Anthony Vallin, aav5195
CMPEN 351

Hangman Lives on MARS the game

Part 2

Architectural

Description

The Bitmap Display Tool is required prior to playing the game. The player is required to set the Bitmap Display settings to the following: Unit Width/Height = 1, Display Width/Height = 256, and Base address for display = 0x10040000 (heap). Additionally, the Keyboard and Display MMIO Simulator, using the default settings is needed for user input.

The game starts by displaying a welcome message on the Bitmap display for two seconds. Immediately after, the *GetWord* function opens/reads a word list text file and writes the data to a word buffer. The *GetWord* function uses the *RandGen* function to generate a random number. The generated number is used to select a word from the word buffer. Afterward, a series of rectangles, resembling a gallows like structure, is drawn by the *DrawRectangle* function. Additionally, the *WordBox* function draws lines that correspond to the total number of letters in the answer.

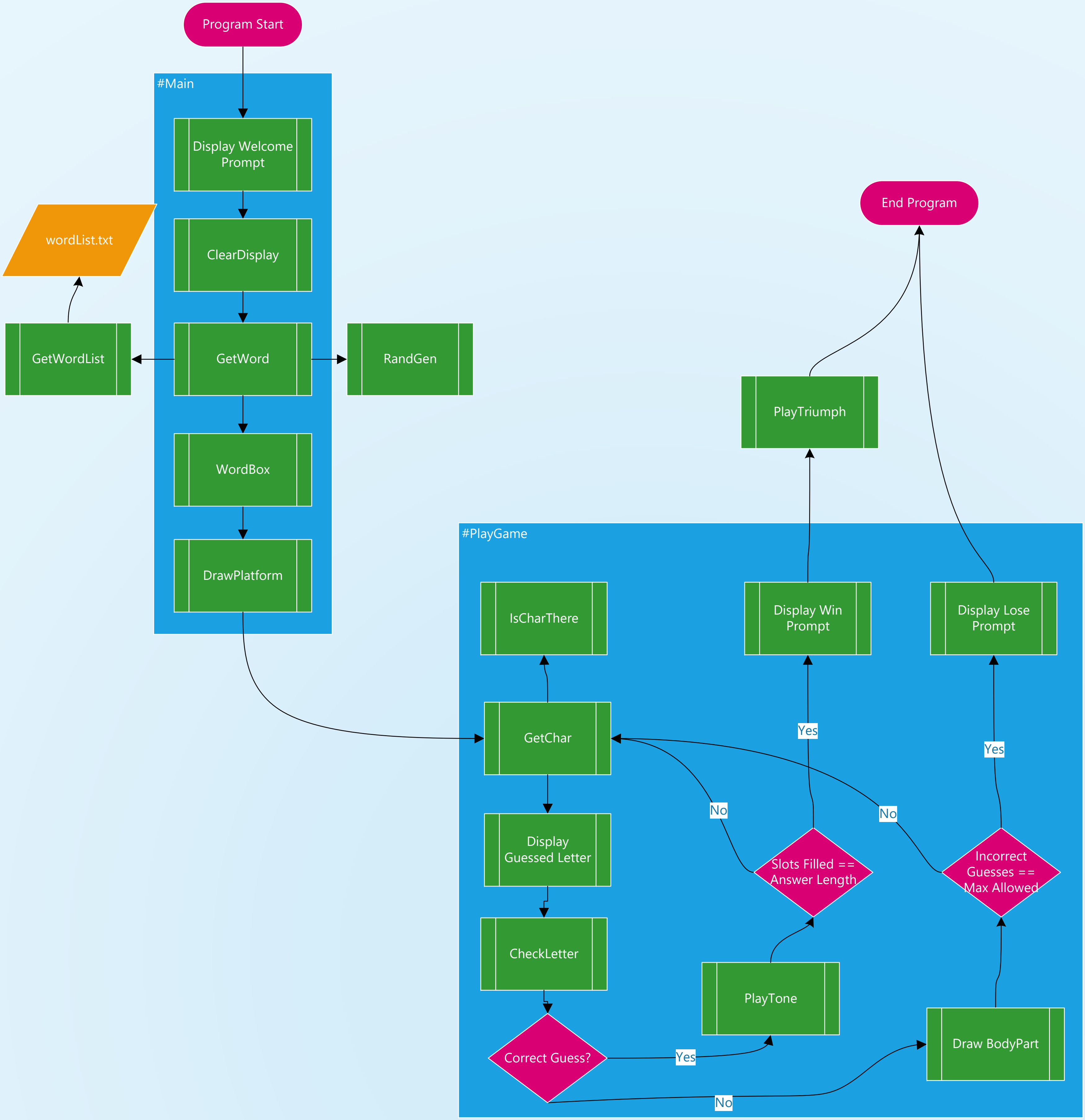
Next, the game displays the standard rules of the game: only capital letters are valid input and a maximum of six guesses are accepted before program termination.

A prompt notifies the user to enter a character in the Keyboard and Display MMIO Simulator. The program checks whether the guessed letter is in the word via the *CheckLetter* function.

- A valid letter is displayed on a line, corresponding to its position in the word. The position of the letter is calculated via an offset and its position in the answer.
- A valid letter plays a tone, notifying the player of their small success.
- An invalid selection causes the program to draw a body part on the Bitmap Display. All body parts (head, body, arms, and legs) are predefined functions.
- An invalid selection does not play a tone. This is to remind the player that in space, no one can hear you scream.
- All valid incorrect and correct guesses are displayed above the scaffolding as to remind the player of previous selections.

A total of six guesses are allowed until the game ends for the player. Failure results in a loser prompt, while success results in a winner prompt.

Flow Chart


Program Start
#Main
Display Welcome Prompt
ClearDisplay
wordList.txt
GetWordList
GetWord
RandGen
WordBox
DrawPlatform
End Program
PlayTriumph
#PlayGame
IsCharThere
Display Win Prompt
Display Lose Prompt
GetChar
Yes
Yes
No
No
Display Guessed Letter
Slots Filled == Answer Length
Incorrect Guesses == Max Allowed
CheckLetter
PlayTone
Draw BodyPart
Correct Guess?
Yes
No

Anthony Vallin, aav5195
CMPEN 351

Test/Verification/Build Plan

Test/Verification

Tests/Verification was accomplished in three phases. The first phase involved testing/verifying functions/modules individually. In other words, every function/method does its job, e.g., *DrawVertLine* draws a vertical line on the Bitmap display. The second phase of testing verified that modules/functions worked with other functions to produce the required output, e.g., the *GetWord* function, using the *GetWordList* function to retrieve a word list from a file text and the *RandGen* function to retrieve a random number, to select a word. The final phase involved testing the entire program in its entirety. In other words, testing various win or lose scenarios inputs:

- Expected results when all inputs are incorrect:
  - All incorrect guesses displayed above the hangman scaffold.
  - No incorrect guesses are displayed on the letter lines underneath the scaffold.
  - Body parts on Bitmap display are drawn in correct order.
  - Game exits and displays a lose prompt when six incorrect guesses attempted.
- Expected results when all inputs are correct:
  - All correct guesses are displayed above the hangman scaffold.
  - All correct guesses are drawn in appropriate positions on lines below the scaffold.
  - Tone sounds when a correct guess is found.
  - No body parts are drawn on the Bitmap.
  - Game exits and displays win prompt and midi song when answer found.
- Expected results when mixed correct and incorrect input:
  - All valid incorrect and correct guesses are displayed above the hangman scaffold.
  - Body parts on Bitmap display are drawn when an incorrect guess attempt.
  - Correct guesses are drawn in appropriate positions on lines below scaffold.
  - Tone sounds when a correct guess is found.
  - Silence when an incorrect guess is attempted.
  - Appropriate exit condition based on whether game is won or lost

Build Plan

To minimize run errors, functions were built modularly. In other words, functions were built and tested immediately after completion. This was done to avoid conflicting issues when two or more functions that interacted produced unexpected output. Looking at you Simon Says labs.

Debug/Issues

I had three issues with the build that were “noteworthy”.

- I had issues with opening and reading from my wordlist.txt file. It took me some time to figure out that problem lay with the file itself. I used Notepad++ to save the file, which encoded the contents to a format not readable by Mars. The failure went away when I saved the file using Microsoft Notepad.
- The second issue revolved around an erroneous horizontal line that appeared above the head of the hanging man. It took a bit of time before I found the problem. There was a typo in one of my variables in the .data section. I changed it from “.word0” to “.word 0” and the erroneous line

went away. However, I couldn’t figure out why that typo didn’t cause other runtime errors since that .data variable was an important part of the code.

- The third issue involved the midi syscall causing my bodyCount branches to fail. An incorrect guess triggered the tone one time but hung afterwards. It didn’t matter whether I used my *Tone* function, the midi syscall directly, or placed the code logic in another location. It just kept hanging the program after the first tone. Ultimately, I couldn’t figure out the cause. The important part is that the failure tone is a feature and not required to run the program successfully.

Changes

1. Initially, I was going to include a hint for the answer but decided against it. This game is not going to be played outside of a project demo, so it didn’t feel like a needed feature. Instead, I implemented a function/method to draw all guesses on the Bitmap display.
2. I changed the dimensions of the Bitmap display’s height and width from 512 to 256. The larger height and width for the display left a lot of empty space. The gameplay looked atheistically better on a smaller display with less white space.