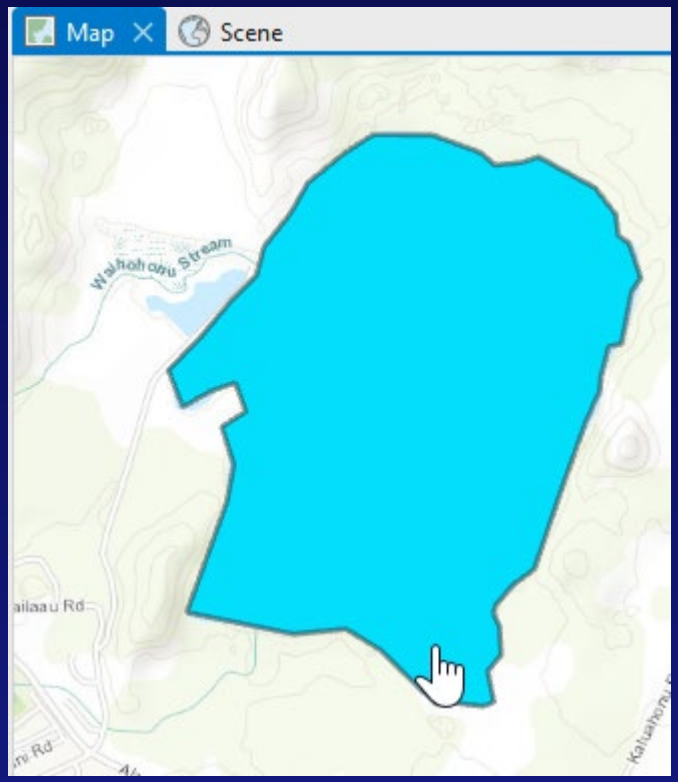
Picture Fill Symbology

- Use SymbolFactory.Instance.ConstructPictureFill

```
QueuedTask.Run<CIMPolygonSymbol>(() =>
{
 var outlineColor = CIMColor.CreateRGBColor(110, 110, 110);
 CIMStroke outline = SymbolFactory.Instance.ConstructStroke
                                    (outlineColor, 2.0, SimpleLineStyle.Solid);
 // picture Fill Symbol
 var pictureFill = SymbolFactory.Instance.ConstructPictureFill(imagePath, 64);
 // picture 'stacked' layers to create Fill Symbol
 List<CIMSymbolLayer> symbolLayers = new() {outline, pictureFill};
 return new CIMPolygonSymbol() { SymbolLayers = symbolLayers.ToArray() };
});
```

Water Animation Symbology

- Water Animation



Water Animation Symbology

- Use SymbolFactory.Instance.ConstructWaterFill

```
QueuedTask.Run<CIMPolygonSymbol>(() =>
{
  var outlineColor = CIMColor.CreateRGBColor(49, 49, 49, 50.0);
  var waterColor = CIMColor.CreateRGBColor(3, 223, 252);
  CIMStroke outline = SymbolFactory.Instance.ConstructStroke(
                    outlineColor, 2.0, SimpleLineStyle.Solid);
  var waterFill = SymbolFactory.Instance.ConstructWaterFill(
                waterColor, WaterbodySize.Small, WaveStrength.Rippled);
  List<CIMSymbolLayer> symbolLayers = new() { outline, waterFill };
  return new CIMPolygonSymbol() { SymbolLayers = symbolLayers.ToArray() };
});
```

Pen and Ink: Ripple Symbology

- Ripple Symbology

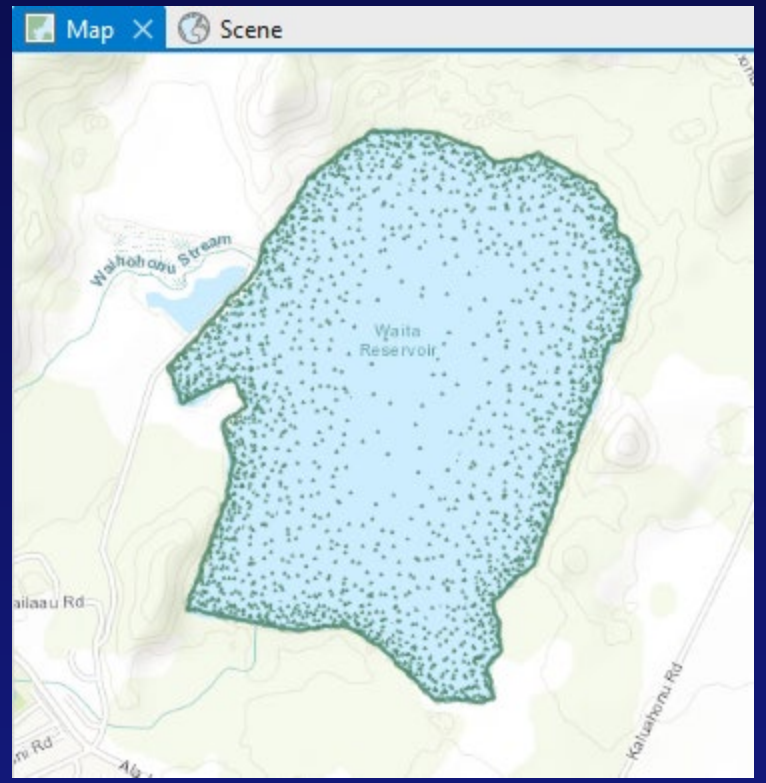
Pen and Ink: Ripple Symbology

- Use SymbolFactory.Instance. ConstructPolygonSymbolWithPenInkRipple

```
QueuedTask.Run<CIMPolygonSymbol>(() =>
{
  //Ripple pen and ink
  var inkColor = CIMColor.CreateRGBColor(13, 24, 54);
  var penInkRipple =
     SymbolFactory.Instance.ConstructPolygonSymbolWithPenInkRipple(inkColor);
  return penInkRipple;
});
```

Pen and Ink: Stipple Symbology

- Stipple Symbology

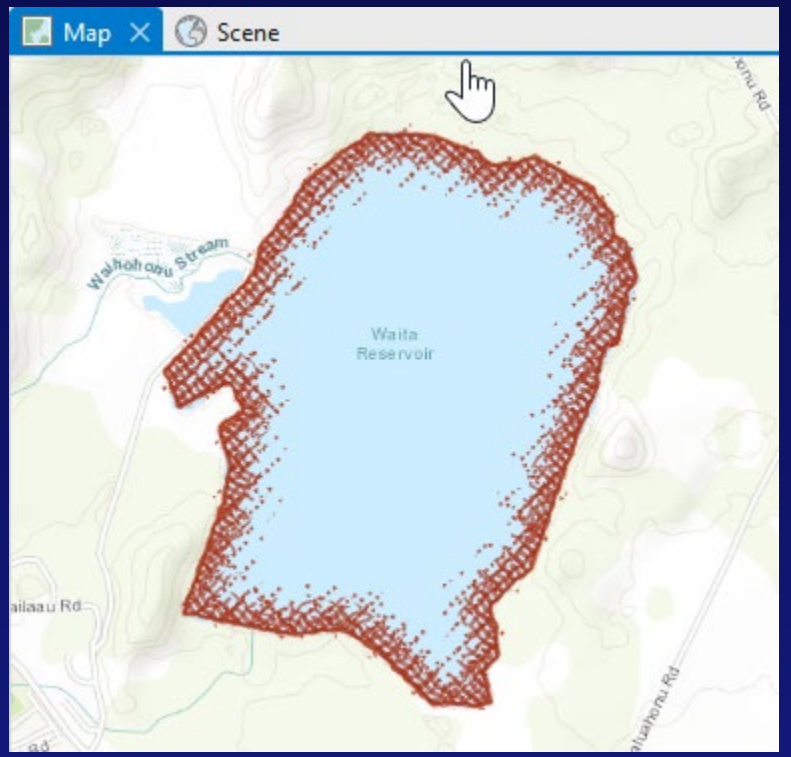


Pen and Ink: Stipple Symbology

- Use SymbolFactory.Instance.ConstructPolygonSymbolWithPenInkStipple

```
QueuedTask.Run<CIMPolygonSymbol>(() =>
{
  //Stipple pen and ink
  var inkColor = CIMColor.CreateRGBColor(78, 133, 105);
  var penInkRipple =
SymbolFactory.Instance.ConstructPolygonSymbolWithPenInkStipple(inkColor,true);
  return penInkRipple;
});
```

Pen and Ink: Cross Hatch Symbology



Pen and Ink: Cross Hatch Symbology

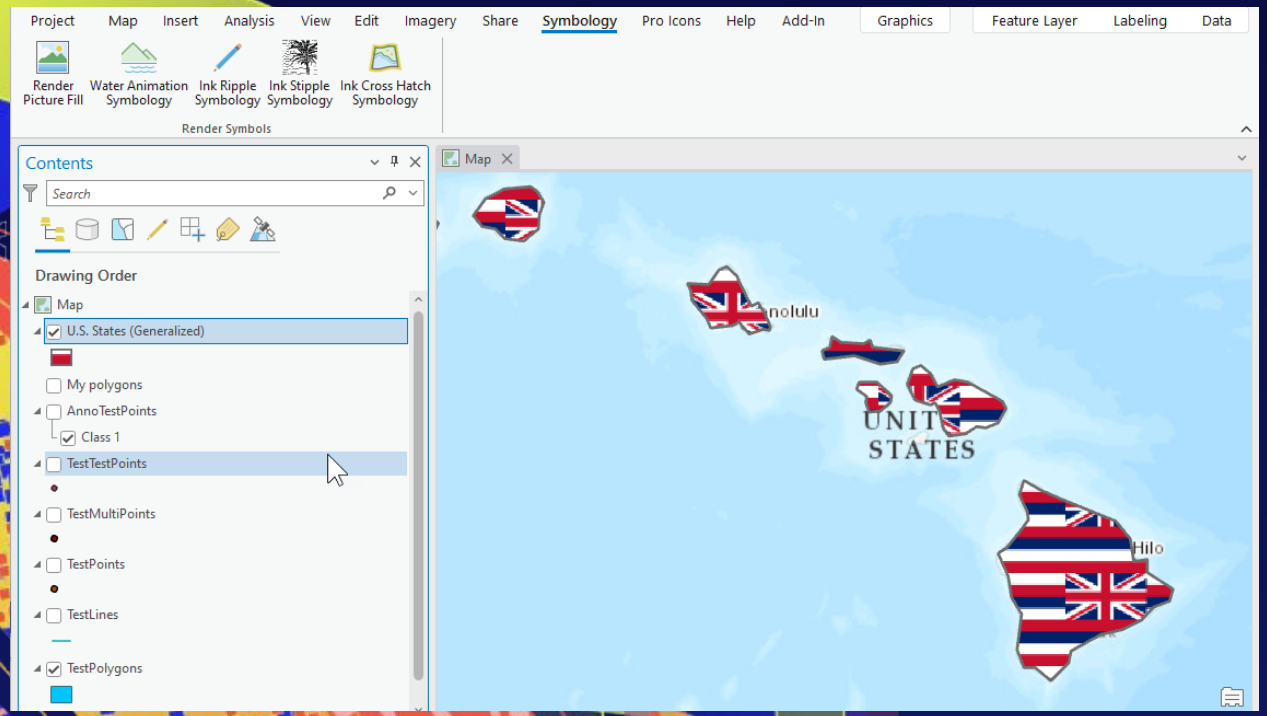
- Use SymbolFactory.Instance.ConstructPolygonSymbolWithPenInkCrossHatch

```
QueuedTask.Run<CIMPolygonSymbol>(() =>
{
  //Cross Hatch pen and ink
  var inkColor = CIMColor.CreateRGBColor(168, 49, 22);
  var penkInkCrossHatch = SymbolFactory.Instance.ConstructPolygonSymbolWithPenInkCrossHatch(inkColor,
true);
  return penkInkCrossHatch;
});
```

Using CIMSymbology in your Renderer

- User the FeatureLayer's GetRenderer and SetRenderer functions:

```
QueuedTask.Run(async () =>
{
  // Get the layer's current renderer
  CIMRenderer renderer = polygonLayer.GetRenderer();
  // Update the symbol of the current simple renderer
  var polySymbol = await CreatePictureFillPolygonAsync();
  (renderer as CIMSimpleRenderer).Symbol = polySymbol.MakeSymbolReference();
  // Update the feature layer renderer
  polygonLayer.SetRenderer(renderer);
});
```



Demo New Symbology

- FeatureTest Project

ArcGIS Pro SDK for .NET – Technical Sessions

NOTE: Session titles are led by “ArcGIS Pro SDK for .NET” in online agenda

Date	Time	Session
Tue, Mar 7	2:00 p.m. - 3:00 p.m.	ArcGIS Pro SDK for .NET: Customizing Layout
	4:00 p.m. - 5:00 p.m.	ArcGIS Pro SDK for .NET: Intermediate Editing 1
	5:30 p.m. - 6:30 p.m.	ArcGIS Pro SDK for .NET: Intermediate Editing 2
Wed, Mar 8	10:30 a.m. - 11:30 a.m.	What's New in the Geodatabase and Utility Network APIs
Thu, Mar 9	10:30 – 11:30 am	ArcGIS Pro SDK for .NET: Intermediate Data Visualization Using Table Controls
	1:00 p.m. - 2:00 p.m.	ArcGIS Pro SDK for .NET: Intermediate Map Visualization Using Time API and Tray Item Template
Fri, Mar 10	8:30 a.m – 9:30 a.m	ArcGIS Pro SDK for .NET: Parcel Fabric API
	10:00 a.m – 11:00 a.m	ArcGIS Pro SDK for .NET: COGO API and Parcel Traverse

- Detailed Agenda: https://www.esri.com/en-us/about/events/devsummit/agenda/detailed

ArcGIS Pro SDK for .NET – Demo theaters

NOTE: Session titles are led by “ArcGIS Pro SDK for .NET” in online agenda

Date	Time	Session
Wed, Mar 8	4:00 p.m. - 4:30 p.m.	ArcGIS Pro SDK for .NET: How to Run GP Tools from Your Add-in
	4:45 p.m. - 5:15 p.m.	ArcGIS Pro SDK for .NET: Customize Galleries and Comboboxes with Templates
	5:30 p.m. - 6:00 p.m.	ArcGIS Pro SDK for .NET: Customizing the Editor Attributes Pane

The Road Ahead: ArcGIS Pro

Date	Time	Session
Thu, Mar 9	2:30 p.m. - 3:30 p.m.	ArcGIS Pro: The Road Ahead

- Detailed Agenda: https://www.esri.com/en-us/about/events/devsummit/agenda/detailed

Intermediate Map Visualization Using Time API and Tray Item Template

- Questions?
    - https://github.com/Esri/arcgis-pro-sdk/wiki/Tech-Sessions#2023-palm-springs