Color Lines is a turn-based board game. Each turn, three randomly colored balls fall on random grids of a 9x9 squares board. Your mission is to re-locate the balls (by moving one of them per turn) to form lines consisting of five or more balls of the same color. The lines can be arranged vertically, horizontally, or diagonally. Each line you build immediately disappears, giving you points. There is basically no end to the game. You lose when the entire board is filled with balls.

Controls: [Q], [A], [O], [P], [SPACE].

Let me explain the code.
Line 0: arrays g and h represent horizontal and vertical directions, respectively. Directions are left, up, right, down, right-down, right-up. Elements 1-4 are used to move the ball by keyboard, elements 3-6 are used to detect lines. Then ball (16x16) and arrow (8x8) sprites are loaded as UDG characters. Playfield (9x9) is populated by invisible ball sprites, we'll change screen attributes to make color balls. Variable z is high score.
Line 1: initializing playfield f(25,25) and temporary t(25,25). Note that we really use 9x9 playfield (indices 9 to 17), other elements are needed for simplification of algorithms. Variable n is a counter of free cells, s is score, u and v are arrow coordinates, c and d are selected ball coordinates. Then I write the game title and high score. Finally, 5 initial balls are born: subroutine 7 creates one.
Line 2: drawing the arrow, asking for keypress, if it is up (Q), down (A), left (O), or right (P) - erasing the previous arrow and updating coordinates, then doing 2 again.
Line 3: if the key pressed is not SPACE or the current cell is empty and no ball has been selected, then write the current score (not the place to do so) and loop to 2.
Line 4: Clean the temporary matrix. If there is a ball in the current cell, unmark the previous ball (remove FLASH attribute) and mark the new one (set FLASH attribute). If no ball was selected I unmark the current one before marking it (avoiding IF statement). Loop to 2.
Line 5: So it is a new destination. Recursive filling of temporary matrix to reach the starting point. If no places were discovered on the current step (variable m), loop to 2. If can be reached, go to 6. Otherwise one more step, loop to 5.
Line 6: Clean the temporary matrix. Move the ball to new destination, reset mark. Subroutine 8 checks whether any line of >=5 balls appeared and removes them. Beep if so, born 3 balls otherwise. Loop to 2.
Line 7: Subroutine, born a new ball. We always know the number of free cells (n), so pick the random cell from that category. If impossible, update high score, beep, and restart the game from line 1.
Line 8: Subroutine, checking whether any line of >=5 balls appear. There are 4 directions (indices 3 to 6): horizontal, vertical, and two diagonals. Counting backward and forward (here extra zeroed elements are acessed), if sum >=5 then mark in the temporary matrix.
Line 9: Subroutine, redraw the playfield (by updating screen attributes). Before that we remove balls marked in temporary matrix (continuing line 8). For other contexts it is harmless (but time wasting). Return from any of subroutines (entry points are 7, 8, and 9).