Lab #10: Player Controlled Character

Introduction

Up to this point, the user inputs in past labs have each resulted in a unique output: a distinct music note or shape to be displayed. However, the operation of a video game depends on both the user input *and* the current state. In this lab, you will implement the simplistic backbone of a top-down video game.

State-Change Approach to Gaming

The operation of a video game can be conceptualized as a series of hierarchical states. For simplicity's sake, we will discuss this in the context of three main 'top-level' states: Initialize, Run, and Pause. These three states can be transitioned to using user-inputs or internal flags that are set during operation. Figure 1 shows a simplistic approach to this design, in which pressing the 'Reset' key will send the game into the Initialize state and pressing the 'Pause' key will enter or exit the Pause state. These three states illustrate the backbone of a simple video game.

Now let us examine what takes place within the 'game' portion, or Run state. During the Run state, each component of the game (characters, enemies, sounds, etc.) is identified by its own set of states. For example, the player character in a top-down video game may be represented by a pair of coordinates that indicate where on the display the character should be. These states can be changed by different user inputs, such as the D-pad's arrow keys. Likewise, the health, inventory, or any other status of any character (player-controlled or otherwise) can be thought of as a set of states. Figure 2 shows a simple diagram of how a player controlled character's position can be conceptualized as a set of states. The final task that must be completed upon each change of any character's state is the updating of the display: if any character has moved, the player needs to see it!

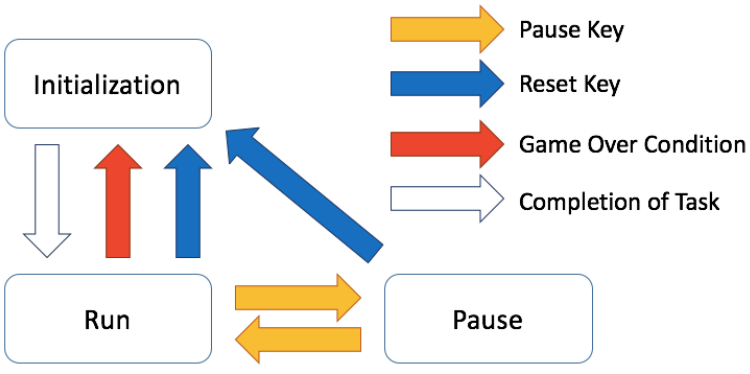

Figure 1: Simple State Diagram of Video Games

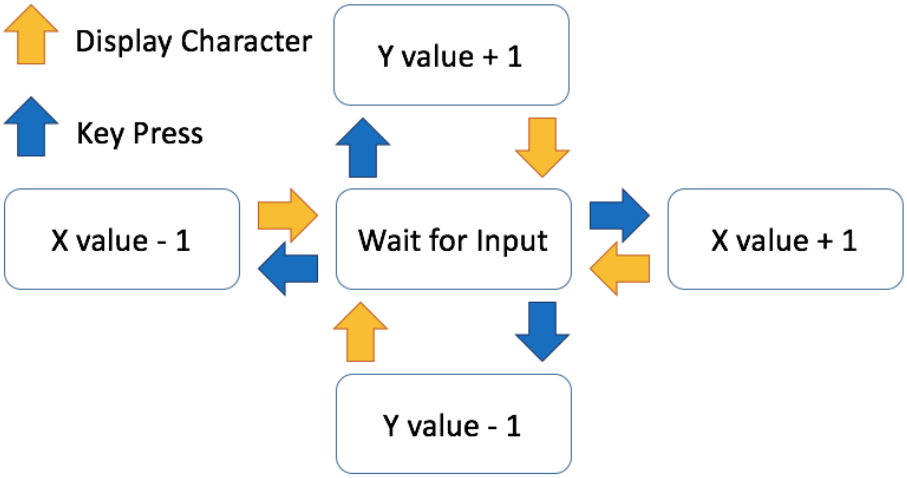

Figure 2: State Diagram of Player Coordinates

Assignment Overview

For this week's lab, your STM32L100RC microcontroller is required to display a player controlled character and play appropriate sounds. The character should move in the directions indicated by the D-pad keys and a short 0.1 second duration beep should sound when they are pressed. When the 'Pause' key is pressed, the game should sound a beep, but no following directional inputs will have an effect on the character displayed or make a sound until the 'Pause' key is pressed again. When the 'Reset' key is pressed, regardless of whether or not the game is paused, the player controlled character should return to the bottom left corner of the LED display matrix and the song played in last week's lab should play.

Pre-Lab Assignment

Modify the code from last week to implement the functionality described above. Give special thought to the following:

1. How will the character's position be displayed and stored?
2. How will you determine which operating state you are in?
3. What needs to happen in the Initialize state for this game?

Laboratory Experiments

1. Record all observations and problems you have in your lab notebook
2. Check all wiring from prior labs before powering up EEBoard and STM32L100RC
3. Observe the movement of the player controlled character and verify it is what you intended
4. If time permits, try to implement a second character that moves in the same manner as the first. If the two characters collide, perform the same action as the 'Reset' key

Laboratory Report

1. Discuss how you stored information regarding the position of your character.
2. What would need to be changed to implement the simple game in Part 4 of the Laboratory Experiments section?
3. What kinds of games can be implemented with a simple control scheme like this? What if we added a button or changed the functionality of the buttons?