# AES Snake Game

In this report, the process of designing the snake game on the Zybo Z7-10 board from Digilent is described. From the initial specifications to the final results. For more information on the project and the board files, visit the project GitHub https://github.com/CasperHermans2000/ZYBO-Snake-Game.

## Specifications

For the mini project, the goal is to make a program similar to the game snake. This game will take place on a grid. The player will be able to move a small “snake” in four directions: Up, down, left and right. You cannot move in the opposite direction of where you’re currently going. On the borders of the screen there will be a line that indicates the edge of the room. When the snake touches this edge, you lose the game.

During the game, dots will appear on the screen. The player must manoeuvre the snake towards these dots. When the snake touches one of these dots, the tail of the snake will become longer. When the head of the snail touches its tail, you also lose the game.

The FPGA will handle the overlay; and control either the HDMI or VGA to display images on a screen. The Microprocessor will handle the GPIO and communicate the changes made by the inputs to the FPGA.

| **Requirement** | **Priority** | **Result** |
|---|---|---|
| Display an image on a monitor | Must | Completed |
| Make a connection between the FPGA and the microcontroller | Must | Completed |
| Implement the basic snake game | Should | Completed |
| Show the current score on the screen | Should | Not completed |
| Add extra rules to the game | Could | Completed |

## Research

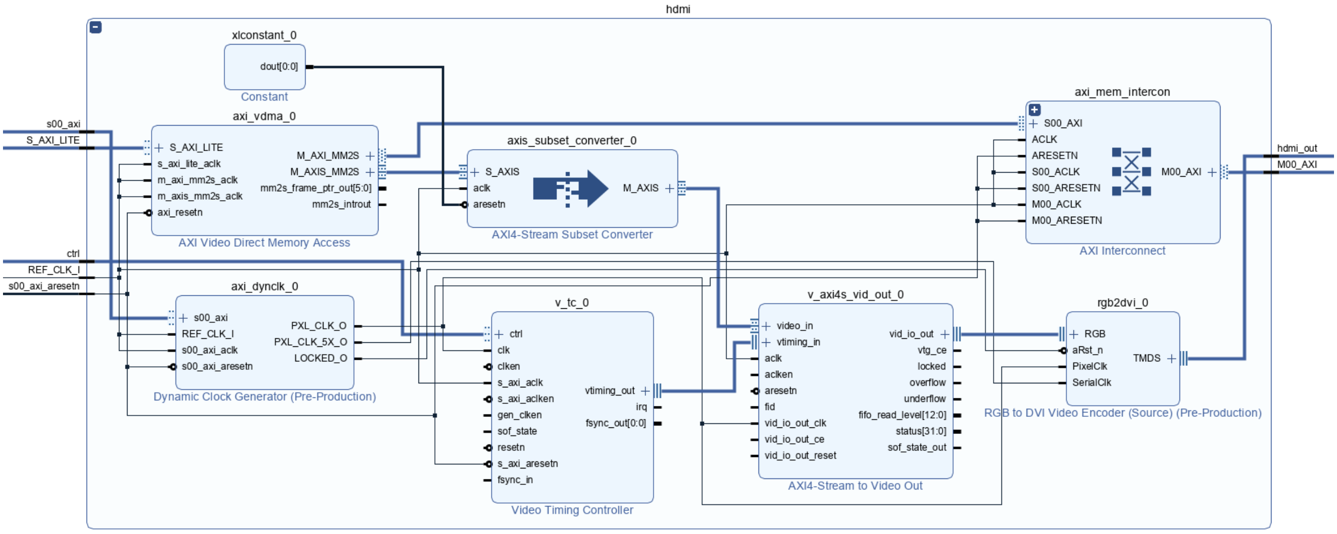
For the HDMI/VGA controller multiple examples are available. One such example is the Zybo Z7 HDMI Input/Output demo from Digilent. This demo demonstrates how the HDMI source on the Zybo Z7 can be used to stream data from the HDMI sink to the HDMI source. [1]

After testing multiple other HDMI examples and libraries, eventually the Zybo Z7 HDMI Output demo from the University of York was found. [2] This demo is an extension of the example made by Digilent and other Digilent Vivado libraries. The libraries used are the Dynamic Clock Generator IP, the RGB to DVI Video Encoder IP and the Transition-minimized Differential Signaling interface.

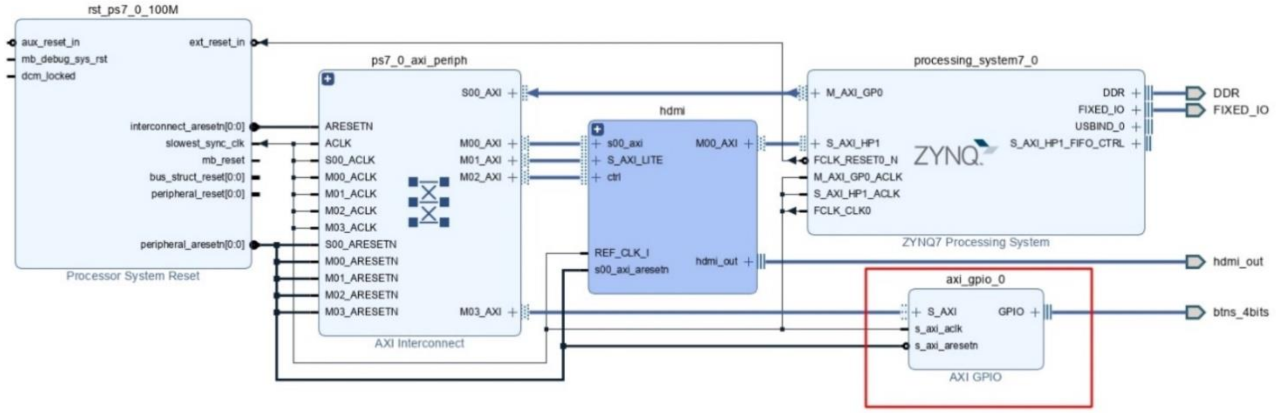
In this demo is also a software example included that shows how to write to the back buffer and switch between the active and inactive frame to animate an image onto the HDMI output. This demo is a good starting point for the project.

## Implementation hardware from library

Using the demo, the following HDMI hierarchy is created. The HDMI output uses the AXI Direct Memory Access (axi_vdma_0) IP to instantiate direct memory access between the memory and the AXI4-Stream target peripherals. To ensure the HDMI has the best possible performance the High Performance AXI 1 is used.



For this project an AXI GPIO IP was added to be able to read the input from the buttons that will eventually control the movement of the snake.



## Design flowchart snake game

When the hardware design was finished and verified, the next task was to write the software for the game. Which basically meant it was finally time to start designing the snake game itself. Since this is quite a big task and it is hard to keep a good overview of the progress it was immediately clear that a draft of the flow of the code was made. For this the flowcharts in “Appendix: Flowchart” have been designed. Although the final code differs a bit from these flowcharts, they had a big contribution to the design phase of the game and the general structure is still the same. The flowchart is structured in a way that the big distinctions of the software parts that are made while brainstorming all have a separate flowchart, to keep the result readable and easier to understand.

## Implementation software

As described before, the library handed example codes to use the hardware design. The first example shows a basic gradient on the screen. By testing this the requirement to display an image on the monitor through HDMI was passed. The next example shows how to program a simple animation of a block. This example provided the base on which the snake game is programmed.

The flowcharts are used a base for the software design. During the programming of the software, some changes were made to the structure to achieve easier to comprehend code.

The code itself is divided into game logic and video logic. The game logic reads the buttons, moves the snake, and checks for collisions. The video logic draws the snake and target, and switches between the active and inactive frames.

The game logic works as follows. The X and Y coordinates of the snake are stored in an array. According to the variable length, the program knows how many blocks it should draw. The program waits for a certain amount of time called a “turn” before doing anything. After this time, it checks if a button was pressed and changes the direction. Then the program tries to lengthen the snake by adding an extra block (a new head) in front of the snake in the chosen direction. If the new block doesn’t touch anything, the program draws the new block and removes the last block of the snake. If the new block touches the target/fruit, it draws the new block and doesn’t delete the last one. If the new block touches the wall or touches the body of the snake itself, the game goes to a game over screen and waits for a new input from a button to restart the game. At the end of the turn, a new turn starts and the program loops.

## Testing

The testing was done by attaching the HDMI TX port with a HDMI cable to a monitor. During the development of the software project, this was done multiple times to test the program and ensure that it works. During this process, the game was slowly improved upon. The Command Shell Console and breakpoints were used to deal with problems and errors.

As shown in the demo, the game works as intended and meets the requirements set at the beginning of the project.

## Results

In the end the project works as was expected at the beginning of the project. There is a fluent snake game displayed through HDMI and is playable with two buttons on the Zybo board. This is done with the use of a flowchart for clarity during the coding process, the HDMI fundamentals in the FPGA, the game itself through software on the microprocessor and the interaction between the two with AXI-stream.

There also are a few points of improvement. The buttons on the Zybo board are quite rigid, and it is hard to make quick movements with the snake when it starts to move faster. Also, there is no leaderboard or score displayed on the screen. This is not implemented because through the library the display is “drawn” pixel by pixel. This would mean that either the letters had to be plotted manually or already existing code for this must exist somewhere. Because it is not a fundamental part of the game it was decided to leave that out. These points could be improved.

All in all, the result of the project has been a success and meets the expectations at the beginning.

## Contribution table

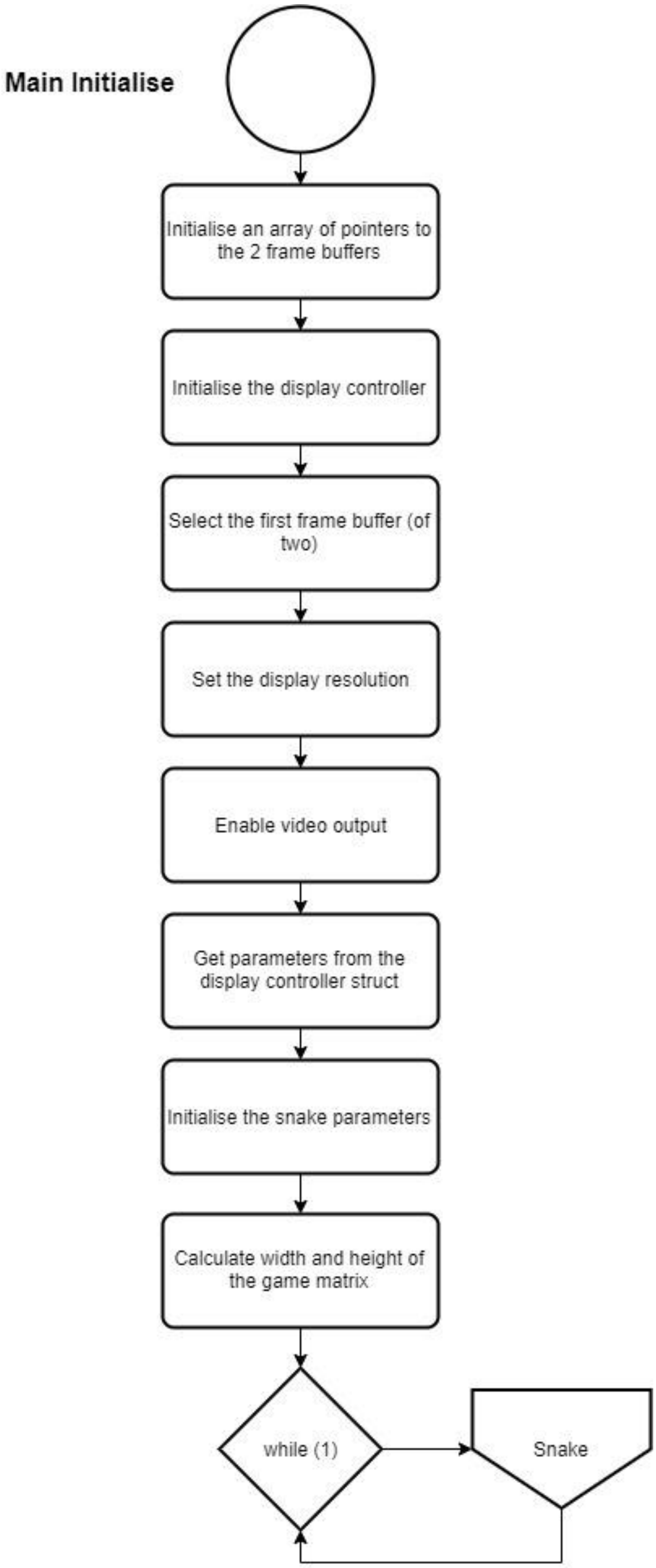
| **Task** | **Contribution** |
|---|---|
| Setup specifications | Casper, Jelle |
| Research on existing HDMI and games work | Casper, Jelle |
| Implementation hardware from HDMI library | Casper, Jelle |
| Test, setup hardware and setup initial software from HDMI library | Jelle |
| Design flowchart snake game | Mostly Jelle, a bit Casper |
| Implementation software | Mostly Casper, a bit Jelle |
| Testing | Casper, Jelle |
| Report | Casper, Jelle |

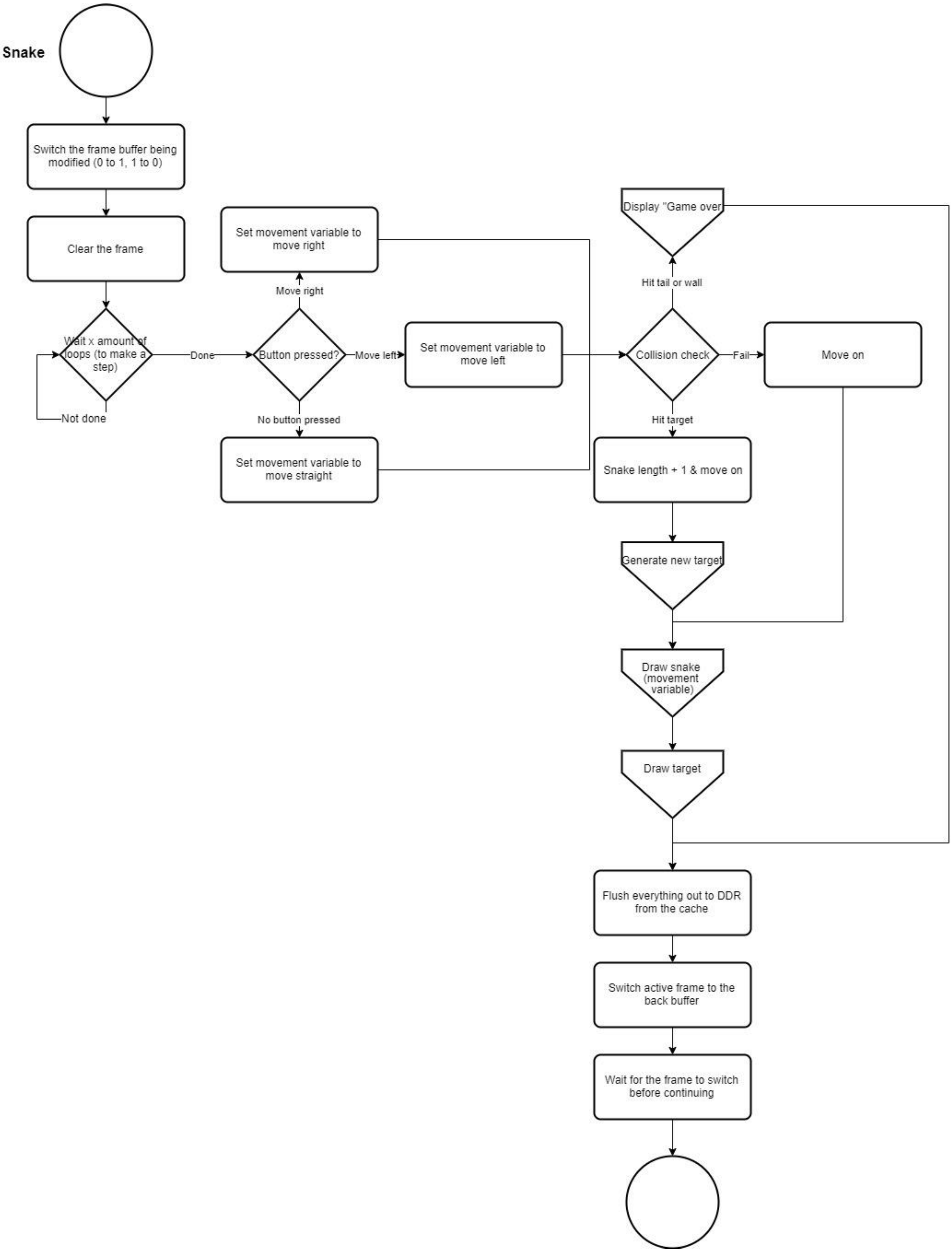
# Resources

[1] „Zybo Z7 HDMI Input/Output Demo," Digilent, [Online]. Available: https://digilent.com/reference/programmable-logic/zybo-z7/demos/hdmi. [Opened 2023].

[2] I. G. Russell Joyce, „Zybo Z7 HDMI Output," University of York, 14 3 2023. [Online]. Available: https://wiki.york.ac.uk/display/RTS/Zybo+Z7+HDMI+Output. [Opened 2023].

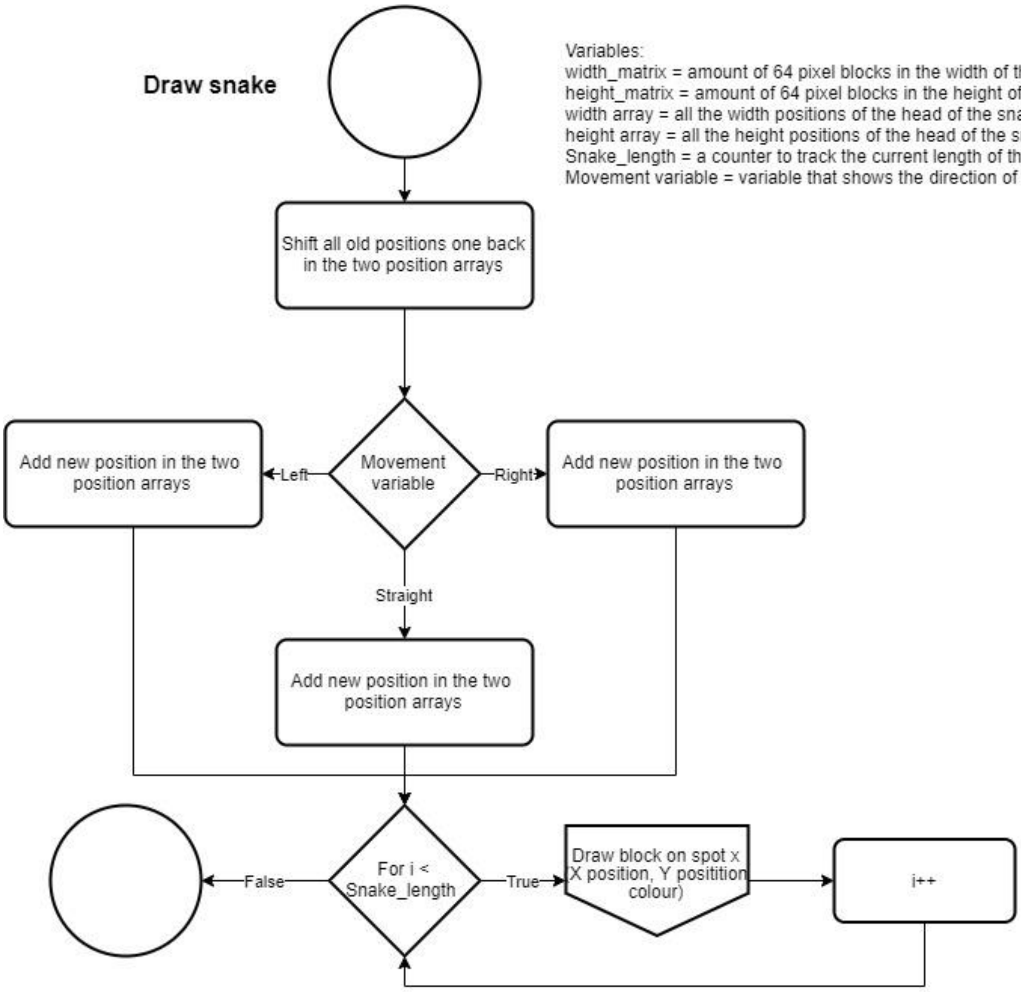
# Appendix: Flowchart


Main Initialise
Initialise an array of pointers to the 2 frame buffers
Initialise the display controller
Select the first frame buffer (of two)
Set the display resolution
Enable video output
Get parameters from the display controller struct
Initialise the snake parameters
Calculate width and height of the game matrix
while (1)
Snake


Snake
Switch the frame buffer being modified (0 to 1, 1 to 0)
Clear the frame
Wait x amount of loops (to make a step)
Not done
Done
Button pressed?
Move right
Set movement variable to move right
Move left
Set movement variable to move left
No button pressed
Set movement variable to move straight
Collision check
Hit tail or wall
Display "Game over
Fail
Move on
Hit target
Snake length + 1 & move on
Generate new target
Draw snake (movement variable)
Draw target
Flush everything out to DDR from the cache
Switch active frame to the back buffer
Wait for the frame to switch before continuing

## Draw snake

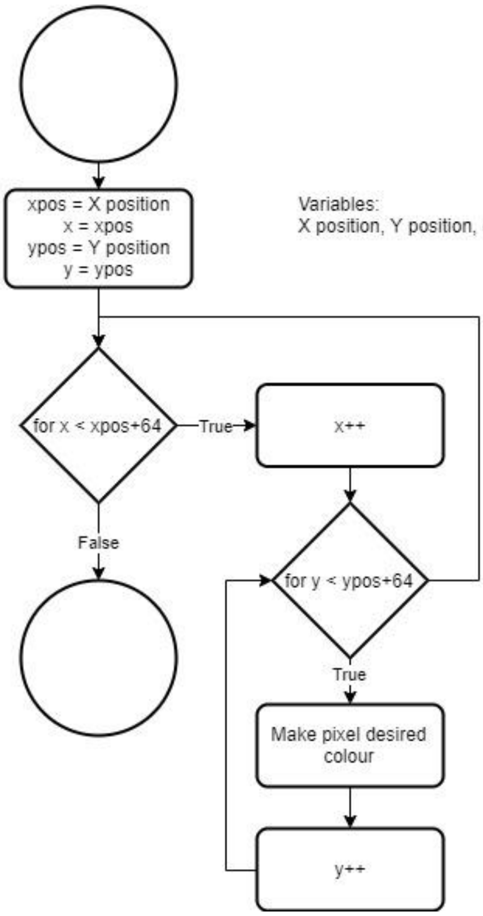
Variables:
width_matrix = amount of 64 pixel blocks in the width of the screen
height_matrix = amount of 64 pixel blocks in the height of the screen
width array = all the width positions of the head of the snake with the current one at 0
height array = all the height positions of the head of the snake with the current one at 0
Snake_length = a counter to track the current length of the snake
Movement variable = variable that shows the direction of the new step of the snake

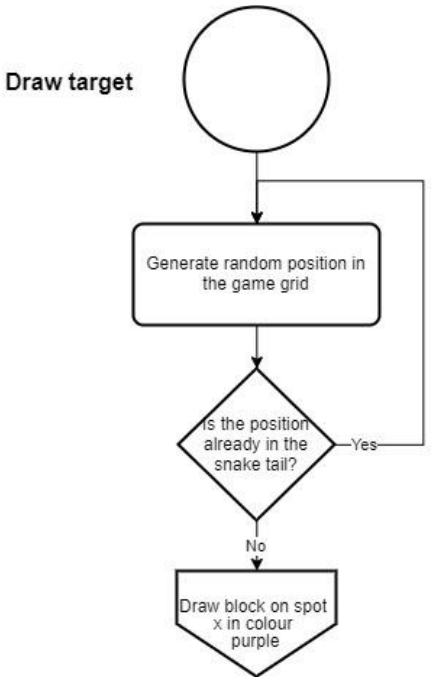


Shift all old positions one back in the two position arrays
Movement variable
Left
Right
Straight
Add new position in the two position arrays
Add new position in the two position arrays
Add new position in the two position arrays
For i < Snake_length
False
True
Draw block on spot x X position, Y positition colour)
i++

## Draw block on spot x

Variables:
X position, Y position, colour




Draw target
Generate random position in the game grid
Is the position already in the snake tail?
Yes
No
Draw block on spot x in colour purple