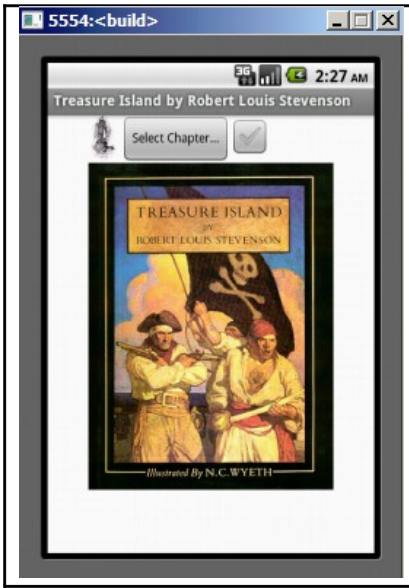
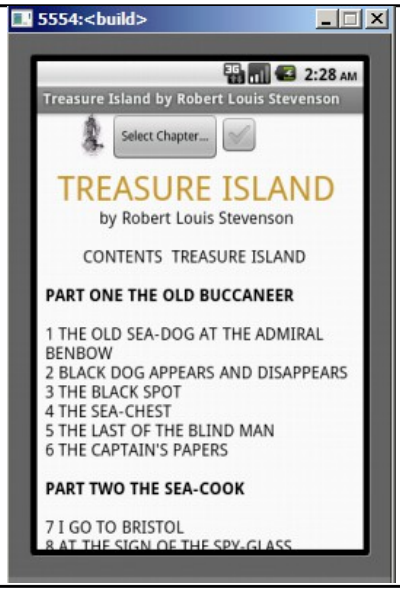
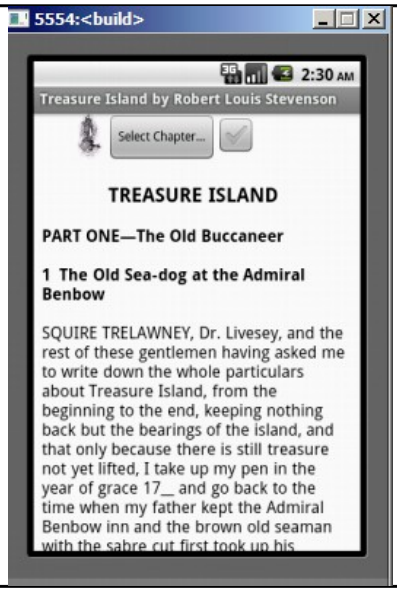
		
Treasure Island - the App e-book cover page	***Treasure Island - the App introduction text***	***Text Version of the e-reader***

Treasure Island - the App...an e-reader Tutorial

What You're Building

Treasure Island - the App demonstrates ways to develop your own Android e-reader app and make the text of a book you write available to friends digitally. Two ways to build the e-reader are explained. Build them both or just build the bunny app. The demonstration text provided with this tutorial comes from the Gutenberg project; read about Project Gutenberg below. This tutorial is for **Intermediate** and **Advanced AI2** users. Tools on your PC may be required to provide some resources. How to use those tools is not explained in detail.

Introduction

I hope you enjoy the novel this tutorial uses to demonstrate how to build the e-reader app. *Treasure Island* was one of my favorites. The book is now in the public domain in the United States and the book's text is available from the Gutenberg project. The example text and tutorial demonstrate how to handle large volumes of text with App Inventor 2. When you finish reading *Treasure Island* , replace Robert Louis Stevenson's text with text from your own book. Writing your book may be difficult; getting your book in digital form suitable for this app might be quite easy.

The full version of the e-reader *Treasure Island - the App*, uses html files created from the entire text of the novel *Treasure Island* by Robert Louis Stevenson. In paper print, *Treasure Island* is a large book, over 300 pages and 34 chapters in six parts. The demonstration text is from the Gutenberg Project and may be freely used. Gutenberg e-books are free in the United States because the book copyright for most of the books they make available has expired. Gutenberg books may not be free of copyright in other countries. Readers outside of the United States: Please check the copyright laws of your country before downloading or redistributing Gutenberg e-books. *Treasure Island* is for the use of anyone anywhere at no cost and with almost no restrictions whatsoever. You may copy the text and finished e-reader, give it away or re-use the *Treasure Island* text under the terms of the Project Gutenberg License included with this project or online at www.gutenberg.org .

A short version of the e-reader app displays five *Treasure Island* chapters. The simplified app uses large blocks of text embedded in the app as a substitute for the html files. The short version is suitable for a short story or as a technique to provide special information to users or another app you develop.

This project is about building an e-reader type app. If you publish you own works using this app, you might want to include a copyright notice within the text to help protect your rights to your original text.

Build the App(s) *Considerations before You Jump into Coding*

The following discussion talks about things you as a developer need to think about before you start coding. If you are uncertain what is discussed here, you probably will better understand after you actually build one of the versions of the e-reader. So read this section, then build and if you need to know why something is done in the coding part, return here.

You create Text pages in App Inventor 2 in several ways. Your book's text pages can be a single html file or multiple html files. An html file (hypertext markup language) is a complex text file that uses special characters to tell a computer how to display the text on a Web page. You need an html editor to create your own book text using html but can build the demonstration app using the html files supplied without any additional programming tools.

You have the option to use simple text files directly to make book text pages instead of using html files. Using the text file method means you will not need an html editor to modify or create your book's text.

Do I have to add separate Screens on my app for each of my book's chapters or pages? No. Both e-reader building methods described here use a single screen.

Can I read e-books from other sources with this app? No, but you might be able to modify the app to read other formatted e-books. App Inventor 2 can read and write files to an Android SDCard using the AI2 **File** control so it is possible to import other resources. The File control has no file selection tools because the File component in AI2 can not presently read the contents of a folder. However, if you know the name of a file on your device and the path to the file, code can be added to retrieve that file for use in the e-reader. The details of how to do that are not explained here and remains a project for you. You can develop this e-reader so it can read multiple books but how is not explained here. The html files in the html version of the app are stored as assets. Files stored in the AI2 assets area can not be modified by the app itself, so if you want to use html files that are not embedded in the app, code is required to retrieve those files from whatever directory you place them.

How are the chapters controlled? What AI2 controls we use to allow users of the app to change chapters or 'flip' through pages depends on the book text creation method. Html or AI2 Labels with Text blocks (the main mechanisms used to develop the two e-readers described here) need different handling. With html, html coding can control/select book chapters. You can change between html book chapter resources with a ListPicker by selecting each html chapter individually. **Call WebViewer.GoToUrl**, and point to each html chapter when the app is constructed if you built the app so each chapter is represented by a separate html.

The app you build needs to know where the resources are stored. They are stored in a different place in your phone/tablet than when used on the PC with AI2 after you install the app than where they are located while you build the app. When HTML documents are uploaded as assets into AI2 during development, you must place the resources in a location AI2 can find when the app is run. This development path location: file:///mnt/sdcard/AppInventor/assets/<NAME OF YOUR HTML FILE>.html is where the html(s) are stored during development of the app. It is a location that is actually on your PC. However, before building the app (the apk) for distribution, this instruction must be replaced with: file:///android_asset/<NAME OF YOUR HTML FILE>.html to allow your e-reader to find the appropriate html files once installed on an Android device. The blocks shown in this tutorial are coded to facilitate this change. The development path blocks are shown as enabled block, the production blocks are shown Disabled. Before you create your compiled app (make the apk), replace the development path blocks with the production blocks.

If you decide to create html chapter links within the html code itself, what chapter is displayed next can be made using html commands from within each html chapter. Readers can change chapters by clicking on links within each chapter.

If you use text blocks (filled with large amounts of text) to create the e-reader app, Chapter1, Chapter2 etc. are associated with an unique global variable. Use labeled buttons or a ListPicker to select the appropriate chapter global variable that will be used to display a particular chapter in a Label. One Label control is sufficient to display all the text of any chapter.

Do you need an image for the cover? Do we want one? Use an Image control, put the cover png or jpg image in the image control. Hide the control and use a button or Listpicker to show the

cover when needed (image.Visible = true). If developing using an html, embed the image in the html or use an Image control. Discussed below, treasure-island.jpg is an image of the cover of the 1911 version of the novel included in the aia as a resource. To fit nicely on most phones, the cover image for your own book should be 320 x 480 pixels in size and in no case larger than 320 pixels in the horizontal dimension.

How can I link the index and other pages of my book from the Cover Page? The tutorial demonstrates several ways to do this: Buttons, ListPicker, html code in the cover page html. I bet you can think of other ways. AI2 is fairly flexible in dealing with text.

How to make an html File from your own Text
To build the full featured e-reader, you need to understand some basics about using and makeing an html file.

A word processor, like Microsoft's Word or the free OpenOffice Writer can convert a text file into a simple html file. A text file is what is created when using a text editor program like *Notepad* on Windows. The text file usually has a name ending in the suffix .txt. Html editing programs like *Kompozer* (free) take a text file and convert it to an html. The editor helps you to change the color of the text, add images etc. How to use a html editing program is up to you; the steps are not described here.Find an editor, load it on your PC and you will discover it is no more difficult to use than a word processing program (and probably more fun).

It is relatively simple to build the full version of the app when you already have an html file to use in your application. Several html chapter files are part of this tutorial. You use the html file or files in several ways. Use a single html to read all the text in your book, all at once. Reading everything works fine and makes your app very small and uses very few blocks. If you make your book with a single html, you probably will not be able to start at or return to a specific chapter should you stop reading from the book before you finish the story. You are able to use your fingers to scroll through the text because the text is displayed in a WebViewer component. If you create an html for each chapter, you can allow you app reader user to return to a spot almost where he/she left off. How close one can return to the where one stops reading depends on how many chapters you have and how many html files you create.

If you do not have an html file of the text you want in your app, you can convert text to html in a word processor. With *OpenOffice Writer*, load your book text in the *Writer*; then **File>Save As > Save as type: > HTML document**. If you have access to an html editor, modify the appearance of the text, add images and do other neat things.

If you create a separate html for each chapter, you can control the 'chapters' three ways:

- 1) use links within the html chapter pages (you create these files outside of AI2)
- 2) or you might use a ListPicker as a table of contents to request individual html files to view
- 3) or you can use Buttons to point to individual html chapters.

Build the full Version of Treasure Island – the App

Treasure Island has 34 chapters. It also has six named parts. All the resources needed for the captioned novel are provided with this tutorial in six html files and a image file loaded as resources in the aia provided. These files are TreasureIsland_1.html through TreasureIsland_6.html. A seventh html provides documentation about the Gutenberg project. If you want all 34 chapters as html files, edit the existing files, saving each chapter as chapter_x.html all the way through 34. The html modification work is for you.

The example demonstrates three ways to control what part (or chapter) of the book is in view on your Android screen:

1) Buttons control methods to access the story's parts. It might mean a lot of buttons if your book has lots of chapters. Quite a problem, the screen is going to fill with buttons and there will not be much room to read the story.

2) A ListPicker control can change the chapters. The event handler in the ListPicker selects the appropriate html chapters. The control allows more choices using less screen space than Buttons and looks more polished.

3) Use a separate html to select the chapters. You do not have to have an html editor to edit simple html files to provide the required chapter links, a text editor like *Notepad* on Windows works fine. Similar tools are available on Macs and LINUX PCs. A 'control' file html is provided with the app and can be modified to suit your book chapters.

```
<!doctype html>

<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
<title>Second Page</title>
</head>
<body>
<p>Contents - Treasure Island<br>
<a href="treasureisland_1.html" target="_blank">Part I</a><br><br>
<a href="treasureisland_2.html" target="_blank">Part II</a><br><br>
<a href="treasureisland_3.html" target="_blank">Part III</a><br><br>
<a href="treasureisland_4.html" target="_blank">Part IV</a><br><br>
<a href="treasureisland_5.html" target="_blank">Part V</a><br><br>
<a href="treasureisland_6.html" target="_blank">Part VI</a><br><br>
<a href="treasureisland.html#down" target="_blank">Go back to main
page anchor</a>
```

```
</p>
</body>
</html>
```

The above example is the file named page2b.html in the full app aia template

Change the parts highlighted in blue in the control file to reference your own html file chapter and chapter title respectively using *Notepad*, then save the file as a resource, replacing any existing “chapter” file. When displayed “in the e-reader,” this file will look like this:

Contents - Treasure Island
Part I

Part II

Part III

Part IV

Part V

Part VI

Go back to main page anchor

4) A fourth method is to use an html editor to provide chapter links within the individual html files. This is doable but requires a knowledge of how an html editor works. The developer would insert links in the contents part of the book and in individual chapters. In the example project, links would go on the TreasureIsland_1.html in the text describing the Contents.

OK, let's go and build it. A template is provided that contains all the resources preloaded. Download the template (see below) and get started building the Project right away. Load the aia using **Project> Import project (aia) from my computer**.

The Design Screen

There are few controls in the e-reader and some of the controls are redundant; they are provided to demonstrate how you can use different methods to change chapters. After you finish the tutorial, you should remove the Button objects or ListPicker or html contents button from the screen. Pick one method to control chapters and your screen will be uncluttered.

This app requires some resources; an image and several html files. The resource files are located in the TresureIsland_Template aia.

These files are preloaded into the resources from the Design screen. The tutorial “Hello Purr for App Inventor 2” explains how to load a resource: http://appinventor.mit.edu/explore/ai2/hellopurr.html . When you load your book’s resources, they need to be loaded into Media The template aia already contains the following files loaded into the Media resources. When you modify the code to present your own text, you load the resources from the Designer screen.

<table>
<tr>
<td>

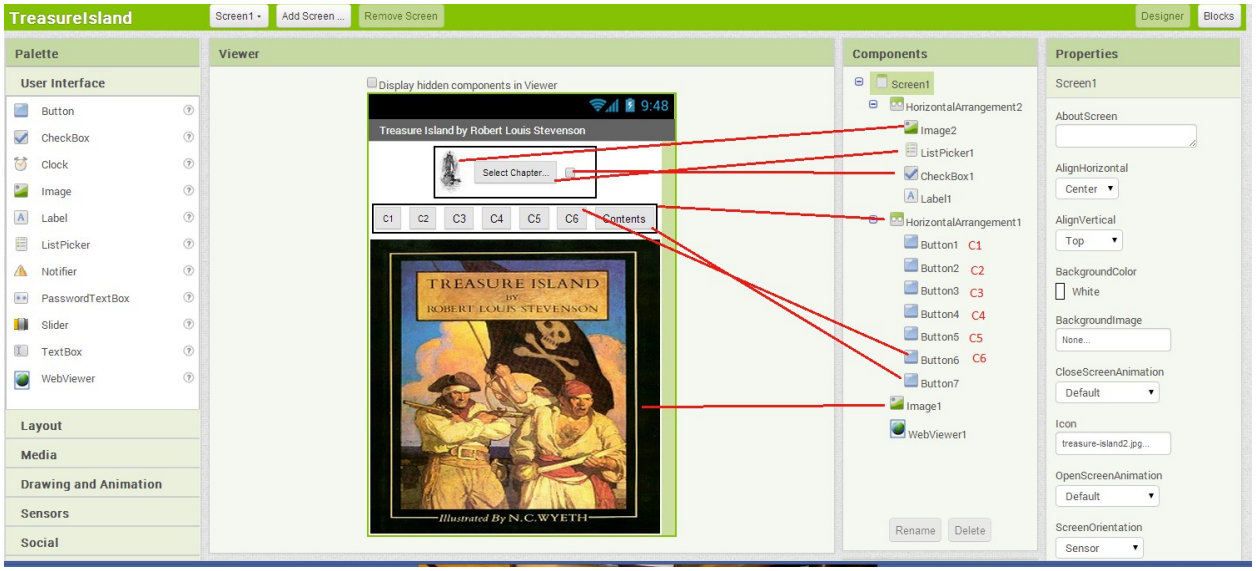
Gutenberg.html
TreasureIsland_1.html
TreasureIsland_2.html
TreasureIsland_3.html
TreasureIsland_4.html
TreasureIsland_5.html
TreasureIsland_6.html
page2b.html
treasure-island2.jpg
TIicon.jpg

</td>
<td>


</td>
</tr>
</table>

- Make your screen look like the following image:

-



- The positioning of the controls is not critical. Use the Designer screen image as a guide. Not every control setting is explained. The following *Properties* were set from the Designer screen. Fonts for Buttons were re-set from 14.0 (the AI2 default text size) to 12.0; Screen1 AlignHorizontal was set to Center; and the text was set in the Designer screen as shown in the image. Set Screen1.ScreenOrientation to **Sensor** (so users can view the novel in both Portrait and Landscape modes on their device). As a developer, you set these to what you want. A different way to set the Properties is shown in the Screen1.Initialize event handler block below.

- This text: “This eBook is for the use of anyone anywhere at no cost and with almost no restrictions whatsoever. You may copy it, give it away or re-use it under the terms of the Project Gutenberg License included with this eBook or online at www.gutenberg.org” should go in the Screen1 Properties> About Screen box on the right side of the Design screen. However, there is a bug in AI2 and this text is actually set using blocks as described below. The text is REQUIRED if you are going to use the Treasure Island htmls in the project and is needed to satisfy the Project Gutenberg License.

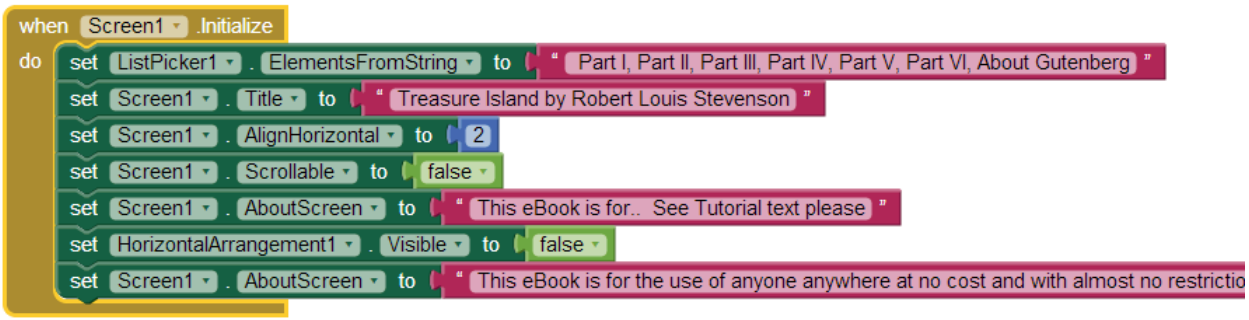
The Blocks

Once all the components are positioned on the Designer screen, start assembling Blocks.

In the Screen1.Initialize blocks :

- set ListPicker1.ElementsFromString to Part I, Part II, Part III, Part IV, Part V, Part VI, About Gutenberg
- set Screen1.Title to Treasure Island by Robert Louis Stevenson

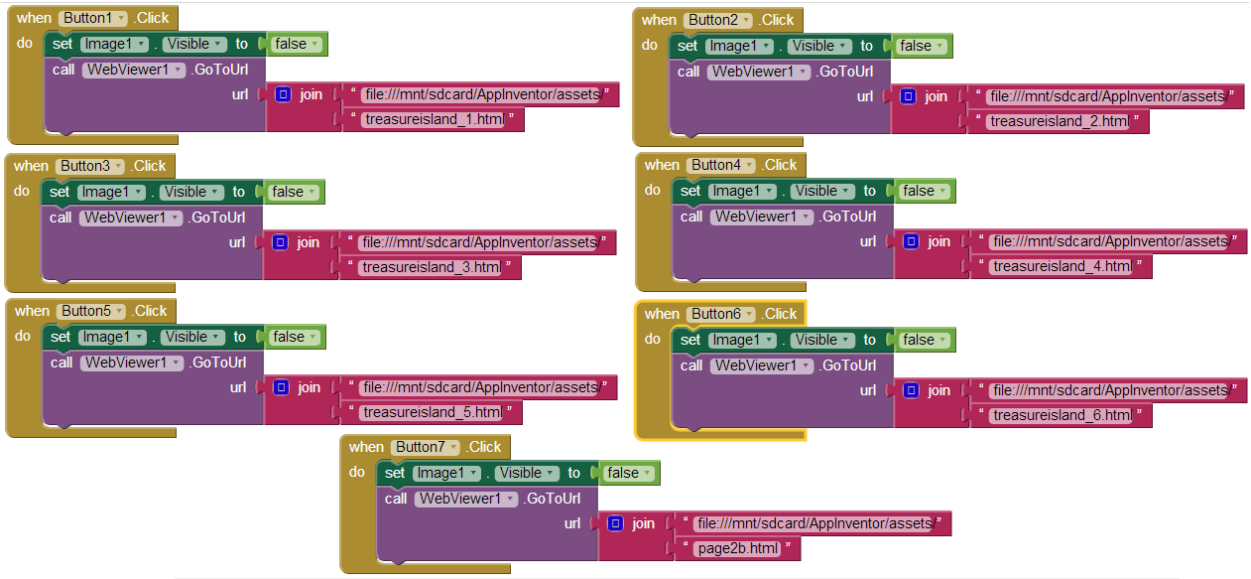
- set Screen1.AlignHorizontal to 2 (The choices are: 1 = left aligned, 2 = horizontally centered, 3 = right aligned).
- set Screen1.Scrollable to false
- set Screen1.AboutScreen to “This eBook is for the use of anyone anywhere at no cost and with almost no restrictions whatsoever. You may copy it, give it away or re-use it under the terms of the Project Gutenberg License included with this eBook or online at www.gutenberg.org” . This notice is required only if you use the Gutenberg text.
- set Screen1.ScreenOrientation to 4 (Other possible settings are: unspecified (-1), landscape (0), portrait (1), sensor (4), and user (2)). You do this so users can view the screen in either landscape or portrait mode.
- The Screen1.Initialize event handler:
-



In the Buttons event handler:

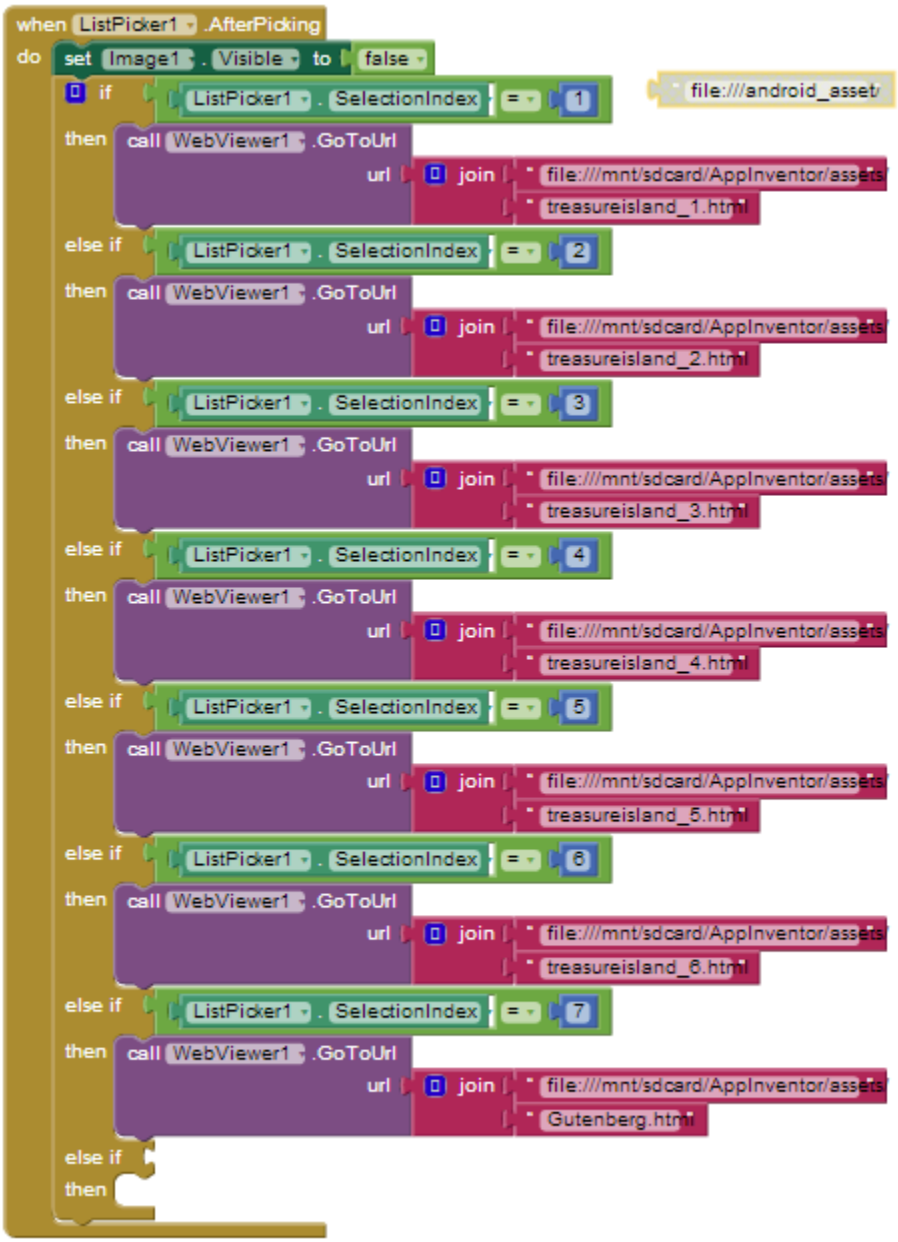
- The development code block is the first join, repeated for each Chapter: file:///mnt/sdcard/AppInventor/assets. Replace this block before you compile the apk with the text in the currently disabled text block shown below floating next to the ListPicker.AfterPicking blocks: file:///android_asset/ .

-



In the ListPicker.AfterPicking event handler:

- The development code block is the first join, repeated for each Chapter: file:///mnt/sdcard/AppInventor/assets. Replace this before you compile the apk with the text in the currently disabled text block shown below: file///android_asset/ .



-

This is ALL the blocks you need for the app and more. The empty puzzle piece is where you would add additional chapters or instructions.

Fast Track

A template aia is provided to get you started. The template allows you start the project with all the resources loaded. The template contains a blank Designer screen but has all the necessary resources pre-loaded. The template contains all the resources and nothing else. tutorial. T

Are You really done?
Make the screen individualized; change the background color; experiment with making the controls rounded. Put your own short story or novel in place of the default text files. Add an Icon to the Screen1 (so when the app compiles, you have a personalized e-reader icon instead of the default icon. A 48x48 pixel image works well. The icon image goes into the Resources. Once the image is there, go to Screen1 > Properties on the Designer screen and select that image in the box below the Icon script. Add a 'book cover' Image .. see the html version description). Add a book cover. Follow the example from the html version of the e-reader. Are you going to write a short story or novel? Add your copyright notice; Screen1 > Properties on the Designer screen, find the AboutScreen box and type something like: Copyright © 2014 by Your Name. In the United States, the **copyright** notice consists of three elements: The © **symbol**, or the word '**Copyright**'; The year of first publication of the **copyrighted** work; and who holds the copyright.

Build the “bunny” Version of Treasure Island – the App

Try a different way to design your e-reader if your book is short. It is easy to use a single Text block attached to a global chapter variable to display large amounts of text. The global texts in turn are displayed in a Label control. A view of the final product for this variation in technique is displayed at the start of this tutorial (the third image)..

The example simple text app demonstrates two methods to control what chapter to view with the e-reader:

1) Buttons control methods to access chapters. If your book has lots of chapters, it might mean a lot of buttons. Quite a problem. The screen is going to be filled with buttons and there will not be much room for the text.

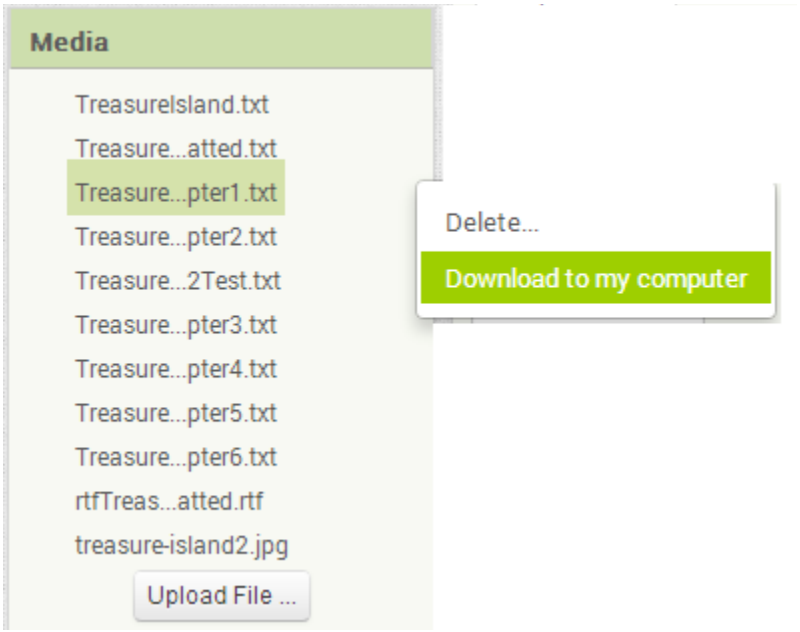
2) A ListPicker control to change chapters shown on the screen. Instead of using the event handler in the buttons to **set Label1.Text to** and **set Label2.text to** to the appropriate chapters, give chapter control to the ListPicker. The control allows more chapter choices using less screen space and looks more polished.

Only a few resources required for this version of the e-reader. None of the ‘resources’ are loaded into the app’s AI2 resources, instead the required text is part of the blocks code. There are five text files provided. The contents of each text file goes into the appropriate global variable. Load the file into a Text block associated with a global chapter variable using a text editor; SelectAll in *Notepad* to select all the text (or use another text editor) and drag and drop

the text to an AI2 Text block connected to each chapter variable. Yes, all that text will fit into the Text block. There probably is a limit to the total size of the text that can be displayed this way. Be aware, it is possible to place the entire text of the example novel into a single label using this methodology.

Let's Build It

A template is provided that pre-loads the five chapter variables. You will not have to drag and drop the example text, but you will eventually have to drag any other text you want in the chapters. The template also contains copies ofthe five text files. These copies are not needed to build the app (they are already loaded in the template) but they are provided so you can understand where they came from. You may want to see what the text looks like. After you load the template aia, you should either delete all the text files in Media (resources) or you can experiment with them using the File component to load these resources into the app's chapter global variables as an exercise. You can move the text files to your PC by left clicking on the text file; then selecting "Download to my computer" as illustrated in the diagram below.



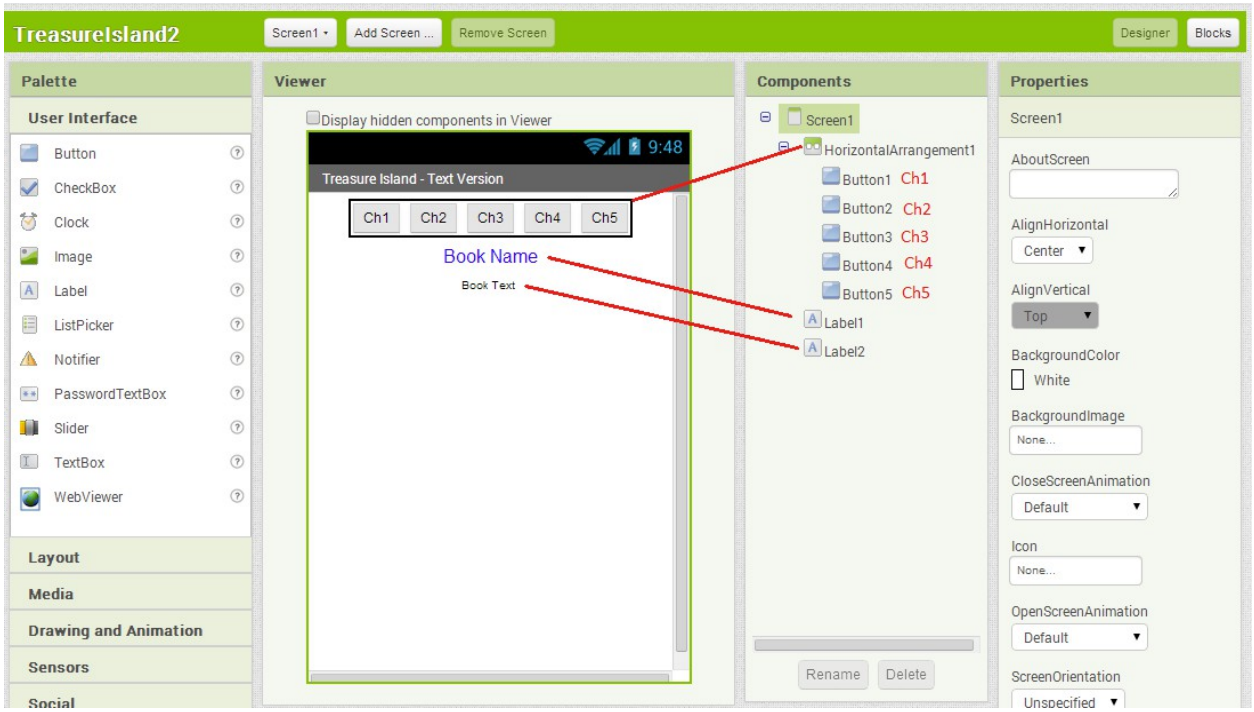
You may want to load the template aia now.

The Design Screen

There are few controls. Two ways to control viewing the book's chapters are shown. You may want to remove the Button objects or ListPicker from the screen after you finish the tutorial. Pick one method and your screen will be uncluttered.

Make your screen look like the following image using the red hints as a guide:

•



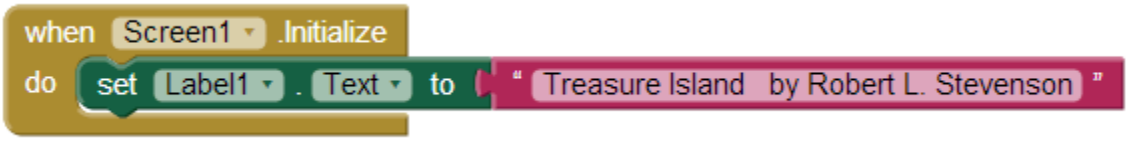
The positioning of the controls is not critical. The image is a only a guide. Not every setting control is explained in the Designer screen image. The following *Properties* are set from the Designer screen. Fonts for Buttons were changed from the AI2 default font size of 14.0 and reset to 12.0; Screen1 AlignHorizontal was set to Center. The text was set in the Designer screen as shown in the image. Set Screen1.ScreenOrientation to Sensor (so users can view the book in both *Portrait* and *Landscape* modes on their device). As a developer, you set these to what you want.

The Blocks

Once all the components are positioned on the Designer screen, start assembling Blocks.

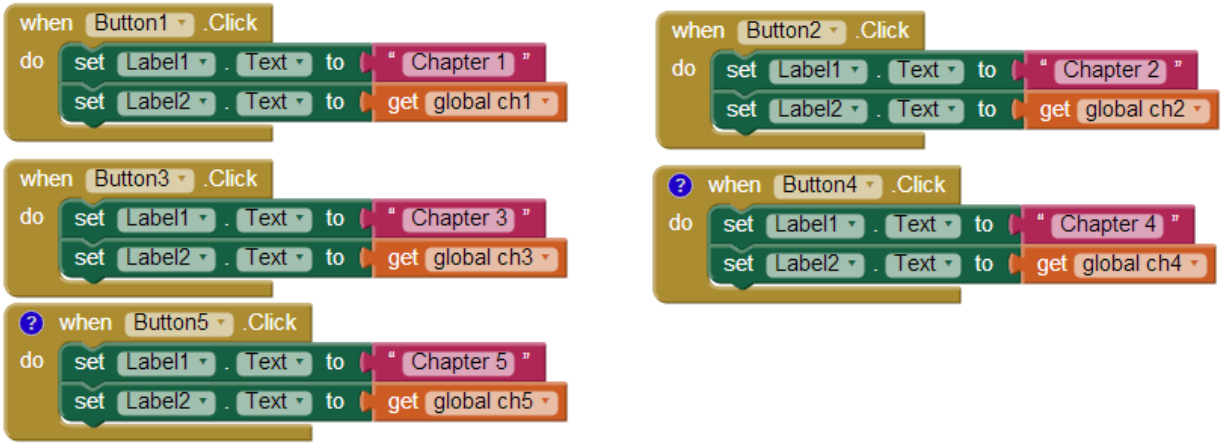
The Screen1.Initialize event handler:

This block contains the book's title. Type it in.

- 

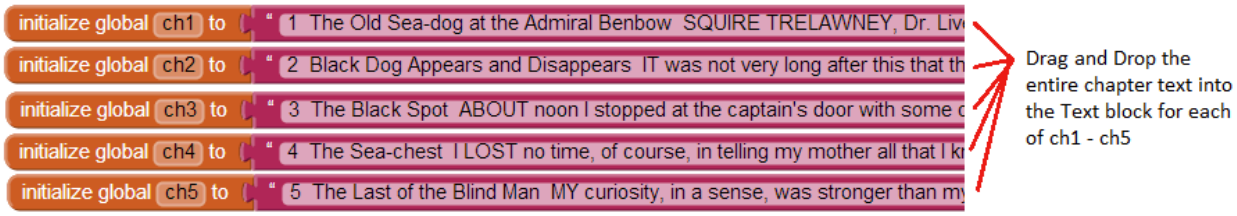
The Buttons event handler:

Chapter titles go here. The text of the chapters is in the global variables as set below.

- 

The Global Variable "Chapters" in the template were made by loading the Treasure Island.txt file (found in the resources of the template aia) into an editing program (like *Notepad* in Windows); selecting the chapter text and dragging and dropping the text into a Text box. Be careful in this step. The Block editor is a bit fussy and it is possible you might have issues with blocks appearing to vanish. The blocks are still there, the editor is just having issues finding them. To return them, either click on the Screen1 button on the browser to reset the blocks, or use the left arrow key on your keyboard to move the cursor to the left (just keep clicking) or left click on the Blocks white screen and select Arrange Blocks Vertically. All three methods will return the Blocks screen to “normal.”

Be aware, if you are using the template, ALL of the chapter “loading” of the global chapter variables has been done for you. This part of the tutorial is here to show how to load your book chapters.

- 

These are all the blocks you need if you want to use Buttons to select chapters. If you rather use a ListPicker to change chapters, then you will need to add a ListPicker control and some additional blocks. Follow the ListPIcker example in the full version of the e-reader app.

There is an alternative way to fill the global chapter text. You can use File control. Add the file control to your project. Now, use **call File.ReadFrom FileName** to read from the individual chapter texts, for example **call File.ReadFrom FileName .** How to do this is not part of the tutorial, however, the blocks you may need to experiment are provided in the template and these demonstration blocks are **Disabled**. Enable them later when you are finished with the tutorial and experiment.

Formatting Plain Text to Behave

Text files, grabbed with *Notepad* use plain text file formatting. What it means, is the text is not formatted with any control characters. The lack of control characters causes an issue when introduced into a Label component on AI2. When the text is dragged and dropped to a label, there is no paragraph formatting. To compensate for this, a developer has the option to either insert a control character (\n) into the text at paragraph breaks. You MUST add the \n's within the purple Text box, one at a time (Sorry, you can not embed the control characters in the text, then drag a text with the \n controls already there. That does not work. AI2 will not recognize the \n when the text is imported that way.. When rendered on the Android screen, the \n will not appear and instead a line feed will result. To get a paragraph break, insert to successive \n characters like thus: \n\n.

Can you automate this process possibly for your own Word or Notepad file? The control character *Notepad* and *Word* uses for paragraphs is ^p^p^p . When the file is viewed, you can not see these characters, however, they are present in the text. Unfortunately you can not replace all those with \n\n . A big gotcha' is the Gutenberg text has no paragraph formatting embedded, so you have to treat the attached files that are garaged in the template resources manually. You have one other possibility, that is to open the unformatted text files, go through the text, determine where a paragraph ends, and set the text to look like you want.

Are You really done?

Make the screen individualized; change the background color; experiment with making the controls rounded. Put your own short story or novel in place of the default text files. Add an Icon to the Screen1 (so when the app compiles, you have a personalized e-reader icon instead of the default icon. A 48x48 pixel image works well. The icon image goes into the Resources. Once the image is there, go to Screen1 > Properties on the Designer screen and select that image in the box below the Icon script. Add a 'book cover' Image .. see the html version description). Add a book cover. Follow the example from the html version of the e-reader. Add your copyright notice; Screen1 > Properties on the Designer screen, find the *AboutScreen* box and type something like: Copyright © 2014 by Your Name. In the United States, the **copyright** notice consists of three elements: The © **symbol**, or the word '**Copyright**'; the year of first publication of the **copyrighted** work; and who holds the copyright.

The TreasureIsland_text_Template.aia contains additional code blocks (the blocks are set to "Disabled") that may help you to add additional features to the app. The extra code blocks allow you to experiment with a "cover' for your book and to experiment with loading chapter files dynamically. How to do these things is up to you.

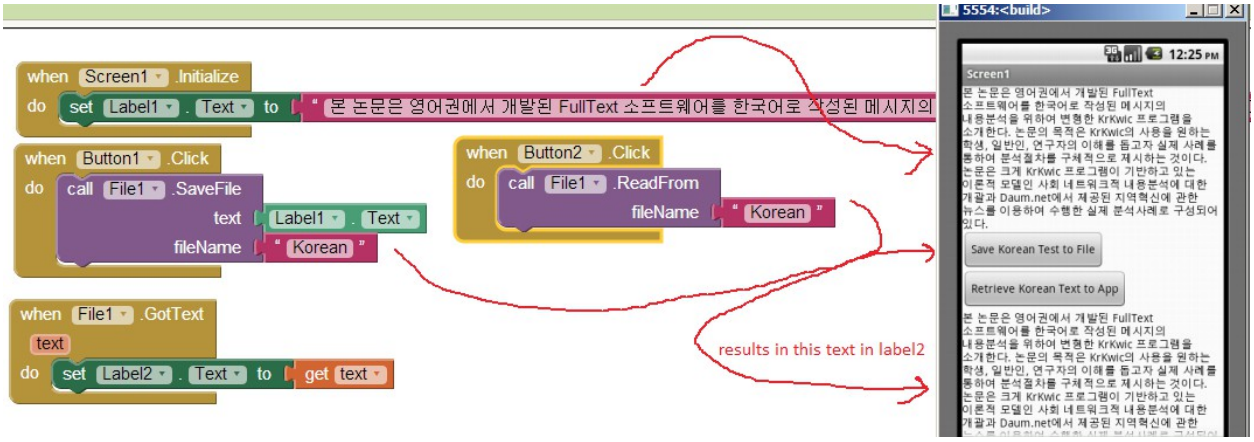
You may want to experiment a lot. Have fun.

Notes

Not all the html files or text files provided in the tutorial are formatted to look "pretty." You can provide more paragraph adjustments etc. as a learning activity or want to provide a slick, finished product.

Extras

English is not your language or you want to write your novel in a different language? Here is a demonstration showing how you might use a different alphabet or language in your ebook. The demo



uses a Korean text file. You can use these techniques in other apps. Drag and drop some Korean character text into the Text box associated with Label1.Text. The Button1.Click sends the label1 text to a file called Korean. Button2.Click retrieves the Korean file and post its contents to label2.

Note, the easiest way to get a text or csv file into your Android using AI2 is to use Windows to get the required text or csv; copy it with an editing program like *Notepad* or by directly copying text from a Web page. Drop the captured text in to a text block within the Block editor. Be aware, this demonstration saves the file within the app. You can also save and retrieve from the SD card ... follow the directions with the File documentation http://ai2.appinventor.mit.edu/reference/components/storage.html#File .

This technique should work for any language in Unicode that reads from left to right (English, Korean, German, Spanish etc) but might not work with languages reading from right to left (Mandarin, Arabic etc.).

Download Source Code

Download the source code to your computer, then open App Inventor, click Projects, choose Import project (.aia) from my computer..., and select the source code you just downloaded.
TreasureIsland_Template.aia (loads all the resources; you do all the coding);
TreasureIsland_text_Template aia (loads all the resources; you do all the coding).

Credits