1.0 Abstract

Would adding the ability to recognize the specific parts put into the sockets and providing vocal interaction from the original Mr. Potato head increase it’s educational value in the long run? That was the question that our group set out to answer. After making our alterations to the original Mr. Potato head, the level of interest increased, as well as the educational value. With the construction of the new Mr. Potato head, it will also benefit the demographic that it is intended for young children. Letting Mr. Potato head interact with kids in a way that would help them to learn puzzle-solving techniques, putting things together, and following directions while entertaining them along the way.

2.0 Design

The goal of the project was to modify Mr. Potato head so he would talk when a part was inserted into a socket and it would react differently if the correct appendages were put in the correct sockets. Our first step was to figure out how to identify different appendages and get that data into a device that could make him speak. This proved to be the most challenging part of the project as the other parts fell into place once we solved this one.

One of our ideas involved having several push buttons inside. This idea would have been extremely easy to implement. When a part was inserted in Mr. Potato Head it would push the button and relay to the cricket that something was inputted. However, this method would not allow us to know what part or more importantly the correct part had been inserted into the socket.

Our next idea was to limit what could be inputted into each slot. By some key mechanism we would only allow the correct part to fit into each hole correctly. Once again this would not give us the execution we desired.

Our last idea involved communication via the microphone jack. With this idea inserting each piece would complete a circuit inside Mr. Potato Head. This circuit would then be wired so that it could identify each part that was inputted. Since we are not communicating with crickets, this method involved us sending the information through the microphone jack. Soon we discovered that our limited intellects did not possess the capability of wiring such a circuit. Furthermore, we still to this day do not comprehend how to in fact send this information via the microphone jack. This idea was hastily
dropped as we came up with a completely revolutionary new idea. We settled on wiring stereo jacks with resistors, which would send information through the crickets.

2.1 Hardware

The first step of our project involved operating on Mr. Potato Head. We began by drilling his holes open just a bit more. Next we glued in stereo sockets to each hole. These sockets were specially wired to be inputted into the crickets. Each cricket would handle two sockets. Next we had to assemble stereo jacks to be inserted into the sockets. We attached resistors to eight different stereo plugs. Each resistor would give each plug
its own unique identifier. Sadly we had to partake in some more manipulation of Mr. Potato Head. Each of his parts had to have the stereo plugs implanted into them. This was easy for some parts as we need merely needed to glue the stereo jack onto the part. However, in some cases we needed to surgically implant the plug and in a few extreme cases we had to do a little amputation. After hours of surgery, everyone was relieved that Mr. Potato Head made it out okay.

Don't tell Mr. Potato Head this. Sadly, he did not make it out of the operation unscathed. As his left arm socket has a short in it. On occasion his body will not recognize the arm it has inside of it and he will lose control. During this time he will yell obscenities that do not make much sense. We only hope he controls himself during our presentation.

### 2.2 Crickets Code

The crickets in the project have the role of sensing when appendages are placed into or removed from sockets. This sensory data is then relayed from the crickets to an interface cricket via infrared and then relayed to the serial port of the desktop computer. The sensory data are simple resistance levels, which are translated by the crickets into digital values (see chart), then encoded with a socket identifier to fit into a single byte for communication. We chose appropriate resistors to give us sufficient separation within the tolerance of the crickets and resistors based on the chart.

![Graph of Cricket Output vs. Resistance](image)

The cricket code has an important role. Because we have 3-4 crickets communicating with one interface cricket, we can't simply stream data constantly from each cricket, or else the interface cricket will become confused and mangle the communications and data. Thus, it is necessary to have some sort of timing to control this cricket infrared mini-network.
The simplest approach in our case has the advantage of offloading some of the complexity from the Visual Basic code to the cricket code. Our design choice was to only send information when the state of a given socket changes: we watch for the resistance level to become infinite (nothing in socket) then change to a recognizable value (known appendage inserted).

This simple actually solves the most of the timing problem: since the data we send for each socket change event is very small (a single eight-bit value), the communication for each has a very short duration. Thus, it is very unlikely that another socket change event will occur during the communications of a previous event and thereby cause a collision. In this unlikely event, we can only rely on the desktop to discard inappropriate values.

Of course, this suggests an immediate improvement: since the target audience (children) is likely to test such boundary cases, it would be desirable to handle them. This would be possible by giving each cricket a "network" address and implementing a simple network communications system. A token ring topology is probably the easiest to implement: each cricket knows the address of the succeeding cricket and listens to the IR communications until it sees it's address come up. At this point, the cricket would begin transmitting its data. When finished would send a recognizable "token" containing the succeeding cricket's address, thus triggering the next cricket in line to begin communications. Thus, each cricket would maintain a queue of socket change events and would wait until the token indicates it's turn to send data.

Another limitation of our prototype is that we send only a single byte for each socket change event. This limits the number of socket/appendage combinations we can have to 255. In our case, we're currently using only 35 (7 sockets and five appendages). For more than 255 combinations, we could instead have the desktop Visual Basic client interpret two or more bytes at a time and have the crickets send corresponding data. However, this would require some communications overhead to delineate or time the begin/end of multi-byte transmissions over the serial port; by using a simple one-byte value we exploit the serial port timing hardware to do this work for us.

A very nice aspect of the design of the system with crickets is that it highlights the power of programmable units in construction: while we certainly could have accomplished the same thing with a serial-port-equipped millimeter and more sophisticated Visual Basic client code, the crickets handle the task with ease. That is, the crickets are over-complicated and over-powered for the task, but this is a fascinating study of the decrease in the cost of technology -- we don't care to optimize the computationally overpowered subsystems because they are relatively cheap (especially when compared to the human costs). On the other hand, the cricket elegantly handles so much of the dirty work that we wonder if some appreciation for the complexity involved is perhaps lost.

2.3 Speech Code
The VB code to make him speak was the last part of the project we did. We didn’t want to write it till we had the rest of him completed because we might then have to go back and rewrite the code if some part of the design changed.

We always had a rough idea of what this code should do which is read in an input and play a sound file based on that input. Once all the rest of him was build and the crickets programmed we knew that the code would read in a byte from the serial port and that byte would be a bit encoding of the socket and appendage that was place in that socket.

VB was picked as the language to write the code in just because it was easiest. The MSCom package that is a part of VB was used to read in the value from the serial port and fmod a shareware package was used to play the sound files. The code at first only played a sound file when an appendage was placed into a socket or removed from one. After playing with him for a while this was deemed to be boring. A timer was added, that would play a random complaint sound file every 20 seconds, was added. This made him more fun to play with since the randomness made him less predictable. Play with him continued and it was decided that playing a message when an appendage was removed was unnecessary since he randomly complained now so that was removed. We tried to record more sound files so we could add to the collection of complaints and not have him repeat himself as often. The key to making him fun to play with was the randomness as this lets you play with him longer before getting bored.

### 3.0 Evaluation

Our original thoughts for the Mr. Potato head were we wanted to be able to play different sound clips based on what appendages were put inside that socket. This meant we would need to be able to tell the different appendages from each other. We had a few ideas on how to do this. One was to put a piece of fiber optics on the appendages and place a colored filter in front of the end that would be plugged into the Mr. Potato head. Then have a light sensor inside him that could read what color the light was and from that tell what limb had been put into that socket. The idea of using a resister placed in the appendages and wiring the socket so it would complete a circuit when an appendage was placed in its socket was also talked about. We just didn’t know how to get the resistance level into a computer in a useful form. The idea of hooking the circuit directly to the microphone jack was talked about, but then we found out you could read directly off the serial port what was beamed from other crickets to the interface cricket by a byte. We knew crickets just produce a value between 0 and 255 based on the level of resistance it got back from the port the sensor was plugged into. So the idea of placing a resister on the appendage and wiring the sockets to a cricket’s sensor port and then writing code to beam that value back to the interface cricket seemed like the easiest solution.

After deciding how we wanted to do appendage detection we need to work out the details of how to to wire the socket and appendage. We came up with the idea to use a stereo jack and plug to complete the circuit. This worked out well and we cut the plastic parts off the appendages that plug into the Mr. Potato head and replaced them with a stereo jacks. Then we made the holes on the Mr. Potato head larger and glued stereo plugs into the
sockets. We then thought about putting the crickets inside the Mr. Potato head but we couldn’t think of a way to mount them so they could be turned on and off inside him so we had to settle for a mess of wires coming out the back of him and the crickets being placed outside.

The key to him keeping people entertained is the randomness of his speech. You never know what funny phrase he is going to say next and you wait for your favorite phrase to be said by him. Putting an appendage in the wrong socket can also be fun since once again he has random phases he utters when you do this. Completing the Mr. Potato head sets him off on another list of random phrases that he will only say once he is fully assembled so you can just listen to him for a while once you have finished putting him together. The computation that makes his speech possible and random adds a great deal of educational and entertainment value to our creation.

When he has no appendages put into him he has a kind of a Hellraiser or Borg look to him from the stereo jacks that we glued on. With him fully put together he has been said to have an Inspector Gadget look as you can see the stereo plugs and resisters that have been placed on the appendages. This look adds something to him and makes him seem a bit futuristic.

4.0 Education

The educational value of the of the original Mr. Potato Head is its ability to allow children to learn how to take several tangible objects, add them in a variety of ways to a larger base, and create a finished model. Children learn the basic layout of human anatomy from this Toy figure. Mr. Potato Head has kept the interest of kids for over 50 years because of the variety of faces you can create with the different tangible pieces; kids never seem to tire of this.

Our group has added on to the Mr. Potato Head’s educational value by inserting voice interaction with the user and the toy. Mr. Potato head is now able to requests certain pieces by saying phrases. The users now have to interpret Mr. Potato Head’s phrase and give him the body part he is indirectly asking for. This action is introducing young children to problem solving, by having them analyze what was said to them, they are able to think logically about what is being asked of them to do or find, and then solve the problem. However, there is more to it than that. There is an original phrase that indirectly asked for a certain body part, the user inserts the piece into the socket they think is correct, then Mr. Potato Head responds with a reaction phrase indirectly telling the user if they are correct or incorrect. This causes the user to interpret the response and come up with their own conclusion to if they were correct or not.

This updated version of Mr. Potato Head is still for children but because of the somewhat fragile resistors on the ends of the stereo plugs, we feel this toy would be designed for children 6 and up when accompanied by an adult. Kids will be able to learn a rough overview of anatomy of a body and face, get much need comprehension, and interpretational skills to give them an edge in their educational process though school by applying what they learned from this updated toy to their schoolwork and everyday life.
Appendix

Part A Cricket Code

; socket identification constants
global [seye smooth snose sarml sarmr searl searr]

; appendage identification constants
global [ear arm mouth eye nose]

; other variables
global [oncea onceb mea meb resa resb temp]

to init

; socket identification constants
;setseye $20
;setsmouth $40
;setsnose $60
;setsarml $80
;setsarmr $a0
;setsearl $c0
;setsearr $e0

; appendage identification constants
setear $01
setarm $02
setmouth $03
seteye $04
setnose $05

; mea and meb identify which socket is plugged into sensora and sensorb
; set these according to the commented-out values above
setmea $c0
setmeb $e0

; get the data started
setoncea 1
setonceb 1

detect

end

to detect

setresa sensora ; resa is the resistor in sensora
setresb sensorb ; resb is resistor in sensorb

; socket a
ifelse oncea = 0 [
    if resa = 0 [
        pushval 0 mea
    ]
]
setoncea 1
]
]
; appendage detection
setoncea identify resa mea
]

; socket b
ifelse onceb = 0 [
  if resb = 0 [
    pushval 0 meb
    setonceb 1
  ]
]
]
; appendage detection
setonceb identify resb meb
]
detect
end

; the case statement to identify appendages
; and send the value if appropriate
; return 0 if sent value, 1 otherwise

to identify :rlev :socket
  if :rlev > 40 [ if :rlev < 44 [
    pushval ear :socket
    output 0
  ]]
  if :rlev > 54 [ if :rlev < 58 [
    pushval arm :socket
    output 0
  ]]
  if :rlev > 60 [ if :rlev < 64 [
    pushval mouth :socket
    output 0
  ]]
  if :rlev > 69 [ if :rlev < 73 [
    pushval eye :socket
    output 0
  ]]
  if :rlev > 91 [ if :rlev < 95 [
    pushval nose :socket
    output 0
  ]]
output 1
end

; This is the output routing
; simple routine to send the value sum
to pushval :app :sock
  settemp :app + :sock
  ; just add another send 0 here if you need
  ; 2 bytes at a time.
send temp
Part B VB Code

' Note this code requires the use of the Microsoft SAPI 5.1 to compile.
' It uses the MSComm Component to communicate to the cricket on comm port 1
Option Explicit
Dim inputBuffer As String
Dim byteBuff() As Byte
Dim flag As Boolean
Dim eyes As Boolean
Dim mouth As Boolean
Dim leftArm As Boolean
Dim rightArm As Boolean
Dim ears As Boolean
Dim nose As Boolean
Dim home As String
Dim wrong(7) As String
Dim complete(15) As String
Dim bitch(10) As String

'Public stream1 As Long          'the stream we open
'Public channel1 As Long         'the channel we play the stream in
Private Sub playFile(filename As String)
    If channel1 <> 0 Then
        Call FSOUND_Stream_Stop(stream1)
        channel1 = 0
    End If
    'Close any opened file
    If stream1 <> 0 Then
        Call FSOUND_Stream_Close(stream1)
        stream1 = 0
    End If
    If mouth And nose And rightArm And leftArm And ears And eyes Then
        stream1 = FSOUND_Stream_OpenFile(home & complete(Int(14 * Rnd)),
                                            FSOUND_2D, 0)
    ElseIf mouth Then
        stream1 = FSOUND_Stream_OpenFile(filename, FSOUND_2D, 0)
    Else
        stream1 = FSOUND_Stream_OpenFile(home & "no mouth.wav", FSOUND_2D, 0)
    End If
    If stream1 = 0 Then
        MsgBox "Error while opening file" & vbCrLf & FSOUND_GetErrorString(FSOUND_GetError)
End If

If stream1 <> 0 Then
    channel1 = FSOUND_Stream_Play(FSOUND_FREE, stream1)
    Sleep (5000)
End If
End Sub

Private Sub actionCmd_Click()
    ' say what is in the text box
    ' spkSpeak.Speak txtText.Text
End Sub

Private Sub Form_Load()
    ' Get our Server Name, and set the form's caption
    Caption = "Potato Head"
    If FSOUND_Init(44100, 32, 0) = 0 Then
        ' Error
        MsgBox "An error occurred initializing fmod" & vbCrLf & FSOUND_GetErrorString(FSOUND_GetError)
    End If
    MSComm1.InputMode = comInputModeBinary
    home = "D:\potato\"
    ' Opens the communications port
    MSComm1.PortOpen = True
    mouth = False
    nose = False
    eyes = False
    rightArm = False
    leftArm = False
    ears = False
    wrong(0) = "wrong hole 1.wav"
    wrong(1) = "freak 2.wav"
    wrong(2) = "get it out.wav"
    wrong(3) = "not funny.wav"
    wrong(4) = "superfreak.wav"
    wrong(5) = "don't put that there.wav"
    wrong(6) = "get it out.wav"
    complete(0) = "bus stop.wav"
    complete(1) = "I'm pretty.wav"
    complete(2) = "finished 1.wav"
    complete(3) = "fat ass wife.wav"
    complete(7) = "up.wav"
    complete(4) = "weareborg.wav"
complete(5) = "demented.wav"
complete(6) = "forbin5.wav"
complete(8) = "help.wav"
complete(9) = "fart.wav"
complete(10) = "smart.wav"
complete(11) = "got soap.wav"
complete(12) = "burp.wav"
complete(13) = "look.wav"
complete(14) = "finished 1.wav"
bitch(0) = "ears 1.wav"
bitch(1) = "arms 2.wav"
bitch(2) = "I'm blind 3.wav"
bitch(3) = "louder I cant.wav"
bitch(4) = "farted.wav"
bitch(5) = "wave coming.wav"
bitch(6) = "no bag tonight.wav"
bitch(7) = "CAN'T BREATHE.wav"
bitch(8) = "breathing.wav"
bitch(9) = "farted.wav"
Timer1.Interval = 6000

End Sub
Private Sub Timer1_Timer()
 Dim partNumber As Integer
      flag = True
 While flag
         partNumber = Int(9 * Rnd)
         If (Not ears) And (partNumber = 0 Or partNumber = 3) Then
            flag = False
         ElseIf (Not rightArm Or Not leftArm) And (partNumber = 1 Or partNumber = 4) Then
            flag = False
         ElseIf (Not eyes) And (partNumber = 2 Or partNumber = 5) Then
            flag = False
         ElseIf (Not nose) And (partNumber = 7 Or partNumber = 6) Then
            flag = False
         ElseIf (Not nose) And partNumber >= 8 Then
            flag = False
            partNumber = 4
         ElseIf nose And eyes And rightArm And ears And leftArm Then
            flag = False
         End If
      Wend
      playFile (bitch(partNumber))
 End Sub
Private Sub Form1_Unload()
' Close the serial port
  MSComm1.PortOpen = False
End Sub

Private Sub MSComm1_OnComm()
  ' If the event messages is a receive event then process the information
  ' Note that any information from the serial port is 16 bits or 2 serial send events on the
  ' Cricket
  If MSComm1.CommEvent = comEvReceive Then
    ' get the 2 bytes and put it into the receive buffer, formatting it as unicode
    inputBuffer = StrConv(MSComm1.Input, vbUnicode)
    inputBuffer = MSComm1.Input
    byteBuff = inputBuffer
    'MsgBox "port = " & byteBuff(0)
    ' select which character was received and process accordingly
    Select Case byteBuff(0)
      ' Right ear put in
      Case 225
        ears = True
        playFile (home & "ears 2.wav")

      ' Right ear taken out
      Case 224
        ears = False
        'playFile (home & "ears 1.wav")
      ' Left ear put in
      Case 34:
        playFile (home & "homer.wav")
        Sleep (6000)
      Case 33
        playFile (home & "blo.wav")
        Sleep (35000)
      Case 193
        ' Clear the input register and text box
        ears = True
        playFile (home & "ears 2.wav")

      ' Left ear taken out
      Case 192
        ears = False
        'playFile (home & "ears 1.wav")
      ' Mouth put in
      Case 67
mouth = True
playFile (home & "mouth.wav")

Case 64
mouth = False
'playFile (home & "no mouth.wav")
'Left arm put in
Case 130
leftArm = True
If rightArm Then
    playFile (home & "arms 5.wav")
Else
    playFile (home & "arm balance.wav")
End If
Case 128
leftArm = False
'playFile (home & "arms 2.wav")
Case 162
rightArm = True
If leftArm Then
    playFile (home & "arms 5.wav")
Else
    playFile (home & "another arm.wav")
End If
Case 160
rightArm = False
'playFile (home & "arms 2.wav")
Case 32
eyes = False
'playFile (home & "I'm blind 3.wav")
Case 36
eyes = True
playFile (home & "eyes 3.wav")

Case 101
nose = True
playFile (home & "nose.wav")

Case 96
nose = False
'playFile (home & "no nose.wav")
Case Else
    Dim filename As String
    playFile (wrong(Int(6 * Rnd)))
End Select
    End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
    'Stop the song that was playing
    If channel1 <> 0 Then
        Call FSOUND_Stream_Stop(stream1)
        channel1 = 0
    End If
    'Close any opened file
    If stream1 <> 0 Then
        Call FSOUND_Stream_Close(stream1)
        stream1 = 0
    End If

    'Make sure you close FMOD on exiting
    '(If you forget this, Visual Basic will crash upon exiting the app in debug mode)
    Call FSOUND_Close
End
End Sub