Thinking about user interface design

1. The vending machine

   The vending machine is always used in the public place; thus, its design of interface should be simple enough for everyone to use. The first strength of this interface is “visible”. Although people may not know the exact brand names of the food they are looking for, the actual products after the display window help users to make up their decision. The second strength is the clear model of machine operation. The transparent interface let users to understand how the machine works without any past experience or manuals. People can easily observe that the product will fall down from the cabinet once put in enough money and match the machine model with their mental model. The visible feature of the vending machine also brings the third advantage: the interface is easy to maintain and repair. The maintainer can easily notice if there is short of any product and refill it. This visible feature of the vending machine is the most successful design point to me. The weakness of the vending machine might be the limitation of payment. Sometime, in the international airport, it’s difficult for the user to find out enough coins with correct currency to use the machine. Another weakness of the vending machine is it’s somehow hard to reach the product for the older adults. To bend over is a dangerous posture of the older adults generally. The other weakness of the vending machine is the limitation of type of food; the interface is designed for the food that can be dropped down from the cabinet without damaging its taste. If we can design the vending machine which metaphors the real cabinet in the kitchen, users can open the door and pick up the food without bending or dropping. This might bring more varieties of food to the current vending machines.

2. The digital wristwatch

   The digital wristwatch is a universal product design, and people from different parts of the world should understand it easily. The strength of my digital wristwatch is its various displays of the time, date, and year. The multiple functions of the watch can help users gather enough information one at a time. The second strength of the watch is the different modes of the time. Users can choose 12-hour time or 24-hour time as they like. The varieties of time display may fit into people with different mental models of the time. The third strength of the watch is the color use made the more useful time information more salient. There is no extra perceptual loading of the users to look at the time even if there are more information then the usual watch. The weakness of the watch is it’s really very difficult to set up the time. There are four
buttons on the watch, but you’ll never know what this is for before trying. The second weakness is the same appearance of the four buttons. Users cannot remember the functions of buttons because they all look like the same. If the designer can use color cue to differentiate the four buttons, the user will be easier to remember how to set up the time. Finally, there are too many modes of the digital wristwatch such as the alarm, the calendar, the timer, and the usual clock. However, we don’t need all of them in the wristwatch. If we can modify the interface as the most 2 or 3 functions we used, the operation of the digital wristwatch should be much easier. Thus, users can build the mental model of the wristwatch based on these basic functions then.

3. The rice cooker

I use the rice cooker a lot. The rice cooker is really popular in Asia, especially for the Asian students who studied abroad. The rice cooker has only two functional controls, one power stick is to turn on the cooker; the other button is to keep heating the food. Thus, it’s really easy to operate the rice cooker. The other strength is the contextual cue of the button. While heating the food, the light of the button will be shining. Thus, users can easily figure out the food is still cooking without checking the food manually. The third strength is the “mapping” of the interface. If users need to turn on the cooker, they have to push the stick downward; then put the lid down. If the food is done, users need to lift the lid, and the stick will spring up. The analogue between the hand movement and power stick will help users understand the rationale of the rice cooker. The weakness of the rice cooker might be the unclear illustration of adding the water. People who don’t often cook might never figure out how much water they need to add in the rice cooker. It’s inconvenient and unreasonable for users to check the volume of water every time. There should be some straightforward aid for the users to operate intuitively. The other weakness is the limitation of the interface. We can only know if the food is well done or not through the power stick and it’s impossible for user to estimate how long it will take from the interface of the rice cooker. Finally, there is a possible bug of the rice cooker. No matter the user turns on the power stick or not, they can use the button to reheat the food. Thus, users sometimes forgot to turn off the reheat button after the rice cooker off, and it’s dangerous to keep heating for a long time. There should be a protective function that users cannot reheat the food after the power stick springing up.

Learning to Listen to Users

1. Graduate student, age 24, male, 8-year computer user

   First, he mentioned the confusion of “fn” key and “ctrl” key on the IBM laptop. We usually used the two keys to combine other keys for different advanced functions; however, the nearby places do make the user confuse them often. One of the possible
interventions is to set these two keys apart to avoid confusion. The other possible intervention is to change the tactile sense of the two keys; thus, user can differentiate the keys without looking.

Second, we discussed the statistic software, STATA, which mainly used for the database analysis. During using this software, there were sometimes some error messages to alarm users the possible mistakes. However, the error messages usually showed something like “ERROR04352” instead of detailed descriptions. Users still don’t know how to fix the problem while having some mistakes. The possible intervention is to make the error message more user-friendly; thus, users can modify the error on their own easily.

Third, we discussed the phenomena that there are more and more unnecessary functions embedded in the cellphone, such as games, calculators, and blog-posting. However, those functions are not useful for all the users which may not only tax the memory of the cellphone, but also make the menu more complicated. The possible solution is to let the users personalize the functions they want after buying the cellphone.

2. Financial secretary, age 22, female, 6-year computer user

First, we discussed the different “silent mode” of the cellphone. In some brands of the cellphone, the “silent mode” implied completely silent. But in some brands of the cellphone, it still rings with the “alarm setting”. The mode error happens a lot for the current cellphone users, since people changed their phone more frequently. The possible solution will be to unify the “mode setting” of the cellphones.

Second, the revision function of the Microsoft Word has been complained for years. Sometimes, the software will help you correct the misspelling; other times, it will change the original meaning to the other ways. Although the user can turn off the automatic revising function, people usually forgot. Thus, if we can change the default of the function as “off”, this might solve some problems.

Third, she mentioned that the different search experience of the search engines. While using the Yahoo search engine, it will automatically open a new window after clicking the hyperlink. But, the Google search engine won’t open a new window while clicking the hyperlink. Thus, users sometimes get confused to do the online search. The possible solution will be the eligibility to set the personalized search preference; thus, users don’t have to change their search habits while using different search engines.

3. Vice director, age 52, male, 30-year computer user

First, he mentioned the recent smart phone, which is the latest multifunctional cellphone, has a function which can “push” the email from the web to the phone daily. However, it’s difficult for the user to find out the important message from the tons of
emails in the smaller-screen cellphone. Further, the memory of the cellphone is considerably small which users need to delete most of the useless messages everyday. The good deed of push-mail function does add the extra costs to users. The possible way to fix this problem is to set up the filter for email. For example, the cellphone only imports the email from some significant others or selects the email which contained some specific keywords. This filter should be customized to the individual user; thus, the issue of information safety can also be secured.

Second, we discussed the update function of the computers. Once connecting to the Internet, all the programs keep searching the possible updates for the users. Thus, the user needs to spend lots of time reviewing if the update is necessary or not. The possible solution is to turn off the automatic update of the software. Although it sometimes doesn’t work at all.

Third, there was also the problem with the “silent mode” of the cellphone. Some cellphones don’t have the function of purely silence, but there are lots of options, such as vibration, vibration and bell, bell after vibration. Users may press the wrong button during the emergent meeting, and the phone might ring unexpectedly. The possible solution is to understand what kinds of modes the user really need. Then, the display of different modes of sounds should be really straightforward and organized. Thus, the user won’t be embarrassed while the cellphone ringing in the important occasion.

**Power Law of Practice**

The participant completed 20 trials with 2 mistakes; the accuracy rate was 90%. Table 1 showed the reaction time of the correct 18 trials, and it was clear to see the reaction time was decreased with the trial practicing. Then, we used the power law of practice and got the leaning constant about 0.5 which was similar to common situations.

\[ T_n = T_1 \times n^{-\alpha} \]

Thus, \( 3712 = 16999 \times 20^{-\alpha} \); \( \alpha = 0.508 \)

From Figure 1, we can see the leaning curve of the participant was following the power function. At the beginning of the experiment, the participant spent lots of time on the first trial to figure out the last 8 letters. After the first trial, the participant kept repeating the 8 letters silently during the experiment. The verbal loop somehow helped the participant memorize the 8 letters backwards. In addition, the repeated finger movement let the participant acquire the skill implicitly.
Table 1. The reaction time of all 18 correct trials

<table>
<thead>
<tr>
<th>Trial</th>
<th>RT (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16999</td>
</tr>
<tr>
<td>2</td>
<td>7238</td>
</tr>
<tr>
<td>3</td>
<td>7863</td>
</tr>
<tr>
<td>4</td>
<td>8169</td>
</tr>
<tr>
<td>5</td>
<td>7006</td>
</tr>
<tr>
<td>6</td>
<td>3925</td>
</tr>
<tr>
<td>7</td>
<td>3779</td>
</tr>
<tr>
<td>8</td>
<td>5791</td>
</tr>
<tr>
<td>9</td>
<td>6285</td>
</tr>
<tr>
<td>10</td>
<td>4455</td>
</tr>
<tr>
<td>11</td>
<td>6219</td>
</tr>
<tr>
<td>12</td>
<td>5091</td>
</tr>
<tr>
<td>13</td>
<td>5556</td>
</tr>
<tr>
<td>14</td>
<td>4838</td>
</tr>
<tr>
<td>15</td>
<td>4433</td>
</tr>
<tr>
<td>16</td>
<td>3390</td>
</tr>
<tr>
<td>17</td>
<td>3353</td>
</tr>
<tr>
<td>18</td>
<td>3712</td>
</tr>
</tbody>
</table>
Choice Reaction Time
1. Using Hick’s Law
   \[ RT = 0 + 1 \log(12) = 1.079, \]
   We will need 1.079 ms to select an item from the menu.

2. (1) Using Fitt’s Law, let’s assume the area of one item is 1 unit. Thus, the static area is 8 units and the dynamic area is 4 units. And we assumed the user did the sequential search vertically.
   --50/50: \[ MT = 0 + 1 \log(0.5\times2 + 0.5\times8)/(1+1) = 0.3979 \text{(ms)} \]
   --75/25: \[ MT = 0 + 1 \log(0.75\times2 + 0.25\times8)/(1+1) = 0.243 \text{(ms)} \]
   --90/10: \[ MT = 0 + 1 \log(0.9\times2 + 0.1\times8)/(1+1) = 0.1139 \text{(ms)} \]
   --The minimum choice time: \[ MT = 0 + 1 \log(1\times2/(1+1)) = \text{nearly 0(ms)} \]
   --The probability which lead the maximum choice time will be 100% in the static area.

3. The limitation of using Hick’s law in the real life is people usually need to make a choice before their movement. It’s unrealistic to assume people know where their target is. Further, the Hick’s law didn’t include the different size of target areas which might influence the movement time in the task.
Source code of question 3

This program is written by e-prime, a kind of common software for experimental psychology task design.

C:\Documents and Settings\Wai-Tat Fu\Desktop\CS465RT.ebs
Generated on: 9/8/2008  15:15:43

This file generated with E-Studio interface.
ALL RIGHTS RESERVED

Legal use of this experiment script requires a full E-Prime or Runtime License.

Author: '    (University of Illinois)

Option CStrings On
Dim ebContext as Context

Sub InitTextDisplayDefaults(theTextDisplay As TextDisplay)
    If theTextDisplay Is Nothing Then Exit Sub
    theTextDisplay.X = "center"
    theTextDisplay.Y = "center"
    theTextDisplay.Width = "100%"
    theTextDisplay.Height = "100%"
theTextDisplay.ForeColor = CColor("black")
theTextDisplay.BackColor = CColor("white")
theTextDisplay.BackStyle = "opaque"
theTextDisplay.BorderColor = CColor("black")
theTextDisplay.BorderColor = CLng("0")
theTextDisplay.XAlign = "center"
theTextDisplay.YAlign = "center"
theTextDisplay.AlignHorizontal = "center"
theTextDisplay.AlignVertical = "center"
theTextDisplay.WordWrap = True
theTextDisplay.ClearAfter = CLogical("No")
theTextDisplay.FontName = "Courier New"
theTextDisplay.FontSize = "18"
theTextDisplay.FontBold = CLogical("Yes")
theTextDisplay.FontItalic = CLogical("No")
theTextDisplay.FontUnderline = CLogical("No")
theTextDisplay.FontStrikeout = CLogical("No")

End Sub

'-------------------------------------------------------------------------------
' Instance Declarations
'-------------------------------------------------------------------------------
Dim Display As DisplayDevice
Dim Sound As SoundDevice
Dim Keyboard As KeyboardDevice
Dim Mouse As MouseDevice
Dim SessionProc As Procedure

Dim List1 As List

Dim proc1 As Procedure

Dim Letter6 As TextDisplay
Dim Letter6EchoClients As EchoClientCollection

Dim Letter4 As TextDisplay
Dim Letter4EchoClients As EchoClientCollection

Dim Letter2 As TextDisplay
Dim Letter2EchoClients As EchoClientCollection

Dim Letter1 As TextDisplay
Dim Letter1EchoClients As EchoClientCollection

Dim Letter3 As TextDisplay
Dim Letter3EchoClients As EchoClientCollection

Dim Letter5 As TextDisplay
Dim Letter5EchoClients As EchoClientCollection

Dim Letter7 As TextDisplay
Dim Letter7EchoClients As EchoClientCollection

Dim Letter8 As TextDisplay
Dim Letter8EchoClients As EchoClientCollection

Dim Instruction As TextDisplay
Dim InstructionEchoClients As EchoClientCollection

Dim Letter9 As TextDisplay
Dim Letter9EchoClients As EchoClientCollection

'--------------------------------------------------------------------------------------------------
' Package Declare Script
'--------------------------------------------------------------------------------------------------

'--------------------------------------------------------------------------------------------------
' User Script
'--------------------------------------------------------------------------------------------------

'--------------------------------------------------------------------------------------------------
' Package Global Script
'--------------------------------------------------------------------------------------------------

'--------------------------------------------------------------------------------------------------
Sub SessionProc_Run(c as Context)
    List1.Run c

    #If RUNTIME_VERSION_MAJOR > 1  Or (RUNTIME_VERSION_MAJOR = 1 And RUNTIME_VERSION_MINOR >= 2) Then
        ' Log clock timing information
        c.SetAttrib "Clock.Information", Clock.Information
    #End If

    c.Log
End Sub

Sub proc1_Run(c as Context)

    Instruction.InputMasks.Reset

    If Keyboard.GetState() = ebStateOpen Then
        InstructionEchoClients.RemoveAll
        Instruction.InputMasks.Add Keyboard.CreateInputMask("{ANY}", ",", CLng(Instruction.Duration), CLng("1"), ebEndResponseActionTerminate, CLogical("Yes"), ",", ",", "ResponseMode:All ProcessBackspace:Yes")
    End If

    Instruction.Run

    Letter8.InputMasks.Reset
If Keyboard.GetState() = ebStateOpen Then
    Letter8EchoClients.RemoveAll
    Letter8.InputMasks.Add Keyboard.CreateInputMask("{ANY}", "z",
    CLng(Letter8.Duration), CLng("1"), ebEndResponseActionTerminate,
    CLogical("Yes"), ", ", ", "ResponseMode:All ProcessBackspace:Yes")
End If

Letter8.Run

Letter8.InputMasks.Reset

If Keyboard.GetState() = ebStateOpen Then
    Letter7EchoClients.RemoveAll
    CLng(Letter7.Duration), CLng("1"), ebEndResponseActionTerminate,
    CLogical("Yes"), ", ", ", "ResponseMode:All ProcessBackspace:Yes")
End If

Letter7.Run

Letter6.InputMasks.Reset

    If Keyboard.GetState() = ebStateOpen Then
        Letter6EchoClients.RemoveAll
            CLng(Letter6.Duration), CLng("1"), ebEndResponseActionTerminate,
            CLogical("Yes"), "", "", "ResponseMode:All Process Backspace:Yes")
    End If

Letter6.Run

Letter5.InputMasks.Reset

    If Keyboard.GetState() = ebStateOpen Then
        Letter5EchoClients.RemoveAll
        Letter5.InputMasks.Add Keyboard.CreateInputMask("{ANY}", "w",
            CLng(Letter5.Duration), CLng("1"), ebEndResponseActionTerminate,
            CLogical("Yes"), "", "", "ResponseMode:All Process Backspace:Yes")
    End If

Letter5.Run

If Keyboard.GetState() = ebStateOpen Then
    Letter5EchoClients.RemoveAll
    Letter5.InputMasks.Reset
End If

Letter5.Run

Letter5.EchoClients.RemoveAll
Letter5.InputMasks.RemoveAll
End If

Letter3.Run

Letter2.InputMasks.Reset

If Keyboard.GetState() = ebStateOpen Then
  Letter2EchoClients.RemoveAll
  Letter2.InputMasks.Add Keyboard.CreateInputMask("{ANY}", "t", CLng(Letter2.Duration), CLng("1"), ebEndResponseActionTerminate, CLogical("Yes"), ",", ",", "ResponseMode:All ProcessBackspace:Yes")

End If

Letter2.Run

Letter1.InputMasks.Reset
If Keyboard.GetState() = ebStateOpen Then
    Letter1EchoClients.RemoveAll
    Letter1.InputMasks.Add Keyboard.CreateInputMask("{ANY}", "s", CLng(Letter1.Duration), CLng("1"), ebEndResponseActionTerminate, CLLogical("Yes"), "", ",", "ResponseMode:All ProcessBackspace:Yes")
End If

Letter1.Run
    c.SetAttrib Letter1.Name & ".DurationError", Letter1.DurationError
    c.SetAttrib Letter1.Name & ".RTTime", Letter1.RTTime
    c.SetAttrib Letter1.Name & ".ACC", Letter1.ACC
    c.SetAttrib Letter1.Name & ".RT", Letter1.RT
    c.SetAttrib Letter1.Name & ".RESP", Letter1.RESP
    c.SetAttrib Letter1.Name & ".CRESP", Letter1.CRESP

Letter9.InputMasks.Reset

If Keyboard.GetState() = ebStateOpen Then
    Letter9EchoClients.RemoveAll
End If

Letter9.Run


----------

Sub InitDevices(c As Context)

----------


----------

Sub InitDevices(c As Context)

----------

Sub InitDevices(c As Context)

----------


c.SetAttrib Letter1.Name & ".DurationError", Letter1.DurationError

c.SetAttrib Letter1.Name & ".RTTime", Letter1.RTTime

c.SetAttrib Letter1.Name & ".ACC", Letter1.ACC

c.SetAttrib Letter1.Name & ".RT", Letter1.RT

c.SetAttrib Letter1.Name & ".RESP", Letter1.RESP

c.SetAttrib Letter1.Name & ".CRESP", Letter1.CRESP

----------

Sub InitDevices(c As Context)

----------

Sub InitDevices(c As Context)
SetOSThreadPriority 3

Set Display = New DisplayDevice
Display.Name = "Display"

Dim DisplayDisplayDeviceInfo As DisplayDeviceInfo
DisplayDisplayDeviceInfo.XRes = 640
DisplayDisplayDeviceInfo.YRes = 480
DisplayDisplayDeviceInfo.ColorDepth = 16
DisplayDisplayDeviceInfo.RefreshRate = 0
DisplayDisplayDeviceInfo.NumPages = 0

'Load values from context if they exist
If c.AttribExists(Display.Name & ".XRes") Then
    DisplayDisplayDeviceInfo.XRes = CLng(c.GetAttrib(Display.Name & ".XRes"))
If c.AttribExists(Display.Name & ".YRes") Then
    DisplayDisplayDeviceInfo.YRes = CLng(c.GetAttrib(Display.Name & ".YRes"))
If c.AttribExists(Display.Name & ".ColorDepth") Then
    DisplayDisplayDeviceInfo.ColorDepth = CLng(c.GetAttrib(Display.Name & ".ColorDepth"))

'Open the device, unless the context values indicate otherwise
Dim DisplayOpen As Boolean
DisplayOpen = True
If c.AttribExists(Display.Name & ".Open") Then DisplayOpen = CLogical(c.GetAttrib(Display.Name & ".Open"))
If DisplayOpen = True Then
    Display.Open DisplayDisplayDeviceInfo
    c.SetAttrib Display.Name & ".RefreshRate", Format$(Display.CalculatedRefreshRate, "0.000")
End If

Set Sound = New SoundDevice
Sound.Name = "Sound"

Dim SoundSoundOutDeviceInfo As SoundDeviceInfo
SoundSoundOutDeviceInfo.Channels = 2
SoundSoundOutDeviceInfo.SamplesPerSecond = 22050
SoundSoundOutDeviceInfo.BitsPerSample = 16
'Load values from context if they exist
If c.AttribExists(Sound.Name & ".Channels") Then
SoundSoundOutDeviceInfo.Channels = CLng(c.GetAttrib(Sound.Name & ".Channels"))
If c.AttribExists(Sound.Name & ".SamplesPerSecond") Then
SoundSoundOutDeviceInfo.SamplesPerSecond = CLng(c.GetAttrib(Sound.Name & ".SamplesPerSecond"))
If c.AttribExists(Sound.Name & ".BitsPerSample") Then
SoundSoundOutDeviceInfo.BitsPerSample = CLng(c.GetAttrib(Sound.Name & ".BitsPerSample"))

Set Keyboard = New KeyboardDevice
Keyboard.Name = "Keyboard"

Dim KeyboardKeyboardDeviceInfo as KeyboardDeviceInfo
KeyboardKeyboardDeviceInfo.CollectionMode = ebPres sesOnly
KeyboardKeyboardDeviceInfo.CapsLock = ebCapsLockOf f
KeyboardKeyboardDeviceInfo.NumLock = ebNumLockOn
'Load values from context if they exist
If c.AttribExists(Keyboard.Name & ".CollectionMode") Then
KeyboardKeyboardDeviceInfo.CollectionMode = CLng(c.GetAttrib(Keyboard.Name & ".CollectionMode"))
If c.AttribExists(Keyboard.Name & ".CapsLock") Then
KeyboardKeyboardDeviceInfo.CapsLock = CLng(c.GetAttrib(Keyboard.Name & ".CapsLock"))
If c.AttribExists(Keyboard.Name & ".NumLock") Then
KeyboardKeyboardDeviceInfo.NumLock = CLng(c.GetAttrib(Keyboard.Name & ".NumLock"))
If c.AttribExists(Keyboard.Name & ".EmulateDeviceName") Then
KeyboardKeyboardDeviceInfo.EmulateDeviceName = c.GetAttrib(Keyboard.Name & ".EmulateDeviceName")

'Open the device, unless the context values indicate otherwise
Dim KeyboardOpen As Boolean
KeyboardOpen = True
If c.AttribExists(Keyboard.Name & ".Open") Then KeyboardOpen = CLogical(c.GetAttrib(Keyboard.Name & ".Open"))
If KeyboardOpen = True Then
    Keyboard.Open KeyboardKeyboardDeviceInfo
End If

Set Mouse = New MouseDevice
Mouse.Name = "Mouse"

Dim MouseMouseDeviceInfo as MouseDeviceInfo
MouseMouseDeviceInfo.OpenMode = ebMouseOpenModeDirect
MouseMouseDeviceInfo.CollectionMode = ebPressesOnly
MouseMouseDeviceInfo.ShowCursor = False
'Load values from context if they exist
If c.AttribExists(Mouse.Name & ".OpenMode") Then
    MouseMouseDeviceInfo.OpenMode = CLng(c.GetAttrib(Mouse.Name & ".OpenMode"))
End If
If c.AttribExists(Mouse.Name & ".CollectionMode") Then
    MouseMouseDeviceInfo.CollectionMode = CLng(c.GetAttrib(Mouse.Name & ".CollectionMode"))
End If
If c.AttribExists(Mouse.Name & ".ShowCursor") Then
    MouseMouseDeviceInfo.ShowCursor = CLogical(c.GetAttrib(Mouse.Name & ".ShowCursor"))
End If
If c.AttribExists(Mouse.Name & ".EmulateDeviceName") Then
    MouseMouseDeviceInfo.EmulateDeviceName = c.GetAttrib(Mouse.Name & ".EmulateDeviceName")
End If

'Open the device, unless the context values indicate otherwise
Dim MouseOpen As Boolean
MouseOpen = True
If c.AttribExists(Mouse.Name & ".Open") Then MouseOpen = CLogical(c.GetAttrib(Mouse.Name & ".Open"))
If MouseOpen = True Then
    Mouse.Open MouseMouseDeviceInfo
End If

SetOSThreadPriority 0
End Sub

'-----------------------------------------------------------------------------
' InitObjects
',
'-----------------------------------------------------------------------------

Sub InitObjects(c As Context)

    Set SessionProc = New Procedure
    SessionProc.Name = "SessionProc"
    SessionProc.Tag = ""
    SessionProc.Subroutine = "SessionProc_Run"

    Set List1 = New List
    List1.Name = "List1"
    List1.Tag = ""

    'Initialization for List1

    Set List1.Order = New SequentialOrder
    Set List1.Deletion = NoDeletion
    List1.ResetEveryRun = False

    ' Create the column headings
    List1.Filename = ""
    List1.LoadMethod = ebLoadMethodEmbedded
    List1.Load

    Set List1.TerminateCondition = Cycles(20)
    Set List1.ResetCondition = Samples(1)
    List1.Reset

    Set proc1 = New Procedure
    proc1.Name = "proc1"
    proc1.Tag = ""
    proc1.Subroutine = "proc1_Run"
Set Letter6 = New TextDisplay
Letter6.Name = "Letter6"
Letter6.Tag = ""

Set Letter6EchoClients = New EchoClientCollection

InitTextDisplayDefaults Letter6

Letter6.Text = "/"
Letter6.Duration = CLng("-1")
Letter6.TimingMode = ebTimingModeEvent
Letter6.PreRelease = Val("0")

Letter6.OnsetSync = 1
Letter6.OffsetSync = 0

Set Letter4 = New TextDisplay
Letter4.Name = "Letter4"
Letter4.Tag = ""

Set Letter4EchoClients = New EchoClientCollection

InitTextDisplayDefaults Letter4

Letter4.Text = "+"
Letter4.Duration = CLng("-1")
Letter4.TimingMode = ebTimingModeEvent
Letter4.PreRelease = Val("0")

Letter4.OnsetSync = 1
Letter4.OffsetSync = 0

Set Letter2 = New TextDisplay
Letter2.Name = "Letter2"
Letter2.Tag = ""

Set Letter2EchoClients = New EchoClientCollection
InitTextDisplayDefaults Letter2

Letter2.Text = "/"
Letter2.Duration = CLng("-1")
Letter2.TimingMode = ebTimingModeEvent
Letter2.PreRelease = Val("0")

Letter2.OnsetSync = 1
Letter2.OffsetSync = 0

Set Letter1 = New TextDisplay
Letter1.Name = "Letter1"
Letter1.Tag = ""

Set Letter1EchoClients = New EchoClientCollection

InitTextDisplayDefaults Letter1

Letter1.Text = "-"
Letter1.Duration = CLng("-1")
Letter1.TimingMode = ebTimingModeEvent
Letter1.PreRelease = Val("0")

Letter1.OnsetSync = 1
Letter1.OffsetSync = 0

Set Letter3 = New TextDisplay
Letter3.Name = "Letter3"
Letter3.Tag = ""

Set Letter3EchoClients = New EchoClientCollection

InitTextDisplayDefaults Letter3

Letter3.Text = "|"
Letter3.Duration = CLng("-1")
Letter3.TimingMode = ebTimingModeEvent
Letter3.PreRelease = Val("0")
Letter3.OnsetSync = 1
Letter3.OffsetSync = 0

Set Letter5 = New TextDisplay
Letter5.Name = "Letter5"
Letter5.Tag = ""

Set Letter5EchoClients = New EchoClientCollection

InitTextDisplayDefaults Letter5

Letter5.Text = ".\nLetter5.Duration = CLng("-1")
Letter5.TimingMode = ebTimingModeEvent
Letter5.PreRelease = Val("0")
Letter5.OnsetSync = 1
Letter5.OffsetSync = 0

Set Letter7 = New TextDisplay
Letter7.Name = "Letter7"
Letter7.Tag = ""

Set Letter7EchoClients = New EchoClientCollection

InitTextDisplayDefaults Letter7

Letter7.Text = "|"
Letter7.Duration = CLng("-1")
Letter7.TimingMode = ebTimingModeEvent
Letter7.PreRelease = Val("0")
Letter7.OnsetSync = 1
Letter7.OffsetSync = 0

Set Letter8 = New TextDisplay
Letter8.Name = "Letter8"
Letter8.Tag = ""

Set Letter8EchoClients = New EchoClientCollection

InitTextDisplayDefaults Letter8

Letter8.Text = "+"
Letter8.Duration = CLng("-1")
Letter8.TimingMode = ebTimingModeEvent
Letter8.PreRelease = Val("0")

Letter8.OnsetSync = 1
Letter8.OffsetSync = 0

Set Instruction = New TextDisplay
Instruction.Name = "Instruction"
Instruction.Tag = ""

Set InstructionEchoClients = New EchoClientCollection

InitTextDisplayDefaults Instruction

Instruction.Text = "In this experiment, please use one hand to type the last 8 letters of the alphabet backwards as fast as you can. The graphic will move after each key press to indicate progress. If you are ready, please press the space bar to begin."
Instruction.Duration = CLng("-1")
Instruction.TimingMode = ebTimingModeEvent
Instruction.PreRelease = Val("0")

Instruction.OnsetSync = 1
Instruction.OffsetSync = 0

Set Letter9 = New TextDisplay
Letter9.Name = "Letter9"
Letter9.Tag = ""
Set Letter9EchoClients = New EchoClientCollection

InitTextDisplayDefaults Letter9

Letter9.Text = "End of this trial.\nPress space bar to continue."
Letter9.Duration = CLng("-1")
Letter9.PreRelease = Val("0")

Letter9.OnsetSync = 1
Letter9.OffsetSync = 0

End Sub

'--------------------------------------------------------------------------
' InitPackages
'--------------------------------------------------------------------------
Sub InitPackages(c As Context)

End Sub

'--------------------------------------------------------------------------
' InitGlobals
'--------------------------------------------------------------------------
Sub InitGlobals(c As Context)

End Sub

'--------------------------------------------------------------------------
' UnInitGlobals
'--------------------------------------------------------------------------
Sub UnInitGlobals()
End Sub

' UnInitDevices

Sub UnInitDevices()
    Display.Close
    Set Display = Nothing
    
    Keyboard.Close
    Set Keyboard = Nothing
    
    Mouse.Close
    Set Mouse = Nothing
End Sub

' UnInitPackages

Sub UnInitPackages()
End Sub

' UnInitObjects

Sub UnInitObjects()
    Set SessionProc = Nothing
    
    Set List1 = Nothing
    
    Set proc1 = Nothing
Set Letter6 = Nothing

Set Letter6EchoClients = Nothing

Set Letter4 = Nothing

Set Letter4EchoClients = Nothing

Set Letter2 = Nothing

Set Letter2EchoClients = Nothing

Set Letter1 = Nothing

Set Letter1EchoClients = Nothing

Set Letter3 = Nothing

Set Letter3EchoClients = Nothing

Set Letter5 = Nothing

Set Letter5EchoClients = Nothing

Set Letter7 = Nothing

Set Letter7EchoClients = Nothing

Set Letter8 = Nothing

Set Letter8EchoClients = Nothing

Set Instruction = Nothing

Set InstructionEchoClients = Nothing

Set Letter9 = Nothing
Set Letter9EchoClients = Nothing

End Sub

'--------------------------------------------------------------------------
' Main
'
'--------------------------------------------------------------------------
Sub Main()

' Create and initialize the default context, data file,
' and provide global access to the context.
Dim c As Context
Set c = New Context
Set c.DataFile = New DataFile
c.PushNewFrame
Set ebContext = c

' Set the log level names
c.SetLogLevelName 1, "Session"
c.SetLogLevelName 2, "Block"
c.SetLogLevelName 3, "Trial"
c.SetLogLevelName 4, "SubTrial"
c.SetLogLevelName 5, "LogLevel5"
c.SetLogLevelName 6, "LogLevel6"
c.SetLogLevelName 7, "LogLevel7"
c.SetLogLevelName 8, "LogLevel8"
c.SetLogLevelName 9, "LogLevel9"
c.SetLogLevelName 10, "LogLevel10"

' Set standard logging items
ebContext.SetAttrib "Experiment", "CS465RT"
Set ebContext.SetAttrib "SessionDate", Date$
Set ebContext.SetAttrib "SessionTime", Time$
Set ebContext.SetAttrib "RandomSeed", PRNG.GetSeed()
' Set default for GroupNumber
c.SetAttrib "Group", "1"

'Initialize global variables for packages
InitGlobals c

CreateDefaultPort

If Basic.OS = ebWin32 Then
    WinActivate "E-Run Experiment Window"
End If

' Get the StartupInfo

' Set the defaults for all of the StartupInfo
If Not c.AttribExists("Subject") Then c.SetAttrib "Subject", "1"
If Not c.AttribExists("Session") Then c.SetAttrib "Session", "1"

' Determine if StartupInfo.UseDefaults exists and is True/False to override prompts for StartupInfo parameters
Dim bStartupInfoUseDefaults As Boolean
bStartupInfoUseDefaults = False
If c.AttribExists("StartupInfo.UseDefaults") Then bStartupInfoUseDefaults = CLogical(c.GetAttrib("StartupInfo.UseDefaults"))
If Not bStartupInfoUseDefaults Then

    Dim vAnswer As Variant

StartupInfo_Begin:

StartupInfoPrompt_Subject:
    vAnswer = AskBox("Please enter the Subject Number (1-32767, 0=No Data Logging)", c.GetAttrib("Subject"))
    If Not IsEmpty(vAnswer) then
        If Not IsNumeric(vAnswer) then
            MsgBox "Please enter an integer value"
            GoTo StartupInfoPrompt_Subject
        ElseIf CLng(vAnswer) < 0 Then
            MsgBox "The value for Subject must not be less than 0"
            GoTo StartupInfoPrompt_Subject
        End If
    End If
ElseIf CLng(vAnswer) < 0 Then
    MsgBox "The value for Subject must not be less than 0"
    GoTo StartupInfoPrompt_Subject
ElseIf CLng(vAnswer) > 32767 Then
    MsgBox "The value for Subject must be not be greater than 32767"
    GoTo StartupInfoPrompt_Subject
End If
Else
    GoTo ExperimentAbort
End if

    c.SetAttrib "Subject", CStr(vAnswer)

StartupInfoPrompt_Session:
    vAnswer = AskBox("Please enter the Session Number (1-32767):", c.GetAttrib("Session"))
    If Not IsEmpty(vAnswer) then
        If Not IsNumeric(vAnswer) then
            MsgBox "Please enter an integer value"
            GoTo StartupInfoPrompt_Session
        ElseIf CLng(vAnswer) < 1 Then
            MsgBox "The value for Session must not be less than 1"
            GoTo StartupInfoPrompt_Session
        ElseIf CLng(vAnswer) > 32767 Then
            MsgBox "The value for Session must be not be greater than 32767"
            GoTo StartupInfoPrompt_Session
        End If
    End If
Else
    GoTo ExperimentAbort
End if

    c.SetAttrib "Session", CStr(vAnswer)

    ' Display the summary
    Dim strSummary As String
    strSummary = "Subject: " & c.GetAttrib("Subject") & "\n"
    strSummary = strSummary & "Session: " & c.GetAttrib("Session") & "\n"
    strSummary = strSummary & "\nContinue with the above startup info?"
Dim nSummaryAnswer As Integer
nSummaryAnswer = MsgBox(strSummary, ebYesNoCancel + ebQuestion, "Summary of Startup Info")
If nSummaryAnswer = ebNo Then
    GoTo StartupInfo_Begin
ElseIf nSummaryAnswer = ebCancel Then
    GoTo ExperimentAbort
End If

End If

' If the attribute Clock.Scale.Override exists
' then use it for to set the Clock.Scale value
If c.AttribExists("Clock.Scale.Override") Then
    Clock.Scale = CDbl(c.GetAttrib("Clock.Scale.Override"))
End If

' Set the Filenames for the data files
Dim strFilenameRecovery As String
Dim strFilename EDAT As String

' If the attribute DataFile.Filename.Override exists
' then use it for the .txt and .edat filenames
If c.AttribExists("DataFile.Filename.Override") Then

    ' Set the default Data Filename
    strFilenameRecovery = CStr(c.GetAttrib("DataFile.Filename.Override")) & ".txt"
    strFilename EDAT = CStr(c.GetAttrib("DataFile.Filename.Override")) & ".edat"
Else

    ' Set the default Data Filename
    strFilenameRecovery = CStr(c.GetAttrib("Experiment")) & "-" 
    CStr(c.GetAttrib("Subject")) & "-" 
    CStr(c.GetAttrib("Session")) & ".txt"
strFilenameEDAT = CStr(c.GetAttrib("Experiment")) & 
"-" & CStr(c.GetAttrib("Subject")) & "-" & CStr(c.GetAttrib("Session")) & ".edat"

End If

'set the name of the data file
C.DataFile.Filename = strFilenameRecovery

' If we are logging data, then prompt to overwrite the data file if it exists
If CLng(c.GetAttrib("Subject")) <> 0 Then
    If FileExists(c.DataFile.Filename) Or FileExists(strFilenameEDAT) Then
        If ebYes <> MsgBox("WARNING: The data file and/or recovery file
already exists:
FILE: " & c.DataFile.Filename & "
Do you want to overwrite?", 
        ebYesNo + ebQuestion) Then
            GoTo ExperimentAbort
        End If
    End If
End If

' initialize all system devices, packages, and objects
InitDevices c
InitPackages c
InitObjects c

' If we are logging data, then open the datafile
If CLng(c.GetAttrib("Subject")) <> 0 Then
    c.DataFile.Open
    c.LogHeader
End If

#if RUNTIME_VERSION_MAJOR > 1 Or (RUNTIME_VERSION_MAJOR = 1
And RUNTIME_VERSION_MINOR >= 2) Then
    ' log clock timing information
    c.SetAttrib "Clock.Information", Clock.Information
#end if
' Start the running of the Experiment
SessionProc.Run c

' Clean up the context and close the datafile
If CLng(c.GetAttrib("Subject")) <> 0 Then
  c.DataFile.Close
  ' Attempt to convert the recovery file into a data file
  Dim nConvert As Long
  nConvert = c.DataFile.Convert(ebProgressSimple)
  If nConvert = 0 Then
    ' Settings in E-Studio are set to not remove E-Recovery file
  Else
    ' The datafile failed to convert!
    MsgBox "ERROR: The datafile did not convert!\nFILE: " & c.DataFile.Filename & "\nIt is recommended that you recover your data with the E-Recovery utility"
    MsgBox c.DataFile.GetLastErrorMessage()
  End If
End If
End If

ExperimentFinish:

  UnInitObjects

  UnInitPackages
  UnInitDevices

  UnInitGlobals

ExperimentAbort:

  ' Clean up the context
  c.PopFrame
  Set c = Nothing
  Set ebContext = Nothing

  DestroyDefaultPort

End Sub
<table>
<thead>
<tr>
<th>Weight</th>
<th>Nested Procedure</th>
</tr>
</thead>
</table>
| 1      | proc1
0       |