Steffen Norgren
A00683006
Set 3F

Doug Penner
A00658271
Set 3F

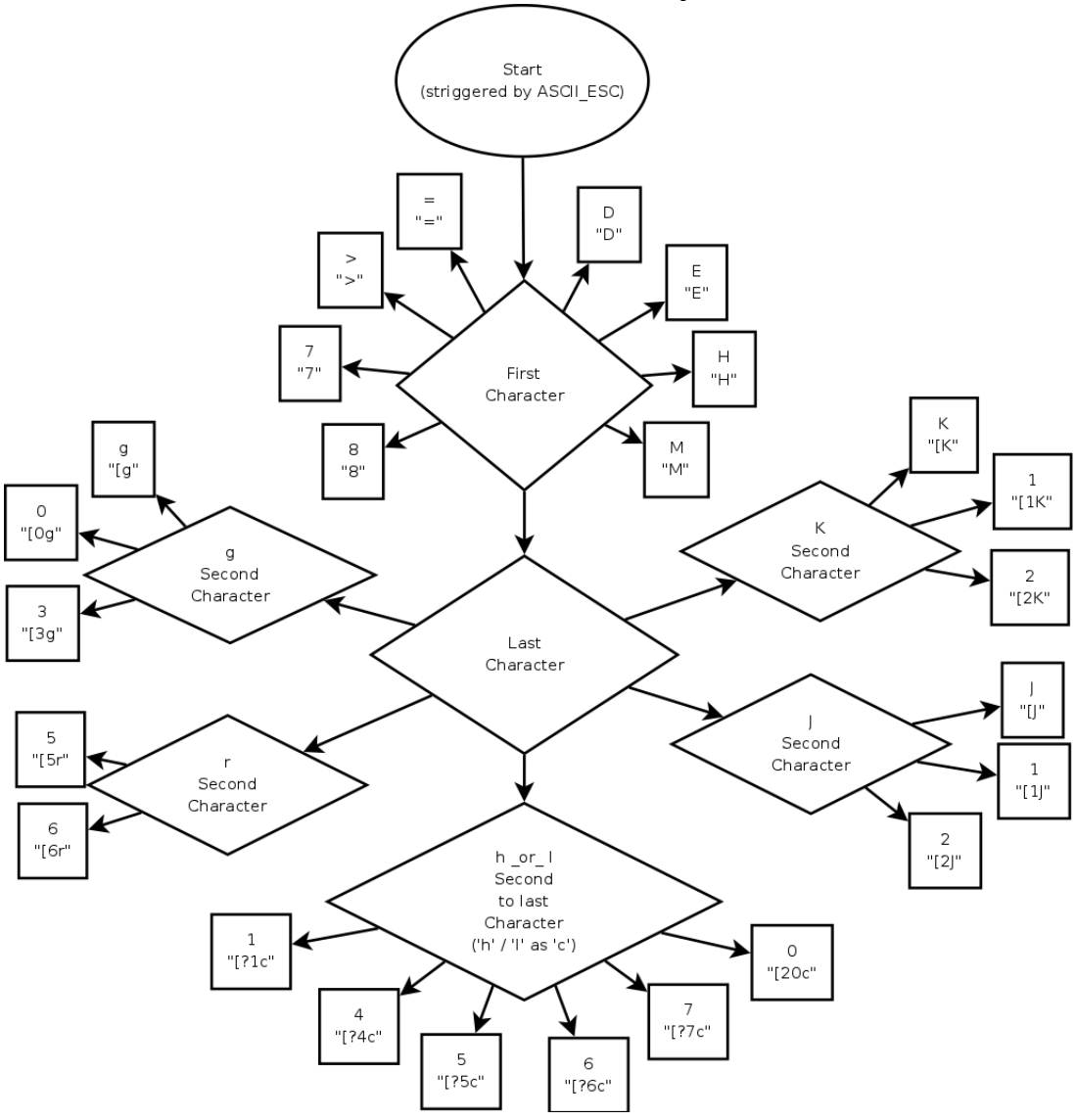
Data Communications
Assignment 2
"VT100 Emulator"

October 24th, 2008

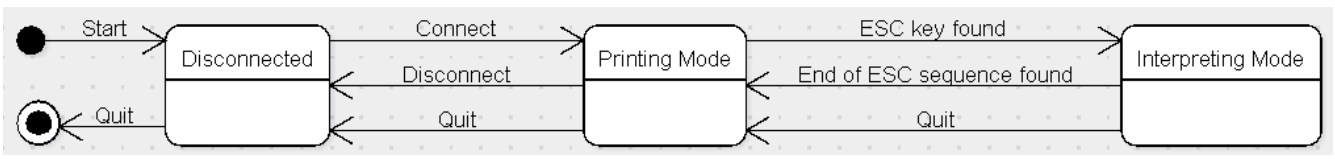
Escape Code String Parsing Case Statement Flow Chart

This case statement is run every time a character is received while the program is in “interpret mode”. If any of the end conditions are met (denoted as a box), the function returns VALID, otherwise it returns PARTIAL, unless it ends with an alpahbetical character in which case it returns INVALID.

```
If a case statement exists that matches the code:
        return VALID
else if the code ends with an aphabetical character:
        return INVALID
else
        return PARTIAL // this will cause the next received character to be appened to the code and
                          the case statement to be called again.
```



State Machine Diagram



Program Sudo-Code

```
function: Main_Program
        verify that the program can run in current version of windows (not DOS or something)
        initialize global variables
        attempt to create main window and class
        enter main message loop

function: Connect
        make sure not already connected (quit if connected)
        create comm connection using "createFile"
        set connected to true
        start Read_Thread

function: Disconnect
        make sure actually connected (quit if not connected)
        set connected to false (causes Read_Thread to die)
        close comm connection file

function: Read_Thread
        while connected
                wait for comm port event
                read string
                interpret each character (one after another)

function: interpret_character
        if program in printing mode
                if character is ESC character
                        put program in interpreting mode
                else if character is special character (LF, CR, etc)
                        call appropriate display function
                else
                        print the character
        else    // in interpreting mode
                append character to escape sequence code (this will build until full)
                run case statement on escape sequence code // see flow chart
                if statement is PARTIAL
                        leave // allowing next character to add onto code until complete
                else if statement is VALID
                        run run appropriate display function // may also cause a response
                        purge escape sequence code
                        put program in printing mode
                else    // INVALID
                        purge escape sequence code
                        put program in printing mode

function: print a character
        if at end of column
                if at last row
                        move all rows up
                else
                        move down one row
                        move to left of column
        else
                move right one position
        adjust caret to new position
        set current position in character array to character
        set current position in formatting array to last used format
        print screen
```

```
function: main window procedure // contains all other windows
        check message
        if the window is being created
                initailize the terminal data
                open the window
        else if the window is being destroyed
                destroy the window
                quit program
        else if a command was entered // menu
                call menu handler function // and pass in menu item that was hit
        else if a character was entered // keyboard letter/number
                send the character over the serial port (using comm send char)
        else if the window is resized
                put size back // window size is determinded by font, thus no resizing ;)
        else if window is closed
                destroy window
        else
                perform default window procedure action

function: child window procedure // where characters are printed
        check message
        if message that the mouse has moved inside the window
                make the window active // sets focus to window
        else if window is resized
                resize the terminal
        else if window needs to be painted
                call window painting function
        else
                perform default window prcedure action

function: window paint
        begin painting
        get old current font
        set background color
        for every row in the character array
                for every column in the row
                        get formatting values from cooresponding cell in formatting array
                        select the font using given formatting values
                        print the current character
                        delete the font
        delete font

function: resize the terminal
        get window rectangle size // entire window
        get client rectangle size // white are of window
        calculate width and height of font character
        calculate how big the screen has to be to fit all the characters // vt100 has fixed char size
        resize window according to calculations
        invaliate the window
```