**README FOR HABILISX**

**Note:**

Project was renamed from AddScatterViewItem to HabilisX. If any artifacts appear for that name, rename to HabilisX

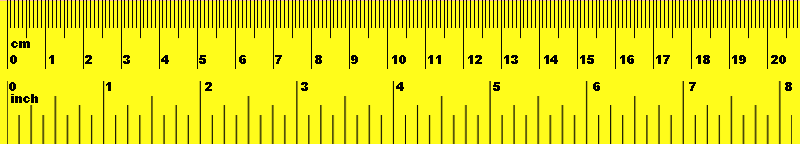
**Tools:**

**Push Pin**

Purpose: Entries that are pinned cannot move, scale, or rotate.  
  
Activation: intersection of Entry with the pin of the Push Pin.

Activation Point: (this.ActualWidth/4),(2\*this.ActualHeight/3)   
  
Default: active

**Ruler:**

Purpose: Ruler will change orientation of the Entry to the closest 90 degrees that matches the ruler, and will “push” it around.  
  
Activation: Intersection of Entry with any point.

Default: Active



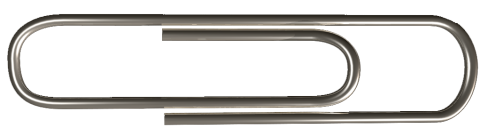
**Magnifying Glass**

Purpose: Magnifying Glass will show the specified attributes of the Entry in larger text to the right of the magnifying glass.

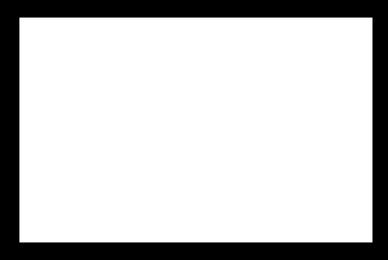
Activation: User drags a filter tile over activation point (on the “lens” ). The attribute associated with the filter tile will appear to the left when active. User entered text is not considered.   
 Activation Point: (this.ActualWidth/4), (1\*this.ActualHeight/3)

Default: Not active. Must drag filter tile to specify which attributes to show.

**Paper Clip**

Purpose: Associates Entries together in a visually organized pile. Entries clipped together, move together.

Activation: User may add filter tiles that determine which entries the paperclip interacts with. When user drags paper clip over entries, they will “stick” to the clip and become organize when user lifts finger/mouseButtonUp. Any intersection with entry will activate the functionality.   
  
Default: Active by default

**Magic Lens**

Purpose: Highlights Entries that match the specified query.

Activation: User drags filter tiles until they intersect to activate the filter. Active filters are shown to the left of the magic lens.

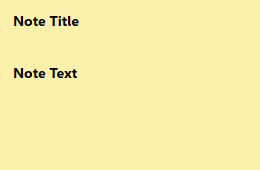
Default: Not active by default. User must specify filters to determine which entries to highlight.

**Filter Tiles**

Purpose: Used to specify which Entries the tools should interact with. User can type search queries into the text box.

Activation: Drag over Magnifying glass, Paper Clip, or Magic Lens. Release mouse.

Default: Tiles will interact with the Magnifying glass since text is ignored. To interact with anything else, a valid query must be entered.

**Note**

Purpose: Annotates an Entry with a user note

Activation: User drags note until intersecting with Entry. The note title will appear next to the entry, while the text will be hidden. User double clicks to show the note in an editable form again.

**To Add Entries to the Workspace:**

To add an Entry to Database:

In Database.cs.init():

Entry paper1 = new Entry();

paper1.addAttribute("title", title);

paper1.addAttribute("authors", authors);

paper1.addAttribute("publicationDate", pubDate);

paper1.addAttribute("pages", 1);

paper1.addAttribute("abstract", "This is the abstract");

this.addEntry(paper1);

Where:

addAttribute:   
 parameter 1: (String) attributeName  
 parameter 2: (object) attributeValue

Program will read all attributes in the database and place buttons to the screen with attributeName for generating FilterTiles. However, buttons will only work if attributeValue is of a supported Type.

Current Supported types: String, List<String>, DateTime, int, List<int>.

**To Support a New Type:**

Follow these steps as shown for List<int>:

In Database.cs

In Database.init();

paper1.addAttribute("intList", intList);

In Entry.cs:

Write new method:   
 private String printIntListAttribute(String att);

In Entry.printAttribute(att)

Add  
 else if (value is List<int>)

{

return printIntListAttribute(att);

}

In iFilter.cs:

Write new class that extends iFilter: IntListFilter.cs

In SurfaceWindow.xaml.cs

Write new method: (look at the methods for other object types, and copy and paste generic areas)  
 void AddIntListFilter\_Click(object sender, RoutedEventArgs e)

* Attach to existing or write new corresponding TextChanged Method

void IntListTile\_MouseMove(object sender, MouseEventArgs e)

In SurfaceWindow.xaml.cs.SurfaceWindow1()

Add

else if (dataSet.allAttributes[str].Equals(typeof(List<int>)))

{

myButton.Background = Brushes.Gray;

myButton.Click += new RoutedEventHandler(AddIntListFilter\_Click);

}