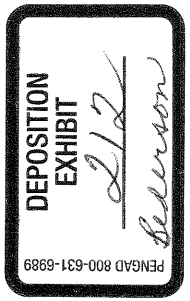
EXHIBIT L

Email.cs




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

namespace LaunchPoint
{
	/// <summary>
	/// Summary description for Email.
	/// </summary>
	public class EmailTile : AppTile
	{
		public EmailTile(string name, QuadTile quadtile, Constants.QuadPosition pos) : base(name, quadtile, pos)
		{
			this.NotificationImage = Util.GetImage(this,  "LaunchPoint.images.email_notif.png");
			ZoomInNode = new EmailAppNode(this);
		}
	}
	public class EmailAppNode: AppNode
	{
		static int MIN_MOUSE_MOVEMENT = Constants.DRAG_THRESHOLD; // 15;
		static int CONTEXT_INCR = 10;
		//static int PAN_INCR = 4;
		ContextMenu context;
		ContextMenu text;
		Timer contextTimer;
		PointF position;
		PCamera camera;
		PNode picked;
		PNode currentItem;

		SizeF cutSize = SizeF.Empty;
		SizeF copySize = SizeF.Empty;
		SizeF pasteSize = SizeF.Empty;
```

```
        SizeF delSize = SizeF.Empty;
        SizeF findSize = SizeF.Empty;
        SizeF searchSize = SizeF.Empty;

        Bitmap contextMenuImage;
        Bitmap textMenuImage;
        float minContextMenuMidSection;
        float minContextMenuCornerSection;
        float minContextMenuWidth;

        private HighlightGlass highlightglass = null;
        //private DefaultHeader emailheader = null;
        //private EmailInboxFooter inboxfooter = null;
        private DetailFooter detailfooter = null;
        private MenuImageNode contextMenuNode = null;
        private MenuImageNode textEditMenuNode = null;
        //                  private InboxList inboxlist;
        private EmailListNode inboxlist = null;
        //private EmailDetail emailDetail;
        private Cursor cursor = null;
        private EmailContentNode emailDetail = null;

        PointF lastOpenOffset;
        bool openedOnce = false;

        Cursor contextCursor;
        Cursor textCursor;

//      private EmailBlueDot emailBlueDot;
        public EmailAppNode(AppTile tile) : base(tile)
        {
                camera = this.AppTile.QuadTile.Landscape.LandscapeCamera;
                context = new ContextMenu(this, false);
                text = new ContextMenu(this, true);
                context.Scale = .7f;
                text.Scale = .7f;
                context.Visible = false;
                text.Visible = false;

                contextCursor = new Cursor();
```

```
                PointF centerContext = PUtil.CenterOfRectangle(context.Bounds);
                contextCursor.SetBounds(centerContext.X-(contextCursor.Width/2), centerContext.Y-(contextCursor.Height/2),
contextCursor.Width, contextCursor.Height);
                contextCursor.ScaleBy(1/.7f, centerContext.X, centerContext.Y);
                contextCursor.Visible = false;
                context.AddChild(contextCursor);
                //context.MouseUp += new PInputEventHandler(context_MouseUp);
                context.AddInputEventListener(new ContextDragEventHandler(this, context, text));

                textCursor = new Cursor();
                textCursor.SetBounds(centerContext.X-(contextCursor.Width/2), centerContext.Y-(contextCursor.Height/2),
contextCursor.Width, contextCursor.Height);
                textCursor.ScaleBy(1/.7f, centerContext.X, centerContext.Y);
                textCursor.Visible = false;
                text.AddChild(textCursor);
                //text.MouseUp += new PInputEventHandler(text_MouseUp);
                text.AddInputEventListener(new ContextDragEventHandler(this, text, context));

                this.TranslateBy(5, 1);
                //AddChild(highlightglass = new HighlightGlass());
                //                AddChild(inboxlist = new InboxList());
                AddChild(inboxlist = new EmailListNode(this));
                //                AddChild(emailDetail = new EmailDetail());
                AddChild(emailDetail = new EmailContentNode(this));
                AddChild(highlightglass = new HighlightGlass());
                //AddChild(emailheader = new DefaultHeader(headerImage));
                //AddChild(inboxfooter = new EmailInboxFooter());
                AddChild(cursor = new Cursor());
                AddChild(detailfooter = new DetailFooter());
//              emailBlueDot = new EmailBlueDot(this);
                // Do this to get the inherited functionality and add to node
//              BlueDot = emailBlueDot;
                InitializeMenu();

                cursor.InitialOffset = new SizeF(50, cursor.InitialOffset.Height);
                cursor.TranslateBy(50, 0);
        }

        class ContextDragEventHandler : PDragEventHandler
        {
```

Email.cs

```
ContextMenu mainMenu, hiddenMenu;
EmailAppNode email;
PointF lastPosition;
public ContextDragEventHandler(EmailAppNode email, ContextMenu mainMenu, ContextMenu hiddenMenu)
{
        this.MinDragStartDistance = Constants.DRAG_THRESHOLD;
        this.mainMenu = mainMenu;
        this.hiddenMenu = hiddenMenu;
        this.email = email;
}
protected override void OnDrag(object sender, PInputEventArgs e) {
        SizeF canvasDelta = e.CanvasDelta;
        SizeF menuDelta = mainMenu.GlobalToLocal(canvasDelta);
        email.highlightglass.TranslateBy(0, canvasDelta.Height);
        email.cursor.TranslateBy(0, canvasDelta.Height);
        mainMenu.TranslateBy(menuDelta.Width, menuDelta.Height);
        hiddenMenu.TranslateBy(menuDelta.Width, menuDelta.Height);
        mainMenu.ResetFadeOutTime();
        email.lastOpenOffset = mainMenu.Offset;
        e.Handled = true;
}
public override void OnMouseDown(object sender, PInputEventArgs e)
{
        base.OnMouseDown (sender, e);

        lastPosition = e.CanvasPosition;
}

public override void OnMouseUp(object sender, PInputEventArgs e)
{
        if (PUtil.DistanceBetweenPoints(e.CanvasPosition, lastPosition) < Constants.DRAG_THRESHOLD)
        {
                switch ((String)e.PickedNode.Tag)
                {
                        case "zoomout":
                                mainMenu.ResetFadeOutTime();
                                email.Back();
                                break;
                        case "zoomin":
                                mainMenu.ResetFadeOutTime();
```

```
                                        email.Forward();
                                        break;
                                default:
                                        if (mainMenu.CenterRect.Contains(e.CanvasPosition)) {
                                                email.ToggleMenus(false);
                                        }
                                        break;
                        }
                }
                else
                {
                        RectangleF itemBounds = email.SnapHighlight();
                        if (!itemBounds.IsEmpty)
                        {
                                itemBounds = email.LocalToGlobal(itemBounds);
                                itemBounds = e.Camera.ViewToLocal(itemBounds);
                                PointF centerItemBounds = PUtil.CenterOfRectangle(itemBounds);
                                PMatrix matrix = mainMenu.Matrix;
                                matrix.OffsetY = centerItemBounds.Y - mainMenu.GlobalBounds.Height /2;
                                mainMenu.AnimateToMatrix(matrix, Constants.DEFAULT_ANIMATION_TIME);
                                hiddenMenu.AnimateToMatrix(matrix, Constants.DEFAULT_ANIMATION_TIME);
                        }
                }
                base.OnMouseUp(sender, e);
        }
}

public void context_MouseUp(object sender, PInputEventArgs e)
{
        switch ((String)e.PickedNode.Tag)
        {
                case "zoomout":
                        context.ResetFadeOutTime();
                        Back();
                        break;
                case "zoomin":
                        context.ResetFadeOutTime();
                        Forward();
                        break;
        }
```

```
}

public void text_MouseUp(object sender, PInputEventArgs e)
{
        switch ((String)e.PickedNode.Tag)
        {
                case "zoomout":
                        text.ResetFadeOutTime();
                        Back();
                        break;
                case "zoomin":
                        text.ResetFadeOutTime();
                        Forward();
                        break;
        }
}

public override void OnMouseDown(UMD.HCIL.Piccolo.Event.PInputEventArgs e)
{
        position = e.CanvasPosition;
        picked = e.PickedNode;

        if (picked is EmailListNode.EmailImage || picked is HighlightGlass)
        {
                if (contextTimer == null)
                {
                        contextTimer = new Timer();
                        contextTimer.Interval = Constants.TAP_HOLD_DELAY; //400;
                        contextTimer.Tick += new EventHandler(RaiseContextMenu);
                }

                contextTimer.Start();
                base.OnMouseDown (e);
        }
}

public override void OnMouseUp(UMD.HCIL.Piccolo.Event.PInputEventArgs e)
{
        inboxlist.ShouldDrag = true;
        if (picked is EmailListNode.EmailImage || picked is HighlightGlass)
```

```
                {
                        contextTimer.Stop();
                        if (context.AnimatingPathOpen)
                        {
                                context.Close();
                        }
                        else if (context.Parent != null)
                        {
                                if (picked is EmailListNode.EmailImage) currentItem = picked;
                                lastOpenOffset = context.Offset;
                                openedOnce = true;
                        }
                        base.OnMouseUp (e);
                }
        }

        protected void RaiseContextMenu(object sender, EventArgs eArgs)
        {
                if (picked is EmailListNode.EmailImage || picked is HighlightGlass)
                {
                        contextTimer.Stop();

                        if (PUtil.DistanceBetweenPoints(PCanvas.CURRENT_PCANVAS.PointToClient(Control.MousePosition),
position) <= MIN_MOUSE_MOVEMENT)
                        {
                                if (context.Parent == null)
                                {
                                        camera.AddChild(context);
                                }

                                if (text.IsOpen)
                                {
                                        text.Close();
                                }

                                PointF centerContext = position;
                                if (picked is EmailListNode.EmailImage || picked is HighlightGlass)
                                {
                                        RectangleF itemBounds = picked.Bounds;
                                        itemBounds = picked.LocalToGlobal(itemBounds);
```

```
                                        itemBounds = camera.ViewToLocal(itemBounds);
                                        PointF itemCenter = PUtil.CenterOfRectangle(itemBounds);
                                        centerContext = new PointF(position.X, itemCenter.Y);
                                    }
                                    inboxlist.ShouldDrag = false;
                                    context.SetOffset(centerContext.X - context.GlobalBounds.Width / 2, centerContext.Y -
context.GlobalBounds.Height /2);
                                    context.AnimateOpen(true).ActivityFinished = new ActivityFinishedDelegate(OpenPathFinished);
                                }
                            }
                        }

                        public override void KeyPressed(Keys keyCode)
                        {
                            SetPickedNodeToPosition();
                            switch (keyCode)
                            {
                                case Keys.Right:
                                    if (context.Visible || text.Visible)
                                    {
                                        MoveMenus(CONTEXT_INCR, 0);
                                    }
                                    break;
                                case Keys.Left:
                                    if (context.Visible || text.Visible)
                                    {
                                        MoveMenus(-CONTEXT_INCR, 0);
                                    }

                                    break;
                                case Keys.Up:
                                    if (context.Visible || text.Visible)
                                    {
                                        MoveMenus(-1);
                                    }
                                    else
                                    {
                                        if (state == Constants.AppState.OVERVIEW) PanEmails(1);
                                    }
                                    break;
```

```
                case Keys.Down:
                        if (context.Visible || text.Visible)
                        {
                                MoveMenus(1);
                        }
                        else
                        {
                                if (state == Constants.AppState.OVERVIEW) PanEmails(-1);
                        }
                        break;
                case Keys.Enter:
                case Keys.D5:
                        ToggleMenus(false);
                        break;
        }
}

public void ToggleMenus(bool animateOpen) {
        if (context.IsOpen) {
                if (text.Parent == null) camera.AddChild(text);
                text.Bounds = context.Bounds;
                text.Offset = context.Offset;
                context.TerminateAllActivities();
                text.AnimateOpen(false);
                context.FadeClosed(0);
        }
        else if (text.IsOpen) {
                text.Close();
        }
        else {
                if (context.Parent == null)
                {
                        camera.AddChild(context);
                }

                if (openedOnce)
                {
                        context.Offset = lastOpenOffset;
                        text.Offset = lastOpenOffset;
                }
```

```
                                else
                                {
                                        openedOnce = true;
                                        RectangleF b = this.LocalToGlobal(Bounds);
                                        b = camera.ViewToLocal(b);
                                        PointF center = PUtil.CenterOfRectangle(b);
                                        context.SetOffset(center.X - context.GlobalBounds.Width / 2-10, center.Y - context.GlobalBounds.Height
/2-2);
                                        text.Offset = context.Offset;
                                        lastOpenOffset = context.Offset;
                                        SetPickedNodeToPosition();

                                        b = picked.LocalToGlobal(picked.Bounds);
                                        b = camera.ViewToLocal(b);
                                        PointF centerItemBounds = PUtil.CenterOfRectangle(b);
                                        //context.OffsetY = centerItemBounds.Y - context.GlobalBounds.Height /2;
                                        //text.OffsetY = context.OffsetY;

                                        OpenPathFinished(null);
                                }

                                if (animateOpen) {
                                        context.AnimateOpen(false);
                                } else {
                                        context.Open();
                                }


                                //if (context.Parent != null && openedOnce)
                                //{
                                //      context.Offset = lastOpenOffset;
                                //      text.Offset = lastOpenOffset;
                                //      context.Open();
                                //}
                        }
                }

                public void MoveMenus(int increment)
                {
                        if (DelayMenus() && currentItem != null)
```

```
        {
            PNode listnode;
            if (state == Constants.AppState.OVERVIEW)
            {
                listnode = inboxlist.GetMainDraggedNode();
            }
            else
            {
                listnode = emailDetail.GetMainDraggedNode();
            }
            int index = listnode.IndexOfChild(currentItem) + increment;
            if (index >= 0 && index < listnode.ChildrenCount)
            {
                currentItem = listnode[index];
                RectangleF itemBounds = currentItem.Bounds;
                itemBounds = currentItem.LocalToGlobal(itemBounds);
                itemBounds = camera.ViewToLocal(itemBounds);
                PointF itemCenter = PUtil.CenterOfRectangle(itemBounds);
                context.OffsetY = itemCenter.Y - context.GlobalBounds.Height /2;
                text.OffsetY = itemCenter.Y - text.GlobalBounds.Height /2;
                RectangleF bnds;
                if (state == Constants.AppState.OVERVIEW)
                {
                    bnds = inboxlist.GetAppNodeLevelBounds((EmailListNode.EmailImage)currentItem);
                }
                else
                {
                    bnds = emailDetail.GetAppNodeLevelBounds((EmailListNode.EmailImage)currentItem);
                }
                SnapHighlightToLine(bnds, false);
                lastOpenOffset = context.Offset;
            }
        }
    }

    public void MoveMenus(float dx, float dy)
    {
        if (this.DelayMenus())
        {
            context.TranslateBy(dx, dy);
```

```
                text.TranslateBy(dx, dy);
                lastOpenOffset = context.Offset;
            }
        }

        public void PanEmails(int increment)
        {
            inboxlist.GetMainDraggedNode().TranslateBy(0, increment * highlightglass.Height);
        }

        public bool DelayMenus()
        {
            bool menuOpen = false;
            if (context.IsOpen)
            {
                menuOpen = true;
                if (context.FadingOut)
                {
                    context.Open();
                }
                else
                {
                    context.ResetFadeOutTime();
                }
            }
            else if (text.IsOpen)
            {
                menuOpen = true;
                if (text.FadingOut)
                {
                    text.Open();
                }
                else
                {
                    text.ResetFadeOutTime();
                }
            }
            return menuOpen;
        }
```

Email.cs

```
protected void OpenPathFinished(PActivity activity)
{
	if (picked is EmailListNode.EmailImage)
	{
		if (state == Constants.AppState.OVERVIEW)
		{
			SnapHighlightToLine(inboxlist.GetAppNodeLevelBounds((EmailListNode.EmailImage)picked));
		}
		else if (state == Constants.AppState.DETAIL)
		{
			SnapHighlightToLine(emailDetail.GetAppNodeLevelBounds((EmailListNode.EmailImage)picked));
		}
	}
}

public override void Back()
{
	// If we're already at overview, go to quad tile
	if (state == Constants.AppState.OVERVIEW)
	{
		if (context.Parent != null)
		{
			context.Close();
			camera.RemoveChild(context);
		}
		if (text.Parent != null)
		{
			text.Close();
			camera.RemoveChild(text);
		}

		openedOnce = false;
		inboxlist.Reset(false);
		this.highlightglass.Reset(false);
		cursor.Reset(false);
		appTile.QuadTile.Landscape.AdjustToCurrent(true);
	}
	else if (state == Constants.AppState.DETAIL)
	{
		AppTile.QuadTile.Landscape.ShellLevel = Constants.ShellLevel.APPLICATION;
```

```
                this.State = Constants.AppState.OVERVIEW;

                emailDetail.Reset(true);
                //(emailDetail.Reset(true)).ActivityFinished = new ActivityFinishedDelegate(BackToOverviewFinished);
                inboxlist.Visible = true;
                inboxlist.Reset(true);
                contextCursor.Visible = false;
                textCursor.Visible = false;
        }
}

public void BackToOverviewFinished(PActivity activity)
{
        //activity.Terminate();
        //SetPickedNodeToPosition();
}

public void SetPickedNodeToPosition()
{
        RectangleF cb = context.LocalToGlobal(context.Bounds);
        SetPickedNodeToPosition(PUtil.CenterOfRectangle(cb));
}

public void SetPickedNodeToPosition(PointF point)
{
        PNode listnode;
        if (this.state == Constants.AppState.OVERVIEW)
        {
                listnode = inboxlist.GetMainDraggedNode();
        }
        else
        {
                listnode = emailDetail.GetMainDraggedNode();
        }

        foreach (PNode node in listnode)
        {
                RectangleF itemBounds = node.Bounds;
                itemBounds = node.LocalToGlobal(itemBounds);
                itemBounds = camera.ViewToLocal(itemBounds);
```

```
                //RectangleF cb = context.LocalToGlobal(context.Bounds);
                if (itemBounds.Contains(point)) //PUtil.CenterOfRectangle(cb)))
                {
                        picked = node;
                        currentItem = picked;
                }
        }
}

public override void Forward()
{
        //base.Forward ();

        if (state == Constants.AppState.OVERVIEW)
        {
                state = Constants.AppState.DETAIL;
                AppTile.QuadTile.Landscape.ShellLevel = Constants.ShellLevel.OBJECT;

                PMatrix emailDetailMatrix = emailDetail.Matrix;
                emailDetailMatrix.OffsetX = 0;
                emailDetailMatrix.OffsetY = 0; //emailheader.Height/2;
                PMatrix inboxListMatrix = inboxlist.Matrix;
                inboxListMatrix.OffsetX = 0;
                inboxListMatrix.OffsetY = - (Bounds.X /2); //- (Bounds.X -emailheader.Height/2);

                inboxlist.AnimateToMatrix(inboxListMatrix, Constants.DEFAULT_ANIMATION_TIME);
                emailDetail.AnimateToMatrix(emailDetailMatrix, Constants.DEFAULT_ANIMATION_TIME);
                //(emailDetail.AnimateToMatrix(emailDetailMatrix, Constants.DEFAULT_ANIMATION_TIME)).ActivityFinished
                //        = new ActivityFinishedDelegate(ForwardToDetailFinished);
                inboxlist.AnimateToMatrix(emailDetailMatrix, Constants.DEFAULT_ANIMATION_TIME);

                contextCursor.Visible = true;
                textCursor.Visible = true;
                highlightglass.Visible = true;
        }
}

public void ForwardToDetailFinished(PActivity activity)
{
```

```
                PointF pt = highlightglass.LocalToGlobal(highlightglass.Bounds.Location);
                pt = camera.ViewToLocal(pt);
                if (pt.Y < 170)
                {
                        this.picked = emailDetail.GetMainDraggedNode()[0];
                        this.currentItem = picked;
                        RectangleF bnds = emailDetail.GetAppNodeLevelBounds((EmailListNode.EmailImage)currentItem);
                        //bnds = new RectangleF(bnds.X, 170, bnds.Width, bnds.Height);
                        this.SnapHighlightToLine(bnds, true);
                }
        }

        protected void InitializeMenu()
        {
                contextMenuImage = Util.GetImage(this, "LaunchPoint.images.contextmenu.png");
                textMenuImage = Util.GetImage(this, "LaunchPoint.images.textentrymenu.png");
                // HardMenuPrep();
        }
        private void HardMenuPrep()
        {

                Graphics graphics = Graphics.FromImage(new Bitmap(1,1));
                cutSize = graphics.MeasureString("cut", Constants.TAHOMA_10);
                copySize = graphics.MeasureString("copy", Constants.TAHOMA_10);
                pasteSize = graphics.MeasureString("paste", Constants.TAHOMA_10);
                delSize = graphics.MeasureString("del", Constants.TAHOMA_10);
                findSize = graphics.MeasureString("find", Constants.TAHOMA_10);
                searchSize = graphics.MeasureString("search", Constants.TAHOMA_10);

                int numSeparators = 4;
                minContextMenuMidSection = Math.Max(copySize.Width, findSize.Width) + (2 *
Constants.CONTEXT_MENU_PADDING);
                // let's assume cut and del don't hold a candle in width to paste and search
                minContextMenuCornerSection = Math.Max(pasteSize.Width, searchSize.Width) + (2 *
Constants.CONTEXT_MENU_PADDING);
                minContextMenuWidth = minContextMenuMidSection + (2*minContextMenuCornerSection) + numSeparators;

        }
        protected override void InternalUpdateBounds(float x, float y, float width, float height)
        {
```

```
                        SizeF destSize = new SizeF(0,0);
                        SizeF sourceSize = new SizeF(0,0);
/*
                        Util.GetBestDestSourceSize(new SizeF(width, height), inboxfooter.Image.Size, ref destSize, ref sourceSize);
                        float factor = 1.0f;
                        if (destSize.Width < sourceSize.Width - Constants.IMAGE_SLOP)
                        {
                                factor = destSize.Width/sourceSize.Width;
                        }
                        inboxfooter.InitialBounds = new RectangleF(x, y + height - (sourceSize.Height * factor), (sourceSize.Width * factor),
(sourceSize.Height * factor));


                        Util.GetBestDestSourceSize(new SizeF(width, height), detailfooter.Image.Size, ref destSize, ref sourceSize);

                        if (destSize.Width < sourceSize.Width - Constants.IMAGE_SLOP)
                        {
                                factor = destSize.Width/sourceSize.Width;
                        }
                        detailfooter.InitialBounds = new RectangleF(x, y + height - (sourceSize.Height * factor), (sourceSize.Width * factor),
(sourceSize.Height * factor));
                        // here's the kicker - initially not visible
                        detailfooter.Visible = false;

                        Util.GetBestDestSourceSize(new SizeF(width, height), emailheader.Image.Size, ref destSize, ref sourceSize);

                        if (destSize.Width < sourceSize.Width - Constants.IMAGE_SLOP)
                        {
                                factor = destSize.Width/sourceSize.Width;
                        }
                        emailheader.InitialBounds = new RectangleF(x, y, (sourceSize.Width * factor), (sourceSize.Height * factor));
*/
//                      Util.GetBestDestSourceSize(new SizeF(width, height), blueDot.Image.Size, ref destSize, ref sourceSize);

//                      blueDot.InitialBounds = new RectangleF(x+ (width/2)-(blueDot.Image.Width/2), y+height-Constants.FOOTER_FUDGE-
blueDot.Image.Height, sourceSize.Width, sourceSize.Height);

//                      cursor.InitialBounds = new RectangleF(blueDot.X + (blueDot.Width/2) - (cursor.Image.Width/2),
//                              blueDot.Y - cursor.Image.Height, cursor.Image.Width, cursor.Image.Height);
                        cursor.InitialBounds = new RectangleF(x, y, cursor.Image.Width, cursor.Image.Height);
```

```
                cursor.Visible = false;
                // Set this before blue dot
                highlightglass.InitialBounds = new RectangleF(x, cursor.InitialBounds.Y, width,
Constants.HIGHLIGHT_GLASS_HEIGHT);

//              emailBlueDot.Cursor = cursor;
//              emailBlueDot.HighlightGlass = highlightglass;

                inboxlist.InitialBounds = new RectangleF(x, y, width, height);
                inboxlist.LayoutChildren();
                SnapObjectToHighlight(false);
                inboxlist.CaptureInitialSlidingOffset();

                emailDetail.InitialBounds = new RectangleF(x, y, width, height);
                emailDetail.InitialOffset = new SizeF(0, height);
                // Hide out of bounds
                emailDetail.TranslateBy(0, height);
                base.InternalUpdateBounds (x, y, width, height);
        }

        public override void Overview(bool animate, bool resetState)
        {
                this.AppTile.QuadTile.Landscape.ShellLevel = Constants.ShellLevel.APPLICATION;

                if (contextMenuNode != null)
                {
                        RemoveChild(contextMenuNode);
                        contextMenuNode = null;
                }
                if (textEditMenuNode != null)
                {
                        RemoveChild(textEditMenuNode);
                        textEditMenuNode = null;
                }


                //inboxfooter.Visible = true;
                //detailfooter.Visible = false;
                //detailfooter.Reset(false);
```

```
                emailDetail.Reset(animate);

                inboxlist.Visible = true;
                inboxlist.Reset(animate);

//              blueDot.Reset(animate);
                this.highlightglass.Visible = true;
                highlightglass.Reset(animate);
                cursor.Visible = false;
                cursor.Reset(animate);
        }
        public override void Detail(bool animate, bool resetState)
        {
                long duration = 0;
                if (animate)
                {
                        duration = Constants.DEFAULT_ANIMATION_TIME;
                }

                if (contextMenuNode != null)
                {
                        RemoveChild(contextMenuNode);
                        contextMenuNode = null;
                }
                if (textEditMenuNode != null)
                {
                        RemoveChild(textEditMenuNode);
                        textEditMenuNode = null;
                }
                AppTile.QuadTile.Landscape.ShellLevel = Constants.ShellLevel.OBJECT;

                //inboxfooter.Visible = false;

                //detailfooter.Visible = true;
                //detailfooter.Reset(true);

                PMatrix emailDetailMatrix = emailDetail.Matrix;
                emailDetailMatrix.OffsetX = 0;
                emailDetailMatrix.OffsetY = 0; //emailheader.Height/2;
                PMatrix inboxListMatrix = inboxlist.Matrix;
```

```
                inboxListMatrix.OffsetX = 0;
                inboxListMatrix.OffsetY = - (Bounds.X /2); //- (Bounds.X -emailheader.Height/2);

                inboxlist.AnimateToMatrix(inboxListMatrix, duration);
                emailDetail.AnimateToMatrix(emailDetailMatrix, duration);
                inboxlist.AnimateToMatrix(emailDetailMatrix, duration);

                cursor.Visible = true;
                highlightglass.Visible = true;
                // If the highlighter has been shortened horizontally, expand to original size
//              highlightglass.SetBounds(highlightglass.InitialBounds.X, highlightglass.Bounds.Y,
//                      highlightglass.InitialBounds.Width, highlightglass.Bounds.Height);

                if (resetState)
                {
                        //blueDot
//                      blueDot.Reset(true);

                        // Cursor
//                      cursor.Reset(true);

                        //Highlightglass
//                      highlightglass.Reset(true);
                }
        }

        /*
        public override void FailedDetailToContext(bool animate)
        {
                if (emailBlueDot.isHome())
                {
                        long duration = 0;
                        if (animate)
                        {
                                duration = Constants.DEFAULT_ANIMATION_TIME;
                        }
                        PMatrix matrix = blueDot.Matrix;
                        PMatrix highlightglassMatrix = highlightglass.Matrix;
                        PMatrix cursorMatrix = cursor.Matrix;
                        float diff = (cursor.Bounds.Y - emailDetail.Bounds.Y ) - emailDetail.Matrix.OffsetY -
```

```
emailDetail.GetHeaderHeight() - emailDetail.GetFirstLineOffset();
                                matrix.OffsetY = - diff;
                                highlightglassMatrix.OffsetY = - diff;
                                cursorMatrix.OffsetY = - diff;
                                blueDot.AnimateToMatrix(matrix, duration);
                                highlightglass.AnimateToMatrix(highlightglassMatrix, duration);
                                PTransformActivity activity = cursor.AnimateToMatrix(cursorMatrix, duration);
                                //                              activity.ActivityFinished = new
ActivityFinishedDelegate(AnimateToFirstLineFinished);
                        }
                        else
                        {
                                SnapHighlight();
                        }
                }
                */
                void AnimateToFirstLineFinished(PActivity activity)
                {
                        //                      while (Root.ActivityScheduler.ActivitiesReference.Count != 0);
                        this.SnapHighlight();
                }

                /*
                public override bool Context(bool animate, bool resetState)
                {
                        // the bounds of the overlapping child in terms of the EmailAppNode
                        EmailText overlappingText = emailDetail.GetIntersectingEmailText(cursor.LocalToParent(cursor.Bounds));
                        if (overlappingText != null)
                        {
                                long duration = 0;
                                if (animate)
                                {
                                        duration = Constants.DEFAULT_ANIMATION_TIME;
                                }
                                this.AppTile.QuadTile.Landscape.ShellLevel = Constants.ShellLevel.CONTEXT;

                                cursor.Visible = false;
                                PMatrix footerMatrix = detailfooter.Matrix;
                                footerMatrix.OffsetY += detailfooter.Height;
                                detailfooter.AnimateToMatrix(footerMatrix, duration);
```

```
                                PMatrix highlightglassMatrix = highlightglass.Matrix;
                                RectangleF overlappingTextBounds = emailDetail.GetAppNodeLevelBounds(overlappingText);

                                this.highlightglass.SetBounds(overlappingTextBounds.X-4, highlightglass.Bounds.Y,
overlappingTextBounds.Width+8, highlightglass.Bounds.Height);
                                ShowContextMenuEasyWay(overlappingText, overlappingTextBounds);
                                //ShowContextMenuHardWay()
                                blueDot.MoveToFront();
                                return true;
                        }
                        else
                        {
                                return false;
                        }
                }

                public override bool Input(bool animate, bool resetState)
                {
                        // the bounds of the overlapping child in terms of the EmailAppNode
                        EmailText overlappingText = emailDetail.GetIntersectingEmailText(cursor.LocalToParent(cursor.Bounds));
                        if (overlappingText != null)
                        {
                                long duration = 0;
                                if (animate)
                                {
                                        duration = Constants.DEFAULT_ANIMATION_TIME;
                                }
                                this.AppTile.QuadTile.Landscape.ShellLevel = Constants.ShellLevel.INPUT;

                                cursor.Visible = false;
                                PMatrix footerMatrix = detailfooter.Matrix;
                                footerMatrix.OffsetY += detailfooter.Height;
                                detailfooter.AnimateToMatrix(footerMatrix, duration);

                                PMatrix highlightglassMatrix = highlightglass.Matrix;
                                RectangleF overlappingTextBounds = emailDetail.GetAppNodeLevelBounds(overlappingText);

                                this.highlightglass.SetBounds(overlappingTextBounds.X-4, highlightglass.Bounds.Y,
overlappingTextBounds.Width+8, highlightglass.Bounds.Height);
```

```
                    ShowEditMenuEasyWay(overlappingText, overlappingTextBounds);
                    //ShowContextMenuHardWay()
                    blueDot.MoveToFront();
                    return true;
                }
                else
                {
                    return false;
                }
            }

            private void ShowContextMenuEasyWay(EmailText overlappingText, RectangleF targetTextBounds)
            {
                if (contextMenuNode != null)
                {
                    contextMenuNode.SetBounds(contextMenuNode.Bounds.X + (contextMenuNode.Bounds.Width/2),
                        contextMenuNode.Bounds.Y + (contextMenuNode.Bounds.Height/2), 0,0);

                    contextMenuNode.MoveToFront();
                    contextMenuNode.AnimateToBounds(contextMenuNode.InitialBounds.X, contextMenuNode.InitialBounds.Y,
contextMenuNode.InitialBounds.Width, contextMenuNode.InitialBounds.Height, Constants.DEFAULT_ANIMATION_TIME);
                }
                else
                {
                    this.contextMenuNode = ShowMenuEasyWay(overlappingText, targetTextBounds, contextMenuImage);
                }
            }
            private void ShowEditMenuEasyWay(EmailText overlappingText, RectangleF targetTextBounds)
            {
                if (textEditMenuNode != null)
                {
                    textEditMenuNode.SetBounds(textEditMenuNode.Bounds.X + (textEditMenuNode.Bounds.Width/2),
                        textEditMenuNode.Bounds.Y + (textEditMenuNode.Bounds.Height/2), 0,0);
                    textEditMenuNode.MoveToFront();

                    textEditMenuNode.AnimateToBounds(textEditMenuNode.InitialBounds.X, textEditMenuNode.InitialBounds.Y,
textEditMenuNode.InitialBounds.Width, textEditMenuNode.InitialBounds.Height, Constants.DEFAULT_ANIMATION_TIME);
                }
                else
```

```
                {
                        this.textEditMenuNode = ShowMenuEasyWay(overlappingText, targetTextBounds, textMenuImage);
                }
        }
        private MenuImageNode ShowMenuEasyWay(EmailText overlappingText, RectangleF targetTextBounds, Bitmap menuImage)
        {
                Bitmap localMenuImage = new Bitmap(menuImage);
                int innerBorderOffset = Constants.CONTEXT_MENU_BORDER_WIDTH+1;
                int innerBorderWidth = contextMenuImage.Width - (2*innerBorderOffset);
                int innerBorderHeight = contextMenuImage.Height- (2*innerBorderOffset);
                RectangleF goalBounds = new RectangleF(targetTextBounds.X + (targetTextBounds.Width/2) -
(localMenuImage.Width/2),
                        targetTextBounds.Y - innerBorderOffset, localMenuImage.Width, localMenuImage.Height);
                Graphics contextMenuGraphics = Graphics.FromImage(localMenuImage);

                // Problem with having transparent window was that I was reusing the image
                // could work fine if I tried again
                //contextMenuGraphics.FillRectangle(Constants.JUST_OFF_HIGHLIGHT_BRUSH, (int)(targetTextBounds.X -
goalBounds.X), (int)(targetTextBounds.Y - goalBounds.Y), (int)targetTextBounds.Width, (int)targetTextBounds.Height);
                contextMenuGraphics.DrawString(overlappingText.Text, Constants.TAHOMA_10, Constants.BLACK_BRUSH,
(int)(targetTextBounds.X - goalBounds.X), (int)(targetTextBounds.Y - goalBounds.Y));
                MenuImageNode menuNode = new MenuImageNode(localMenuImage);
                menuNode.InitialBounds = goalBounds;
                menuNode.SetBounds(goalBounds.X + (goalBounds.Width/2), goalBounds.Y + (goalBounds.Height/2),0,0);
                //contextMenuNode.UsesTransparency = true;
                AddChild(menuNode);
                //contextMenuNode.TransparencyLocation = new Point((int)(targetTextBounds.X - goalBounds.X)+1,
(int)(targetTextBounds.Y - goalBounds.Y));
                menuNode.AnimateToBounds(goalBounds.X, goalBounds.Y, goalBounds.Width, goalBounds.Height,
Constants.DEFAULT_ANIMATION_TIME);
                return menuNode;

        }

        private void ShowContextMenuHardWay(RectangleF overlappingChildBounds)
        {
                // Build the menu
                // Need at least as wide as the dot
                float minRequired = Math.Max(blueDot.Width, minContextMenuWidth);
                int menuWidth = (int) Math.Max(minRequired, (overlappingChildBounds.Width +
```

```
(2*Constants.CONTEXT_MENU_BORDER_WIDTH)));
                        // need at least as high as the bootom of dot to top of text
                        int menuHeight = (int) (blueDot.Height + emailBlueDot.HighlightglassYDiff +
(2*Constants.CONTEXT_MENU_BORDER_WIDTH));

                        Bitmap contextMenuBitmap = new Bitmap(menuWidth, menuHeight);
                        Graphics contextMenuGraphics = Graphics.FromImage(contextMenuBitmap);
                        contextMenuGraphics.FillRectangle(Constants.MED_GRAY_BRUSH, 0,0, menuWidth, menuHeight);
                        int innerBorderOffset = Constants.CONTEXT_MENU_BORDER_WIDTH;
                        int innerBorderWidth = menuWidth - (2*innerBorderOffset);
                        int innerBorderHeight = menuHeight - (2*innerBorderOffset);
                        contextMenuGraphics.FillRectangle(Constants.HIGHLIGHT_BRUSH, innerBorderOffset, innerBorderOffset,
innerBorderWidth, innerBorderHeight);
                        // Pink is used to denote the tansparent region
                        contextMenuGraphics.FillRectangle(Constants.JUST_OFF_HIGHLIGHT_BRUSH, Math.Max(innerBorderOffset,
(int)(innerBorderOffset + (innerBorderWidth/2) - (overlappingChildBounds.Width/2))), innerBorderOffset, (int)overlappingChildBounds.Width,
(int)overlappingChildBounds.Height);
                        contextMenuGraphics.FillRectangle(Constants.DARK_GRAY_BRUSH, innerBorderOffset , (int)(innerBorderOffset +
highlightglass.Height), innerBorderWidth, (int)(innerBorderHeight-highlightglass.Height));

                        // draw borders
                        contextMenuGraphics.DrawRectangle(Constants.DARK_GRAY_PEN, 0,0, menuWidth-1, menuHeight-1);
                        contextMenuGraphics.DrawRectangle(Constants.DARK_GRAY_PEN, innerBorderOffset-1,innerBorderOffset-1,
innerBorderWidth, innerBorderHeight);
                        // draw text
                        float leftover = menuWidth - ((2*minContextMenuCornerSection) + minContextMenuMidSection +4);
                        int pad = Constants.CONTEXT_MENU_PADDING;
                        float rawCornerSize = minContextMenuCornerSection - (2*pad);
                        float rawMidSize = this.minContextMenuMidSection - (2*pad);
                        int separatorSize = 1;
                        if (leftover > 0)
                        {
                                pad += (int)(leftover / 3)/2;
                        }
                        float newCornerSize = (int)rawCornerSize + (2*pad);
                        float newMidSize = (int)rawMidSize + (2*pad);
                        //upper text
                        int yOffset = (int) (Constants.CONTEXT_MENU_BORDER_WIDTH - cutSize.Height)/2;
                        int xOffset = separatorSize;
                        int itempad = (int)(newCornerSize - cutSize.Width)/2;
```

```
                    xOffset += itempad;
                    contextMenuGraphics.DrawString("cut", Constants.TAHOMA_10, Constants.BLACK_BRUSH, xOffset, yOffset);
                    xOffset += (int)cutSize.Width + itempad;
                    contextMenuGraphics.DrawLine(Constants.DARK_GRAY_PEN, xOffset, 0, xOffset,
Constants.CONTEXT_MENU_BORDER_WIDTH-1);
                    contextMenuGraphics.DrawLine(Constants.DARK_GRAY_PEN, 0, (int)newCornerSize,
Constants.CONTEXT_MENU_BORDER_WIDTH, (int)newCornerSize);
                    xOffset += separatorSize;
                    itempad = (int)(newMidSize - copySize.Width)/2;
                    xOffset += itempad;
                    contextMenuGraphics.DrawString("copy", Constants.TAHOMA_10, Constants.BLACK_BRUSH, xOffset, yOffset);
                    xOffset += (int)copySize.Width +itempad;
                    contextMenuGraphics.DrawLine(Constants.DARK_GRAY_PEN, xOffset, 0, xOffset,
Constants.CONTEXT_MENU_BORDER_WIDTH-1);
                    xOffset += separatorSize;
                    itempad = (int)(newCornerSize - pasteSize.Width)/2;
                    xOffset += itempad;
                    contextMenuGraphics.DrawString("paste", Constants.TAHOMA_10, Constants.BLACK_BRUSH, xOffset, yOffset);
                    contextMenuGraphics.DrawLine(Constants.DARK_GRAY_PEN, menuWidth, (int)newCornerSize, menuWidth -
Constants.CONTEXT_MENU_BORDER_WIDTH, (int)newCornerSize);

                    yOffset = (int)(menuHeight - cutSize.Height) - yOffset;
                    xOffset = separatorSize;
                    itempad = (int)(newCornerSize - delSize.Width)/2;
                    xOffset += itempad;
                    contextMenuGraphics.DrawString("del", Constants.TAHOMA_10, Constants.BLACK_BRUSH, xOffset, yOffset);
                    xOffset += (int)delSize.Width + itempad;
                    contextMenuGraphics.DrawLine(Constants.DARK_GRAY_PEN, xOffset, menuHeight-
Constants.CONTEXT_MENU_BORDER_WIDTH, xOffset, menuHeight);
                    contextMenuGraphics.DrawLine(Constants.DARK_GRAY_PEN, 0, menuHeight - (int)newCornerSize,
Constants.CONTEXT_MENU_BORDER_WIDTH, menuHeight -  (int)newCornerSize);
                    xOffset += separatorSize;
                    itempad = (int)(newMidSize - findSize.Width)/2;
                    xOffset += itempad;
                    contextMenuGraphics.DrawString("find", Constants.TAHOMA_10, Constants.BLACK_BRUSH, xOffset, yOffset);
                    xOffset += (int)copySize.Width +itempad;
                    contextMenuGraphics.DrawLine(Constants.DARK_GRAY_PEN, xOffset, menuHeight-
Constants.CONTEXT_MENU_BORDER_WIDTH, xOffset, menuHeight);
                    xOffset += separatorSize;
                    itempad = (int)(newCornerSize - searchSize.Width)/2;
```

```
                    xOffset += itempad;
                    contextMenuGraphics.DrawString("search", Constants.TAHOMA_10, Constants.BLACK_BRUSH, xOffset, yOffset);
                    contextMenuGraphics.DrawLine(Constants.DARK_GRAY_PEN, menuWidth, menuHeight-(int)newCornerSize,
menuWidth - Constants.CONTEXT_MENU_BORDER_WIDTH, menuHeight-(int)newCornerSize);


                    // If you want to do it this way, create a new type of image node and override paint so that
                    // transparency is only done if the bounds.width == image.width (so you don't get weird holes
                    // in the animation
                    //      this.contextMenuNode = new ContextMenuNode(contextMenuBitmap);
                    this.contextMenuNode = new MenuImageNode(contextMenuBitmap);
                    contextMenuNode.UsesTransparency = true;
                    AddChild(contextMenuNode);
                    RectangleF goalBounds = new RectangleF(overlappingChildBounds.X + (overlappingChildBounds.Width/2) -
(contextMenuNode.Width/2),
                            overlappingChildBounds.Y - Constants.CONTEXT_MENU_BORDER_WIDTH, contextMenuNode.Width,
contextMenuNode.Height);
                    contextMenuNode.TransparencyLocation = new Point((int)contextMenuNode.Bounds.Width/2,
(int)Constants.CONTEXT_MENU_BORDER_WIDTH + 1);
                    contextMenuNode.SetBounds(goalBounds.X + (goalBounds.Width/2), goalBounds.Y + (goalBounds.Height/2),0,0);
                    contextMenuNode.AnimateToBounds(goalBounds.X, goalBounds.Y, goalBounds.Width, goalBounds.Height,
Constants.DEFAULT_ANIMATION_TIME);
                }
                */

                public void SnapHighlightToPosition(RectangleF sourceRectangle, RectangleF destinationRectangle, bool yOnly, bool animate)
                {
                    // what would it take to put cursor in horizontal center

                    float xdiff = 0;
                    if (!yOnly)
                    {
                        // Both of these bounds are in EmailAppNode coordiate space
                        float centerDestination = (destinationRectangle.X + (destinationRectangle.Width/2));
                        float centerSource = (sourceRectangle.X + (sourceRectangle.Width/2));
                        // align source center to destination center
                        xdiff = centerSource - centerDestination;
                    }
                    // align source to destination top
                    float ydiff = sourceRectangle.Top - destinationRectangle.Top;
```

```
            /*
                                    PMatrix matrix = blueDot.Matrix;
                                    matrix.OffsetX -= xdiff;
                                    matrix.OffsetY -= ydiff;
                                    blueDot.AnimateToMatrix(matrix, Constants.DEFAULT_ANIMATION_TIME/2);
            */
            if (animate)
            {
                    PMatrix matrix = cursor.Matrix;
                    matrix.OffsetX -= xdiff;
                    matrix.OffsetY -= ydiff;
                    cursor.AnimateToMatrix(matrix, Constants.DEFAULT_ANIMATION_TIME);
                    matrix = highlightglass.Matrix;
                    matrix.OffsetY -= ydiff;
                    highlightglass.AnimateToMatrix(matrix, Constants.DEFAULT_ANIMATION_TIME);
            }
            else
            {
                    cursor.TranslateBy(-xdiff, -ydiff);
                    highlightglass.TranslateBy(0, -ydiff);
            }
    }

    public void SnapHighlightToPosition(RectangleF sourceRectangle,
            RectangleF destinationRectangle, bool yOnly)
    {
            SnapHighlightToPosition(sourceRectangle, destinationRectangle, yOnly, true);
    }

    public void SnapPositionToObject(PNode obj, RectangleF sourceRectangle,
            RectangleF destinationRectangle, bool yOnly, bool animate)
    {
            float xdiff = 0;
            long duration = 0;
            if (animate)
            {
                    duration = Constants.DEFAULT_ANIMATION_TIME/2;
            }
            if (!yOnly)
            {
```

```
                float centerDestination = (destinationRectangle.X + (destinationRectangle.Width/2));
                float centerSource = (sourceRectangle.X + (sourceRectangle.Width/2));
                // align source center to destination center
                xdiff = centerSource - centerDestination;
            }
            // align source to destination top
            float ydiff = sourceRectangle.Top - destinationRectangle.Top;

            PMatrix matrix = obj.Matrix;
            matrix.OffsetX -= xdiff;
            matrix.OffsetY -= ydiff;
            if (animate)
            {
                obj.AnimateToMatrix(matrix, duration);
            }
            else
            {
                obj.Matrix = matrix;
            }

        }
        public void SnapObjectToHighlight(bool animate)
        {
            if (State == Constants.AppState.OVERVIEW)
            {
                PNode obj = inboxlist.GetMainDraggedNode();
                SnapObjectToHighlight(obj, animate);
            }
        }
        public void SnapObjectToHighlight(PNode node, bool animate)
        {
            RectangleF cursorEmailAppBounds = cursor.LocalToParent(cursor.Bounds);
            RectangleF sourceRectangle = RectangleF.Empty;
            if (State == Constants.AppState.OVERVIEW)
            {
                sourceRectangle = inboxlist.GetIntersectingEmailItemBounds(cursorEmailAppBounds);
                if (!sourceRectangle.IsEmpty)
                {
                    this.SnapPositionToObject(node, new RectangleF(sourceRectangle.X,
                        sourceRectangle.Y, sourceRectangle.Height, sourceRectangle.Width), cursorEmailAppBounds,
```

```
true, animate);
                                }
                        }
                }
                public RectangleF SnapHighlight()
                {
                        RectangleF cursorEmailAppBounds = cursor.LocalToParent(cursor.Bounds);
                        RectangleF targetRectangle = RectangleF.Empty;
                        bool yOnly = false;
                        if (State == Constants.AppState.DETAIL)
                        {
                                targetRectangle = emailDetail.GetIntersectingEmailItemBounds(cursorEmailAppBounds);
                                yOnly = true;
//                              targetRectangle = emailDetail.GetClosestIntersectingEmailTextBounds(cursorEmailAppBounds);
                        }
                        else if(State == Constants.AppState.OVERVIEW)
                        {
                                //if (emailBlueDot.isHome())
                                //{
                                //      SnapObjectToHighlight(true);
                                //}
                                //else
                                //{
                                        targetRectangle = inboxlist.GetIntersectingEmailItemBounds(cursorEmailAppBounds);
                                        yOnly = true;
                                //}
                        }
                        if (!targetRectangle.IsEmpty)
                        {
//                                      SnapHighlightToPosition(new RectangleF(cursorEmailAppBounds.X, cursorEmailAppBounds.Y-1,
cursorEmailAppBounds.Width, cursorEmailAppBounds.Height), targetRectangle, yOnly);
                                SnapHighlightToPosition(new RectangleF(cursorEmailAppBounds.X, cursorEmailAppBounds.Y,
cursorEmailAppBounds.Width, cursorEmailAppBounds.Height), targetRectangle, yOnly);
                        }
                        return targetRectangle;
                }
                public void SnapHighlightToLine(RectangleF lineRectangle, bool animate)
                {
                        RectangleF cursorEmailAppBounds = cursor.LocalToParent(cursor.Bounds);
                        SnapHighlightToPosition(cursorEmailAppBounds, lineRectangle, true, animate);
```

```
        }
        public void SnapHighlightToLine(RectangleF lineRectangle)
        {
                SnapHighlightToLine(lineRectangle, true);
                //RectangleF cursorEmailAppBounds = cursor.LocalToParent(cursor.Bounds);
                //SnapHighlightToPosition(cursorEmailAppBounds, lineRectangle, true);
        }
        protected override void Paint(PPaintContext paintContext)
        {

                Graphics graphics = paintContext.Graphics;
                // Draw a white background on which to layer things
                graphics.FillRectangle(Constants.WHITE_BRUSH, Bounds.X, Bounds.Y, Bounds.Width-1, Bounds.Height-1);
        }
        public HighlightGlass HighlightGlass
        {
                get
                {
                        return this.highlightglass;
                }
        }
}

class EmailInboxFooter : ImageNode
{
        public EmailInboxFooter() : base()
        {
                Image = Util.GetImage(this, "LaunchPoint.images.emailinboxfooter.png");
                this.DoScale = true;
        }
}
class DetailFooter : ImageNode
{
        public DetailFooter() : base()
        {
                Image = Util.GetImage(this, "LaunchPoint.images.detailfooter.png");
                this.DoScale = true;
        }
}
```

Email.cs

```
class InboxList : ImageNode
{
        public InboxList() : base()
        {
                Image = Util.GetImage(this, "LaunchPoint.images.emaillist.png");
                usesTransparency = true;
                transparencyLocation = new Point(0,0);
                this.AddInputEventListener(new InboxListDragEventHandler());
        }
        class  InboxListDragEventHandler : PDragEventHandler
        {
                protected override void OnDrag(object sender, PInputEventArgs e)
                {
                        InboxList list = (InboxList)sender;
                        SizeF s = e.GetDeltaRelativeTo(list);
                        s = list.LocalToParent(s);
                        list.TranslateBy(0, s.Height);
                }

        }
}
class EmailDetail : ImageNode
{
        public EmailDetail() : base()
        {
                Image = Util.GetImage(this, "LaunchPoint.images.emaildetail.png");
                usesTransparency = true;
                transparencyLocation = new Point(0,Image.Height-1);
        }

}

public class HighlightGlass : PNode
{
        Brush blueBrush = Constants.HIGHLIGHT_BRUSH;
        RectangleF initialBounds = RectangleF.Empty;
        SizeF initialOffset = new SizeF(0,0);
        public HighlightGlass() : base()
        {
```

```
            Color bcolor = ((SolidBrush)blueBrush).Color;
            blueBrush = new SolidBrush(Color.FromArgb(120, bcolor.R, bcolor.G, bcolor.B));
            //this.AddInputEventListener(new HighlightGlassDragEventHandler());
}
protected override void Paint(PPaintContext paintContext)
{
            paintContext.Graphics.FillRectangle(blueBrush, Bounds.X, Bounds.Y, Bounds.Width, Bounds.Height);
            base.Paint (paintContext);
}
public RectangleF InitialBounds
{
            set
            {
                        initialBounds = value;
                        Bounds = value;
            }
            get { return initialBounds; }
}
public SizeF InitialOffset
{
            set { initialOffset = value;}
            get { return initialOffset; }
}
public void Reset (bool animate)
{
            Bounds = InitialBounds;
            PMatrix matrix = Matrix;
            matrix.OffsetX = InitialOffset.Width;
            matrix.OffsetY = InitialOffset.Height;
            long duration = 0;

            if (animate) {
                        duration = Constants.DEFAULT_ANIMATION_TIME;
                        this.AnimateToMatrix(matrix, duration);
            } else {
                        this.Matrix = matrix;
            }
}
class  HighlightGlassDragEventHandler : PDragEventHandler
{
```

```
        }
}
public class EmailBlueDot : BlueDot
{
        private EmailAppNode emailAppNode;
        private HighlightGlass highlightGlass;
        private Cursor cursor;
        // The y diff between blue dot and highlightglass
        float glassYDiff = 0;
        float cursorYDiff = 0;
        float cursorXDiff = 0;

        public EmailBlueDot(EmailAppNode node) : base()
        {
                EmailAppNode = node;
                this.AddInputEventListener(new BlueDotDragEventHandler());
        }
        // If still in starting gate, considered "home"
        public bool isHome()
        {
                return (MatrixReference.OffsetX == 0 && Math.Abs(MatrixReference.OffsetY) < Height);
        }

        public EmailAppNode EmailAppNode
        {
                set { emailAppNode = value; }
                get { return emailAppNode; }
        }
        public float HighlightglassYDiff
        {
                set { glassYDiff = value; }
                get { return glassYDiff; }
        }
        public float CursorXDiff
        {
                set { cursorXDiff = value; }
                get { return cursorXDiff; }
        }
        public float CursorYDiff
```

```
{
        set { cursorYDiff = value; }
        get { return cursorYDiff; }
}

public HighlightGlass HighlightGlass
{
        set
        {
                highlightGlass = value;
                if (!highlightGlass.InitialBounds.IsEmpty && !InitialBounds.IsEmpty)
                {
                        this.glassYDiff = InitialBounds.Y - highlightGlass.InitialBounds.Y;
                }
        }
        get { return highlightGlass;}
}
public Cursor Cursor
{
        set
        {
                cursor = value;
                if (!cursor.InitialBounds.IsEmpty && !InitialBounds.IsEmpty)
                {
                        this.cursorYDiff = InitialBounds.Y - cursor.InitialBounds.Y;
                        this.cursorXDiff = InitialBounds.X - cursor.InitialBounds.X;
                }
        }
        get { return cursor;}
}
public override RectangleF InitialBounds
{
        get
        {
                return base.InitialBounds;
        }
        set
        {
                base.InitialBounds = value;
        }
```

```
        }
        protected override void Paint(PPaintContext paintContext)
        {
                if (emailAppNode.AppTile.QuadTile.Landscape.Camera.ViewScale ==
emailAppNode.AppTile.QuadTile.Landscape.ApplicationScale)
                {
                        base.Paint (paintContext);
                }
        }

        public override void Clicked()
        {
        //      EmailAppNode.Forward();
//              base.Clicked ();
                /*
                if (Parent is AppNode)
                {
                        AppNode appParent = (AppNode)Parent;
                        if (appParent.State == Constants.AppState.OVERVIEW)
                        {
                                appParent.State = Constants.AppState.DETAIL;
                        }
                        else if (appParent.State == Constants.AppState.DETAIL)
                        {
                                appParent.State = Constants.AppState.CONTEXT;
                        }
                        // Once in context, blue dot is dragged
                }
        */
        }
        public override void TranslateBy(float dx, float dy)
        {
                base.TranslateBy (dx, dy);
                if (cursor != null) cursor.TranslateBy(dx, dy);
                // highlighter only goes up and down
                if (highlightGlass != null) highlightGlass.TranslateBy(0, dy);
        }

        // This is here to eat up the onclick event for the main blue dot
        public override void OnClick(PInputEventArgs e)
```

```
{
}

/*
                public override void OnClick(PInputEventArgs e)
                {
                        if (Parent is AppNode)
                        {
                                AppNode appParent = (AppNode)Parent;
                                if (appParent.State == Constants.AppState.OVERVIEW)
                                {
                                        Clicked(e);
                                }
                                else if (appParent.State == Constants.AppState.DETAIL)
                                {
                                        Clicked(e);
                                }

                        }

                }
*/
protected override void InternalUpdateBounds(float x, float y, float width, float height)
{

        if (this.glassYDiff != 0)
        {
                RectangleF glassBounds = highlightGlass.Bounds;
                highlightGlass.SetBounds(glassBounds.X, y - glassYDiff, glassBounds.Width, glassBounds.Height);
        }
        if (this.cursorYDiff != 0)
        {
                RectangleF cursorBounds = cursor.Bounds;
                cursor.SetBounds(cursorBounds.X, y - cursorYDiff, cursorBounds.Width, cursorBounds.Height);
        }
        if (this.cursorXDiff != 0)
        {
                RectangleF cursorBounds = cursor.Bounds;
                cursor.SetBounds(x - cursorXDiff, cursorBounds.Y, cursorBounds.Width, cursorBounds.Height);
        }
```

```
                base.InternalUpdateBounds (x, y, width, height);
        }


//              class  BlueDotDragEventHandler : PDragEventHandler
class BlueDotDragEventHandler : UMD.HCIL.Piccolo.Event.PBasicInputEventHandler
        {
                PointF mousePressedPos;
                bool dragging = false;
/*
        protected override void OnStartDrag(object sender, PInputEventArgs e)
                {
                        PDebug.Log("Enter\t OnStartDrag");
                        base.OnStartDrag(sender, e);
                }
                protected override void OnEndDrag(object sender, PInputEventArgs e)
                {
                        PDebug.Log("Enter\t OnEndDrag");
                }
                protected override void StartDragActivity(PInputEventArgs e)
                {
                        base.StartDragActivity(e);
                        DragActivity.StepInterval = 10;
                }
                protected override bool ShouldStartDragInteraction(PInputEventArgs e)
                {
                        return PUtil.DistanceBetweenPoints(MousePressedCanvasPoint, e.CanvasPosition)
                                >= Constants.DRAG_THRESHOLD;
                }
                */
        public override void OnMouseDown(object sender, PInputEventArgs e)
        {
                mousePressedPos = e.CanvasPosition;
                base.OnMouseDown (sender, e);
        }


        public override void OnMouseDrag(object sender, PInputEventArgs e)
        {
```

```
            if (! dragging && PUtil.DistanceBetweenPoints(mousePressedPos, e.CanvasPosition) >= Constants.DRAG_THRESHOLD)
            {
                dragging = true;
            }
            else if (dragging)
            {
                OnDrag(sender, e);
            }
            base.OnMouseDrag (sender, e);
        }


//          protected override void OnDrag(object sender, PInputEventArgs e)
    private void OnDrag(object sender, PInputEventArgs e)
            {
            e.TopCamera.Canvas.PaintImmediately();

                EmailBlueDot dot = (EmailBlueDot)sender;

                // If In overview mode, can move blue dot to a particular email
                // If in edit mode, can move blue dot to line
                if (dot.EmailAppNode.State == Constants.AppState.OVERVIEW)
                {
                    SizeF s = e.GetDeltaRelativeTo(dot);
                    s = dot.LocalToParent(s);
                    // Don't go below original position, less than means "above"
                    if (dot.MatrixReference.OffsetY + s.Height <= dot.InitialOffset.Height)
                    {
                        dot.TranslateBy(0,s.Height);
                    }
                }
                else if (dot.EmailAppNode.State == Constants.AppState.DETAIL )
                {
                    SizeF s = e.GetDeltaRelativeTo(dot);
                    s = dot.LocalToParent(s);



                    // Don't go below original position
```

```
// Can only drag up while home
if (dot.isHome() && (dot.MatrixReference.OffsetY + s.Height <= dot.InitialOffset.Height) )
{
        dot.TranslateBy(0,s.Height);
}
else
{

        if (Math.Abs(s.Height) > Math.Abs(s.Width))
        {
                // less than means "above"
                if (dot.MatrixReference.OffsetY + s.Height <= dot.InitialOffset.Height)
                {
                        dot.TranslateBy(0,s.Height);
                }
        }
        else
        {
                dot.TranslateBy(s.Width, 0);
        }
}

/*
        if (direction == Constants.NavigationDirection.NONE)
        {
                if (Math.Abs(s.Height) > Math.Abs(s.Width))
                {
                        direction = Constants.NavigationDirection.VERTICAL;
                        dot.SetBounds(dot.Bounds.X, dot.Bounds.Y+s.Height,
                                dot.Bounds.Width,
                                dot.Bounds.Height);
                }
                else if (Math.Abs(s.Height) < Math.Abs(s.Width))
                {
                        direction = Constants.NavigationDirection.HORIZONTAL;
                        dot.SetBounds(dot.Bounds.X+s.Width,
                                dot.Bounds.Y, dot.Bounds.Width,
                                dot.Bounds.Height);
                }
        }
```

```
                                        else if (direction == Constants.NavigationDirection.VERTICAL)
                                        {
                                                dot.SetBounds(dot.Bounds.X, dot.Bounds.Y+s.Height,
                                                        dot.Bounds.Width,
                                                        dot.Bounds.Height);
                                        }
                                        else if (direction == Constants.NavigationDirection.HORIZONTAL)
                                        {
                                                dot.SetBounds(dot.Bounds.X+s.Width,
                                                        dot.Bounds.Y, dot.Bounds.Width,
                                                        dot.Bounds.Height);
                                        }
                                        */
                                // for point, only move in x or y

                        }
                //e.Camera.Repaint();
//              e.TopCamera.Canvas.PaintImmediately();

                }

                public override void OnMouseUp(object sender, PInputEventArgs e)
                {
                        EmailBlueDot dot = (EmailBlueDot)sender;
                        if (!dragging)
                //      if (this.Dragging)
                        {
                                dot.Clicked();
                        }
                        else
                        {
                                dot.EmailAppNode.SnapHighlight();
                        }
                        dragging = false;
                        base.OnMouseUp (sender, e);
                }
        }

}
class MenuImageNode : ImageNode
```

```
        {
            public MenuImageNode(Bitmap image) : base(image)
            {
                DoScale = true;
            }
        }
}
```