EXHIBIT K

```
using System;
using System.Collections;
using System.Drawing;
using System.Windows.Forms;
using UMD.HCIL.Piccolo;
using UMD.HCIL.Piccolo.Event;
using UMD.HCIL.Piccolo.Util;
using UMD.HCIL.Piccolo.Nodes;
using UMD.HCIL.PiccoloX.Nodes;

namespace LaunchPoint
{
	/// <summary>
	/// Summary description for EmailListNode.
	/// </summary>
	public class EmailListNode : PRectClip
	{
		private EmailAppNode emailAppNode = null;
		private RectangleF initialBounds = RectangleF.Empty;
		private PNode slidingNode = new PNode();
		private SizeF initialOffset = SizeF.Empty;
		private RectangleF lastLayoutBounds = RectangleF.Empty;
		private PointF mouseDownCanvasPosition = PointF.Empty;
		private SizeF initialSlidingOffset = SizeF.Empty;
		private SliderDragEventHandler sliderDragEventHandler;
		public EmailListNode(EmailAppNode parent) : base()
		{
			emailAppNode = parent;
			AddChild(slidingNode);
			//	slidingNode.AddInputEventListener(new SliderDragEventHandler());
			AddInputEventListener(sliderDragEventHandler = new SliderDragEventHandler());
			for (int i = 1; i <= 59; i++)
			{
				slidingNode.AddChild(new EmailImage(Util.GetImage(this, "LaunchPoint.images.email" + i + ".png")));
			}
		}

		public bool ShouldDrag
		{
			set { sliderDragEventHandler.ShouldDrag = value; }
```

```
        }
        public PNode GetMainDraggedNode()
        {
                return slidingNode;
        }
        /*
        public override void OnMouseDown(PInputEventArgs e)
        {
                this.mouseDownCanvasPosition = e.CanvasPosition;
                base.OnMouseDown (e);
        }

        public override void OnClick(PInputEventArgs e)
        {
                base.OnClick (e);
                if (e.PickedNode is EmailImage)
                {
                        if (PUtil.DistanceBetweenPoints(this.mouseDownCanvasPosition, e.CanvasPosition) <=
Constants.DRAG_THRESHOLD)
                        {
                                emailAppNode.SnapHighlightToLine(GetAppNodeLevelBounds((EmailImage)e.PickedNode));
                        }
                }
        }
        public override void OnMouseUp(PInputEventArgs e)
        {
                this.mouseDownCanvasPosition = PointF.Empty;
                base.OnMouseUp (e);
        }
        */


        public EmailAppNode EmailAppNode
        {
                get { return emailAppNode; }
        }
        public RectangleF InitialBounds
        {
                set
                {
```

```
                initialBounds = value;
                Bounds = value;
            }
            get { return initialBounds; }
        }
        public SizeF InitialOffset
        {
            set
            {
                initialOffset = value;
            }
            get { return initialOffset; }
        }

        public RectangleF GetIntersectingEmailItemBounds(RectangleF cursorRect)
        {
            RectangleF bestIntersection = RectangleF.Empty;
            RectangleF bestAppLevelBounds = RectangleF.Empty;
            PNodeList children = slidingNode.ChildrenReference;
            PNode each = null;
            RectangleF eachAppLevelBounds;
            RectangleF eachIntersection = RectangleF.Empty;
            // First child is header
            for (int i = 0; i < children.Count; i ++)
            {
                each = children[i];
                eachAppLevelBounds = GetAppNodeLevelBounds((EmailImage)each);
                eachIntersection = cursorRect;
                eachIntersection.Intersect(eachAppLevelBounds);
                    //PUtil.IntersectionOfRectangles(cursorRect, eachAppLevelBounds);
                if (!eachIntersection.IsEmpty)
                {
                    // Dont' want to do anything special with the header
                    //                                  if (each is EmailImage)
                    //                                  {
                    if (!bestAppLevelBounds.IsEmpty)
                    {
                        if (bestIntersection.Width * bestIntersection.Height < eachIntersection.Width *
eachIntersection.Height)
                        {
```

```
                                        bestAppLevelBounds = eachAppLevelBounds;
                                        bestIntersection = eachIntersection;
                                }
                        }
                        else
                        {
                                bestAppLevelBounds = eachAppLevelBounds;
                                bestIntersection = eachIntersection;
                        }
                        //                                              }
                }
        }
        return new RectangleF(bestAppLevelBounds.X,bestAppLevelBounds.Y - 1,
                bestAppLevelBounds.Width, bestAppLevelBounds.Height);
}

public RectangleF GetIntersectingEmailItemBounds(RectangleF cursorRect, ref PNode node)
{
        RectangleF bestIntersection = RectangleF.Empty;
        RectangleF bestAppLevelBounds = RectangleF.Empty;
        PNodeList children = slidingNode.ChildrenReference;
        PNode each = null;
        RectangleF eachAppLevelBounds;
        RectangleF eachIntersection = RectangleF.Empty;
        // First child is header
        for (int i = 0; i < children.Count; i ++)
        {
                each = children[i];
                eachAppLevelBounds = GetAppNodeLevelBounds((EmailImage)each);
                eachIntersection = cursorRect;
                eachIntersection.Intersect(eachAppLevelBounds);
                //PUtil.IntersectionOfRectangles(cursorRect, eachAppLevelBounds);
                if (!eachIntersection.IsEmpty)
                {
                        // Dont' want to do anything special with the header
                        //                                      if (each is EmailImage)
                        //                                      {
                        if (!bestAppLevelBounds.IsEmpty)
                        {
                                if (bestIntersection.Width * bestIntersection.Height < eachIntersection.Width *
```

```
eachIntersection.Height)
                                        {
                                                node = each;
                                                bestAppLevelBounds = eachAppLevelBounds;
                                                bestIntersection = eachIntersection;
                                        }
                                }
                                else
                                {
                                        bestAppLevelBounds = eachAppLevelBounds;
                                        bestIntersection = eachIntersection;
                                }
                                //                                              }
                        }
                }
                return new RectangleF(bestAppLevelBounds.X,bestAppLevelBounds.Y - 1,
                        bestAppLevelBounds.Width, bestAppLevelBounds.Height);
        }

        public RectangleF GetAppNodeLevelBounds(EmailImage image)
        {
                return LocalToParent(slidingNode.LocalToParent(image.LocalToParent(image.Bounds)));
        }
        public void CaptureInitialSlidingOffset()
        {
                PMatrix slidingMatrix = slidingNode.Matrix;
                initialSlidingOffset.Width = slidingMatrix.OffsetX;
                initialSlidingOffset.Height = slidingMatrix.OffsetY;
        }

        public void Reset(bool animate)
        {
                PMatrix slidingMatrix = slidingNode.Matrix;
                slidingMatrix.OffsetX = initialSlidingOffset.Width;
                slidingMatrix.OffsetY = initialSlidingOffset.Height;
                PMatrix matrix = Matrix;
                matrix.OffsetX = 0;
                matrix.OffsetY = 0;

                long duration = 0;
```

```
        if (animate) {
                duration = Constants.DEFAULT_ANIMATION_TIME;
                slidingNode.AnimateToMatrix(slidingMatrix, duration);
                AnimateToMatrix(matrix, duration);
        } else {
                slidingNode.Matrix = slidingMatrix;
                this.Matrix = matrix;
        }
}

public override void LayoutChildren()
{
        if (lastLayoutBounds.IsEmpty || !lastLayoutBounds.Equals(Bounds))
        {
                slidingNode.SetBounds(Bounds.X, Bounds.Y, Bounds.Width, Bounds.Height);
                float offsetY = 0;
                PNodeList children = slidingNode.ChildrenReference;
                PNode each = null;

                // Assume height of first text is the height of all text
                for (int i = 0; i < children.Count; i ++)
                {
                        each = children[i];
                        each.SetBounds(Bounds.X, Bounds.Y + offsetY, each.Width, each.Height);
                        offsetY += each.Height;
                }
                lastLayoutBounds = Bounds;
        }
}
public class EmailImage : ImageNode
{
        public EmailImage(Bitmap image) : base(image)
        {
                //      this.Pickable = false;
                //usesTransparency = true;
                //transparencyLocation = new Point(0,0);
        }
}
class SliderDragEventHandler : PDragEventHandler
```

```
{
        bool shouldDrag = true;

        public SliderDragEventHandler()
        {
                this.MinDragStartDistance = Constants.DRAG_THRESHOLD;
        }
        public bool ShouldDrag
        {
                set
                {
                        shouldDrag = value;
                }
        }
        protected override void StartDragActivity(PInputEventArgs e)
        {
                base.StartDragActivity(e);
                DragActivity.StepInterval = 10;
        }
        protected override bool ShouldStartDragInteraction(PInputEventArgs e)
        {
                return shouldDrag && base.ShouldStartDragInteraction(e);
                //return PUtil.DistanceBetweenPoints(MousePressedCanvasPoint, e.CanvasPosition) >=
Constants.DRAG_THRESHOLD;
        }
        protected override void OnStartDrag(object sender, PInputEventArgs e)
        {
                base.OnStartDrag (sender, e);
                if (e.PickedNode is EmailImage)
                {
                        DraggedNode = e.PickedNode.Parent;
                }
        }
        protected override void OnEndDrag(object sender, PInputEventArgs e)
        {
                base.OnEndDrag(sender, e);
        }
        protected override void OnDrag(object sender, PInputEventArgs e)
        {
                //e.TopCamera.Canvas.PaintImmediately();
```

```
                    SizeF s = e.GetDeltaRelativeTo(DraggedNode);
                    s = DraggedNode.LocalToParent(s);
                    // only drag vertically
                    DraggedNode.OffsetBy(0, s.Height);
                }
                public override void OnMouseUp(object sender, PInputEventArgs e)
                {
                    if (Dragging)
                    {
                        ((EmailListNode)sender).EmailAppNode.SnapObjectToHighlight(DraggedNode, true);
                    }
                    base.OnMouseUp (sender, e);
                }

            }

        }
}
```