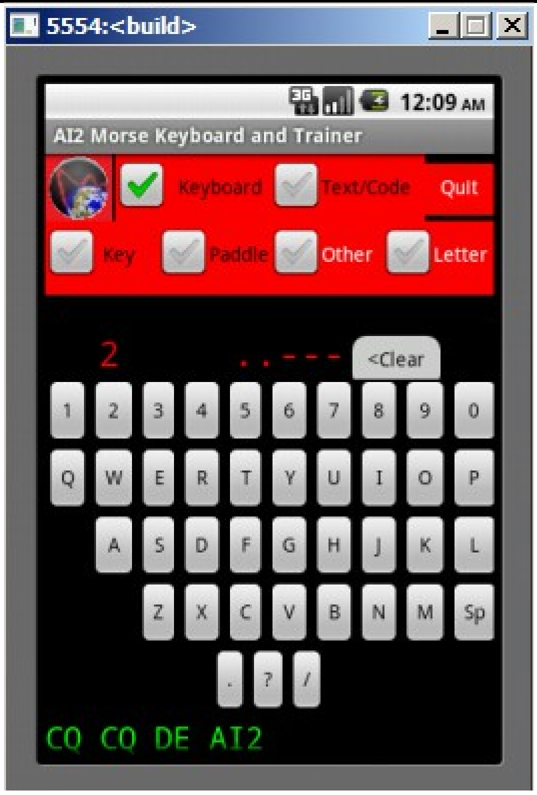
CQ de AI2 Morse Code -- a tutorial demonstrating how to use layouts instead of screens and the *Any components*

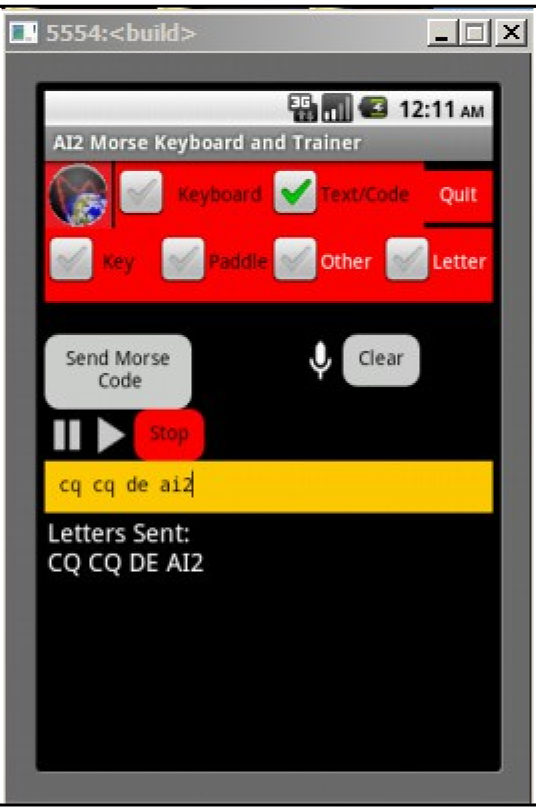
Morse code .. the dots and dashes comes to life with the sound of the “dits and dahs.” The Morse code app uses lots of buttons, layouts and sound files to help users learn Morse code. “CQ,” in radio parlance, is a general call to anyone…”please talk to me.” “de” is shorthand for “this is.” Amateur Radio operators still preserve and use the early form of digital communication called Morse code and use cw (continuous wave) to transmit messages back and forth on specially allocated radio frequencies.

Originally, Morse code (often called *cw*) was transmitted (encoded) with a hand key, essentially a touch switch. The dit, produced by a very short touch, the dah with a slightly longer touch of the key. It was soon discovered one could transmit cw faster using a “bug” or paddle that transmitted a dot automatically when pressed in one direction and a dah when pressed in the other. Years later, keyboards were introduced. This AI2 app is an opportunity to experiment with all three techniques of generating Morse code tones using a modern cell phone. The app allows you to produce a “canned” (previously prepared) Morse message too. The message can be paused or aborted.

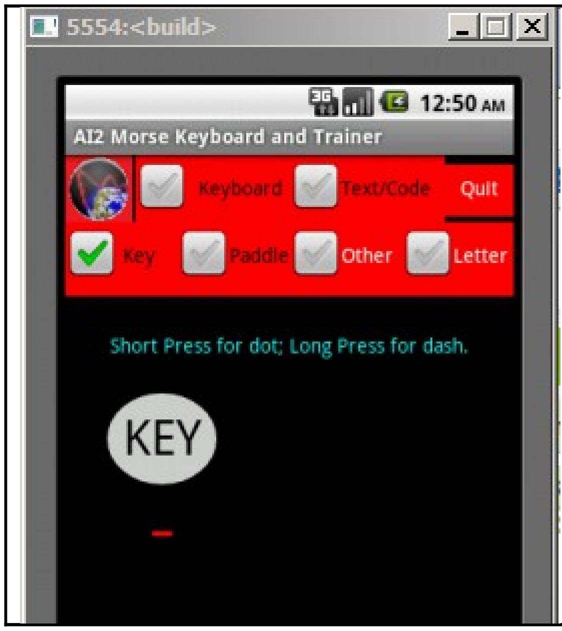
The App Screens



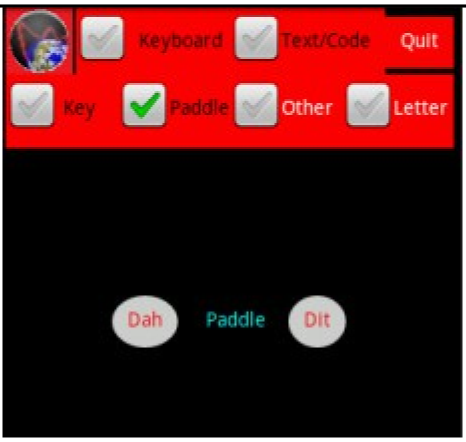
The Keyboard Screen



The Text to Code Player Screen


5554:<build>
12:50 AM
AI2 Morse Keyboard and Trainer
Keyboard
Text/Code
Quit
Key
Paddle
Other
Letter
Short Press for dot; Long Press for dash.
KEY

A simple telegraph key



A simple paddle key

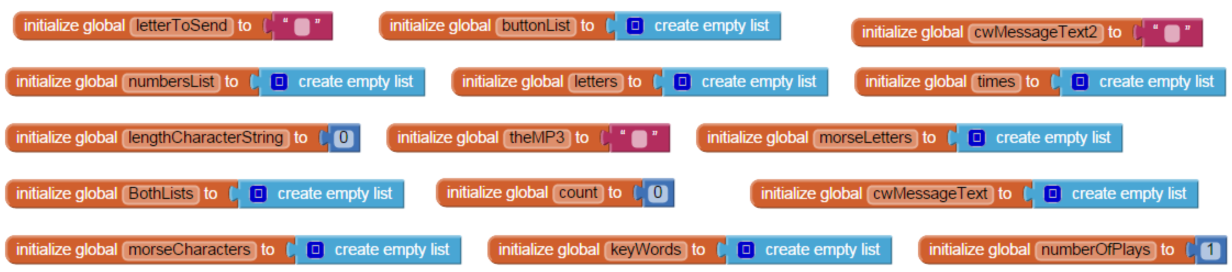
The KEY invokes a dot with a short press; it invokes a dash with a long press, on a phone, this is difficult to find the correct rhythm to send a Morse character (A mechanical key is much easier). The left Paddle invokes a dash, the right a dot. Most users find the paddle easier to use than the simple key.

Logic and Design

The app is a very primitive artificial intelligence, creating the dots and dashes of Morse code, creating the sounds that make up Morse code and displaying graphic images of the messages produced.

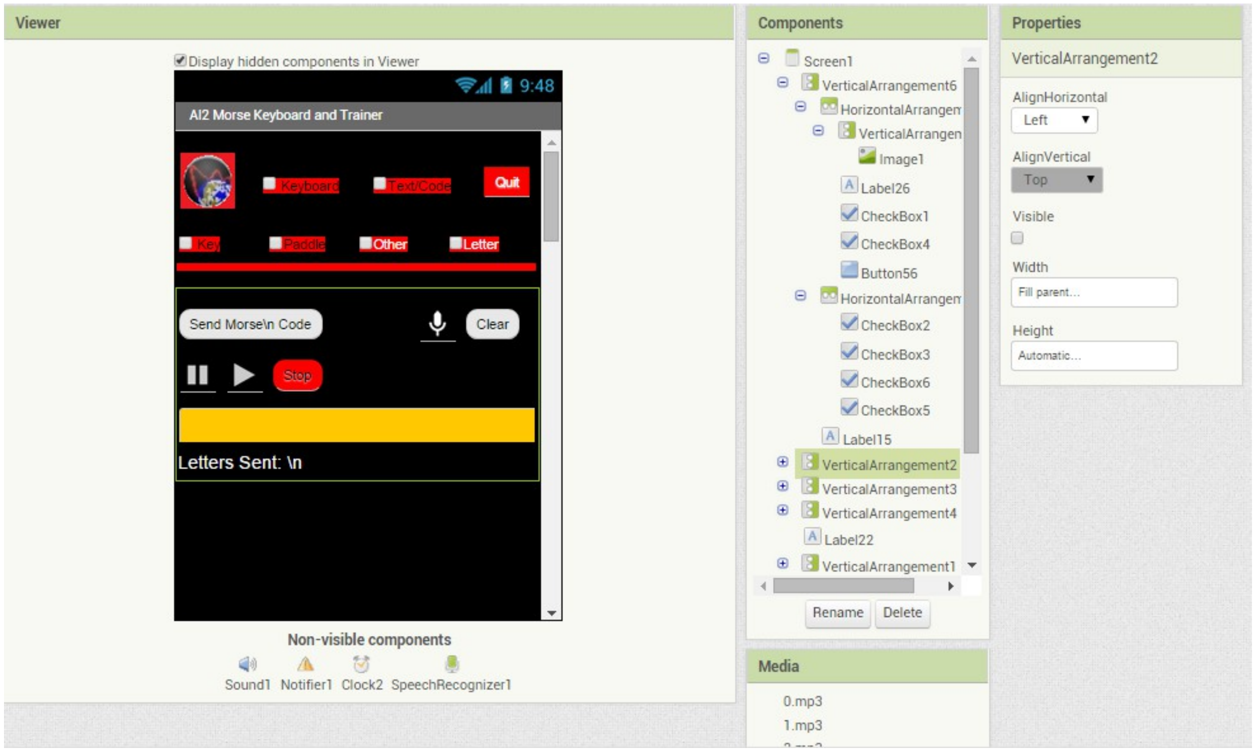
A single AI2 Screen is used; six Vertical Arrangements are displayed or hidden using their Visibility property (set variously to true or false) simulating additional screens.

The variables required are:



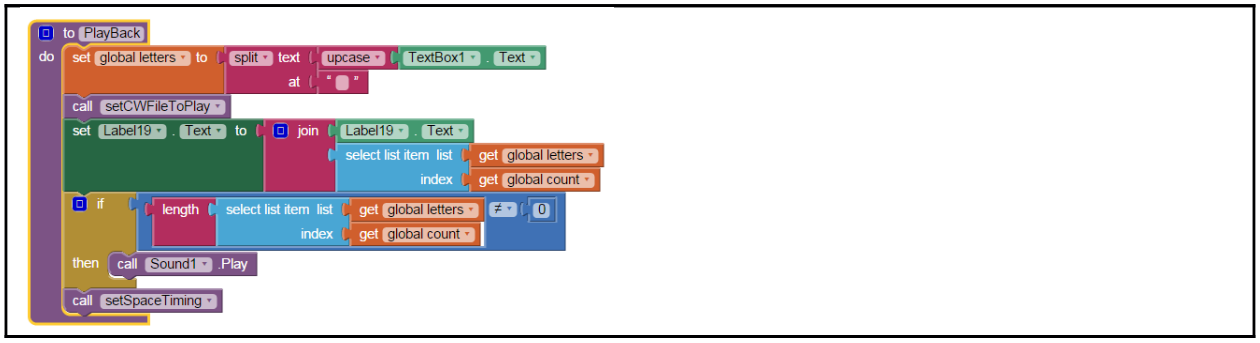
```
initialize global letterToSend to " "
initialize global buttonList to create empty list
initialize global cwMessageText2 to " "
initialize global numbersList to create empty list
initialize global letters to create empty list
initialize global times to create empty list
initialize global lengthCharacterString to 0
initialize global theMP3 to " "
initialize global morseLetters to create empty list
initialize global BothLists to create empty list
initialize global count to 0
initialize global cwMessageText to create empty list
initialize global morseCharacters to create empty list
initialize global keyWords to create empty list
initialize global numberOfPlays to 1
```

The *Designer* screen looks like this. It makes use of a variety of Vertical and Horizontal Layouts and a Table Arrangement. Make your own screen to suit your preferences.

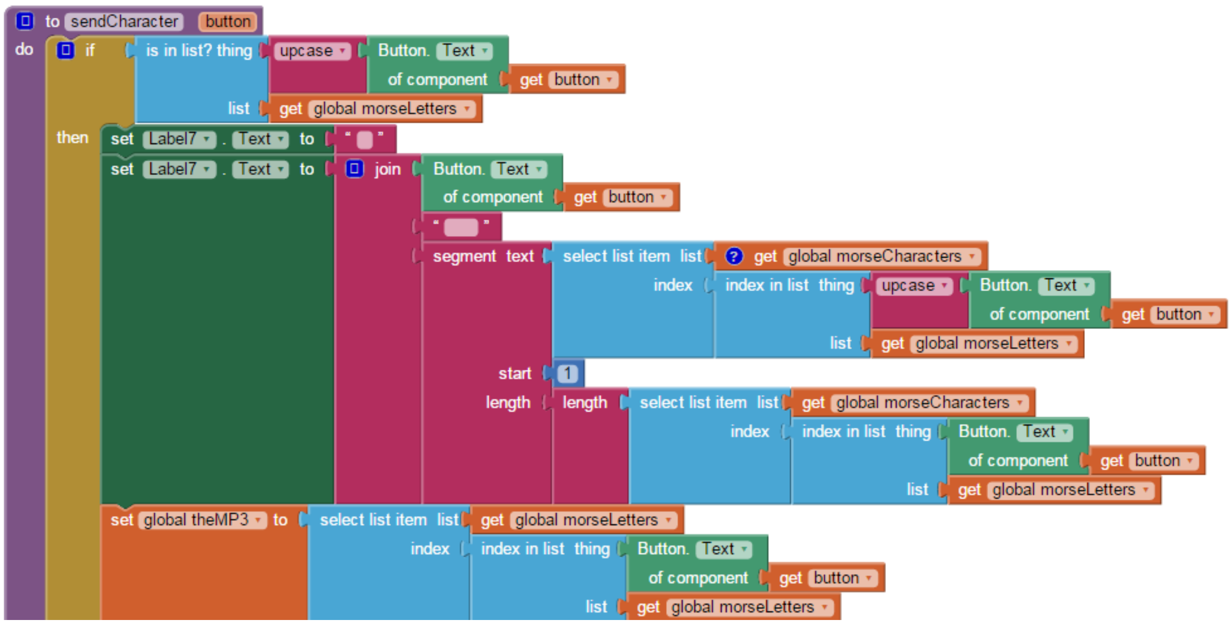


VerticalArrangement2
AlignHorizontal: Left
AlignVertical: Top
Visible: ☐
Width: Fill parent...
Height: Automatic...

The Code .. Two Procedures do most of the work within the app . *PlayBack* and *sendCharacter* contain the core coding.

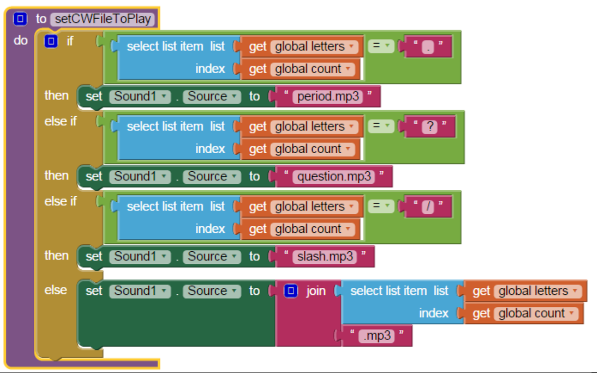


```
to PlayBack
do  set global letters to  split text  upcase  TextBox1 . Text
                                 at  " "
    call setCWFileToPlay
    set Label19 . Text to  join  Label19 . Text
                                 select list item  list  get global letters
                                                   index get global count
    if    length  select list item  list  get global letters   ≠  0
                                    index get global count
    then  call Sound1 .Play
    call setSpaceTiming
```

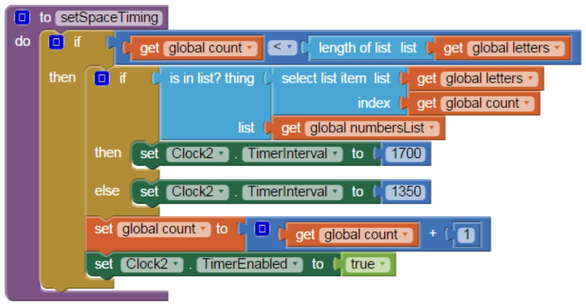


```
to sendCharacter button
do  if  is in list? thing  upcase  Button.Text
                                     of component  get button
                list  get global morseLetters
    then  set Label7.Text to " "
          set Label7.Text to  join  Button.Text
                                      of component  get button
                                    " "
                                    segment  text  select list item  list  get global morseCharacters
                                                                     index  index in list  thing  upcase  Button.Text
                                                                                                            of component  get button
                                                                                           list  get global morseLetters
                                             start  1
                                             length  length  select list item  list  get global morseCharacters
                                                                               index  index in list  thing  Button.Text
                                                                                                              of component  get button
                                                                                                     list  get global morseLetters
    set global theMP3 to  select list item  list  get global morseLetters
                                            index  index in list  thing  Button.Text
                                                                           of component  get button
                                                                  list  get global morseLetters
```

Several other procedures supplement them. *Playback* is used on the Text to Code screen to start the app displaying characters and playing the appropriate mp3 files. Text can be input into the Text field using the virtual keyboard or the mike input. The playback can be paused, resumed or stopped/aborted.



```
to setCWFileToPlay
do  if  select list item  list  get global letters  =  "."
                          index  get global count
    then  set Sound1.Source to "period.mp3"
    else if  select list item  list  get global letters  =  "?"
                               index  get global count
    then  set Sound1.Source to "question.mp3"
    else if  select list item  list  get global letters  =  "/"
                               index  get global count
    then  set Sound1.Source to "slash.mp3"
    else  set Sound1.Source to  join  select list item  list  get global letters
                                                         index  get global count
                                      ".mp3"
```



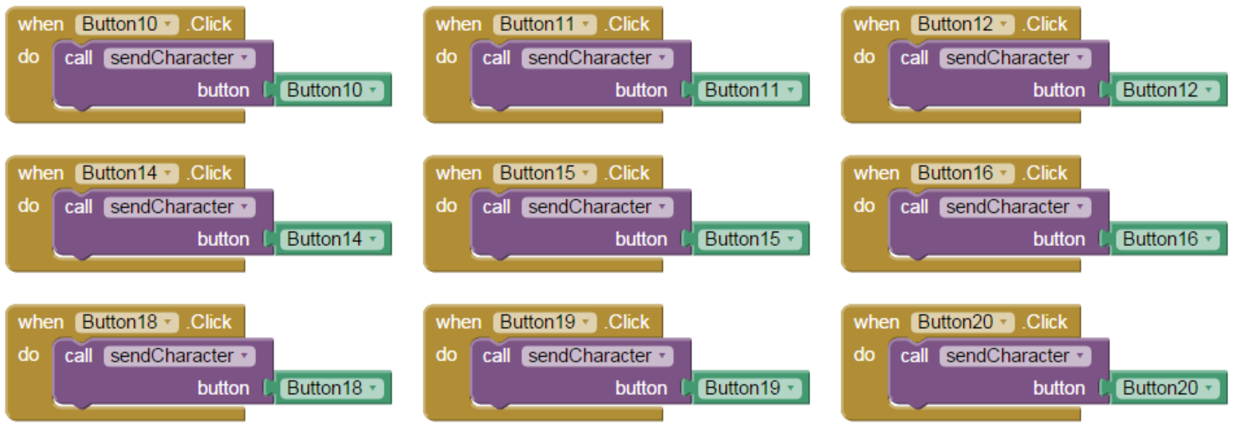
```
to setSpaceTiming
do  if  get global count  <  length of list  list  get global letters
    then  if  is in list? thing  select list item  list  get global letters
                                                   index  get global count
                          list  get global numbersList
          then  set Clock2.TimerInterval to 1700
          else  set Clock2.TimerInterval to 1350
          set global count to  get global count  +  1
          set Clock2.TimerEnabled to true
```

The setCWFileToPlay Procedure *The setSpaceTiming Procedure*

The *setCWFileToPlay* procedure selects the appropriate mp3 file to play based on the characters input into the textbox field. The period (.), question mark (?) and slash (/) are prohibited in file names. The mpg files are named for the letter/character they represent so these characters must be treated specially as they are named period,question and slash.mpg.

The *setSpaceTiming* procedure recognizes that it takes longer to play some code character sound files than others; the mp3's are longer. The Clock2.TimerInterval is set to provide a delay in ms between the times a file is invoked and playing of the subsequent mp3. Numerals are composed of the most dot/dash elements and, in general, take the longest time to 'send'.

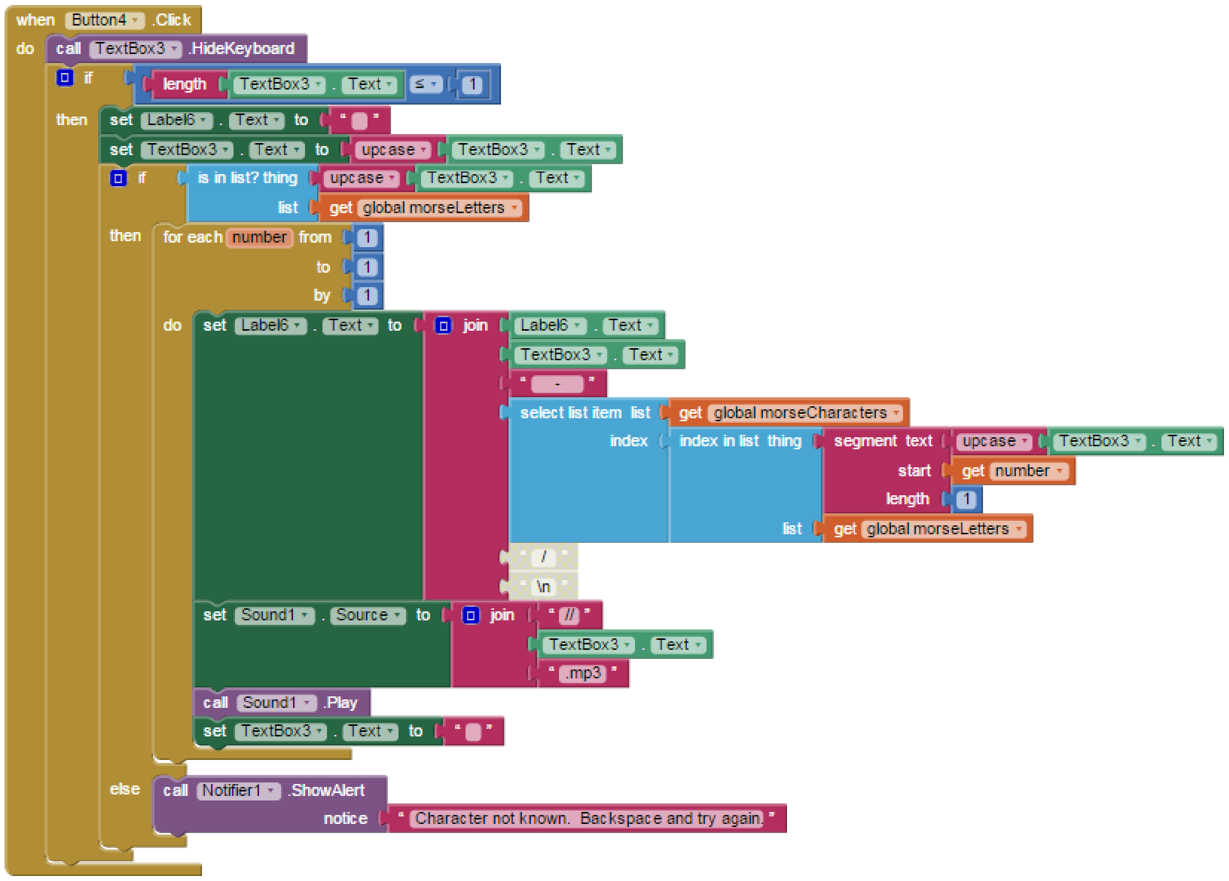
The forty keyboard buttons are set up like this:



```
when Button10 .Click
do  call sendCharacter
                button  Button10

when Button11 .Click
do  call sendCharacter
                button  Button11

when Button12 .Click
do  call sendCharacter
                button  Button12

when Button14 .Click
do  call sendCharacter
                button  Button14

when Button15 .Click
do  call sendCharacter
                button  Button15

when Button16 .Click
do  call sendCharacter
                button  Button16

when Button18 .Click
do  call sendCharacter
                button  Button18

when Button19 .Click
do  call sendCharacter
                button  Button19

when Button20 .Click
do  call sendCharacter
                button  Button20
```

These are only a small sample of the Buttons involved in the Morse Keyboard. The event handler contains a procedure call invoked by each button. Consult the aia to understand the assignment of buttons to characters. Buttons5-29 are the characters, Buttons41-50 are the numerals 1...0 and Buttons57,58,59 are the period, question mark and slash.

The algorithm used to display a single character is the Button4.Click event handler code:

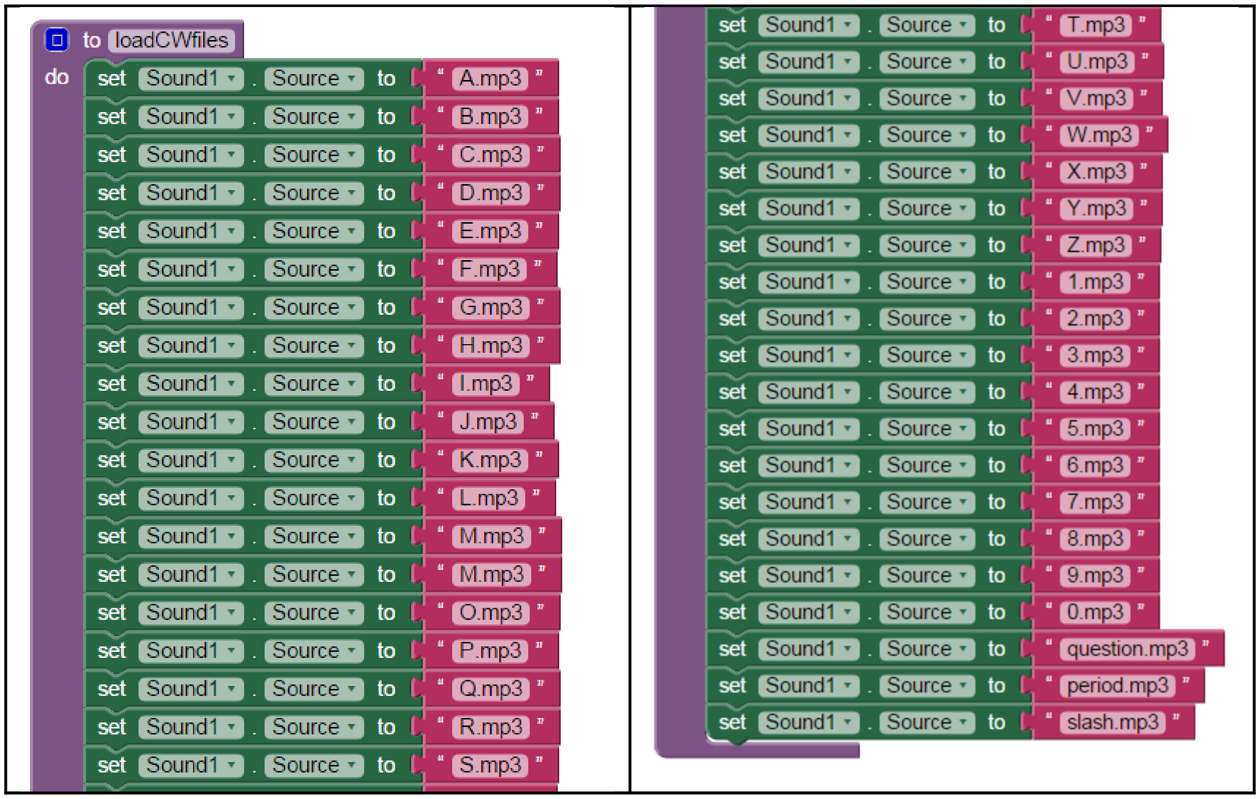


```
when Button4 .Click
do  call TextBox3 .HideKeyboard
    if    length TextBox3 . Text ≤ 1
    then  set Label6 . Text to " "
          set TextBox3 . Text to upcase TextBox3 . Text
          if    is in list? thing upcase TextBox3 . Text
                            list  get global morseLetters
          then  for each number from 1
                                to 1
                                by 1
                do  set Label6 . Text to join  Label6 . Text
                                               TextBox3 . Text
                                               " - "
                                               select list item  list  get global morseCharacters
                                                                 index index in list  thing segment  text   upcase TextBox3 . Text
                                                                                                     start  get number
                                                                                                     length 1
                                                                                      list  get global morseLetters
                                               " / "
                                               " \n "
                    set Sound1 . Source to join  " // "
                                                 TextBox3 . Text
                                                 " .mp3 "
                    call Sound1 .Play
                    set TextBox3 . Text to " "
          else  call Notifier1 .ShowAlert
                                notice " Character not known.  Backspace and try again. "
```

The Morse Characters and Sounds

The dots and dashes most people associate with Morse when they see a graphical representation of the code are a substitute for the alternating short and long tones of the code. A mp3 sound file is necessary for each letter of the alphabet, numeral and punctuation. These mp3 files were obtained from *Wikipedia* as created by Joe deRose and released as public domain. The oog files were converted to mp3 format.

After coding this project, it became evident all the code can be represented by two mp3 sound files, the sound files representing the letters E and T . This could be implemented with more complicated coding and timing routines...a project for you.

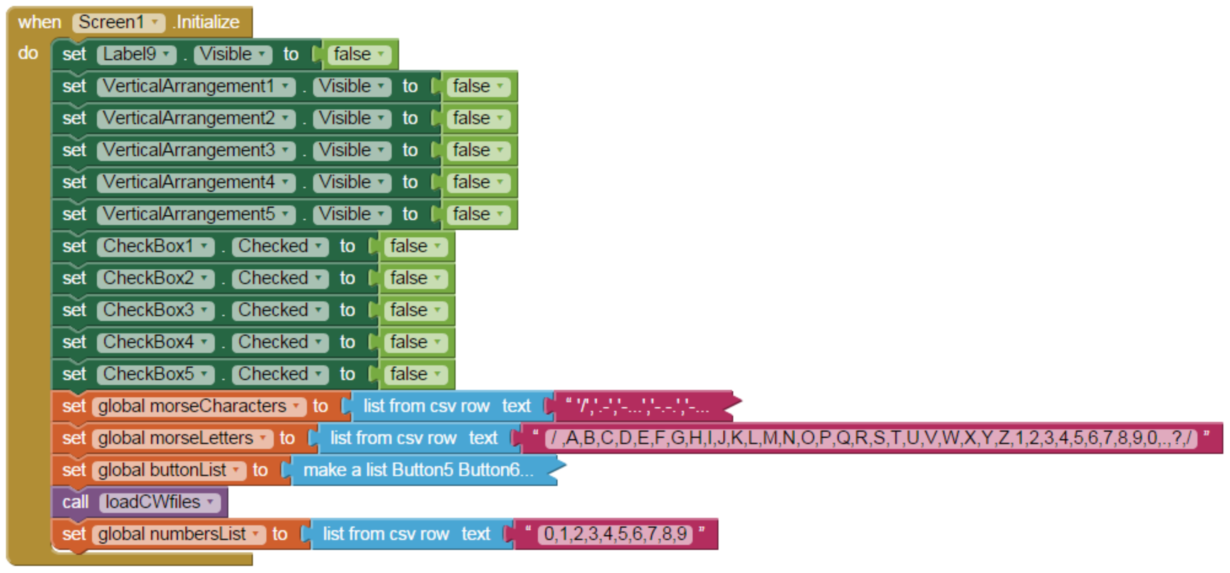


```
to loadCWfiles
do  set Sound1 . Source to "A.mp3"
    set Sound1 . Source to "B.mp3"
    set Sound1 . Source to "C.mp3"
    set Sound1 . Source to "D.mp3"
    set Sound1 . Source to "E.mp3"
    set Sound1 . Source to "F.mp3"
    set Sound1 . Source to "G.mp3"
    set Sound1 . Source to "H.mp3"
    set Sound1 . Source to "I.mp3"
    set Sound1 . Source to "J.mp3"
    set Sound1 . Source to "K.mp3"
    set Sound1 . Source to "L.mp3"
    set Sound1 . Source to "M.mp3"
    set Sound1 . Source to "M.mp3"
    set Sound1 . Source to "O.mp3"
    set Sound1 . Source to "P.mp3"
    set Sound1 . Source to "Q.mp3"
    set Sound1 . Source to "R.mp3"
    set Sound1 . Source to "S.mp3"
```

```
    set Sound1 . Source to "T.mp3"
    set Sound1 . Source to "U.mp3"
    set Sound1 . Source to "V.mp3"
    set Sound1 . Source to "W.mp3"
    set Sound1 . Source to "X.mp3"
    set Sound1 . Source to "Y.mp3"
    set Sound1 . Source to "Z.mp3"
    set Sound1 . Source to "1.mp3"
    set Sound1 . Source to "2.mp3"
    set Sound1 . Source to "3.mp3"
    set Sound1 . Source to "4.mp3"
    set Sound1 . Source to "5.mp3"
    set Sound1 . Source to "6.mp3"
    set Sound1 . Source to "7.mp3"
    set Sound1 . Source to "8.mp3"
    set Sound1 . Source to "9.mp3"
    set Sound1 . Source to "0.mp3"
    set Sound1 . Source to "question.mp3"
    set Sound1 . Source to "period.mp3"
    set Sound1 . Source to "slash.mp3"
```

The mp3 sound files are included in the aia file that demonstrates the app.

The sound files, lists of the letters/characters and their “dot-dash” representations and various settings are loaded when the app loads using the *loadCWfiles* procedure shown above and in the Screen1.Initialize block.



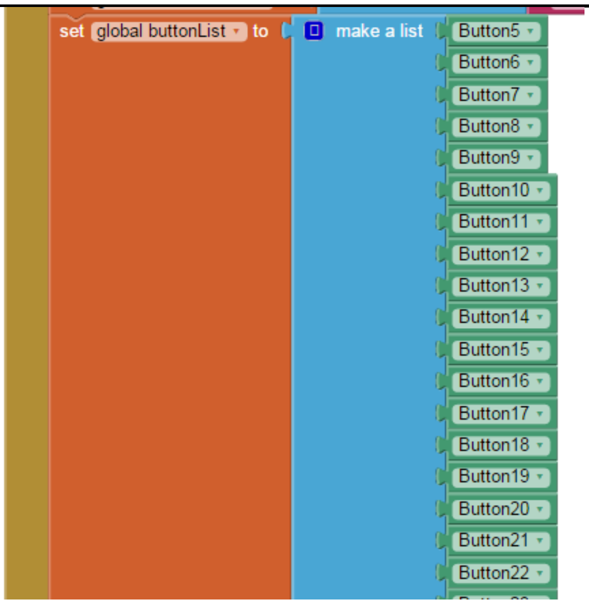
```
when Screen1 .Initialize
do  set Label9 . Visible to false
    set VerticalArrangement1 . Visible to false
    set VerticalArrangement2 . Visible to false
    set VerticalArrangement3 . Visible to false
    set VerticalArrangement4 . Visible to false
    set VerticalArrangement5 . Visible to false
    set CheckBox1 . Checked to false
    set CheckBox2 . Checked to false
    set CheckBox3 . Checked to false
    set CheckBox4 . Checked to false
    set CheckBox5 . Checked to false
    set global morseCharacters to list from csv row text " '/','.-','-...','-.-.','-...
    set global morseLetters to list from csv row text " / ,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,1,2,3,4,5,6,7,8,9,0,.,?,/ "
    set global buttonList to make a list Button5 Button6...
    call loadCWfiles
    set global numbersList to list from csv row text " 0,1,2,3,4,5,6,7,8,9 "
```

The *morseCharacters* collapsed string is:
'/','.-','-...','-.-.','-..','.','..-.','--.','....','..','.---','-.-','.-..','--','-.','---','.--.','--.-','.-.','...','-','..-','...-','.--','-..-','-.--','--..
','.----','..---','...--','....-','.....','-....','--...','---..','----.','-----',.-.-.-',..--..',-...-'
The letters/numerals/punctuation (*morseLetters*) are:

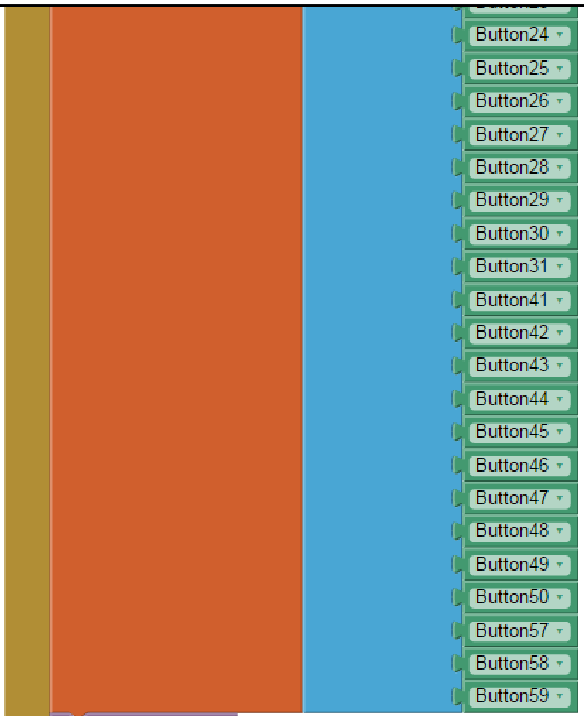
/ ,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,1,2,3,4,5,6,7,8,9,0,.,?,/

Both strings are stored internally in the app as a csv (comma separated value) file and are converted with an AI2 block into a List.

The buttonList looks like this:



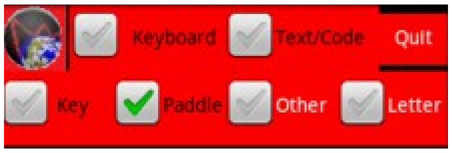
```
set global buttonList to make a list Button5
                                     Button6
                                     Button7
                                     Button8
                                     Button9
                                     Button10
                                     Button11
                                     Button12
                                     Button13
                                     Button14
                                     Button15
                                     Button16
                                     Button17
                                     Button18
                                     Button19
                                     Button20
                                     Button21
                                     Button22
```

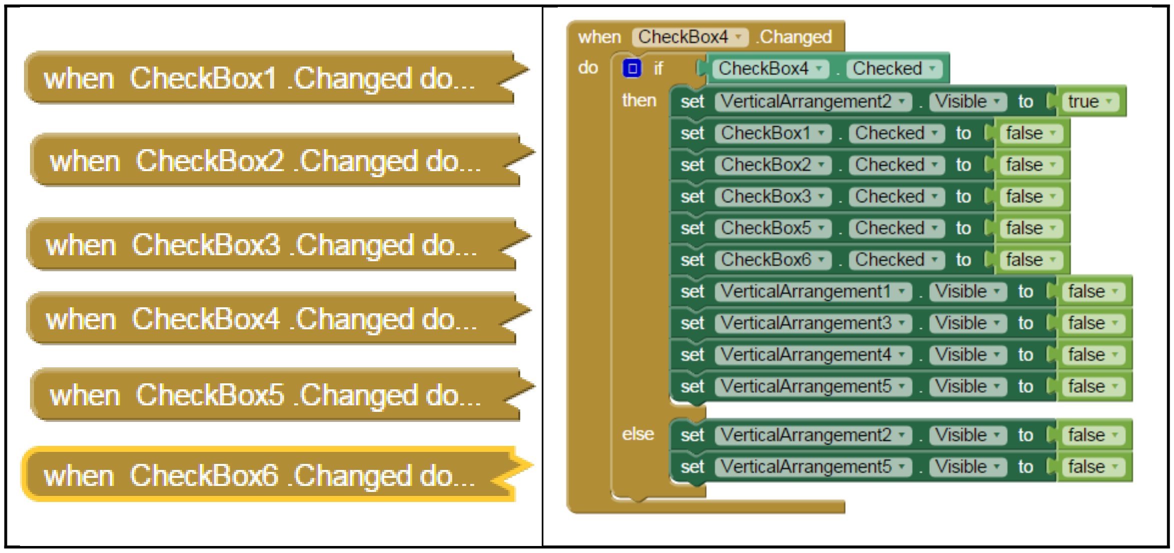


```
Button24
Button25
Button26
Button27
Button28
Button29
Button30
Button31
Button41
Button42
Button43
Button44
Button45
Button46
Button47
Button48
Button49
Button50
Button57
Button58
Button59
```

Where the green buttons are the last object in the Buttons control.

An “array” of six check boxes is used as the app’s menu. All the check boxes follow essentially the identical form. When one box is active, it unchecks the other check boxes and hides Vertical Layouts not needed while displaying the selected Vertical Layout.



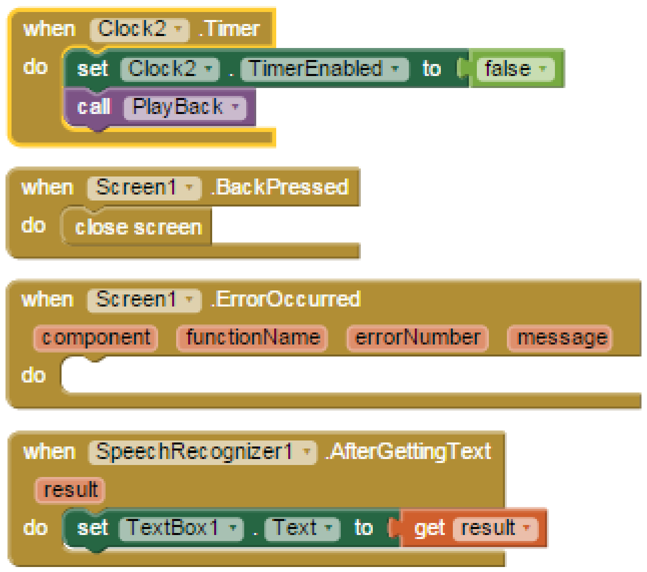


```
when CheckBox1 .Changed do...
when CheckBox2 .Changed do...
when CheckBox3 .Changed do...
when CheckBox4 .Changed do...
when CheckBox5 .Changed do...
when CheckBox6 .Changed do...
```

```
when CheckBox4 .Changed
do  if  CheckBox4 . Checked
    then  set VerticalArrangement2 . Visible to true
          set CheckBox1 . Checked to false
          set CheckBox2 . Checked to false
          set CheckBox3 . Checked to false
          set CheckBox5 . Checked to false
          set CheckBox6 . Checked to false
          set VerticalArrangement1 . Visible to false
          set VerticalArrangement3 . Visible to false
          set VerticalArrangement4 . Visible to false
          set VerticalArrangement5 . Visible to false
    else  set VerticalArrangement2 . Visible to false
          set VerticalArrangement5 . Visible to false
```

The above is an example of one of the six check boxes. Be aware, each is slightly different but all contain the blocks hiding arrangements and selectively setting the Checked property of the other check boxes, effectively creating a GroupBox of check boxes.

More blocks are needed:



```
when Clock2 .Timer
do  set Clock2 . TimerEnabled to false
    call PlayBack

when Screen1 .BackPressed
do  close screen

when Screen1 .ErrorOccurred
 component  functionName  errorNumber  message
do

when SpeechRecognizer1 .AfterGettingText
 result
do  set TextBox1 . Text to get result
```

A speech recognizer is use with the Text to Code screen. The Clock2 is used with the Player routine and uses recursion to time the playing of the sound files.

Be aware, there are several more blocks that implement the code paddle and key that are not represented in this tutorial but are present in the source aia.

Is there a source file? **Download Source Code**

Download the source code to your computer, then open App Inventor 2, click Projects, choose Import project (.aia) from my computer..., and select the source code you just downloaded.

There is one aia file: **Morse.aia** (the complete project) . The links to the Google folder with the aia is shown on the main tutorial blog page.

Coding Advice

1) *Android* and *App Inventor* are text case sensitive. When coding names of objects, images and things, remember *blue.png* is NOT *Blue.png* or *blue.jpg* for example.
2) Save your work frequently. Either create a new aia or (very easily) *Projects>Save project as* saves a copy to the MIT server.
3) Stop coding when you are tired. OK, don't and see what happens.
4) You do not have to replicate the Designer screen.
5) Please do not copy this app and attempt to sell it on *Google Play*, THIS tutorial is copyrighted, if you want something for *Google Play* use the principles you learned and code your own app.
6) For your own personal use, have fun with *CQ de AI2* It is free for personal use. *Do learn some basic programming techniques and learn the basics of Morse code.*
7) Have fun.