Assembly Project: Dr Mario

Defne Eris & Dimitrios Gkiokmema

April 2, 2025

1 Instruction and Summary

1. Which milestones were implemented? Milestones 1-5
    (a) Milestone 1, drawing the container
    (b) Milestone 2, implementing keyboard controls and moving the capsule inside the bottle
    (c) Milestone 3, implemented the collision detection algorithm so that 4 in a row or column get deleted
    (d) For Milestone 4 and 5 we implement gravity, so that each second that passes will automatically move the capsule down one row.
    (e) Assuming that gravity has been implemented, the speed of gravity increase gradually over time, or after the player completes a certain number of rows.
    (f) When the player has reaches the "game over" condition, we display a Game Over screen in pixels on the screen. The player restarts the game if a "retry" option is chosen by the player. Retry starts a brand new game (no state is retained from previous attempts).
    (g) We Added sound effects for different conditions like rotating and dropping capsules, removing a row of squares, for beating a level and the game over condition.
    (h) If the player presses the keyboard key p, we display a "Paused" message on screen until they press p a second time, at which point the original game will resume.
    (i) We have implemented a panel on the side that displays a preview of the next capsule that will appear.
    (j) The panel mentioned above to displays a preview of the next 4-5 capsules, and have this preview update with each new capsule
    (k) We have drawn Dr. Mario and the viruses on the side panels, as in Figure 2.1
2. How to view the game:
    (a) View dimensions: 256 x 256
    (b) Pixel dimensions: 4 x 4

Figure 1: caption

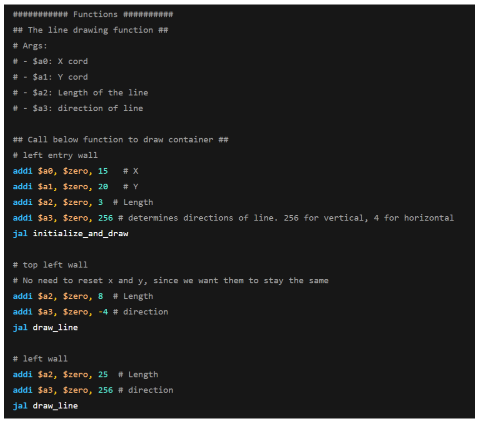
3. Game Summary:
    - Run game, click on bitmap for assembly to access the keyboard so that you can play
    - Keys: a left, s down, d right, w turns right, e turns left, q quits, p pauses/unpauses
    - If 'GAME OVER', press r to restart game or q to quit.

2 Attribution Table

Defne Eris	Dimitrios Gkiokmema 1010372286
Wrote code for the main keyboard controls	Wrote initial gravity
Drawing the viruses	Wrote initial deletion
Generating random colored viruses	Drawing container
Generating random colors for the pills	Drawing Dr. Mario (I used Defne's code)
Displaying and shifting future pills	GAME OVER condition
Merging the code files	Speeding up gravity
Wrote code for blocking keyboard movements	Pause/unpause
	Sound effects

3 Game Features

1. Drawing Lines: Created a function that given x, y, length, and direction variables, draws a line.



```
########### Functions ##########
## The line drawing function ##
# Args:
# - $a0: X cord
# - $a1: Y cord
# - $a2: Length of the line
# - $a3: direction of line

## Call below function to draw container ##
# left entry wall
addi $a0, $zero, 15   # X
addi $a1, $zero, 20   # Y
addi $a2, $zero, 3  # Length
addi $a3, $zero, 256 # determines directions of line. 256 for vertical, 4 for horizontal
jal initialize_and_draw

# top left wall
# No need to reset x and y, since we want them to stay the same
addi $a2, $zero, 8  # Length
addi $a3, $zero, -4 # direction
jal draw_line

# left wall
addi $a2, $zero, 25  # Length
addi $a3, $zero, 256 # direction
jal draw_line
```

Figure 2: Draw Line Example Usage

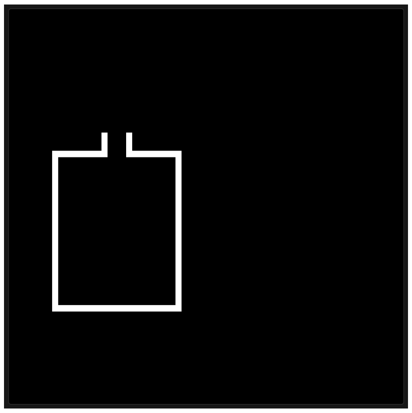
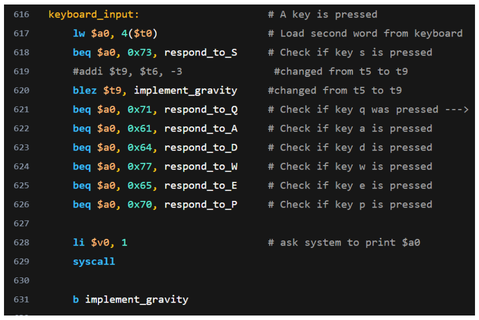


Figure 3: Draw Line Result

2. Keyboard Input: Created in-game functionality for all necessary keyboard inputs

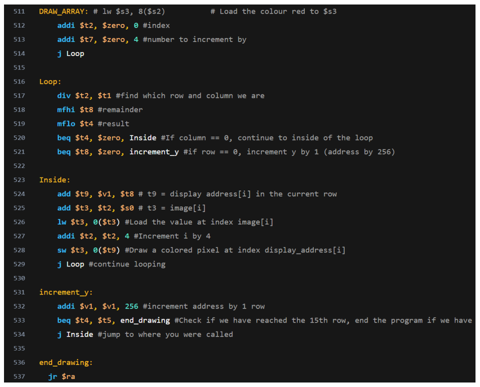


```
keyboard_input:                     # A key is pressed
    lw $a0, 4($t0)                  # Load second word from keyboard
    beq $a0, 0x73, respond_to_S     # Check if key s is pressed
    #addi $t9, $t6, -3               #changed from t5 to t9
    blez $t9, implement_gravity     #changed from t5 to t9
    beq $a0, 0x71, respond_to_Q     # Check if key q was pressed --->
    beq $a0, 0x61, respond_to_A     # Check if key a is pressed
    beq $a0, 0x64, respond_to_D     # Check if key d is pressed
    beq $a0, 0x77, respond_to_W     # Check if key w is pressed
    beq $a0, 0x65, respond_to_E     # Check if key e is pressed
    beq $a0, 0x70, respond_to_P     # Check if key p is pressed

    li $v0, 1                       # ask system to print $a0
    syscall

    b implement_gravity
```

Figure 4: Keyboard Input Code

3. Drawing Viruses: created code to read array values (0 and 1) and fill a pixel with a predetermined colour each time a 1 was encountered. This code was later updated to display the colour entered in the arrays directly.

4. Generating Random Viruses: Generates a red, blue, and yellow virus at random locations.



```
DRAW_ARRAY: # lw $s3, 8($s2)          # Load the colour red to $s3
    addi $t2, $zero, 0 #index
    addi $t7, $zero, 4 #number to increment by
    j Loop

Loop:
    div $t2, $t1 #find which row and column we are
    mfhi $t8 #remainder
    mflo $t4 #result
    beq $t4, $zero, Inside #If column == 0, continue to inside of the loop
    beq $t8, $zero, increment_y #if row == 0, increment y by 1 (address by 256)

Inside:
    add $t9, $v1, $t8 # t9 = display address[i] in the current row
    add $t3, $t2, $s0 # t3 = image[i]
    lw $t3, 0($t3) #Load the value at index image[i]
    addi $t2, $t2, 4 #Increment i by 4
    sw $t3, 0($t9) #Draw a colored pixel at index display_address[i]
    j Loop #continue looping

increment_y:
    addi $v1, $v1, 256 #increment address by 1 row
    beq $t4, $t5, end_drawing #Check if we have reached the 15th row, end the program if we have
    j Inside #jump to where you were called

end_drawing:
  jr $ra
```

Figure 5: Draw Array Example Usage

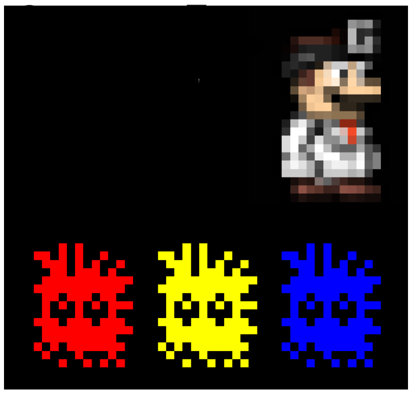
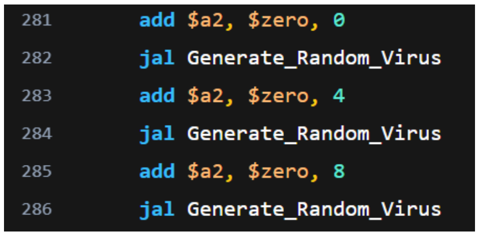


Figure 6: Draw Array Result



```
        add $a2, $zero, 0
        jal Generate_Random_Virus
        add $a2, $zero, 4
        jal Generate_Random_Virus
        add $a2, $zero, 8
        jal Generate_Random_Virus
```

Figure 7: Generate Viruses Example Usage

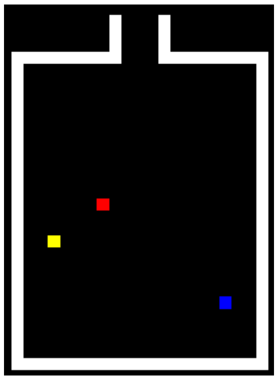


Figure 8: Generate Viruses Result

5. Generating Random Pills: Generates a pill with random colours.

```
        j GENERATE_PILL
```

Figure 9: Generate Viruses Example Usage

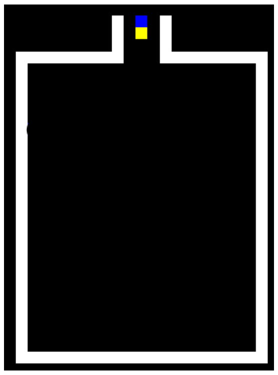
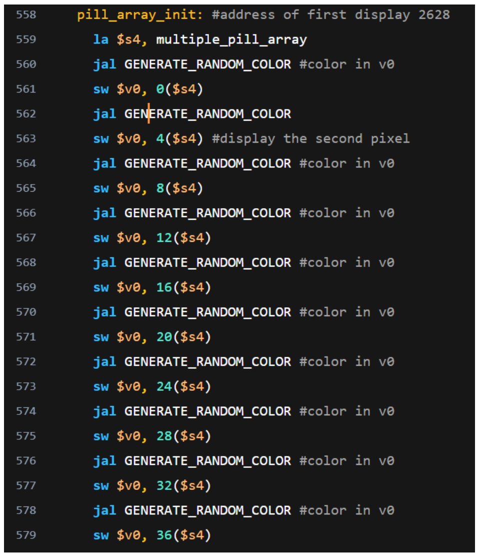


Figure 10: Generate Viruses Result

6. Displaying Future Pills: Created and displayed the next three
7. Collision Detection: implemented a way to prevent collisions
8. Gravity: brings down everything in the bottle impacted by gravity.
9. Game Over: Created a way to check if the bottle entrance is blocked, and end the game if so.



```
pill_array_init: #address of first display 2628
  la $s4, multiple_pill_array
  jal GENERATE_RANDOM_COLOR #color in v0
  sw $v0, 0($s4)
  jal GENERATE_RANDOM_COLOR
  sw $v0, 4($s4) #display the second pixel
  jal GENERATE_RANDOM_COLOR #color in v0
  sw $v0, 8($s4)
  jal GENERATE_RANDOM_COLOR #color in v0
  sw $v0, 12($s4)
  jal GENERATE_RANDOM_COLOR #color in v0
  sw $v0, 16($s4)
  jal GENERATE_RANDOM_COLOR #color in v0
  sw $v0, 20($s4)
  jal GENERATE_RANDOM_COLOR #color in v0
  sw $v0, 24($s4)
  jal GENERATE_RANDOM_COLOR #color in v0
  sw $v0, 28($s4)
  jal GENERATE_RANDOM_COLOR #color in v0
  sw $v0, 32($s4)
  jal GENERATE_RANDOM_COLOR #color in v0
  sw $v0, 36($s4)
```

Figure 11: Generate Viruses Example Usage

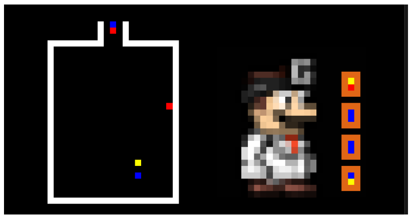
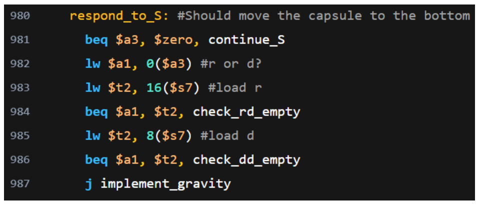


Figure 12: Generate Viruses Result

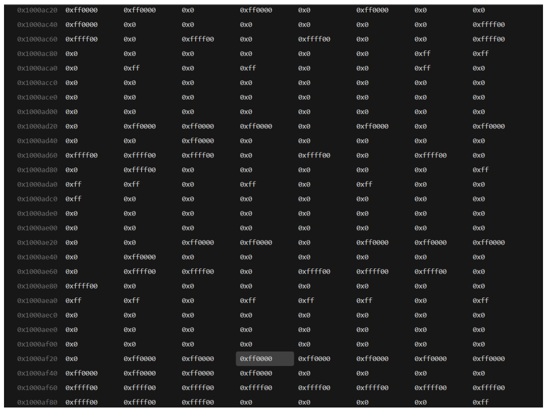


```
respond_to_S: #Should move the capsule to the bottom
  beq $a3, $zero, continue_S
  lw $a1, 0($a3) #r or d?
  lw $t2, 16($s7) #load r
  beq $a1, $t2, check_rd_empty
  lw $t2, 8($s7) #load d
  beq $a1, $t2, check_dd_empty
  j implement_gravity
```

Figure 13: Checks for collisions before moving

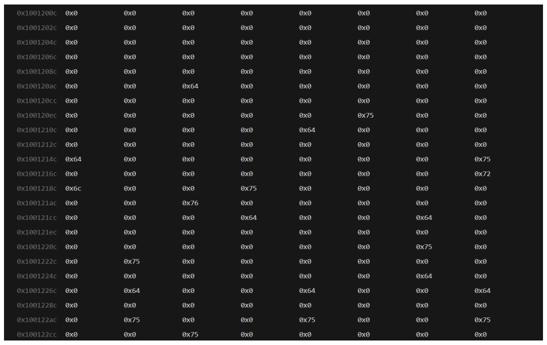
4 Memory Diagrams

1. In our .data section, we spare memory for the grid array that stores the positions of all the blocks in the bottle, the dr mario, virus, and bottle drawings, and our capsule array



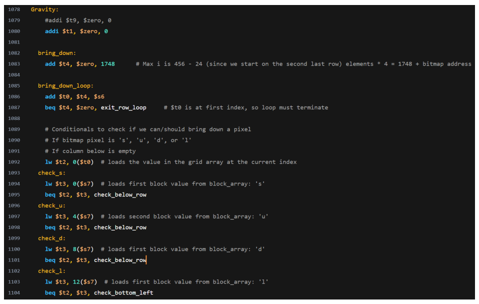
```
0x1000ac20 0xff0000   0xff0000   0x0        0xff0000   0x0        0xff0000   0x0        0x0
0x1000ac40 0xff0000   0x0        0x0        0x0        0x0        0x0        0x0        0xffff00
0x1000ac60 0xffff00   0x0        0xffff00   0x0        0xffff00   0x0        0x0        0xffff00
0x1000ac80 0x0        0x0        0x0        0x0        0x0        0x0        0xff       0xff
0x1000aca0 0x0        0xff       0x0        0xff       0x0        0x0        0xff       0x0
0x1000acc0 0x0        0x0        0x0        0x0        0x0        0x0        0x0        0x0
0x1000ace0 0x0        0x0        0x0        0x0        0x0        0x0        0x0        0x0
0x1000ad00 0x0        0x0        0x0        0x0        0x0        0x0        0x0        0x0
0x1000ad20 0x0        0xff0000   0xff0000   0xff0000   0x0        0xff0000   0x0        0xff0000
0x1000ad40 0x0        0x0        0xff0000   0x0        0x0        0x0        0x0        0x0
0x1000ad60 0xffff00   0xffff00   0xffff00   0x0        0xffff00   0x0        0xffff00   0x0
0x1000ad80 0x0        0xffff00   0x0        0x0        0x0        0x0        0x0        0xff
0x1000ada0 0xff       0xff       0x0        0xff       0x0        0xff       0x0        0x0
0x1000adc0 0xff       0x0        0x0        0x0        0x0        0x0        0x0        0x0
0x1000ade0 0x0        0x0        0x0        0x0        0x0        0x0        0x0        0x0
0x1000ae00 0x0        0x0        0x0        0x0        0x0        0x0        0x0        0x0
0x1000ae20 0x0        0x0        0xff0000   0xff0000   0x0        0xff0000   0xff0000   0xff0000
0x1000ae40 0x0        0xff0000   0x0        0x0        0x0        0x0        0x0        0x0
0x1000ae60 0x0        0xffff00   0xffff00   0x0        0xffff00   0xffff00   0xffff00   0x0
0x1000ae80 0xffff00   0x0        0x0        0x0        0x0        0x0        0x0        0x0
0x1000aea0 0xff       0xff       0x0        0xff       0xff       0xff       0x0        0xff
0x1000aec0 0x0        0x0        0x0        0x0        0x0        0x0        0x0        0x0
0x1000aee0 0x0        0x0        0x0        0x0        0x0        0x0        0x0        0x0
0x1000af00 0x0        0x0        0x0        0x0        0x0        0x0        0x0        0x0
0x1000af20 0x0        0xff0000   0xff0000   0xff0000   0xff0000   0xff0000   0xff0000   0xff0000
0x1000af40 0xff0000   0xff0000   0xff0000   0xff0000   0x0        0x0        0x0        0x0
0x1000af60 0xffff00   0xffff00   0xffff00   0xffff00   0xffff00   0xffff00   0xffff00   0xffff00
0x1000af80 0xffff00   0xffff00   0xffff00   0x0        0x0        0x0        0x0        0xff
```

Figure 16: Memory For The Virus Drawings



```
0x1001200c 0x0    0x0    0x0    0x0    0x0    0x0    0x0    0x0
0x1001202c 0x0    0x0    0x0    0x0    0x0    0x0    0x0    0x0
0x1001204c 0x0    0x0    0x0    0x0    0x0    0x0    0x0    0x0
0x1001206c 0x0    0x0    0x0    0x0    0x0    0x0    0x0    0x0
0x1001208c 0x0    0x0    0x0    0x0    0x0    0x0    0x0    0x0
0x100120ac 0x0    0x0    0x64   0x0    0x0    0x0    0x0    0x0
0x100120cc 0x0    0x0    0x0    0x0    0x0    0x0    0x0    0x0
0x100120ec 0x0    0x0    0x0    0x0    0x0    0x75   0x0    0x0
0x1001210c 0x0    0x0    0x0    0x0    0x64   0x0    0x0    0x0
0x1001212c 0x0    0x0    0x0    0x0    0x0    0x0    0x0    0x0
0x1001214c 0x64   0x0    0x0    0x0    0x0    0x0    0x0    0x75
0x1001216c 0x0    0x0    0x0    0x0    0x0    0x0    0x0    0x72
0x1001218c 0x6c   0x0    0x0    0x75   0x0    0x0    0x0    0x0
0x100121ac 0x0    0x0    0x76   0x0    0x0    0x0    0x0    0x0
0x100121cc 0x0    0x0    0x0    0x64   0x0    0x0    0x64   0x0
0x100121ec 0x0    0x0    0x0    0x0    0x0    0x0    0x0    0x0
0x1001220c 0x0    0x0    0x0    0x0    0x0    0x0    0x75   0x0
0x1001222c 0x0    0x75   0x0    0x0    0x0    0x0    0x0    0x0
0x1001224c 0x0    0x0    0x0    0x0    0x0    0x0    0x64   0x0
0x1001226c 0x0    0x64   0x0    0x0    0x64   0x0    0x0    0x64
0x1001228c 0x0    0x0    0x0    0x0    0x0    0x0    0x0    0x0
0x100122ac 0x0    0x75   0x0    0x0    0x75   0x0    0x0    0x75
0x100122cc 0x0    0x0    0x75   0x0    0x0    0x0    0x0    0x0
```

Figure 17: Example Memory For The Grid Array



```
Gravity:
    #addi $t9, $zero, 0
    addi $t1, $zero, 0

  bring_down:
    add $t4, $zero, 1748      # Max i is 456 - 24 (since we start on the second last row) elements * 4 = 1748 + bitmap address

  bring_down_loop:
    add $t0, $t4, $s6
    beq $t4, $zero, exit_row_loop     # $t0 is at first index, so loop must terminate

    # Conditionals to check if we can/should bring down a pixel
    # If bitmap pixel is 's', 'u', 'd', or 'l'
    # If column below is empty
    lw $t2, 0($t0)  # loads the value in the grid array at the current index
  check_s:
    lw $t3, 0($s7)  # loads first block value from block_array: 's'
    beq $t2, $t3, check_below_row
  check_u:
    lw $t3, 4($s7)  # loads second block value from block_array: 'u'
    beq $t2, $t3, check_below_row
  check_d:
    lw $t3, 8($s7)  # loads first block value from block_array: 'd'
    beq $t2, $t3, check_below_row
  check_l:
    lw $t3, 12($s7)  # loads first block value from block_array: 'l'
    beq $t2, $t3, check_bottom_left
```

Figure 14: Implements Gravity

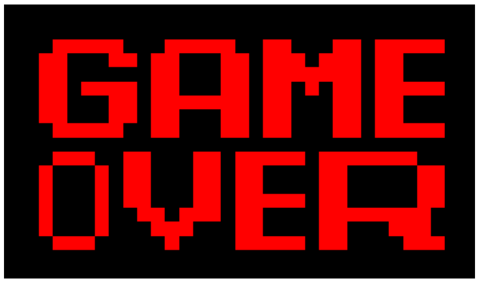
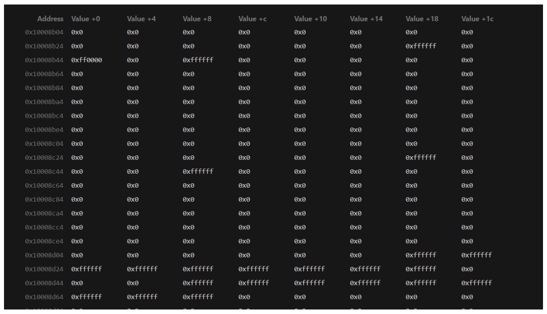
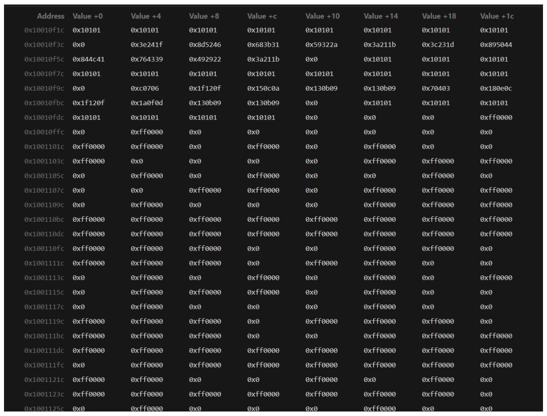


Figure 15: Game Over condition



```
   Address  Value +0   Value +4   Value +8   Value +c   Value +10  Value +14  Value +18  Value +1c
0x10008b04  0x0        0x0        0x0        0x0        0x0        0x0        0x0        0x0
0x10008b24  0x0        0x0        0x0        0x0        0x0        0x0        0xffffff   0x0
0x10008b44  0xff0000   0x0        0xffffff   0x0        0x0        0x0        0x0        0x0
0x10008b64  0x0        0x0        0x0        0x0        0x0        0x0        0x0        0x0
0x10008b84  0x0        0x0        0x0        0x0        0x0        0x0        0x0        0x0
0x10008ba4  0x0        0x0        0x0        0x0        0x0        0x0        0x0        0x0
0x10008bc4  0x0        0x0        0x0        0x0        0x0        0x0        0x0        0x0
0x10008be4  0x0        0x0        0x0        0x0        0x0        0x0        0x0        0x0
0x10008c04  0x0        0x0        0x0        0x0        0x0        0x0        0x0        0x0
0x10008c24  0x0        0x0        0x0        0x0        0x0        0x0        0xffffff   0x0
0x10008c44  0x0        0x0        0xffffff   0x0        0x0        0x0        0x0        0x0
0x10008c64  0x0        0x0        0x0        0x0        0x0        0x0        0x0        0x0
0x10008c84  0x0        0x0        0x0        0x0        0x0        0x0        0x0        0x0
0x10008ca4  0x0        0x0        0x0        0x0        0x0        0x0        0x0        0x0
0x10008cc4  0x0        0x0        0x0        0x0        0x0        0x0        0x0        0x0
0x10008ce4  0x0        0x0        0x0        0x0        0x0        0x0        0x0        0x0
0x10008d04  0x0        0x0        0x0        0x0        0x0        0x0        0xffffff   0xffffff
0x10008d24  0xffffff   0xffffff   0xffffff   0xffffff   0xffffff   0xffffff   0xffffff   0x0
0x10008d44  0x0        0x0        0xffffff   0xffffff   0xffffff   0xffffff   0xffffff   0xffffff
0x10008d64  0xffffff   0xffffff   0xffffff   0x0        0x0        0x0        0x0        0x0
```

Figure 18: Memory For The Bottle Drawing



```
   Address  Value +0   Value +4   Value +8   Value +c   Value +10  Value +14  Value +18  Value +1c
0x10010f1c  0x10101    0x10101    0x10101    0x10101    0x10101    0x10101    0x10101    0x10101
0x10010f3c  0x0        0x3e241f   0x8d5246   0x683b31   0x59322a   0x3a211b   0x3c231d   0x895044
0x10010f5c  0x844c41   0x764339   0x492922   0x3a211b   0x0        0x10101    0x10101    0x10101
0x10010f7c  0x10101    0x10101    0x10101    0x10101    0x10101    0x10101    0x10101    0x10101
0x10010f9c  0x0        0xc0706    0x1f120f   0x150c0a   0x130b09   0x130b09   0x70403    0x180e0c
0x10010fbc  0x1f120f   0x1a0f0d   0x130b09   0x130b09   0x0        0x10101    0x10101    0x10101
0x10010fdc  0x10101    0x10101    0x10101    0x10101    0x0        0x0        0x0        0xff0000
0x10010ffc  0x0        0xff0000   0x0        0x0        0x0        0x0        0x0        0x0
0x1001101c  0xff0000   0xff0000   0x0        0xff0000   0x0        0xff0000   0x0        0x0
0x1001103c  0xff0000   0x0        0x0        0x0        0x0        0xff0000   0xff0000   0xff0000
0x1001105c  0x0        0xff0000   0x0        0xff0000   0x0        0x0        0xff0000   0x0
0x1001107c  0x0        0x0        0xff0000   0xff0000   0x0        0xff0000   0xff0000   0xff0000
0x1001109c  0x0        0xff0000   0x0        0x0        0x0        0xff0000   0xff0000   0xff0000
0x100110bc  0xff0000   0xff0000   0xff0000   0xff0000   0xff0000   0xff0000   0xff0000   0xff0000
0x100110dc  0xff0000   0xff0000   0xff0000   0xff0000   0xff0000   0xff0000   0xff0000   0xff0000
0x100110fc  0xff0000   0xff0000   0xff0000   0x0        0x0        0xff0000   0xff0000   0x0
0x1001111c  0xff0000   0xff0000   0xff0000   0x0        0xff0000   0xff0000   0x0        0x0
0x1001113c  0x0        0xff0000   0x0        0xff0000   0x0        0xff0000   0x0        0xff0000
0x1001115c  0x0        0xff0000   0xff0000   0xff0000   0x0        0xff0000   0x0        0x0
0x1001117c  0x0        0xff0000   0x0        0x0        0x0        0xff0000   0x0        0x0
0x1001119c  0xff0000   0xff0000   0xff0000   0x0        0xff0000   0xff0000   0xff0000   0x0
0x100111bc  0xff0000   0xff0000   0xff0000   0x0        0x0        0xff0000   0xff0000   0xff0000
0x100111dc  0xff0000   0xff0000   0xff0000   0xff0000   0xff0000   0xff0000   0xff0000   0xff0000
0x100111fc  0x0        0xff0000   0xff0000   0xff0000   0xff0000   0xff0000   0xff0000   0xff0000
0x1001121c  0xff0000   0xff0000   0x0        0x0        0xff0000   0x0        0xff0000   0x0
0x1001123c  0xff0000   0xff0000   0xff0000   0xff0000   0xff0000   0xff0000   0xff0000   0x0
0x1001125c  0x0        0xff0000   0x0        0x0        0x0        0xff0000   0x0        0x0
```

Figure 19: Memory For The Dr Mario and Caplsules