Arjun Dalwadi

Tetris – Project Scope, Timeline, and Testing Strategy

Scope:

This project is going to use the LCD touchscreen and the user button to play Tetris. We have been provided with file sets which easily facilitate communication with the LCD touchscreen. These file sets use the peripherals: GPIO, NVIC, I2C3, SPI5, and the LTDC. The GPIO, I2C3, and SPI5 are used to communicate with the LCD touchscreen, and the NVIC is used to allow the touchscreen to be interrupt driven. I will also integrate the user button, which will be communicated with using the GPIO peripheral and be interrupt driven using the NVIC. A timer TIM2 will also be used to control the blocks' fall rate and keep track of elapse playing time. The RNG peripheral will be used to randomly select one of the 7 blocks in the gameplay screen. The code I will add to the project shall have:

- 3 screens on the LCD: the "start game" screen, the "gameplay" screen, and the "game over" screen
    - "Start Game" Screen: has a 'start' button and displays all 7 Tetris blocks
    - "Gameplay" Screen: the actual gameplay
    - "Game Over" Screen: displays elapsed game time in seconds
- An 8x10 unit game field (8 units wide, 10 units tall)
- 7 unique rotatable Tetris blocks (a button press is a 90-degree clockwise rotation)
- Touching the screen on the left/right side will shift the live block left/right one unit
- Blocks will fall one level every second (approved by Xavion). 3 seconds per level felt slow.

If time permits, I will add the keeping track of the singles, doubles, and Tetris' and display them in the "game over" screen as well as the elapsed game time.

Timeline:

Week of 11/18: Configure and initialize all necessary peripherals

Week of 11/25: 3 screens designed, screen navigation, and main gameplay logic

Week of 12/02: Stress test all features, add extra credit feature

Week of 12/09: Wrap up final testing and documentation revising

Testing and Verification:

At each milestone listed in the **Timeline** section, I will commence a round of testing which ensures expected/desired behavior. This will either be done by writing short testing functions and running the code, or by reusing old projects' implementations of similar features. I will debug as we have been, using the IDE's built-in debugging tools. Each week will be 40% development and 60% testing, and I will not move on to the next milestone until the previous one was perfected. That way, I can test the overall system as well as I build up the higher-level code, such as the main game logic. This is the Waterfall model, and I believe for a project of this type, where I have been given much of the low-level programming and am tasked with the higher-level functionality, Waterfall is the best model.