Code description

The program starts off by first reading the voltage at PA0 (ADC channel 0). The default 10 bit ADC resolution of analogRead() is used as the onboard 12 bit ADC's full precision is not required. The centre of the turret is generated by mapping the raw 10 bit ADC values between 20 and 235, using which the outline of the turret is generated (using 3 graphical lines) and loaded into frame_objects matrix (size: n x 5). Frame_objects matrix is used to store the attributes of the objects to be displayed in the general format {object_type(0 for circle, 2 for point to point line),centre_x/xstart,centre_y/ystart,radius/xend,yend}. If the time elapsed since the last target spawn is greater than 15secs, the target generation block is used. First, the radius of the target, which are all circles, is randomly generated between 15 and 25. Then the centre's x coordinate is randomly generated between 30 and 180, this is done so that the number of targets is also randomized, as targets are continuously generated as long as the right most x coordinate of the target is less than 255 (max raw input for 8 bit DAC). Now, the program enters the clean up routine, which deletes targets whose top y coordinate is below the bottom of the screen (0 raw value for DAC) and bullets that have gone above the top of the screen i.e. bottom y coordinate > 255. The routine also checks for any intersections between a target and a bullet and if any is found, a 70 HZ square wave is generated at the speaker to alert the player of a hit and both the objects are deleted. A similar scenario occurs when a collision occurs between the turret and a target, where instead of a tone a melody is played to signify that the game has ended and the program enters an infinite loop. After the cleanup, bullets are generated based on whether the user pushed a button, which is checked using a falling edge hardware interrupt attached to PA2. The bullet generation routine also checks if a cool-off/reloading time of 2.5 secs has passed before entertaining the user's request to fire the turret. The bullets and targets are not directly loaded into the frame objects array as the program moves their y coordinates in different directions and at different speeds across the screen. Once the y coordinates have been updated the mobjs matrix(for targets) and bullets matrix(for bullets) are loaded into the frame_objects matrix using load(). Now the program enters the rendering phase, in which the frame_objects matrix is sorted using the bottom y coordinates of the objects (scanning starts from the bottom) in ascending order using selection sort. After frame_objects is sorted, the program enters a loop in which the scanning takes place starting from y = 0 to y = 255. In each iteration, shapes from frame_objects are checked for intersections with the horizontal line located at the then current y coordinate, this is done by checking whether the bottom y coordinate of the shape is below the current y and the top y coordinate is above the current y coordinate. If the first statement is not satisfied, the scanning of the shapes stops, as the shapes are already been arranged based on their bottom y coordinate and all subsequent shapes will have their y coordinates above the current y. For the second statement, the loop simply skips to the next shape. Further down the loop, the intersection points for the shapes are calculated by calling read_primitives(), which processes the attribute array for the shape and loads the intersection points into the point array. After all shapes are finished scanning, the program heads to the DAC interface section. First, the current y coordinate is used to set the voltage on channel 1 using port A and B's output data registers, which are assigned values based on the pin mappings of the individual bit pins of the 8 bit DAC to the GPIO pins of the STM32 blue/blackpill. Once the y coordinate is set, the x values from the point array are rapidly displayed on channel 2 to generate a single horizontal slice of the frame. The loop continues until the entire frame is horizontally scanned from bottom to top. For the next frame, the positions of the objects are again updated and the cycle repeats.