****

**Tic-Tac-Toe**

**Final Report**

**Fall 2021**

**12/2/2021**

**College of Engineering and Computing**

**Department of Electrical and Computer Engineering**

**ECE 287: Digital Systems**

**Hartup, Dave Ph.D.**

*Zack Burton*

*Riley Taylor*

**Index**

*Work Accomplished* [*pg 3-4*](#wazc1p25zc1x)

*Operation and Features* [*pg 5*](#trd9oge924f4)

*Future work* [*pg 6*](#362n6muzwo84)

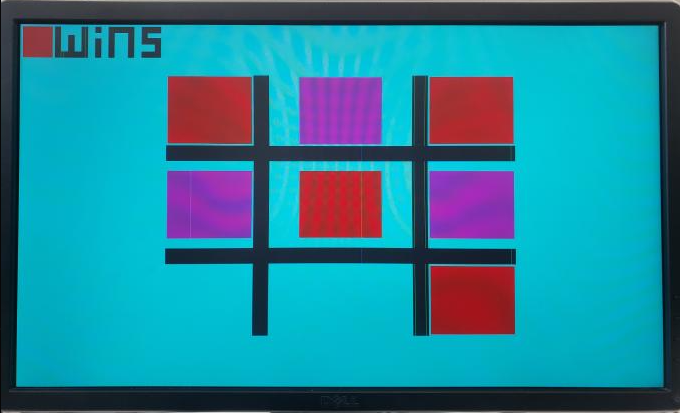
*Github* [*pg 6*](#8m9p8xt9pt5k)

*Gantt Chart* [*pg 7*](#ixcrxmgabfmr)

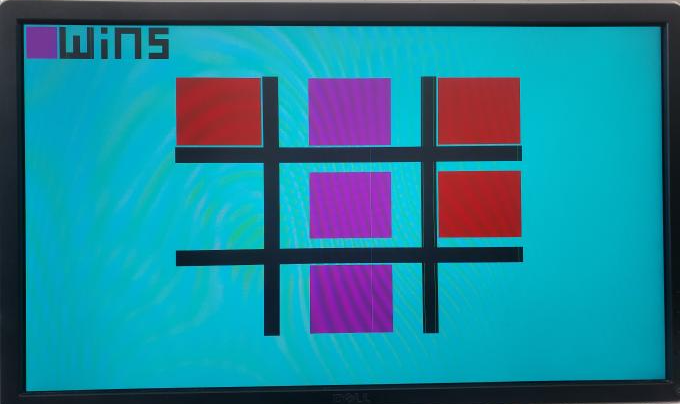
*Appendix* [*pg 8-24*](#lm3gcaojakt)

**Work Accomplished**

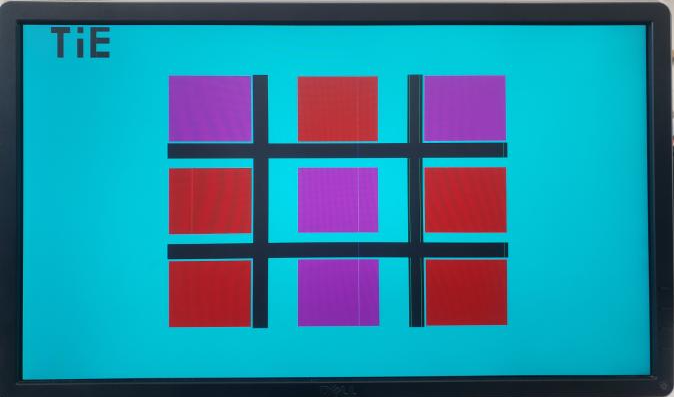
After utilizing around seventy lab hours we have completely finished a working game of tic-tac-toe using VHDL in Quartus Prime using the Altera DE2-115 board. The game code has the implementation of the original tic-tac-toe game rules, consisting of a two player turn-based system with three different winning conditions. The first condition is if player one wins, indicated by a red square followed by “Wins” in the top left corner of the screen.



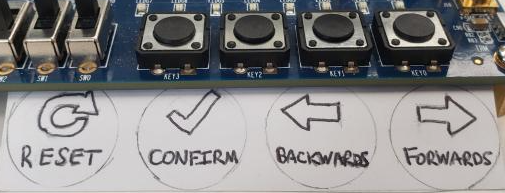
The second win condition is if player two wins, indicated by a magenta square followed by “Wins” in the top left corner of the screen.



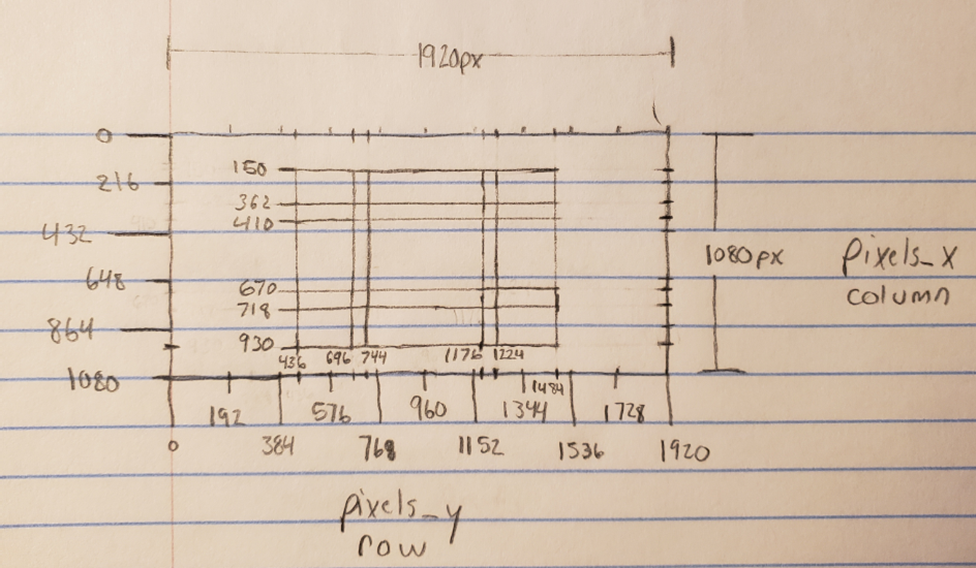
The third win condition is if the board is full and the player can no longer choose a square to place a piece, then “Tie” will display on the top left of the screen.



To aid the user in playing the game, we have included four different buttons. Two of the buttons are for moving the visual indicator to different squares on the screen. One button is used to confirm the player’s choice to place their piece. The last button is a full game reset, which resets the visuals and leaves the game board intact. The buttons were made by creating an updated symbol in the project schematic.

*Buttons and descriptions*

We have also successfully implemented the use of the DE2-115 board’s VGA port, with the help and usage of Scott’s DigiKey VGA code.[[1]](#footnote-0) After some minor adjustments to Scott’s schematic, the VGA allowed us the ability to display the game on a monitor screen, making the game itself more visually appealing. The board outlines were created by using a visual breakdown of the dimensions of the screen on a piece of paper to ensure our board was centered correctly.



Our project progress was consistently updated with the use of Gitlab, and later transferred on GitHub for public access. To progress and accomplish goals in the project, we took advantage of a Gantt chart and have completed around eighty five percent of it.

**Operation and Features**

The code’s structure is sequential with the use of signals and variables. Our signals are used to keep track of the game conditions as it is played, while variables are mainly used as counters to progress the game. We also made use of button states to keep track of each button press. The VGA is controlled by several if statements that include parameters so images can be created on the screen, which covers most of our code. Integration of the board’s rising edge clock allowed us to record our conditions and counters so they could be properly used inside the process. Finally, our reset state made it possible to clean the game board and restart the game by the use of a physical button.

*Features completely working:*

1. VGA
2. All four DE2-115 buttons

1. Moving visual indicator forwards one space

2. Moving visual indicator backwards one space

3. Player choice confirmation

4. Game reset

1. Two LED lights on the DE2-155 board

1. LEDR0 was used as the debugger for our move forward button.It would light up when the button was pressed. This let us know that our boolean button state was working properly with the clock’s rising edge.

2. LEDR1 was used as the debugger for our confirmation button. It would light up when the button was pressed. This let us know that our boolean state was working properly with the clock’s rising edge. The code for both leds was not commented out.

4. Game conditions display

1. Red wins
2. Magenta wins
3. Tie

*Features Partially Working:*

1. 7 segment display code used for debugging that has now been commented out. We used the display as a counter for each section of the board to make sure everything was counting correctly with button confirmations.
2. Smooth and perfect transition from each space on the tic-tac-toe board (random jumping at random times). Seemingly random, the block will jump to another block with no negative impact to the game. We even used our debugger stated above to verify.

*Features Not Working:*

1. All features included are working.

**Future Work**

In the future we were planning on making another game or multiple games depending on the time restriction which could be swapped to by flipping the sw17-sw14 switches to their on state. We first thought of doing a 5x5 grid to enhance the tic-tac-toe experience. Later on we thought about making a party pack game. This would consist of games that you could find at any family gathering, similar to how the “Jackbox” game is set up. A menu screen would have to be worked on as well so the user could navigate the different games visually when moving the switches to their on state.

Confetti celebration with variations of the winner’s color is something that we were very close to accomplishing. When a player won we would have small pixels of that player's color fall from the screen at a set clock rate and zig-zag pattern.

The last thing we decided we could work on is a tie condition met before the game board is completely full. We did not think of this until our demonstration when a student mentioned it to us. This would be a very good idea to implement later on and could be done after our counter hits a set number. The game would check some if statements to see if a player only needs one more block to win, if so no tie would happen but if both players need more than one block the game would indicate a tie.

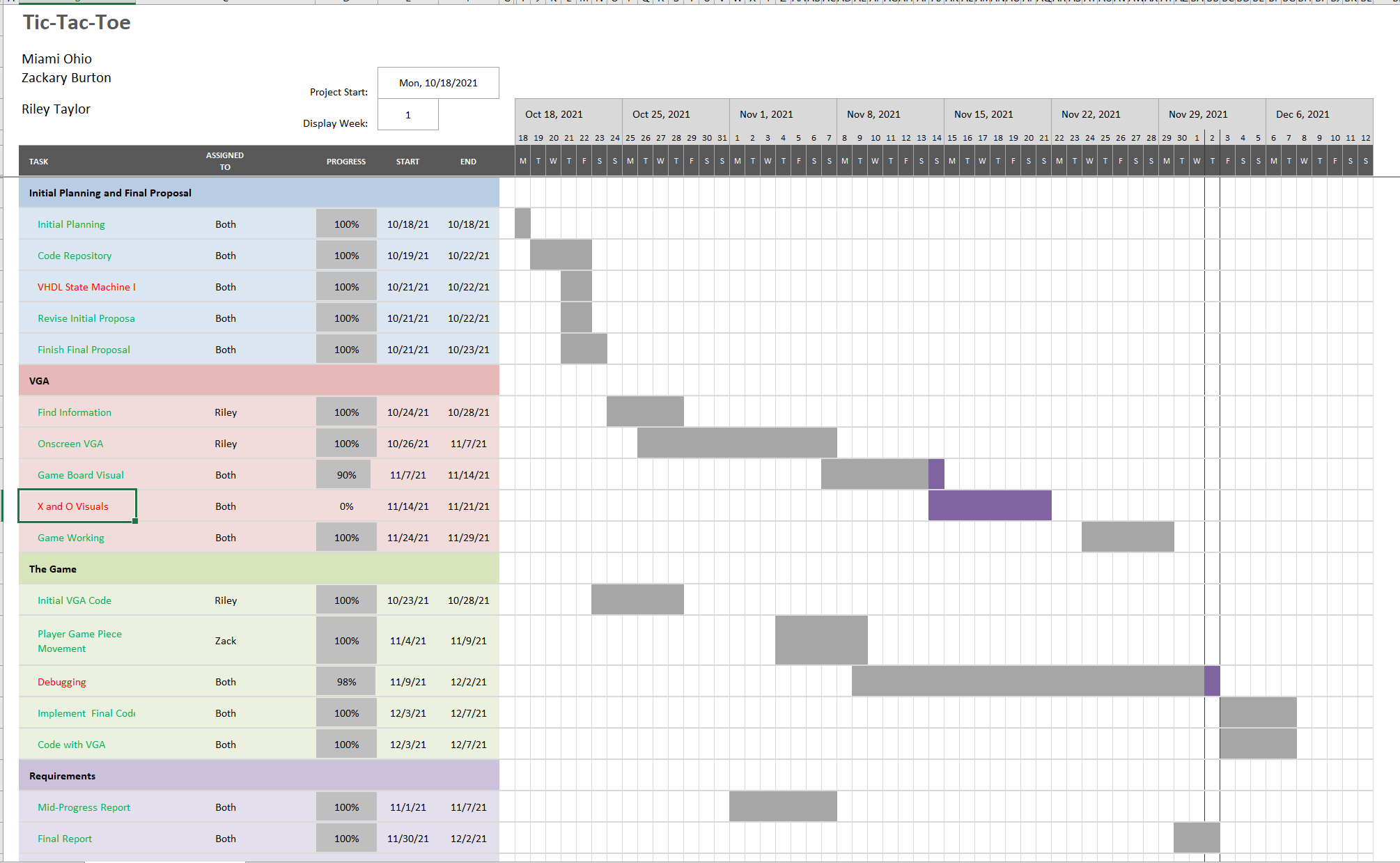
**GitHub**

<https://github.com/taylo550Riley/ECE287tictactoe>

The GitHub repository consists of the following:

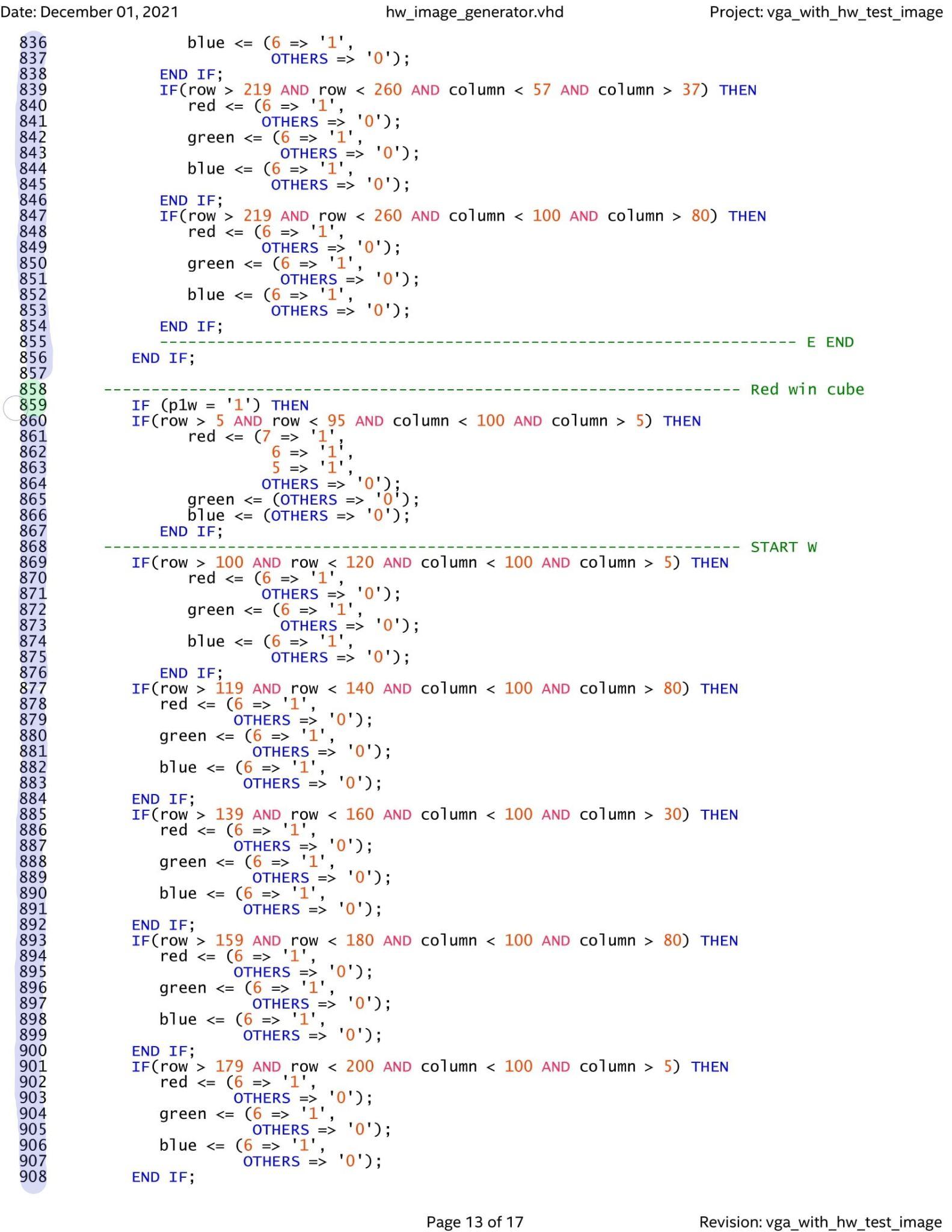
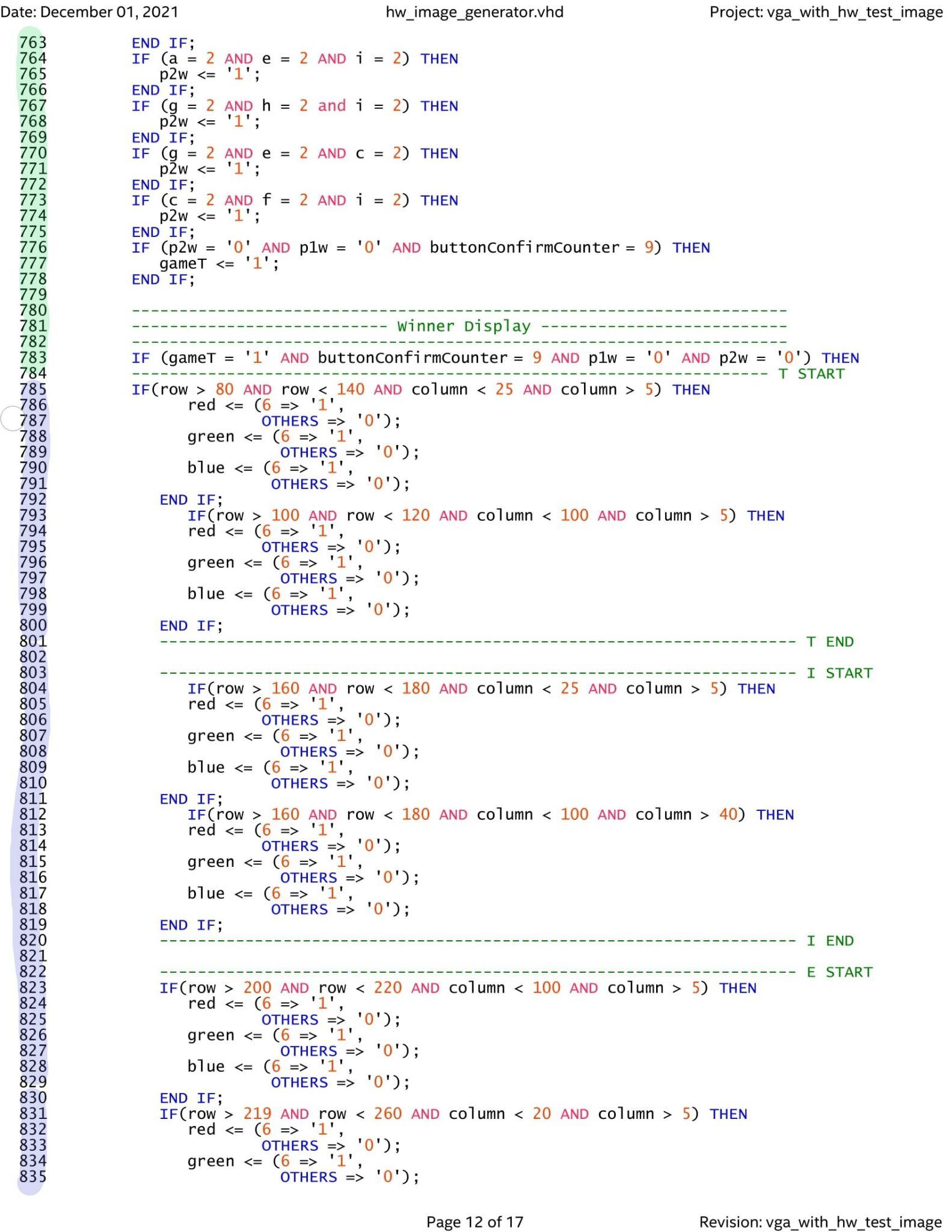
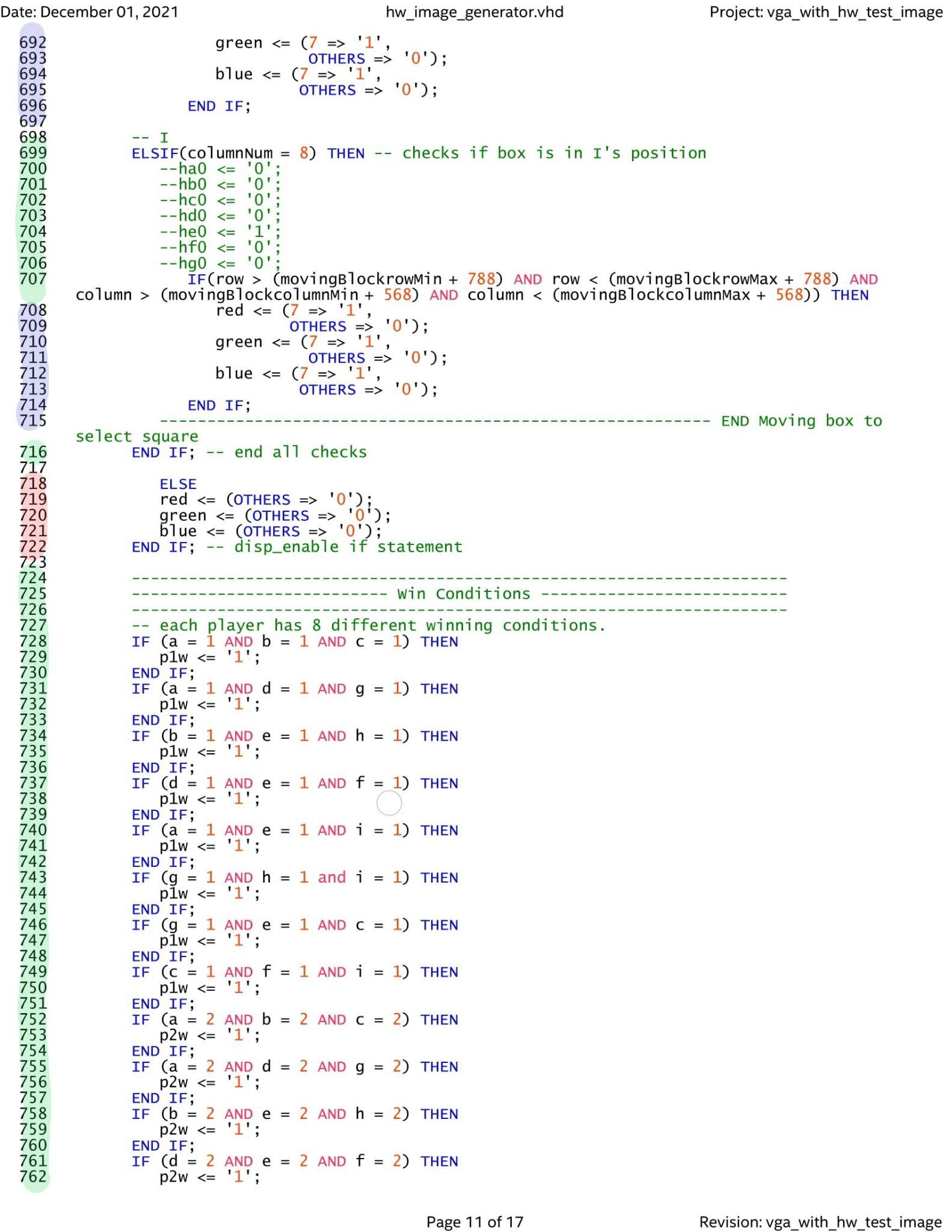
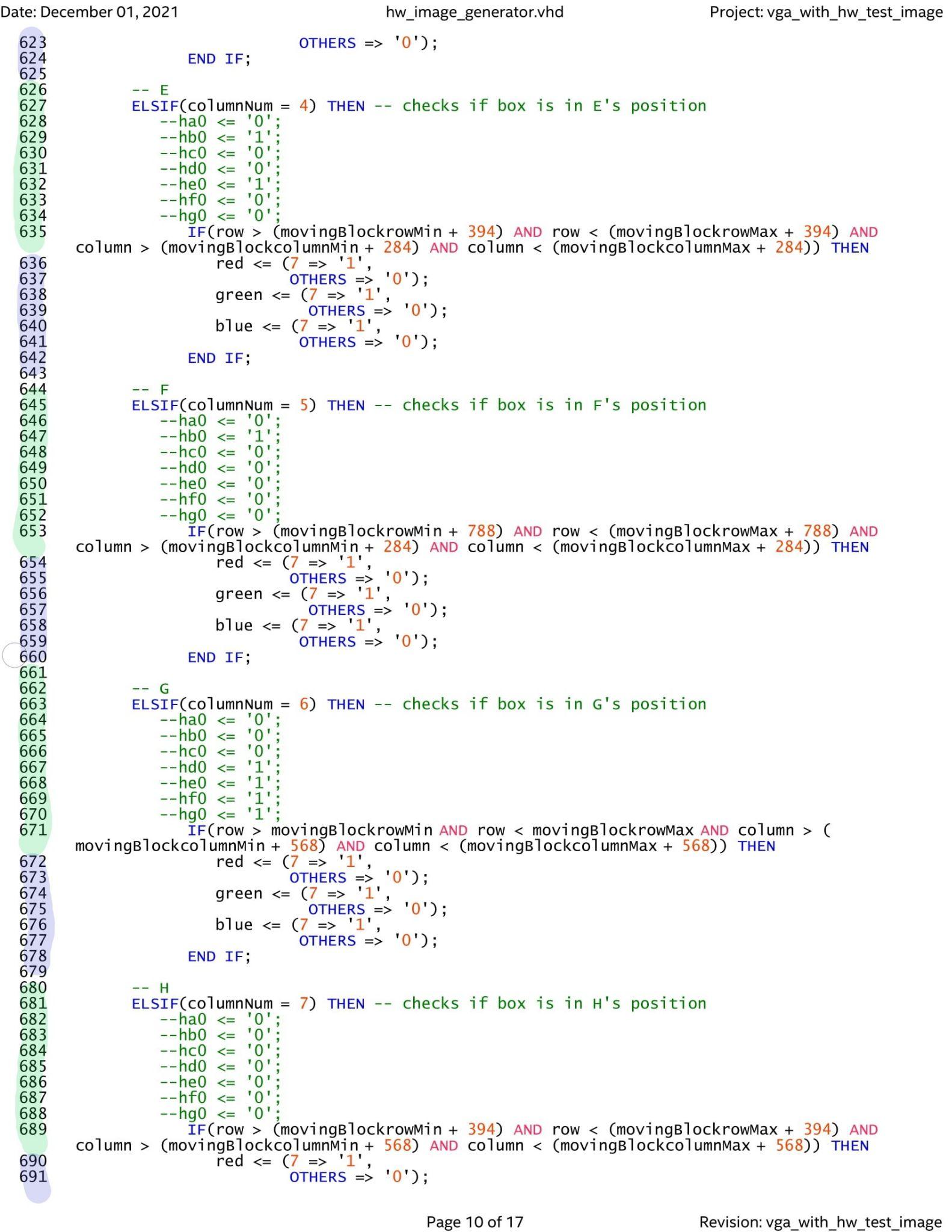
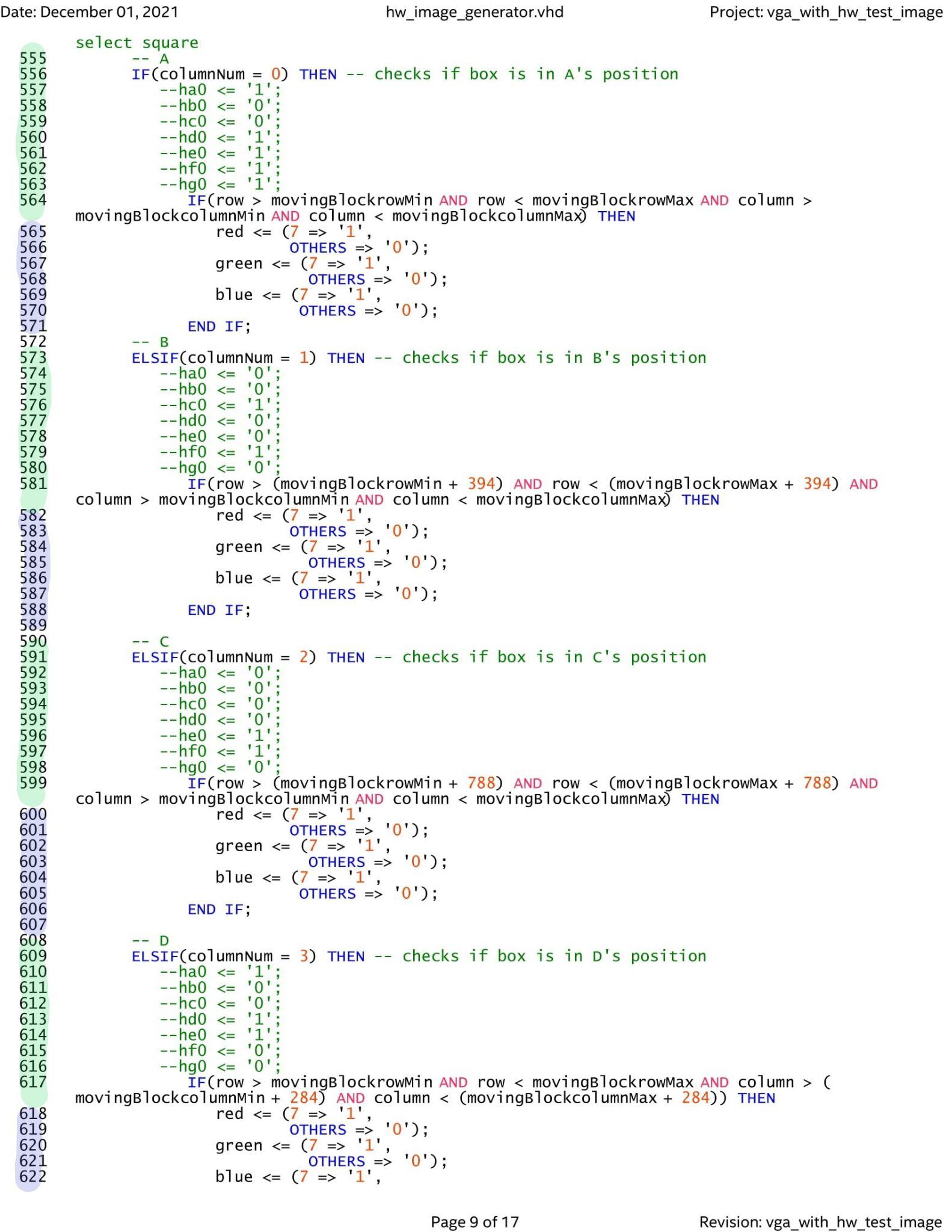
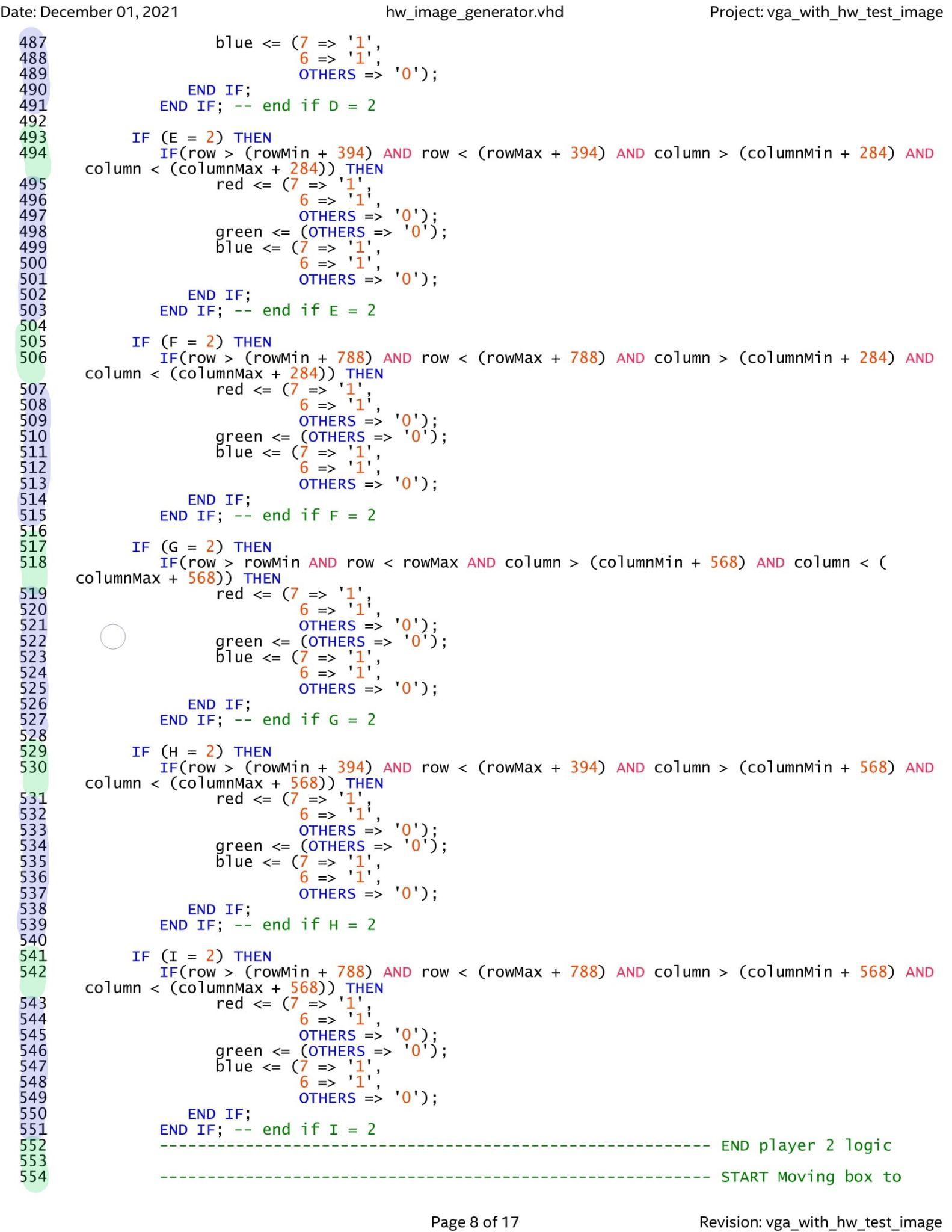
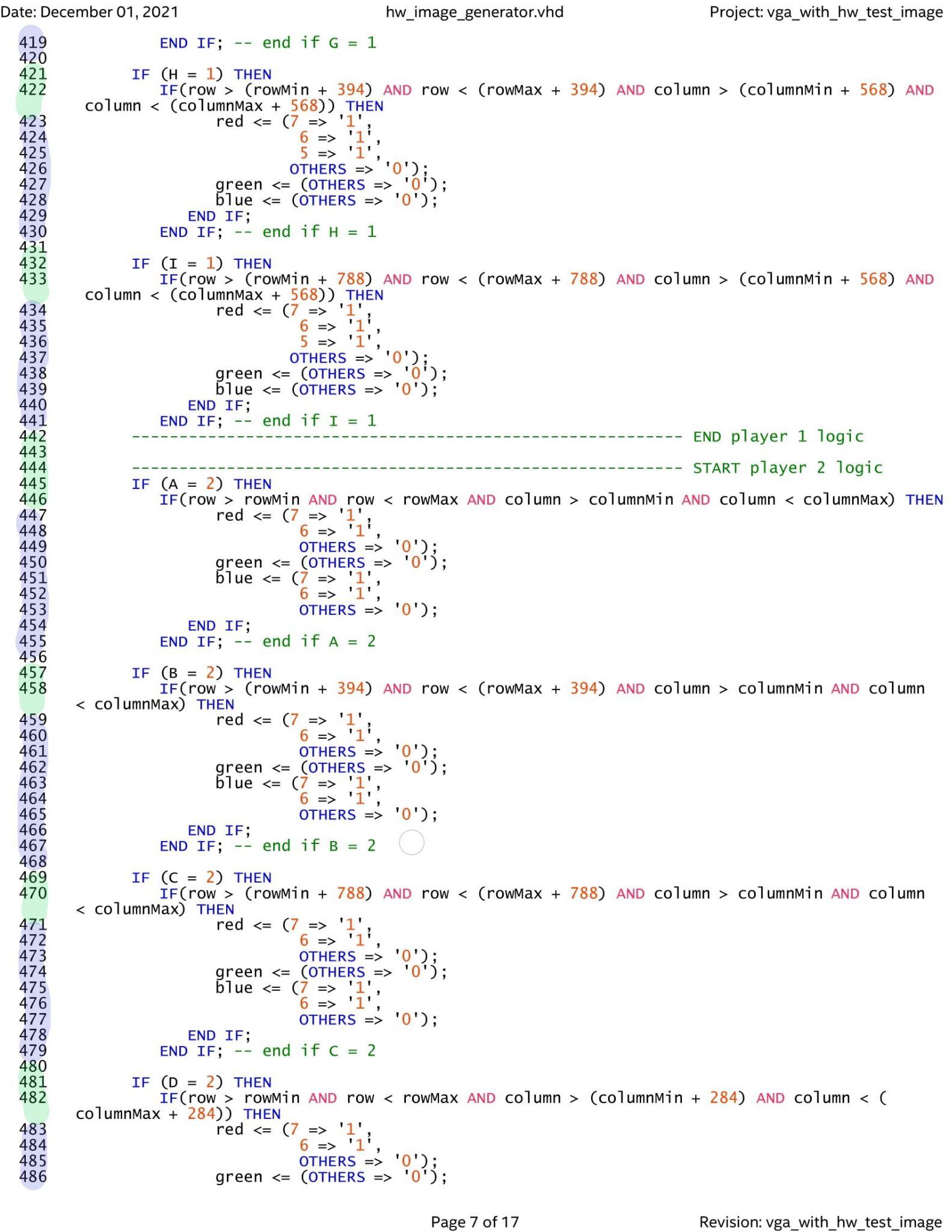
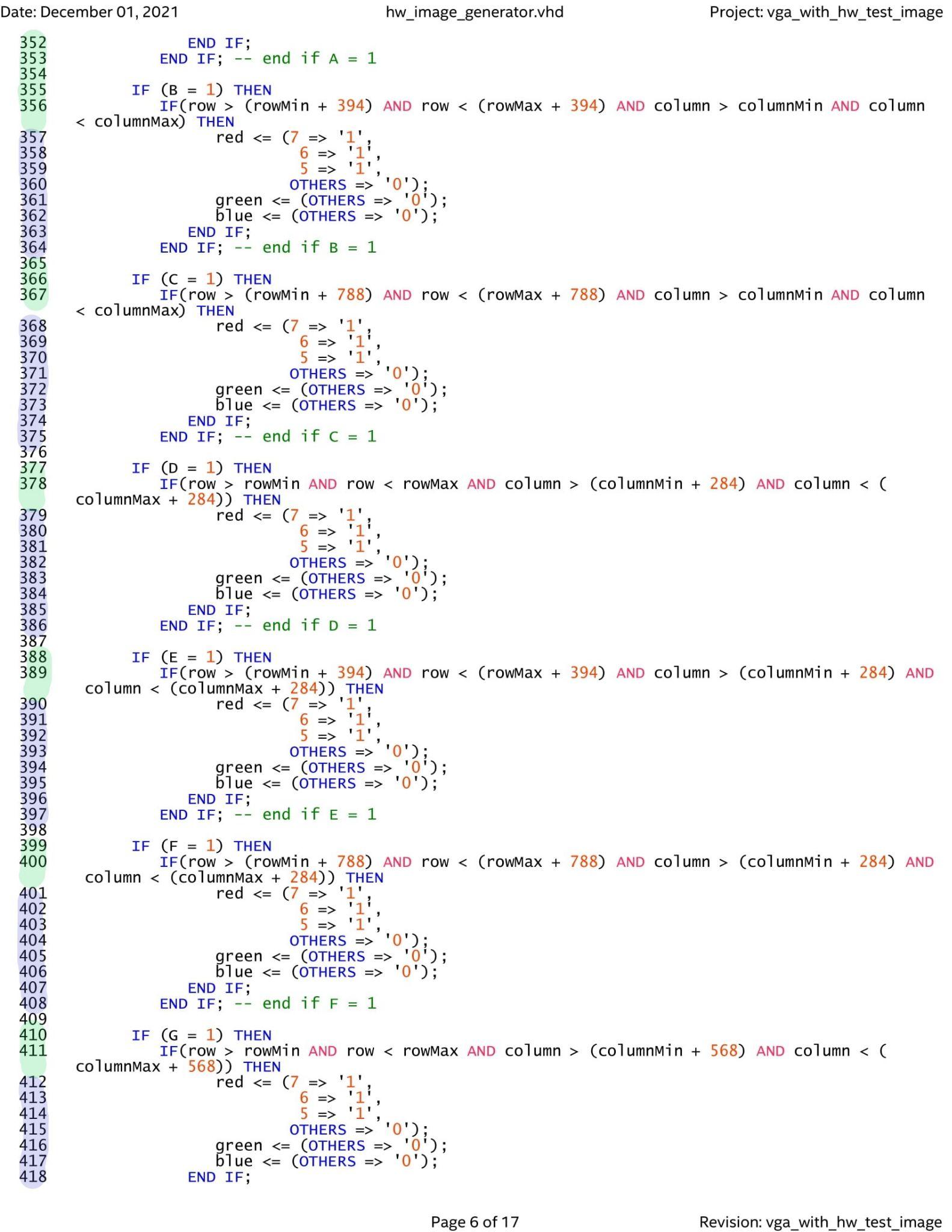
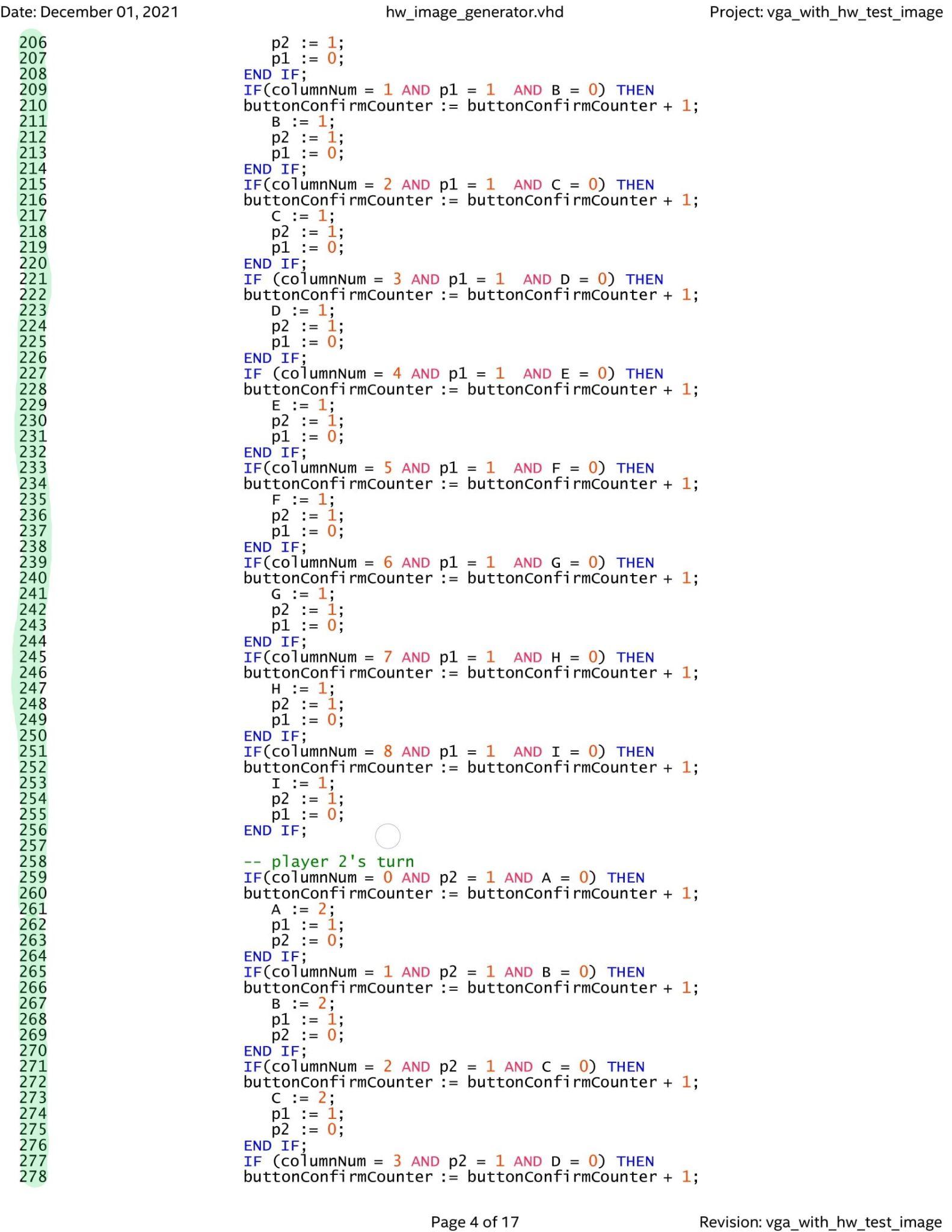
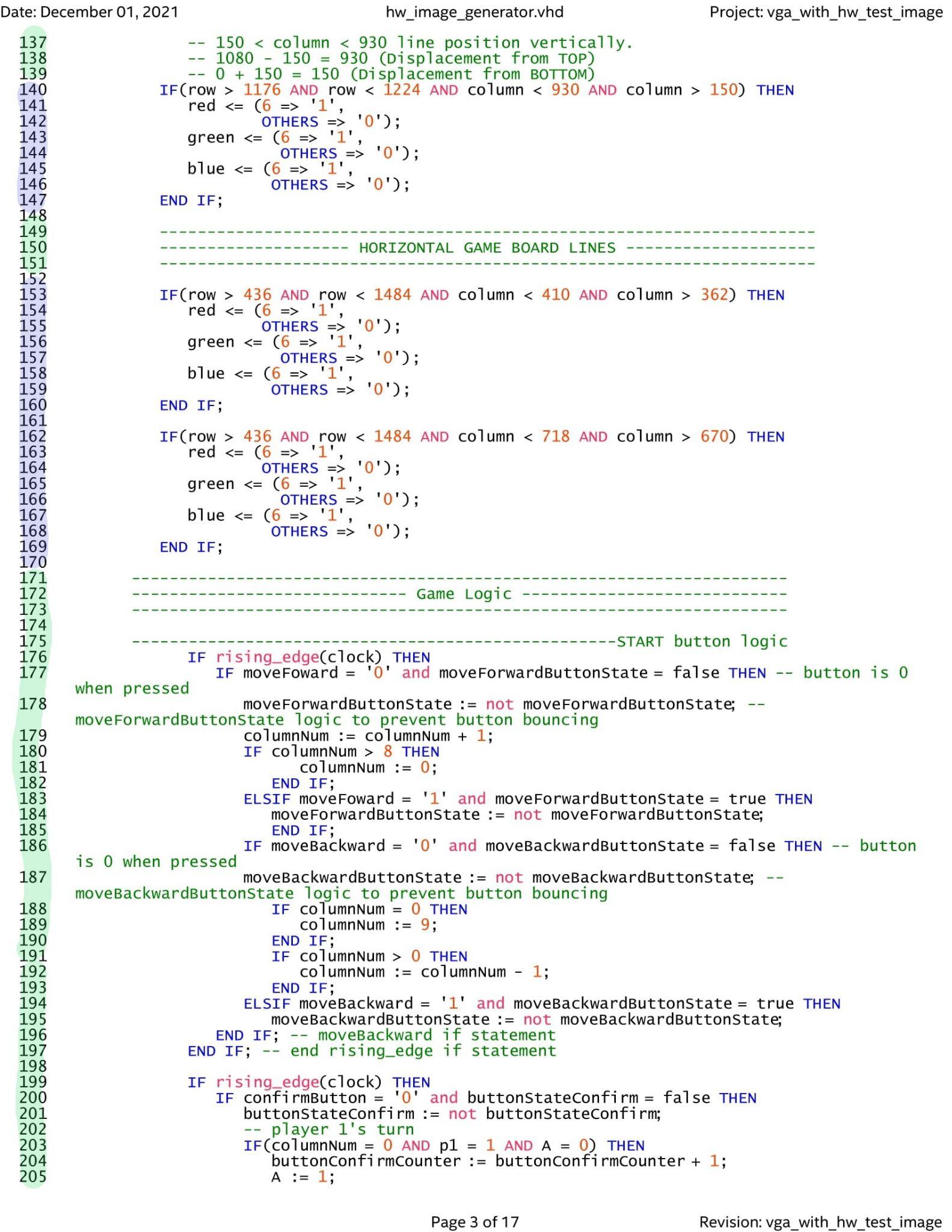
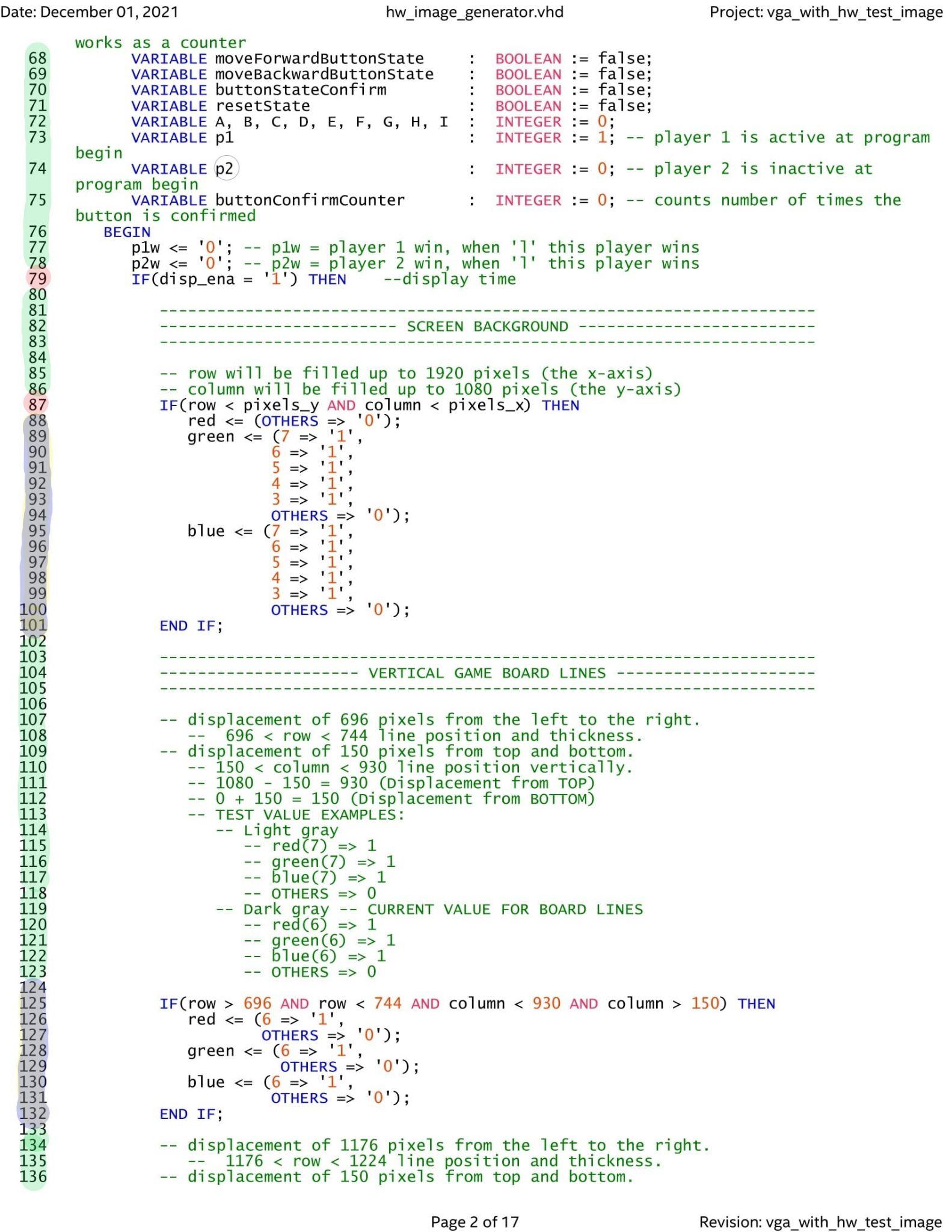
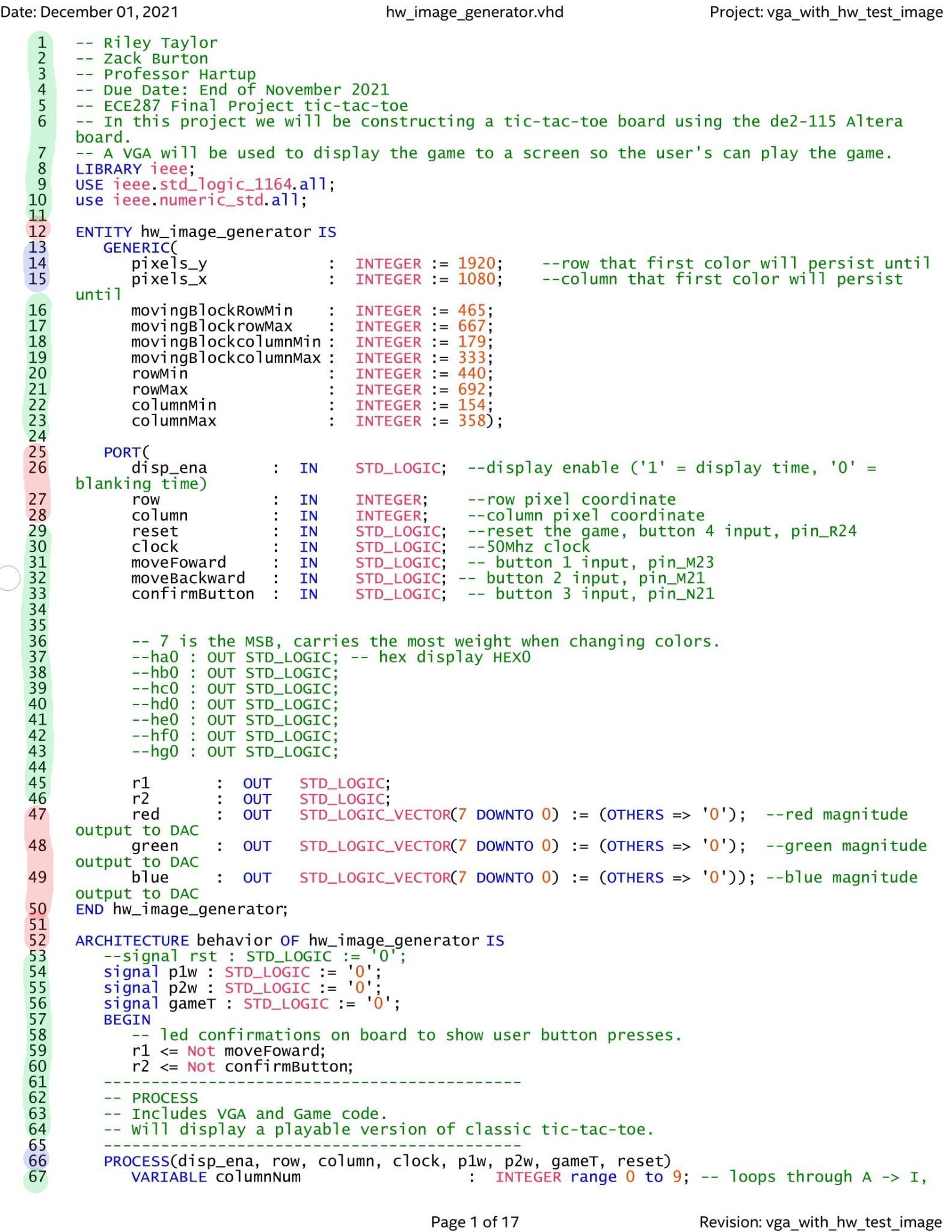
* Full VGA code and game code
* Completed Gantt chart
* The initial proposal, mid-progress report, and final report

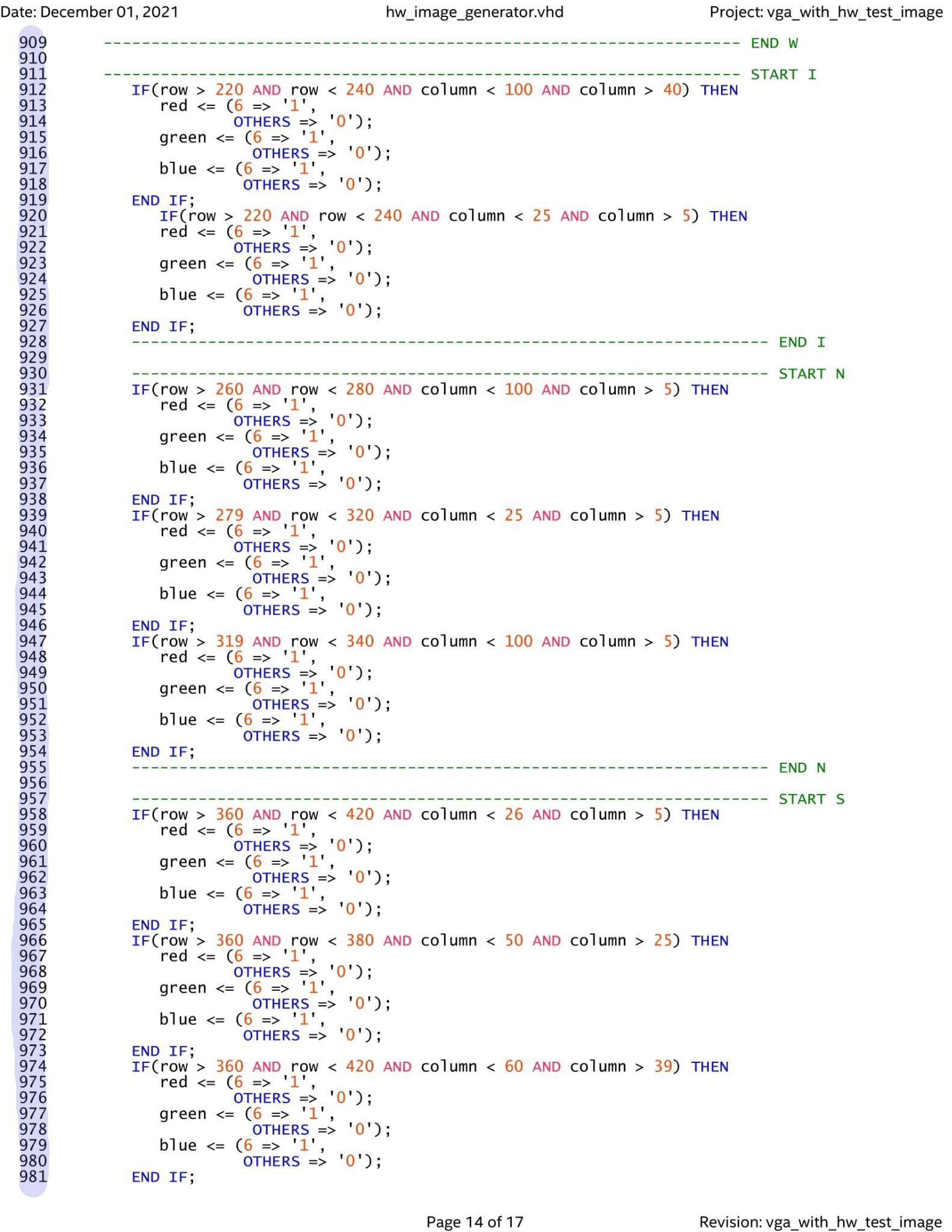
**Gantt chart**

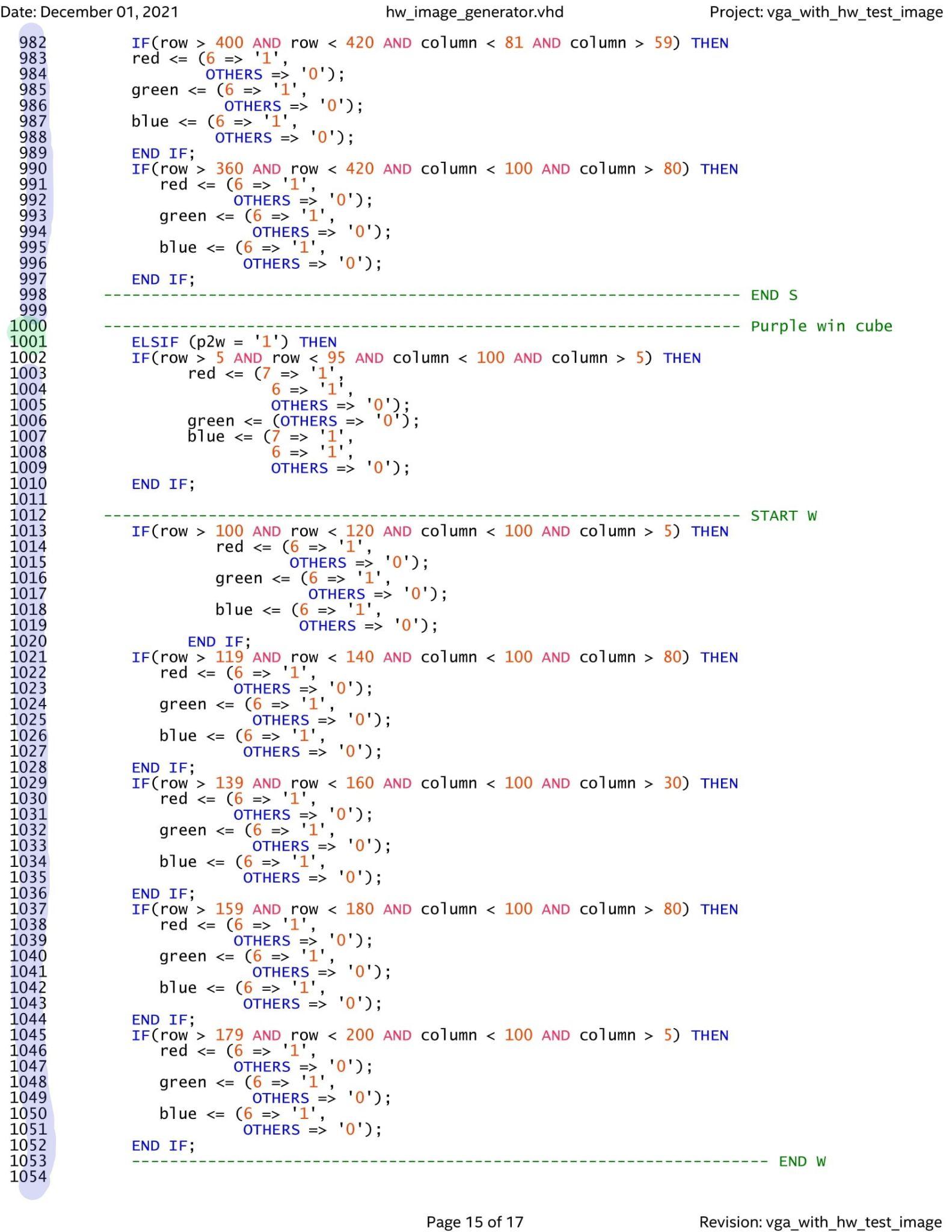


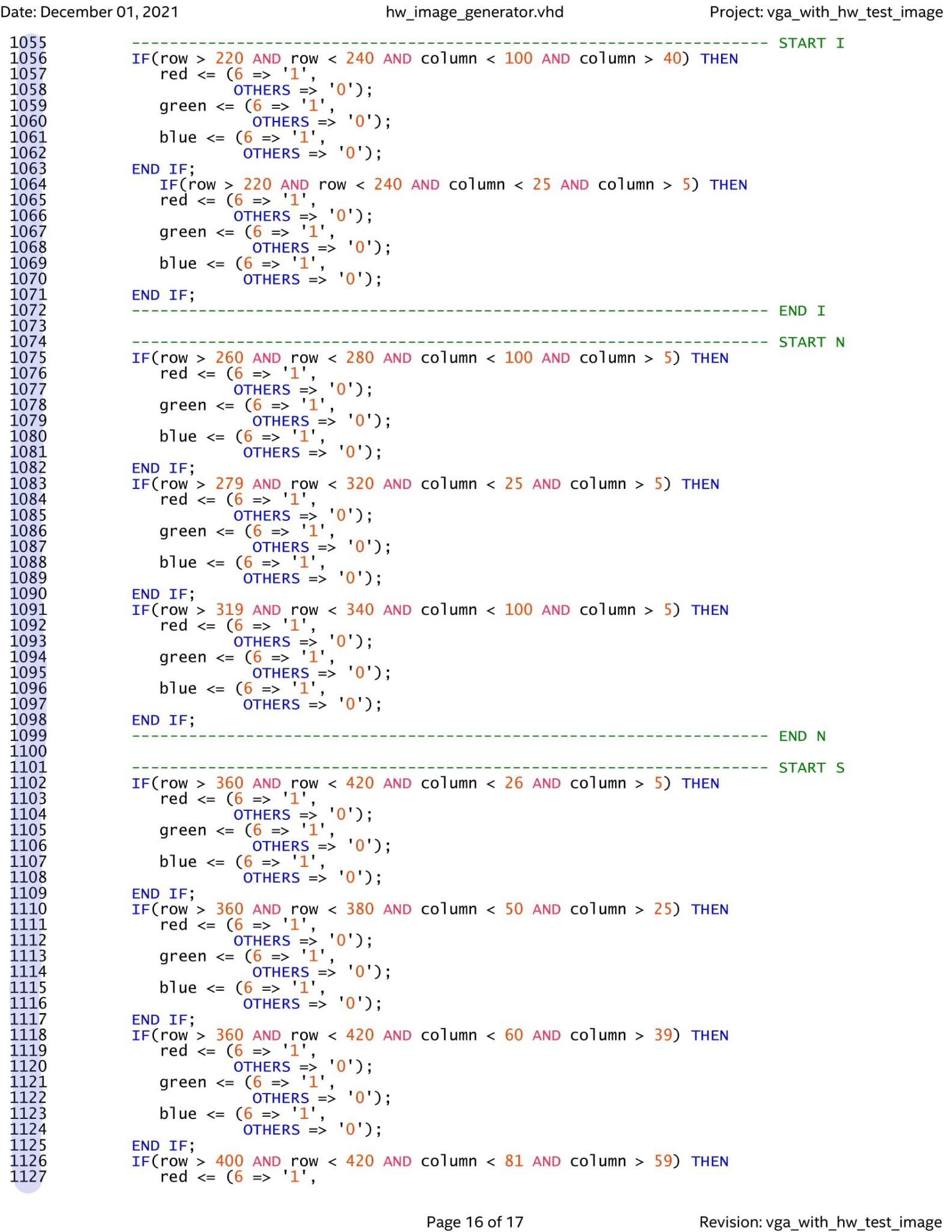
* Green Color Coding- The green represents all the tasks we have accomplished hundred percent, within the time we gave ourselves. We accomplished eighty five percent of the tasks on time that we assigned ourselves at the beginning of the project.
* Yellow Color Coding- The yellow represents all the tasks that are getting close to their due date and have not been finished. However due to this being our final report we do not have any yellow color coding.
* Red Color Coding- The red represents tasks that were not accomplished during the time given or tasks that were shelved. We have three tasks that are red. One is the state machine which we made red due to it being unrealistic and shelved early on in the project. The second was our X and O visuals; they were also shelved due to the complexity of the code. We could have made them however we feel that the player would understand their piece with color coded squares; this also would reduce the amount of code required to make the board pieces. Lastly, our debugging task is red due to our project still having minor, seemingly random visual bugs when the player moves them back and forth.

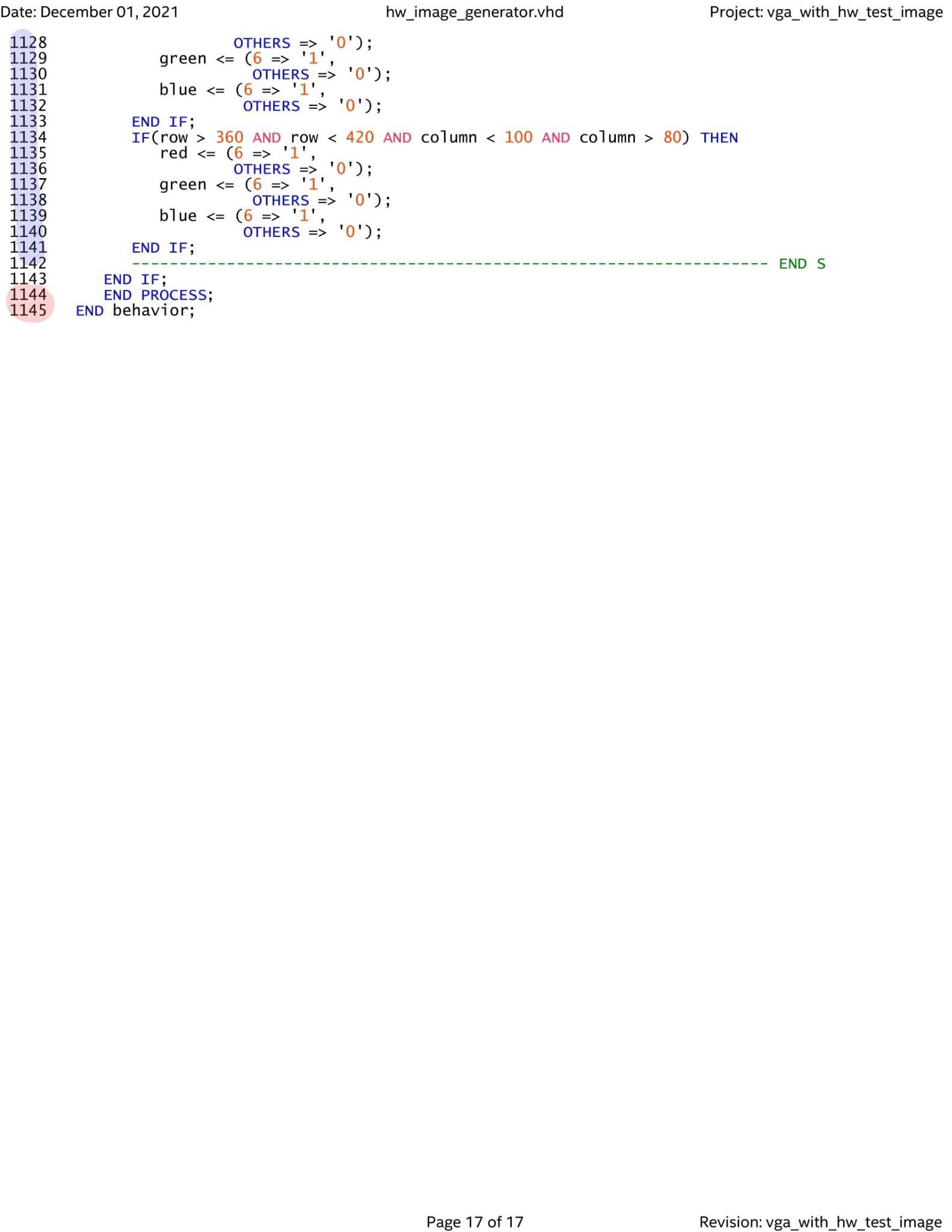
**Appendix**











1. Scott\_1767, 1995-2020 Digi-Key Electronics, <https://forum.digikey.com/t/vga-controller-vhdl/12794>.

   Last name was not shown. [↑](#footnote-ref-0)